INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA MECÂNICA




DESENVOLVIMENTO DE UMA PROPOSTA DE MODELO DE GESTÃO E PRODUÇÃO ÁGIL ORIENTADO A COMUNIDADES DE PROJETOS ABERTOS (OPEN DESIGN)

FRANCISCA ISABEL MARTINS MADUREIRA
outubro de 2023

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

DESENVOLVIMENTO DE UMA PROPOSTA DE MODELO DE GESTÃO E PRODUÇÃO ÁGIL ORIENTADO A COMUNIDADES DE PROJETOS ABERTOS (*OPEN DESIGN*)

Francisca Isabel Martins Madureira

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica

P.PORTO


P.PORTO

DESENVOLVIMENTO DE UMA PROPOSTA DE MODELO DE GESTÃO E PRODUÇÃO ÁGIL ORIENTADO A COMUNIDADES DE PROJETOS ABERTOS (OPEN DESIGN)

Francisca Isabel Martins Madureira

1180639

Dissertação apresentada ao Instituto Superior de Engenharia do Porto para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizada sob a orientação do Doutor Hélio Cristiano Gomes Alves de Castro e coorientação do Doutor Rok Vrabič

2023

Instituto Superior de Engenharia do Porto

Departamento de Engenharia Mecânica


P.PORTO

AGRADECIMENTOS

Primeiramente, quero expressar a minha profunda estima e apreço ao meu orientador, Dr. Hélio Castro. A sua dedicação, ensinamentos e os valores que me transmitiu foram fundamentais para a concretização desta dissertação. Agradeço-lhe por todo o cuidado, atenção, ajuda e disponibilidade durante o desenvolvimento da minha dissertação.

Dirijo um especial reconhecimento à minha família, o verdadeiro pilar da minha existência. À minha mãe e irmãos, os meus melhores amigos, que sempre estiveram ao meu lado e me acudiram em todos os momentos, sei que a magnitude do vosso amor é incomparável.

À minha avó, que embora ausente fisicamente, sei que iluminou e que acompanhou todo o meu percurso. Muito do que sou deve-se a ela, os meus momentos mais felizes foram com ela que os passei. Estará para sempre nos meus pensamentos, um amor incondicional.

Ao meu namorado Artur, as palavras não são suficientes para conseguir me expressar. Foi fundamental nesta etapa, sendo o meu porto seguro em todas as horas. A presença incansável e o apoio inabalável foram essenciais. Nunca me deixou pensar em desistir, sendo sempre a minha força e o meu apoio.

À Bárbara, à minha parceira da faculdade e companheira de vida, agradeço por ter iluminado e ter tornado estes 5 anos do meu percurso académico mais leves e felizes. No meio de tudo, as memórias que construímos juntas estarão para sempre no meu coração. Ambas sabemos aquilo que vivemos, as lutas que enfrentamos e tudo ainda o que está para vir, e sei que é e será sempre juntas.

A todos os meus melhores amigos, Sofia, Camila, Joana, Sandy, André, Inês, Andreia, Babi, Diana, Txics e Rede, a minha eterna estima. A vossa amizade e presença tornaram este percurso mais alegre e reconfortante. Agradeço-lhes por serem aqueles amigos que, aconteça o que acontecer, sempre permanecem ao meu lado.

A vocês e por vocês, que de alguma forma enriquecem a minha vida, o meu muito obrigada.

página propositadamente em branco

RESUMO

Este trabalho tem como objetivo desenvolver uma proposta de modelo de gestão e produção ágil orientado a comunidades de projetos abertos (*Open Design*). *Open Design* é um movimento que procura promover a colaboração e a transparência na criação de projetos, e neste caso, aplicando-se os princípios ágeis. A agilidade surgiu como uma estratégia de resposta e adaptação à complexidade e ambientes incertos. A combinação desses dois conceitos permite uma melhor adaptação às mudanças e entrega de valor para o cliente de forma rápida e eficiente. Nesta dissertação, explora-se a convergência entre a agilidade e o conceito de Open Design, que enfatiza a colaboração, a participação e a transparência na criação de produtos, serviços e sistemas. Comunidades de *Open Design* reúnem-se para desenvolver projetos de forma colaborativa e aberta, compartilhando conhecimentos, recursos e ferramentas com o foco na inovação e objetivos comuns. Ao unir a abordagem de Open Design à gestão e produção ágil, este estudo visa melhorar a eficiência e eficácia dos projetos, especialmente aqueles que envolvem a colaboração de múltiplas partes interessadas. A integração dos princípios ágeis e das práticas de comunicação aberta e colaborativa permite criar um ambiente dinâmico, flexível e adaptável, orientado para o utilizador final. Isso possibilita que as equipas se ajustem rapidamente a mudanças e imprevistos, minimizando riscos e maximizando a eficiência. Além disso, a abertura e a colaboração entre as comunidades fornecem acesso a um vasto conjunto de especialistas e ideias. A experiência e a iteração contínua permitem melhorias constantes nos projetos. Salienta-se que a gestão eficaz de projetos abertos (Open Design) exige a compreensão e a adoção dos princípios da agilidade. Os objetivos desta dissertação incluem a compreensão das dinâmicas de gestão em comunidades de projetos abertos, a identificação dos desafios de gestão que enfrentam e a avaliação do impacto desses desafios na eficiência dos projetos. Além disso, examinam-se as métricas de sucesso em comunidades de código aberto e propõem-se a aplicação de um modelo de gestão ágil em uma plataforma de código aberto para avaliar sua relevância e eficácia. A metodologia adotada envolve uma análise detalhada da literatura existente e casos de estudo para compreender a dinâmica específica das comunidades de projetos abertos. Esta dissertação procura contribuir para a compreensão e o aprimoramento da gestão de projetos em comunidades de Open Design, promovendo a inovação e a eficácia desses projetos e, assim, fornecendo uma base sólida para a melhoria contínua no domínio da engenharia.

PALAVRAS-CHAVE

Open Design; *Innovation*; Produção Ágil; *Open Source*; Gestão Ágil de Projeto.

página propositadamente em branco

ABSTRACT

This work aims to develop a proposal for an agile management and production model oriented towards open project communities (Open Design). Open Design is a movement that seeks to promote collaboration and transparency in project creation, and in this case, applying agile principles. Agility emerged as a strategy for responding and adapting to complexity and uncertain environments. The combination of these two concepts allows for better adaptation to changes and delivers value to the client quickly and efficiently. In this dissertation, the convergence between agility and the concept of Open Design, which emphasizes collaboration, participation, and transparency in the creation of products, services, and systems, is explored. Open Design communities come together to develop projects collaboratively and openly, sharing knowledge, resources, and tools with a focus on innovation and common objectives. By joining the Open Design approach to agile management and production, this study aims to improve the efficiency and effectiveness of projects, especially those involving collaboration from multiple stakeholders. Integrating agile principles and practices of open and collaborative communication creates a dynamic, flexible, and adaptable environment, oriented towards the end user. This allows teams to quickly adjust to changes and unforeseen events, minimizing risks and maximizing efficiency. Moreover, the openness and collaboration among communities provide access to a vast array of specialists and ideas. Experience and continuous iteration allow for constant improvements in projects. It is emphasized that effective management of open projects (Open Design) requires understanding and adopting the principles of agility. The objectives of this dissertation include understanding the management dynamics in open project communities, identifying the management challenges they face, and assessing the impact of these challenges on project efficiency. Additionally, success metrics in open-source communities are examined, and the application of an agile management model on an open-source platform is proposed to evaluate its relevance and effectiveness. The adopted methodology involves a detailed analysis of existing literature and case studies to understand the specific dynamics of open project communities. This dissertation seeks to contribute to the understanding and enhancement of project management in Open Design communities, promoting the innovation and effectiveness of these projects, and thereby providing a solid foundation for continuous improvement in the engineering domain.

KEYWORDS

Open Design; Innovation; Agile Production; Open Source; Agile Project Management.

página propositadamente em branco

ÍNDICE

página propositadamente em branco

ÍNDICE DE FIGURAS

página propositadamente em branco

ÍNDICE DE TABELAS

página propositadamente em branco

LISTAS DE SIGLAS E SÍMBOLOS

Lista de Siglas

API	Application Programming Interface
APM	Agile Project Management
CI/CD	Continuous Integration/Continuous Delivery
DSR	Design Science Research
FDD	Feature Driven Development
IA	Inovação Aberta
ISEP	Instituto Superior de Engenharia do Porto
LSD	Lean Software Development
OD	Open Design
OHWR	Open Source Hardware Repository
OI	Open Innovation
OS	Open Source
OSD	Open Source Definition
OSHW	Open Source Hardware
OSI	Open Source Innovation
OSS	Open Source Software
PI	Propriedade Intelectual
PM	Project Management
PME´S	Pequenas e Médias Empresas
PR	Pull Request
TDD	Test-Driven Development
XP	Extreme Programming

página propositadamente em branco

1. INTRODUÇÃO

A presente dissertação faz parte do programa de Mestrado em Engenharia Mecânica, na área de Gestão Industrial, do Departamento de Engenharia Mecânica do Instituto Superior de Engenharia do Porto (ISEP).

Este capítulo apresenta uma contextualização dos temas abordados nesta dissertação, bem como os seus respetivos objetivos. Para além disso, também é descrita brevemente a metodologia e estrutura da dissertação.

1.1. Contextualização

A necessidade das empresas se adaptarem a ambientes em constante mudança e se tornarem mais competitivas, transforma a agilidade essencial para as empresas. As estratégias para alcançar esse conceito incluem a capacidade de resposta rápida e eficiente às necessidades do cliente e o uso de novas tecnologias para aumentar a flexibilidade e rapidez no mercado.

A metodologia ágil de gestão de projetos surgiu como uma resposta às limitações da abordagem tradicional de planeamento e controlo, que se mostrava ineficiente ao lidar com o ritmo acelerado das mudanças no mercado.

Por outro lado, o *Open Design* é uma abordagem que enfatiza a colaboração, participação e transparência da comunidade no projeto e desenvolvimento de produtos, serviços e sistemas, permitindo que a comunidade esteja ativamente envolvida, proporcionando-lhes a oportunidade de compartilhar ideias e *feedback*. Comunidades de projetos abertos são comunidades que se unem para desenvolver projetos de forma colaborativa e aberta, como redes e plataformas digitais e sociais. Estas são caracterizadas por compartilhar conhecimentos, recursos e ferramentas a fim de alcançar objetivos comuns e sempre com o pensamento na inovação. A colaboração entre os membros da comunidade garante que as soluções desenvolvidas sejam relevantes e adaptadas às necessidades reais dos utilizadores.

Juntar a abordagem de *Open Design* à gestão e produção ágil é uma forma eficaz de melhorar a eficiência e eficácia dos projetos, especialmente aqueles que envolvem a colaboração e a contribuição de várias partes interessadas. Ao combinar os princípios e práticas da agilidade com a colaboração de comunidades de projetos abertos (*Open Design*), é possível criar um ambiente dinâmico, flexível, adaptável e orientado ao usuário final, que permite à equipa adaptar-se rapidamente às mudanças e imprevistos. Dessa forma, o risco é minimizado e a eficiência do projeto aumenta. Ao utilizar metodologias ágeis e técnicas de comunicação abertas e colaborativas, as equipas podem trabalhar em conjunto para definir e alcançar objetivos comuns, garantir ainda a transparência e inclusão e melhorar a qualidade do projeto.

Além disso, *openness* e colaboração entre as comunidades permitem acesso a um amplo leque de especialista e ideias, e a experiência e iteratividade permitem à equipa desenvolver e melhorar continuamente o projeto. É importante ainda salientar que as equipas e organizações que utilizam esta abordagem precisam de estar preparadas para lidar com a complexidade e a incerteza inerentes a projetos abertos, e para garantir que todos os envolvidos estejam alinhados em torno dos objetivos e valores do projeto. Dessa forma, a sensibilização de todos sobre a agilidade é crucial.

1.2. Objetivos

O objetivo principal é investigar as características de gestão que uma comunidade de projetos abertos precisa para ter sucesso e, com base nessas descobertas, desenvolver uma proposta de modelo de gestão ágil para projetos desenvolvidos em comunidades digitais e abertas, onde os membros das comunidades são os principais agentes de impulso e motivação dos projetos em questão, utilizando os fundamentos, valores e técnicas da agilidade, além das abordagens e metodologias ágeis como base para a construção do modelo. Este objetivo geral pode ser decomposto em metas especificas:

- Compreender a dinâmica de gestão e produção nas comunidades de projetos abertos, identificando as peculiaridades que as distinguem de outros modelos de gestão.
- Identificar os principais desafios de gestão enfrentados por comunidades de projetos abertos e avaliar o impacto desses desafios na eficiência e eficácia dos projetos desenvolvidos.
- Realizar uma análise detalhada das métricas utilizadas para avaliar o sucesso em comunidades de projetos *open source*, enfatizando a sua relevância e aplicabilidade ao modelo de gestão ágil proposto.
- Aplicar o modelo proposto ao contexto de uma plataforma de código aberto e avaliar a sua relevância e eficácia na gestão de projetos *open source* nessa plataforma.
- Discutir as implicações do modelo proposto para a gestão de projetos *open source* e avaliar a sua contribuição para a inovação na gestão desses projetos na plataforma estudada.

1.3. Metodologia

A primeira fase da pesquisa envolve uma análise dos estudos já publicados, através da recolha de artigos científicos, para compreender a teoria que sustenta a investigação. O objetivo é sintetizar os principais conceitos e ferramentas teóricas relevantes para o trabalho. A revisão da bibliográfica também tem como objetivo avaliar o estado de arte da investigação no tema em questão.

No âmbito do *Open Design* e *Open Source*, a metodologia de casos de estudo emerge como uma ferramenta crucial para examinar a gestão e produção em comunidades de projetos abertos. Esta abordagem permite uma investigação detalhada das dinâmicas específicas que caracterizam tais comunidades, facilitando a compreensão dos desafios e oportunidades inerentes ao *design* e desenvolvimento abertos. Os casos de estudo, ao centrarem-se em situações reais e contextos específicos, proporcionam perspetivas valiosas sobre como o *design* aberto e o *open source* interagem e se influenciam mutuamente. Esta metodologia não só enriquece o conhecimento acerca do fenómeno estudado, mas também oferece a possibilidade de expandir aprendizagens e práticas para outros ambientes ou contextos semelhantes no domínio do *Open Design* e *Open Source.*

1.4. Estrutura da Dissertação

A dissertação é organizada por 5 principais capítulos.

No primeiro capítulo, o da Introdução, é apresentado de forma resumida o tema abordado na dissertação. Os objetivos principais que serão alcançados ao longo do trabalho serão descritos e,

finalmente, a metodologia usada para compreender o estado de arte relativamente ao tema será explicada.

O segundo, da Revisão Bibliográfica, são explorados conceitos-chave e literatura relevante para o estudo. O capítulo aborda os conceitos fundamentais que dão suporte teórico à dissertação são apresentados, conceitos como *open design*, *open innovation*, gestão ágil de projetos e gestão ágil de projeto orientado a comunidades. A ideia não é fornecer uma descrição exaustiva destes conceitos, mas sim incluir os detalhes necessários para uma compreensão, complementando com um conjunto de referências que cobrem a teoria necessária para o desenvolvimento do modelo apresentado.

O terceiro capítulo (Avaliação das Comunidades *Open Source*), concentra-se na avaliação e análise das comunidade de código aberto. Aborda as métricas relevantes para avaliar o sucesso destas comunidades, explora o Modelo *Onion* e a estrutura das comunidades Open Source, e examina em detalhe a Comunidade *GitHub* e o *Open Hardware Repository*. Além disso, discute o processo de recolha de dados e os dados extraídos das plataformas.

No quarto capítulo (Análise dos Dados Obtidos e Discussão), os dados recolhidos nas etapas anteriores são aqui apresentados e analisados. A discussão centra-se na interpretação dos resultados, oferecendo *insights* e compreensões sobre as implicações dos mesmos e do seu impacto no contexto global do estudo.

No capítulo final (Conclusão) são apresentadas as conclusões derivadas do estudo, proporcionando uma síntese dos principais resultados. Para além disso, são discutidas as limitações do estudo e possíveis direções para trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

Neste capítulo, apresentam-se os princípios e conceitos fundamentais que servem como base para a investigação em foco. A revisão bibliográfica abrange uma série de áreas críticas que moldam o contexto da dissertação. Inicialmente, explorou-se o universo das Comunidades de Projetos Abertos, também conhecidas como Open Design, destacando a sua relevância e impacto na inovação colaborativa. Em seguida, examinou-se a Inovação Aberta, destacando os conceitos de *Openness* e *Open* como impulsionadores da transparência e colaboração. Aprofundou-se a Gestão Ágil de Projetos como resposta à dinâmica do mercado, abordando as suas metodologias ágeis. Finalmente, explorou-se como a Gestão Ágil se conecta às Comunidades de Projetos Abertos, adaptando-se à gestão eficiente em ambientes colaborativos.

A revisão bibliográfica estabelece um alicerce sólido para compreender as interações entre comunidades abertas, gestão ágil e a cultura de compartilhamento de informações que impulsiona a inovação em ambientes mutáveis.

2.1. Open Design

Open Design (OD) é um processo impulsionado pela comunidade em rápido crescimento que depende de plataformas digitais para gerar ideias e soluções numa área ou num campo específico. O *Open Design* significa projetos digitais de acesso aberto que podem ser adaptados de forma livre para atender aos requisitos e, posteriormente, podem ser usados pelos consumidores para fabricar produtos por métodos de produção comerciais e prontos a usar (Avital, 2011). Este pode ser encontrado em vários setores, incluindo *software*, *hardware*, ciência, tecnologias e até negócios. Uma das plataformas mais conhecidas do *Open Design* é a *Wikipédia*, uma enciclopédia aberta com conteúdo que pode ser editado (Castro et al., 2019).

O termo "*open"* aparece em vários conceitos e termos diferentes. Embora o uso deste termo em *Open Design* não implique necessariamente domínio público, este pode abranger várias formas de propriedade intelectual e direitos que podem ser assumidos (Castro et al., 2019). Quanto mais aberto for o processo de *design*, mais este se vai encaixar naquilo que é pertencer a um domínio público. Já o termo "*design*" pode ser usado de quatro maneiras diferentes: como substantivo referente ao campo do *design* em geral, como verbo referente ao ato de projetar, como substantivo referente a um plano ou intenção e como substantivo referente ao produto finalizado (Tooze et al., 2014).

O *Open design* é acessível à comunidade, no sentido de se poder participar, usar e construir sobre os resultados já produzidos, o que o torna uma força democratizante na sociedade (Aitamurto et al., 2015). Esta abordagem de *design* promove a inovação e evita a estagnação. É baseado no conceito do *open source design* e envolve o uso da internet para redefinir os direitos da propriedade intelectual e industrial e direitos, para facilitar o desenvolvimento colaborativo de *hardware* e *software*. O número de projetos e plataformas de *open source of software design* indica que essa área é mais ativa e difundida do que o *hardware design* (Castro et al., 2019).

Os princípios do *Open Design* são frequentemente estabelecidos por organizações como a *Open Source Initiative* e o *Open Design Working Group*, que estabelecem condições para o uso e distribuição de produtos de *Open Design* (Tooze et al., 2014).

Segundo Aitamurto et al. (2015), num processo de *Open Design* é fornecido acesso público para contribuir com o processo de *design* e para o acesso ao produto final, bem como a quaisquer dados e conteúdos desenvolvidos durante o processo, incluindo especificações técnicas e outras informações. Esta abordagem permite transparência durante o processo de *design*. Assim é definido o *Open Design* como uma prática que combina a *openness* do produto e processo e ainda combina com a definição de *Open Source Innovation* que é uma das categorias de práticas dentro do *Open Design* (Bonvoisin et al., 2021).

A estrutura de *Open Design* envolve um processo contínuo de *design*, ajuste e *redesign* que pode levar à transformação. Este processo conta com o envolvimento da comunidade e requer colaboração e partilha de processos de aprendizagem entre os seus membros. Apoiado numa plataforma digital, o *Open Design* é um processo generativo com alto potencial de crescimento através do desenvolvimento técnico e práticas de diálogo. Iniciativas e movimentos abertos, como *Open Source Iniative* e *Open Source Hardware*, são exemplos de iniciativas de partilha de alto nível que operam sem limites de tempo ou lugar e priorizam o compromisso com a comunidade e a sociedade como um só (Castro & Ávila, 2020).

O *Open Design* compreende quatro etapas – planeamento do produto, *design* colaborativo, criação e desenvolvimento de modelos e otimização, como se pode observar pela Figura 1 (Rebensdorf et al., 2015).

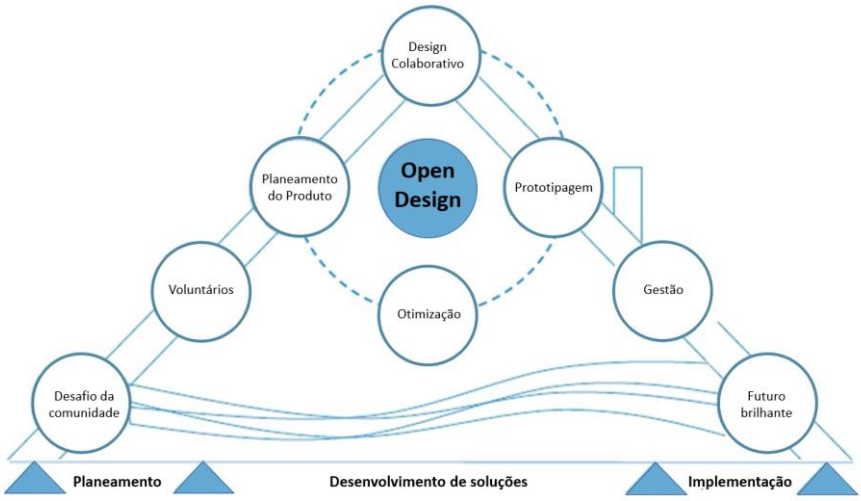


Figura 1 - Etapas do processo *Open Design*. Adaptado de Rebensdorf et al. (2015)

Alguns exemplos de produtos desenvolvidos através da abordagem *Open Design* incluem *Farmbot*, uma empresa privada que usa esta abordagem para a criação de produtos, *Open Source Imaging Iniative*, uma comunidade que aplica o *Open Design* na área da *Open Science*, e *e-NABLE*, uma comunidade que recorre ao *Open Design* para iniciativas sociais (Castro & Ávila, 2020).

As várias questões subjacentes do *Open Design* podem ser organizadas e compreendidas em quatro camadas conceituas interdependentes, como se segue (Avital, 2011):

- ***Object layer*** - Refere-se aos planos e diretrizes de *design* que moldam a criação de objetos de *design*. Esta camada inclui a disponibilidade de *blueprints* de *Open Design*, que são modelos de *design* personalizáveis e expansíveis que permitem livre acesso através de repositórios públicos *online* sob uma licença de acesso aberto;

- ***Process layer*** - Refere-se aos métodos e ferramentas usados para produzir objetos de *design*. Inclui o fabrico de *Open Design*, que se refere ao uso de máquinas comerciais prontas a usar para criar produtos personalizados sem a necessidade de moldes ou máquinas especializadas;
- ***Practice layer*** - Refere-se às práticas e comportamentos que moldam o desenvolvimento dos processos de *design*. Inclui a cultura de *Open Design*, que abrange a terminologia, os padrões profissionais, as técnicas, as regras e os valores que fazem parte da área deste;
- ***Infrastructure layer*** - Refere-se aos sistemas e tecnologias subjacentes que suportam a viabilidade das práticas de *design*. Inclui a infraestrutura de *Open Design*, que abrange a estrutura legal, a estrutura de mercado e a arquitetura técnica que regulam e influenciam as atividades de *Open Design* e o desenvolvimento futuro.

O *Open Design* envolve a criação de desenvolvimentos conjuntos que são protegidos de perder a sua abertura e a capacidade de continuar a ser desenvolvido, através do uso de um tipo específico de licença. A *Open Design Foundation* estabelece diretrizes para essas licenças que são semelhantes aos direitos fornecidos pelas licenças do *Open Source Software*, como se mostra através da Figura 2. Para ser eficaz, uma licença *Open Design* deve exigir documentação contínua e disponível gratuitamente do processo de desenvolvimento, identificação clara do sistema *Open Design* quando distribuído junto com sistemas proprietários e documentação e acessibilidade de quaisquer modificações feitas no sistema (Tooze et al., 2014).

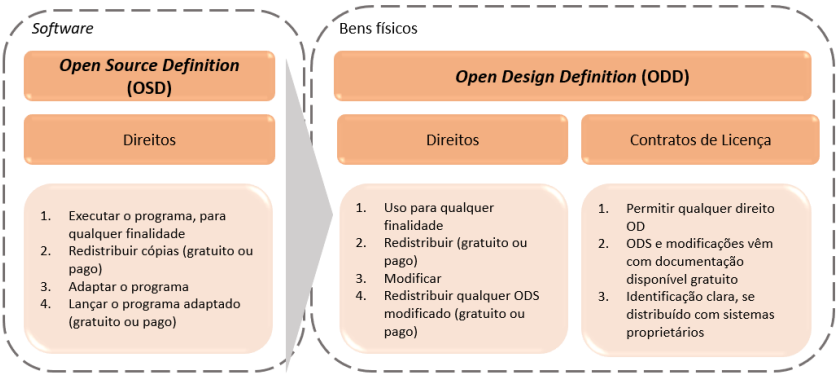


Figura 2 - Direitos de acesso e uso do open design. Adaptado de Wulfsberg et al. (2011)

O princípio fundamental do *Open Design,* que também pode ser referido como *Open Content* ou *Free Culture Works,* é lançar os conteúdos muito cedo, de acordo com a frequência de ocorrência. Esta abordagem tem dois benefícios: deteção precoce de erros e falhas e respetiva correção numa fase inicial, e melhor qualidade e velocidade de produção (Rebensdorf et al., 2015).

As práticas de *Open Design* têm o potencial de contribuir com o processo de *design* de várias formas. Quando as soluções são colaborativas e cocriadas, mais soluções para o desafio de *design* podem ser apresentadas do que num processo fechado. Essa variedade de opções ajuda um *designer* a encontrar a solução ideal, economizando tempo e dinheiro, e a entrada de utilizadores e *designers* pode melhorar o resultado (Rebensdorf et al., 2015).

Para facilitar a comunicação clara sobre o *Open Design*, é útil ter um conjunto de termos definidos que possam ser usados consistentemente. Assim, o conjunto de termos relacionados ao *design* inclui (Tooze et al., 2014):

- ***Open design solution*** - Um conjunto de planos e instruções que permite que outros possam usar as informações do projeto ao fazer ou modificar o projeto sem restrições;

- ***Open design contribution*** - Qualquer contribuição, em qualquer formato, para um processo de *design* que seja disponibilizado para uso por outros sem restrições;
- ***Open design process*** - Desenvolvimento de uma solução ou soluções de *design* que são criadas pela entrada de contribuições de *Open Design* e resulta numa solução ou soluções de *Open Design*;
- ***Open designing*** - Envolvência no *design* de qualquer coisa por um processo de *Open Design*;
- ***Open design project*** - Qualquer projeto que segue um processo de *Open Design*.

De forma a entender os componentes abertos de um processo de *design* e a sua respetiva explicação, pode ser útil a sua representação visual das várias partes e possíveis variações. A Figura 3 ilustra esse conceito, onde mostra os modelos simplificados de como as soluções de *open* e *non-open design* podem ser criadas através de contribuições abertas e não abertas. As contribuições representam-se por segmentos, enquanto as soluções são representadas pelos círculos preenchidos. As setas indicam a direção do tempo, já os círculos a cinza destacam os casos em que todas as contribuições são publicadas publicamente como parte do desenvolvimento de uma *open solution* (Tooze et al., 2014).

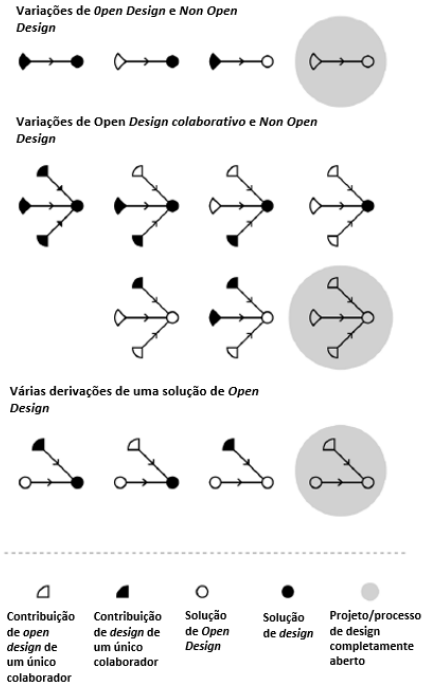


Figura 3 - Diversas partes e variações do Open Design. Adaptado de Tooze et al. (2014)

O uso do *Open Design* pode ser uma oportunidade valiosa para empreendedores e empresas ágeis para a obtenção de uma forte presença no mercado e potencialmente até superarem os atuais líderes de mercado. O *Open Design* pode facilitar na mudança dos modelos tradicionais de negócios "*push*" para os modelos "*pull*" mais modernos (Avital, 2011).

2.2. Open Innovation

No mercado atual, com a diante crescente concorrência global e com o aumento dos custos de Pesquisa e Desenvolvimento (P&D) e a redução dos ciclos de vida dos produtos, o modelo de inovação fechada já não é mais suficiente para o sucesso das empresas. Dessa forma, para se

manterem competitivas, as empresas sabem que é necessário recorrer a fontes externas de conhecimento e colaborar com indivíduos, empresas e outras organizações com conhecimentos relevantes como parte do seu processo de inovação (Saebi & Foss, 2015).

A propagação de tecnologias de informação e rede, assim como os custos reduzidos de comunicação e coordenação, tornaram mais fácil para as empresas procurar e alavancar conhecimento externo de todo o mundo através de parcerias e colaborações. A perceção pela parte das empresas de que o conhecimento necessário para a inovação é amplamente disperso e difícil de transferir levou à adoção de modelos de inovação aberta (Afuah & Tucci, 2012; Saebi & Foss, 2015).

A Inovação Aberta é uma estratégia que envolve a partilha de informações proprietárias ao público, permitindo que um número maior de pessoas participe dos processos de solução de problemas (Forbes & Schaefer, 2017). O paradigma da Inovação Aberta assume o princípio de que a inovação não deve ser baseada apenas em esforços internos e isolados das empresas.

A premissa básica de Inovação Aberta é abrir o processo de inovação. Uma das definições mais comuns é o uso intencional de adquirir e partilhar conhecimento para acelerar o desenvolvimento de novos produtos e serviços internamente, bem como expandir a utilização dessas inovações para mercados externos. Por sua vez, no modelo fechado de inovação, os projetos de pesquisa são lançados a partir da base da ciência e tecnologia, como se observa na Figura 4 (H. Chesbrough, 2012).

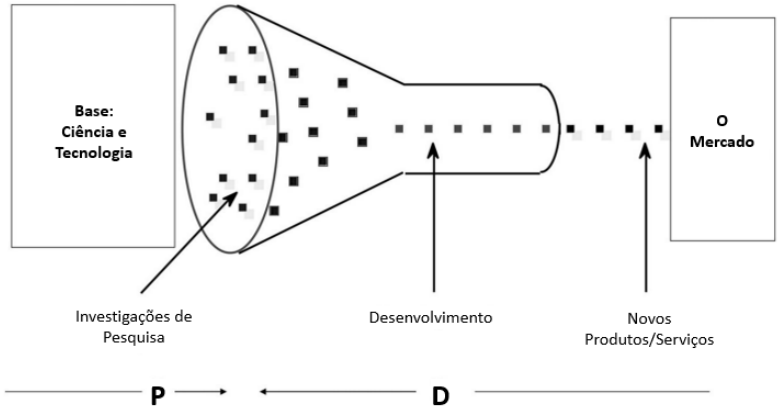


Figura 4 - Modelo de inovação fechado. Adaptado de H. Chesbrough (2012)

Os projetos progridem através do processo de desenvolvimento, e alguns projetos são interrompidos enquanto outros são selecionados para trabalho posterior e os projetos de sucesso são escolhidos para entrar no mercado. Esse processo tradicional de inovação é fechado, uma vez que os projetos só podem entrar de uma forma, pela base interna da empresa, e só podem sair por uma via, com a entrada no mercado (H. Chesbrough, 2012).

No modelo de Inovação Aberta, ao contrário, os projetos podem entrar ou sair em vários pontos e de várias maneiras, como se pode observar na Figura 5. Aqui, os projetos podem ser lançados a partir de fontes de tecnologia internas ou externas, e a nova tecnologia pode entrar no processo em várias fases. Uma abordagem aberta pode promover o processo de inovação da empresa em

diferentes fases através de diversas formas, como *spin-off*[1] de novos projetos, licenciamento de propriedade intelectual (PI) de ideias não desenvolvidas, busca de financiamento externo de fontes públicas ou privadas, entre outros (Melo et al., 2020).

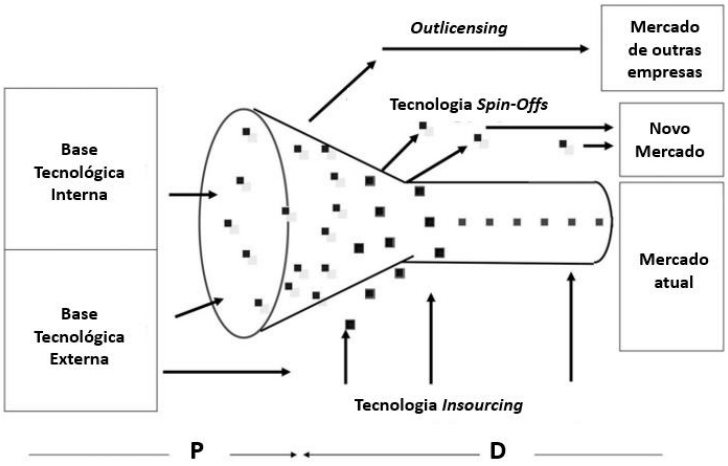


Figura 5 - Modelo de inovação aberta. Adaptado de H. Chesbrough (2012)

Existem diversos estudos que são inspirados pela mudança organizacional, em que as barreiras organizacionais e a inércia precisam de ser ultrapassadas, aquando da mudança de Inovação Fechada para a Inovação Aberta. A Tabela 1 resume as principais diferenças entre o paradigma de inovação fechada e o paradigma de inovação aberta (Greco et al., 2017).

Tabela 1 - Paradigma entre a Inovação Aberta e a Inovação Fechada. Adaptado de Greco et al., (2017)2

Inovação Aberta	Inovação Fechada
Tanto o conhecimento externo quanto o interno são igualmente importantes	Conhecimento externo complementa o conhecimento interno de uma empresa
É essencial o desenvolvimento de modelos de negócios que efetivamente transformem a pesquisa e o desenvolvimento em valor	Investir e contratar os melhores pode impulsionar a inovação dentro da empresa
Gerir o fluxo de conhecimento e tecnológico é crucial	Spillovers3 podem ser prejudiciais e é importante prevenir a saída de conhecimento
A maior parte do conhecimento desenvolvido em qualquer área existe fora dos limites de uma empresa	Os esforços internos de P&D de uma empresa podem fornecer uma vantagem competitiva sem a necessidade de colaboração externa
A gestão de propriedade intelectual também pode desempenhar um papel proativo e diferenciado	A gestão de propriedade intelectual pode ser usada como medida defensiva

Na Figura 6, pode-se observar o processo para adoção de Inovação Aberta, sendo que este processo possui três etapas: *unfreezing, moving* e *institutionalizing* (Melo et al., 2020).

[1] *Spin-offs* são empresas criadas a partir de uma empresa existente como forma de explorar novas oportunidades.
[2] Tradução do autor
[3] No contexto de inovação fechada, um *spillover* pode se referir a como a inovação pode afetar outras empresas ou indústrias.

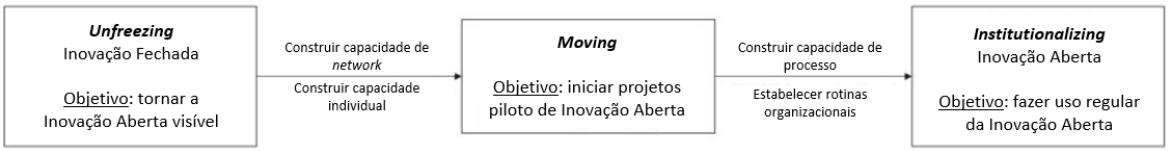


Figura 6 - Modelo de transição de inovação fechada para inovação aberta. Adaptado de Melo et al. (2020)

Segundo H. W. Chesbrough (2003), a Inovação Aberta é definida como uma prática em que uma empresa utiliza fontes de ideias, tanto internas quanto externas, e tanto caminhos internos quanto externos de forma a levar essas ideias para o mercado. Lichtenthaler (2008) também refere que o conceito IA consiste em confiar nas capacidades da empresa em realizar as tarefas fundamentais de gestão de tecnologia, tanto interna como externamente, incluindo a aquisição de tecnologia e o seu uso, junto com o desenvolvimento de novos processos inovadores. A Inovação Aberta pode fornecer uma base de conhecimento mais ampla e criar oportunidades estratégicas para as empresas melhorarem a eficácia da inovação (Chen & Liu, 2019; Hung & Chou, 2013).

A abordagem de Inovação Aberta é frequentemente usada no *Social Product Development*, onde dados e ferramentas são disponibilizados ao público. Embora isso possa representar um risco, os benéficos potenciais são enormes. Um exemplo bem conhecido é o *Goldcorp Challenge*, lançado por Rob McEwan, que depois da sua empresa, a *Goldcorp*, lutou para obter lucro. Tradicionalmente, na indústria de exploração de ouro, os dados geológicos eram considerados confidenciais e guardados em segurança. Rob McEwan, inspirado pelo sucesso do sistema operacional de *Open Source Linux*, divulgou uma grande quantidade de dados para o público e como consequência da Inovação Aberta, a *Goldcorp* tornou-se uma empresa multibilionária (Forbes & Schaefer, 2017). A Figura 7 representa o modelo conceitual de Inovação Aberta.

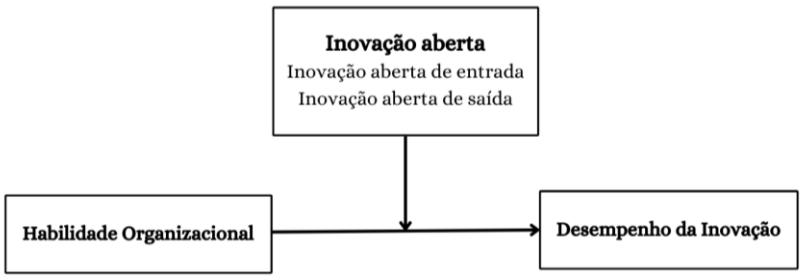


Figura 7 - Modelo conceitual de Inovação Aberta. Adaptado de Li et al. (2020)

Na Tabela 2 estão referidas algumas das definições de Inovação Aberta. As definições existentes referentes ao conceito enfatizam três pontos essenciais para a compreensão deste. Em primeiro lugar, Inovação Aberta engloba uma variedade de práticas que facilitam tanto a entrada como a saída deliberada de conhecimento. Em segundo lugar, a Inovação Aberta bem-sucedida requer um certo nível de permeabilidade dos limites organizacionais e do processo de inovação, de acordo com a pesquisa. Em terceiro lugar, as definições de Inovação Aberta são geralmente amplas para incluir uma variedade de estratégias diferentes, conforme descrito por Huizingh (2011), que descreve a Inovação Aberta como um “"guarda-chuva" que abrange, conecta e integra uma variedade de atividades já existentes"(p.3). Assim, as definições referidas na Tabela 2, são inclusivas para que diferentes tipos de iniciativas se encaixem no conceito de Inovação Aberta.

Tabela 2 - Definições de Inovação Aberta. Adaptado de Saebi & Foss (2015)

Estudo (ano)	Definição de Inovação Aberta
(H. Chesbrough, 2006, p. 1)	"A Inovação aberta é o uso de fluxos de conhecimento intencionais para acelerar a inovação interna e ampliar os mercados para uso externo da inovação, respetivamente. [Este paradigma] pressupõe que as empresas podem e devem usar ideias externas, bem como ideias internas e caminhos internos e externos para o mercado, ao procurarem avançar suas tecnologias."
(Gassmann & Enkel, 2004, p. 2)	"Inovação aberta significa que a empresa precisa abrir suas fronteiras sólidas para permitir que o conhecimento valioso flui de fora para dentro, a fim de criar oportunidades para processos de inovação cooperativos com parceiros, clientes e/ou fornecedores. Também inclui a exploração de ideias e propriedade intelectual para levá-las ao mercado mais rápido do que os concorrentes conseguem."
(Dittrich & Duysters, 2007, p. 512)	"O sistema é chamado de aberto porque as fronteiras do funil de desenvolvimento de produtos são permeáveis. Algumas ideias dos projetos de inovação são iniciadas por outras partes antes de entrar no funil interno; outros projetos deixam o funil e são desenvolvidos mais adiante por outras partes."
(Perkmann & Walsh, 2007, p. 259)	"Isso significa que a inovação pode ser considerada como resultado de redes interorganizacionais distribuídas, em vez de uma única empresa."
(West & Gallagher, 2006, p. 320)	"Definimos a inovação aberta como o estímulo sistemático e a exploração de uma ampla variedade de fontes internas e externas de oportunidades de inovação, a integração consciente dessa exploração com as capacidades e recursos da empresa, e a exploração ampla dessas oportunidades através de múltiplos canais."

A Inovação Aberta abrange as dimensões de entrada e saída do processo de inovação. A Inovação Aberta de Entrada refere-se a uma variedade de estratégias que incorporam conhecimento externo para uma empresa, como examinar o ambiente externo à procura de ideias, adquirir tecnologia com base no mercado, envolver-se em *crowdsourcing*[4], participar em concursos de inovação, formar parcerias, participar de alianças de pesquisa e desenvolvimento e ingressar em redes para coordenar atividades inovadoras. A Inovação Aberta de Entrada envolve a procura de novos e diversos conhecimentos e tecnologias fora dos limites de uma empresa para resolver problemas e aumentar a capacidade de inovação. Esta abordagem permite que as empresas possam aceder a conhecimento e tecnologias que não possuem ou que seriam muito dispendiosas, demoradas ou difíceis de desenvolver internamente (Greco et al., 2016).

Ao incorporar conhecimento externo, as empresas podem encontrar a resolução a certos problemas. Estudos anteriores ainda mostraram que as empresas que adquirem recursos externos são mais propensas a aumentar a sua capacidade de inovação quando trabalham além das fronteiras organizacionais e tecnológicas (Hung & Chou, 2013; Wassmer et al., 2017).

Uma empresa pode utilizar a Inovação Aberta de Entrada como um recurso estratégico, uma vez que ajuda na obtenção de recursos e uma base de conhecimento mais ampla. Quando uma empresa possui um amplo nível de Inovação Aberta de Entrada, não está só sintonizada com as oportunidades no seu ambiente tecnológico, mas também mais proativa em alavancar essas oportunidades através de fontes internas e externas de conhecimento e tecnologia (Rothaermel & Alexandre, 2009). Assim, a Inovação Aberta de Entrada refere-se a uma multiplicidade de

[4] *Crowdsourcing* é um termo que se refere ao processo de obtenção de contribuições de uma grande quantidade de pessoas, geralmente através da Internet, para realizar uma tarefa ou resolver um problema.

estratégias de Inovação Aberta e essas mesmas estratégias podem exigir diferentes modelos de negócios subjacentes (Saebi & Foss, 2015).

Por outro lado, a Inovação Aberta de Saída envolve uma empresa que usa deliberadamente o conhecimento tecnológico fora dos seus limites. Essa abordagem permite que uma empresa comercialize o seu conhecimento tecnológico ou colabore com outra organização independente para explorá-lo (H. Chesbrough et al., 2006). As empresas que são excessivamente protetoras da sua propriedade intelectual podem perder oportunidades. Quando uma empresa não possui o conhecimento de mercado necessário ou recursos complementares para utilizar as suas tecnologias internamente, explorar a tecnologia externamente permite que ela comercialize ativos de propriedade intelectual não utilizados e use o seu conhecimento tecnológico existente fora de seus limites (Li et al., 2020).

A Inovação Aberta de Saída envolve as atividades de inovação para impulsionar as capacidades tecnológicas existentes externas, enquanto a Inovação Aberta de Entrada refere-se à inclusão de conhecimento externo nos processos internos de uma empresa. Esta abordagem permite que uma empresa comercialize conhecimento tecnológico ou co-explore esse conhecimento com outra organização independente, aumenta a taxa de sucesso da inovação e reduz os custos de P&D para as empresas (H. Chesbrough et al., 2006).

2.2.1. Openness

Openness tornou-se um tema popular nas discussões sobre tecnologia, aparecendo em conceitos como *Open Source*, Inovação Aberta e *Open Design*. As qualidades associadas à *openness* facilitam a criatividade, a inovação e a prosperidade (Avital, 2011).

O termo *openness* refere-se à medida em que algo é facilmente acessível para visualização, modificação e uso. Isso inclui a partilha de conteúdo e informações, permitindo que sejam feitas mudanças e melhorias e permitindo a sua reutilização. De uma perspetiva de sistemas, *opennes* relaciona-se com a transparência e permeabilidade das fronteiras. No entanto, não é apenas uma característica técnica, mas também um conceito que está presente numa sociedade saudável e próspera. Este promove a partilha, a colaboração, a tolerância, a equidade, a justiça e a liberdade. A ascensão da aplicação do *openness*, ou "Open-X", tornou-se uma tendência importante que afeta todos os níveis da sociedade e materializou-se como Inovação Aberta, *Open Source* e *Open Design* (Avital, 2011).

Os três paradigmas apresentam-se na Tabela 3 como uma visão geral preliminar para apontar as suas diferentes proposições de valor e impulso, o cerne do conceito *openness* e os principais envolvidos. Compreender essas megatendências pode ajudar no planeamento para o futuro e na formação de ações atuais de forma a antecipar esse futuro (Avital, 2011).

Tabela 3 - Três Paradigmas do Open-X. Adaptado de Avital (2011)

	Inovação Aberta	Open Source	Open Design
Proposta de valor e Impulso	Conhecimento distribuído	Desenvolvimento distribuído	Fabricação distribuída
Cerne do conceito openness	Visualizar	Modificar	Usar
Principais envolvidos	Organizações	Comunidade de criadores	Consumidores

O conceito *openness* num processo é uma questão da procura pela interação com pessoas externas, que podem ser clientes, cidadãos ou funcionários de outras empresas. Pelo contrário, a *openness* de um projeto é uma questão de gestão de informações e propriedade intelectual aplicada aos objetos intermediários e aos resultados de um processo de design. Neste conceito, o foco não é só sobre o que os colaboradores podem fazer dentro do processo de *design* comum com as informações que podem aceder desse processo, mas mais sobre o que eles podem fazer fora desse processo com essas informações, ou seja, replicar o produto e vendê-lo por conta própria (Bonvoisin et al., 2021).

Openness parece ser um conceito gradual e multidimensional (Balka et al., 2010; Bonvoisin & Mies, 2018) que não está claramente definido. Existem três aspetos inerentes ao conceito *openness*: transparência, acessibilidade e replicabilidade. Na Tabela 4 apresentam-se as definições destes três fatores de *openness*. Embora a estrutura abordada por Balka et al. (2014) seja útil para entender este conceito, permanece ainda assim um tanto confusa e multifacetada, e não há evidências concretas de que reflita com precisão o que é praticado na verdade. Por exemplo, no fator de replicabilidade, embora se aborde o termo comunidade, ainda não está claro o que é a comunidade (Bonvoisin et al., 2021).

Tabela 4 - Fatores openness. Adaptado de Balka et al. (2014)

Fator de openness	Definição
Transparência	"[…] a possibilidade e o direito de "olhar" para o design. Para ver que eles são distintos, basta pensar na prática de engenharia reversa (possibilidade, mas muitas vezes não correta) ou de designs sendo compartilhados livremente em formatos complicados (correta, mas efetivamente poucas possibilidades)."[5]
Acessibilidade	"[…] é definida como a possibilidade e o direito da comunidade de aceder e modificar informações de projeto. A possibilidade depende de um formato de design acionável, o direito pode ser instituído por uma licença aberta, por exemplo, a General Public License ou Creative Commons. A acessibilidade não exige que cada membro da comunidade tenha habilidades, ferramentas e outros pré-requisitos que o tornem efetivamente capaz de se envolver no codesenvolvimento."[6]
Replicabilidade	"[…] definida como a possibilidade e o direito da comunidade e produzir partes do design. A replicabilidade é criada pela disponibilidade de componentes individuais (fazer ou comprar), o fornecimento de informações necessárias sobre o layout e montagem do projeto e pela ausência de barreiras legais que impeçam a autoprodução ou a automontagem."[7]

2.2.2. Open

- **Produção Aberta**

A Produção Aberta é um conceito que permite às empresas a aplicação do critério de *openness* a todo o processo de criação de valor. Estes novos padrões de criação de valor permitem às pequenas empresas aproveitarem a economia de baixo para cima, combinado os três fatores de produção – trabalho, matéria-prima e capital – num único *stakeholder*. Ou seja, permite a criação de pequenas

[5] Tradução do autor
[6] Tradução do autor
[7] Tradução do autor

empresas que podem contribuir para a criação de valor de forma mais flexível e descentralizada (Basmer et al., 2015).

À medida que a sociedade muda o seu foco industrial para um baseado no conhecimento, o valor do conhecimento e da informação aumenta em relação aos fatores de produção tradicionais. Muitos produtos consistem, cada vez mais, em ativos intangíveis e muitas etapas do processo de desenvolvimento do produto podem ser executadas virtualmente. O trabalho torna-se mais independente da localização à medida que a criação de valor se move para a esfera virtual. Essa mudança para a criação de valor cooperativa, descentralizada e auto-organizada exige que as empresas se adaptam e cedam a novos padrões de produção (Basmer et al., 2015).

A Produção Aberta envolve o uso da emergência para criar valor dentro do processo de produção e é caracterizada pelo conceito *openness*, que permite a interação entre outros elementos e é necessária para os sistemas serem viáveis a longo prazo. Do ponto de vista económico, a Produção Aberta permite que os indivíduos possuam o mesmo nível de influência que as empresas nas áreas de pesquisa e desenvolvimento, produção e marketing, de forma a possuírem a oportunidade de criação de valor líquido. O site de estatísticas públicas da *Wikimedia Foundation*, *Wikimedia Statistics*, fornece dados mensais sobre a *English Language Wikipedia*. Esses dados podem ser usados para analisar a comunidade Open Design como uma forma de Produção Aberta, conforme exemplificado pelo exemplo da *Wikipedia* (Castro & Ávila, 2020).

A estrutura de Produção Aberta oferece ferramentas para projetar a criação de valor nos níveis normativos, estratégico e operacional dentro do contexto do conceito *openness*. *Openness* é um fator-chave nas estruturas e processos de produção e reflete-se nas estruturas de sistemas de valor e nos artefactos de criação de valor, bem como nas formas colaborativas de desenvolvimento, produção e marketing, como se pode observar na Figura 8 (Tooze et al., 2014).

Figura 8 - Aspetos-chave da criação de valor empresarial na Produção Aberta. Adaptado de Tooze et al. (2014)

De forma a implementar efetivamente a filosofia de Produção Aberta, os seguintes requisitos devem ser considerados no projeto de criação de valor (Tooze et al., 2014):

- Vontade de renunciar ao controlo total e o “desejo do domínio”;
- Organização e coordenação hierárquicas, e a exploração de mecanismos de auto-organização;
- Potencial para apoiar processos de desenvolvimento e produção fora dos domínios da empresa;
- Capacidade de redesenhar com flexibilidade as configurações de criação de valor;

- Divisão intensiva de conhecimento e trabalho;
- Ação global;
- Intensificação dos incentivos à participação através da transparência;
- Fornecimento de sistemas híbridos de produto-serviço e experiências de cocriação em vez de produtos acabados;
- Classificação igual de todos os envolvidos no sistema de criação de valor;
- Eliminação da distinção entre produtor e consumidor;
- Reconhecimento do cliente como um recurso criador de valor do sistema de produção e um participante proativo nos processos de criação de valor.

- ***Open Source***

Open Source refere-se a *hardware*, *software*, processos ou outras coisas não proprietárias e isentas de *royalties*. O termo "*open*" significa a permissão ao acesso, passagem ou visão através de um espaço vazio e o termo "*source"* refere-se à origem ou local onde algo pode ser obtido (Castro & Ávila, 2020). Os quatro princípios do conceito *Open Source* referem-se ao direito de todos poderem usar, estudar, modificar e distribuir objetos (Bonvoisin et al., 2016).

O foco principal do *Open Source* consiste na capacidade de realizar modificações e depende de um processo de desenvolvimento distribuído. Essa abordagem de criação de *software* surgiu na indústria de *software* como uma alternativa ao modelo tradicional em que o *software* é desenvolvido por profissionais dentro de uma empresa, protegido por medidas legais e técnicas e depois licenciado mediante o pagamento de uma taxa. Pelo contrário, no modelo de *Open Source*, o *software* é desenvolvido através de voluntários independentes (Avital, 2011).

O modelo de *Open Source* permite que qualquer pessoa possa aceder e modificar o *source code*, que pode ser redistribuído nos mesmos termos. Isso incentiva a melhoria contínua, a adaptação e a expansão através da colaboração distribuída. Com a adoção de recursos externos de desenvolvimento, um projeto pode aproveitar um conjunto mais amplo de criatividade e inovação. Inspirados pelo impacto de projetos de alto perfil como *Linux* e *Mozilla Firefox*, os princípios do modelo de desenvolvimento, licenciamento e distribuição de *Open Source* promovem a proliferação de projetos de *Open Source* de todos os tipos, desde o desenvolvimento de conteúdo digital, por exemplo *Wikipedia*, até veículos e bebidas, a impressoras 3D, por exemplo RepRap (Avital, 2011).

Além disso, os responsáveis por desenvolver OSHW podem projetar o seu hardware sob uma licença *Creative Commons Attritbution-Share Alike* 2.5 usando *Arduino*, uma plataforma de prototipagem de *Open Source* que usa microcontroladores *AVR Atmega*. Assim como, o projeto *OScar* que visa desenvolver um carro de acordo com o *Open Source*, com a participação aberta da internet (Kim & Shin, 2016).

Para que o *software* seja considerado *Open Source*, este deve ser licenciado de forma a permitir modificações e exija redistribuição gratuita do *software* sob a mesma licença. Conforme descrito pela *Open Source Definition* (OSD) da *Open Source Initiative*. Essas licenças que são baseadas principalmente na lei de direitos de autor, são compatíveis com o OSD e são usadas para *software*, a área onde as práticas de *Open Source* se originaram (Bruijn, 2010).

- ***Open Source Innovation***

A *Open Source Innovation* é uma estratégia inovadora *enclosed* que exige que os participantes do mercado partilhem os seus conhecimentos de forma voluntária e participem de uma forma pouco ortodoxa de transferência de conhecimento. A ideia principal é utilizar recursos privados para produzir bens públicos. Dentro do projeto de *Open Source Innovation* (OSI), os resultados obtidos são de livre acesso, não são limitados de forma alguma e não estão vinculados a determinados participantes. No entanto, existem certos termos para aqueles que usam o *Open Source Innovation* que consideram aspetos económicos, sociais e legais (Redlich, 2010).

A *Open Source Innovation* é definida pela revelação gratuita de informações sobre um novo *design* com a intenção de desenvolvimento colaborativo de um projeto único ou de um número limitado de projetos relacionados para exploração comercial ou não comercial (Raasch et al., 2009). Existe ainda a distinção entre abordagens de *Open Source Innovation* que são impulsionadas por uma empresa ou comunidade. Uma abordagem orientada pela empresa para a abertura do processo envolve os indivíduos externos em atividades de *co-design* e a procura por contribuições externas para o avanço dos processos internos (Raasch et al., 2009).

Na prática, é feita a distinção entre *Open Source Software* e *Open Source Hardware* como duas formas de *Open Souce Innovation* (Vallance et al., 2001).

- ***Open Source Software* e *Hardware***

O desenvolvimento de produtos de *Open Source* baseia-se no modelo de desenvolvimento de *Open Source Software* (OSS), e através de um número crescente de projetos está constantemente a evoluir.

Open Source Hardware (OSHW) é um termo composto, pelo que é necessário entender o conceito original de *Open Source Software* de forma a compreender como o *Open Source Hardware* funciona. O *Open Source Software* não se diferencia pela linguagem de programação, ambiente operacional ou domínio do aplicativo, mas sim pela(s) licença(s) que concedem o uso, a distribuição, e mais importante, ainda, os direitos de aceder e modificar o *software´s source code* (Tiemann, 2009). Esses direitos permitem que qualquer pessoa estude, sem limitações, o funcionamento do *software* lançado sob uma licença de código aberto, que o possa melhorar e redistribuía o resultado do trabalho (Lock, 2013).

Uma definição de OSHW fornecida pela *Open Source Hardware Association* (OSHWA), que foi derivada da definição de *Open Source* geralmente aceite da *Open Source Innovation* (Open Source Initiative, 2022) , refere que "Open Source Hardware (OSHW) é um termo para objetos semelhantes a artefactos – máquinas, dispositivos ou outros objetos físicos – com projetos disponibilizados abertos para qualquer pessoa estudar, modificar, compartilhar e usar."(Open Source Hardware Association, 2022)[8]. Por outras palavras, um produto de OSH é um artefacto físico cuja documentação é liberada sob uma licença que concede a qualquer pessoa com direitos de produção e distribuição, e é detalhado o suficiente, para permitir a qualquer pessoa o estudar e desenvolvê-lo ainda mais (Bonvoisin & Mies, 2018).

A transparência permite que os utilizadores tenham controlo total da tecnologia. O *design* de produtos de OSH é altamente eficiente por vários motivos. Em primeiro lugar, a comunidade de OSH compreende um grande número de pessoas heterogéneas, mas com ideias semelhantes, de

[8] Tradução do autor

todo o mundo que colaboram voluntariamente em diferentes projetos. Em segundo lugar, a informação pode circular livremente. As pessoas partilham ideias, aprendem umas com as outras e desenvolvem as ideias umas das outras. Por último, OSH promove a sustentabilidade quando se trata da eficiência de recursos, aptidão tecnológica e independência do usuário (Moritz et al., 2018).

Na verdade, existem muitas definições do que constituiu o *Open Source Hardware.* O consenso é que o *Open Source Hardware* é um *design de hardware* eletrónico que está disponível gratuitamente sob uma das licenças de *Open Source* legalmente vinculadas e reconhecidas. O *Open Source Hardware* inclui esquemas, diagramas e regras de *design* que podem usados, estudados e modificados sem qualquer restrição e podem ser copiados e redistribuídos de forma modificada ou não modificada, sem restrições ou com restrições mínimas que apenas possuam a garantia que outros destinatários possam fazer o mesmo (Acosta et al., 2009).

Open Source Hardware (OSHW) e *Open Source Software* (OSS) incluem quatro fases de desenvolvimento: *design*, produção, distribuição e atualizações. Embora ambos aproveitem a máxima utilização da abordagem de *Open Source*, durante a fase de projeto, a produção e distribuição de *Open Source Software* geralmente é mais fácil e menos dispendiosa do que a de *Open Source Hardware*. A razão disso é que o OSS pode ser distribuído digitalmente pela internet, enquanto o OSHW requer a produção física e o armazenamento de máquinas, dispositivos e outros itens físicos. Além disso, a atualização do OSS normalmente envolve simplesmente atualizar e compartilhar o código revisado online, enquanto as atualizações de *hardware* exigem a substituição física de componentes ou unidades inteiras, o que pode ser realmente caro (Kim & Shin, 2016).

Open Source Software deve atender a certos critérios relacionados à distribuição incluindo (Castro & Ávila, 2020):

- Redistribuição gratuita;
- Disponibilidade do *Open Source*;
- Permissão para criar trabalhos derivados;
- Preservação da integridade do Open Source do autor;
- A não discriminação contra pessoas ou grupos;
- A não discriminação contra áreas de atuação;
- Distribuição da licença;
- A licença não deve ser específica para um produto;
- A licença não deve restringir outro *software*;
- A licença deve ser neutra em termos de tecnologia.

Quando se compara entre o OSS e o OSHW, o OSHW é visto como o mais desfavorável devido à dificuldade de copiar e modificar produtos físicos em comparação com o *software*. No entanto, existem vários modelos e iniciativas de negócios de OSWH bem-sucedidos, empresas como *SparkFun Electronics* e *Raspberry Pi*, bem como as tecnologias de *Field-Programmable Gate Array* (FPGA). Isso torna o processo de desenvolvimento de *hardware* aproximado ao desenvolvimento de *software*. Além disso, os conceitos de *Open Source* são aplicados em vários campos, incluindo impressoras 3D, plataformas de prototipagem eletrônica, carros, próteses, robôs e outros projetos de *design* socialmente relevantes. *RepRap* (*Replicating Rapid-prototyper)*, é uma impressora 3D económica que é capaz de se reproduzir (Kim & Shin, 2016).

Existem várias razões pelas quais o *Open Source Hardware* pode ser vantajoso. Um dos motivos é por permitir a reutilização do *design*, isto é, outros podem aceder e modificar facilmente o *design* para as suas próprias necessidades e para o fim que quer dar ao seu uso. Dessa forma, pode levar a uma gama mais ampla de aplicações e a um uso mais eficiente dos recursos e a atrair a contribuição de outros. Para além disso, leva a um ritmo mais rápido de desenvolvimento e um senso de comunidade mais forte em torno do projeto (van der Bij, 2018).

Outro benefício do *Open Source Hardware* é pelo facto de permitir a *peer review,* uma vez que especialistas de outro mundo podem rever e fornecer feedback sobe o *design*, além de permitir que os *designs* se tornem melhores. Além disso, o *Open Source Hardware* pode facilitar a disseminação do conhecimento e ajuda os projetos a garantirem que os seus projetos são desenvolvidos exatamente como desejam. Também pode tornar mais fácil para as instituições públicas a gerirem os seus financiamentos e permite que pequenas empresas com bom apoio local participem do processo de desenvolvimento (van der Bij, 2018).

- ***Open Data***

Open Data, que se refere à disponibilização de dados gratuitamente para qualquer um aceder, reutilizar ou partilhar, é cada vez mais comum globalmente. No contexto de pesquisa, isto significa que os dados podem ser partilhados e reutilizados com a aprovação ética apropriada, sem quaisquer barreiras, como custos ou requisitos de permissão. Estes dados referem-se a conjuntos de dados originais, não apenas descobertas ou resultados publicados, que podem ser incluídos em revisões sistemáticas ou meta-análises (Chauvette et al., 2019).

Apesar dos cientistas geralmente terem a opinião de que os dados publicados são pertencentes a toda a comunidade científica, alguns editores afirmam ter controlo de direitos de autor sobre os dados e não permitem sua reutilização sem permissão. Pelo que isso pode dificultar o avanço das pesquisas na era digital (Murray-Rust, 2008).

Os dados fornecem a evidência para aquilo que é publicado de conhecimento científico, que é a base para todo e qualquer progresso. Quanto mais dados forem disponibilizados abertamente de forma útil, maior será o nível de transparência e reprodutibilidade e, portanto, maior será a eficiência do processo científico, cujos benefícios serão para a sociedade (Molloy, 2011).

O *Open Data* permite que a comunidade aceda a informação gratuita, permitindo uma sociedade mais evoluída e mais culta e ainda permite às empresas se tornarem mais competitivas. Disponibilizar dados leva a uma maior transparência, participação e inovação para toda a sociedade. Para além disso, é visto como um movimento muito semelhante ao *Open Source*.

O aumento na disponibilidade de iniciativos de *Open Data* é visto principalmente devido à crescente pressão imposta pelos governos sobre todo o tipo de organizações públicas. Existem várias motivações para encorajar as organizações públicas a publicar dados, como por exemplo, o maior crescimento político e económico e a contribuição para valores públicos como transparência e responsabilidade (Kapoor et al., 2015).

O principal objetivo das plataformas de *Open Data* tem sido promover o acesso aos dados e incentivar o desenvolvimento de ferramentas e aplicativos para envolver e servir a comunidade em geral. Assim, incentiva-se ao envolvimento cívico, com a oferta de oportunidades para cidadãos, organizações do setor público, empresas e desenvolvedores independentes de usar o fluxo sistematicamente atualizado de open data (Kapoor et al., 2015).

Existem duas abordagens principais para tornar os dados disponíveis ao público. A primeira abordagem envolve o fornecimento de informações legíveis por humanos em vez de *raw data*. Isso geralmente envolve a transformação desses dados em documentos que contêm valores agregados, visualizações ou outras simplificações para tornar os dados mais fáceis de serem compreendidos. Embora essa abordagem possa ser útil em alguns casos, pode também não conter todas as informações desejadas ou permitir que os utilizadores obtenham perceções para além daquilo que é pretendido pelo autor do relatório. Também não permite a reutilização de *raw data* (Braunschweig et al., 2012).

A segunda abordagem envolve a publicação de *raw data* em que as ferramentas de *software* os processam. Isto permite aos utilizadores aceder diretamente aos dados e podem personalizá-los de acordo com as suas próprias necessidades, embora possa exigir um certo nível de especialização. No entanto, esta abordagem tem a vantagem de oferecer suporte à reutilização dos dados para uma ampla variedade de casos de uso (Braunschweig et al., 2012).

2.3. Gestão ágil de projetos

A gestão ágil de projetos é uma metodologia voltada para a produção ágil. Ser ágil requer uma mudança comportamental que afeta a forma de pensar e agir dos membros da equipa na empresa (Sherehiy et al., 2007). Na verdade, existem diversas definições sobre gestão ágil de projetos (Mkoba & Marnewick, 2020).

O Manifesto Ágil (Beck et al., 2001) é a referência pioneira para a implementação do conceito ágil, uma vez que descreve os principais valores de gestão ágil de projetos para gerir projetos de desenvolvimento de software com mais eficácia. Os quatro princípios fundamentais declarados pelo Manifesto Ágil são (Cervone, 2011):

- Indivíduos e interações sobre processos e ferramentas;
- Demonstração do *software* em vez de documentação detalhada;
- Colaboração do cliente invés da negociação de contratos;
- Resposta às mudanças ao invés do seguimento de um plano.

Numa gestão ágil de projetos, a documentação e o planeamento são reduzidos ao mínimo para facilitar a flexibilidade e permitir uma resposta rápida ao ambiente em mudança. A flexibilidade ajuda a gerir a complexidade, reduz a incerteza técnica e de mercado e aumenta a eficiência sob requisitos instáveis. Esta metodologia é voltada para a aprendizagem. A equipa de projeto aprende continuamente através de testes frequentes e *feedback* recorrente do cliente, o que permite que as novas informações e conhecimentos sejam rapidamente incorporados na próxima iteração, com a respetiva entrega do produto que o cliente realmente precisa (Žužek et al., 2020).

O interesse pela agilidade está cada vez mais a aumentar, mas as definições neste campo permanecem inconsistentes, incompletas e sem clareza (Azanha et al., 2017; Conforto et al., 2016). Highsmith (2009) define a agilidade em termos de cinco objetivos principais: inovação contínua, adaptabilidade do produto, prazos de entrega reduzidos, adaptabilidade de pessoas e processos e resultados confiáveis (Highsmith, 2009). Augustine (2005) descreveu a agilidade como a habilidade de fornecer benefícios para o cliente, enquanto se adapta à incerteza e ao caráter dinâmico do projeto, reconhecendo e reagindo a mudanças. Conforto et al. (2014) definiram o APM como uma metodologia fundamentada em conjunto de princípios, visando tornar a administração de projetos

mais fácil, ágil e iterativa, com o objetivo de obter resultados superiores (custo, prazo e qualidade) com menos esforço a nível de gestão e maior grau de inovação e benefícios para o cliente. Cooper & Sommer (2016) caracterizaram o APM como uma técnica de planeamento ou gestão de projetos, criada para incluir uma equipe de desenvolvimento, juntamente com o cliente, visando alcançar rapidamente um resultado funcional e operável.

Conforto et al. (2016) conduziram uma revisão sistemática da literatura e aplicaram a metodologia de semântica de quadro para desenvolver o que eles denominaram de definição abrangente de agilidade. De acordo com a definição proposta, agilidade é a capacidade de uma equipa de projeto adaptar o seu plano de projeto para atender às necessidades e demandas dos clientes, partes interessadas e do mercado ou tecnologia, a fim de melhorar o desempenho do projeto e do produto num ambiente de projeto dinâmico e inovador. Os autores enfatizaram que a agilidade deve ser vista como uma característica de uma equipa de projeto e não como uma metodologia ou prática específica. A pesquisa também destacou que os dois componentes cruciais da agilidade na gestão de projetos são a capacidade de modificar o plano de projeto rapidamente e a participação ativa do cliente.

Azanha et al. (2017), no seu estudo, descobriram que, embora as definições de APM possam variar, há um consenso entre a maioria dos autores de que o núcleo do APM envolve a procura de flexibilidade, simplicidade, iterações em intervalos curtos e adição de valor incremental (Žužek et al., 2020).

A gestão ágil de projetos está profundamente alinhada com os princípios do desenvolvimento ágil de *software*, mas ligeiramente adaptada para a gestão de projetos. Isso reflete-se em certos aspetos da abordagem ágil de gestão de projetos. Por exemplo, a gestão ágil de projetos enfatiza dois conceitos-chave: minimizar o risco, concentrando-se em iterações curtas de entregas bem definidas e priorizando a comunicação direta com os parceiros de desenvolvimento em vez da extensa documentação do projeto. Esses conceitos são importantes para permitir que uma equipa de projeto responda rapidamente aos requisitos imprevisíveis e que mudam rapidamente, comuns em muitos projetos de desenvolvimento (Cervone, 2011).

De acordo com R. M. Wideman (2006), a gestão ágil de projetos é "uma estrutura conceitual de gestão de projetos para a realização de projetos de desenvolvimento de software em que o destaque é alterado do planeamento para a execução". R. M. Wideman (2006) e Sheffield & Lemétayer (2013) definem a gestão ágil de projetos como um processo de evolução contínua baseado em repetições de entregas curtas, com a finalidade de lidar com as incertezas no meio e as alterações rápidas nas necessidades do projeto. Além disso, argumentam que a gestão ágil de projetos é orientada para o valor e não por plano.

A abordagem ágil para gerir projetos, práticas, ferramentas e técnicas interligadas, tem-se mostrado adaptável a qualquer ambiente dinâmico e em constante mudança, sendo assim adotada em projetos com diferentes características (Conforto et al., 2014). O conceito ágil evoluiu de uma abordagem específica de desenvolvimento de *software* para uma maneira inovadora e versátil de gerir projetos que várias indústrias usam para melhorar o desempenho dos seus projetos (Goldstein & Euchner, 2017; Ito, 2017; Pellizzoni et al., 2019).

A abordagem ágil e a abordagem tradicional apresentam visões distintas. Na Tabela 6, encontra-se o resumo das características das abordagens de gestão de projetos para apresentar as principais diferenças entre as abordagens ágeis e tradicionais.

A Tabela 5 tem como objetivo a classificação e identificação das diferentes abordagens no que diz respeito à gestão de projetos. É de notar que, nos diversos trabalhos apresentados na literatura, todas as abordagens de gestão de projetos sofrem mudanças e adaptações, de acordo com as necessidades do ambiente, ainda que, no caso do ágil, alguns autores enfatizam que as práticas devem ser implementadas sem modificação para promover uma transformação cultural das organizações (Cooper & Sommer, 2016; Schwaber, 2004; Schwaber & Sutherland, 2016).

Tabela 5 - Características das Abordagens da Gestão Ágil de Projetos. Adaptado de Azenha et al., (2021, p. 96)

Características	Ágil	Tradicional
Planning Horizon	Planeamento a curto prazo focado em objetivos de iteração	Planeamento a longo prazo focado em todo o ciclo de vida do projeto
Planeamento do projeto	Planeamento *lean* no ciclo de iterações reavaliado e melhorias constantes	Planeamento aprimorado para todo o ciclo de vida do projeto
Detalhes das atividades	Detalhamento imprevisível, não linear e imensurável com entregas fomentais e constantes	Detalhamento revisível, linear e mensurável com entregas integrais ou faseadas e documentação formal aprimorada
Project Scope	Visão direcionada para o deve ser construído. Representação ampla e alegórica dos objetivos e resultados previstos de forma dúbia e desafiadora	Baseado na especificação detalhada do que será construído. Descrição vigorosa e formal dos objetivos e resultados previstos dos documentos contratuais
Scope Conformance	As mudanças são identificadas e o planeamento é ajustado de acordo com estas para cada iteração	Os desvios são identificados e a as atividades ajustadas para manter o planeamento
Controlo e Monitorização	Feito de artefactos e dispositivos visuais físicos e/ou virtuais, como pósteres, murais e fotos, com reuniões curtas e frequentes de equipa	Realizado através de indicadores de desempenho, cronogramas, documentação formal, revisões de desempenho e auditorias, com reuniões de equipa extensas e pouco frequentes
Estilo de gestão	Flexível, variável e adaptável. Não é necessária a presença da figura de um gerente de projetos. Equipas multidisciplinares, autogeridas e com baixa hierarquia	Mecânico, formal e burocrático. Forte presença do gerente de projetos. Altamente especializado, trabalho realizado com equipas especializadas

Azenha et al. (2021) no seu caso de estudo, averiguou que todas as empresas que foram abordadas indicaram que a abordagem híbrida na gestão ágil de projetos é ideal para projetos inovadores, aqueles que envolvem um alto grau de incerteza e que não podem ser realizados sem algum nível de planeamento devido às restrições do ambiente organizacional ou dos setores de negócios.

A informação é um aspeto importante, embora seja apenas um dos desafios ao implementar os princípios ágeis. A criação de uma equipa para liderar e alavancar a gestão ágil, a denominada *Agile Team*, e a forma como essa equipa comunica com os *stakeholders* externos e os colaboradores (internos) da empresa, ao longo da gestão ágil de projetos, são os principais desafios ao implementar as práticas ágeis dentro de uma empresa com sucesso. Após a implementação de uma gestão ágil, recomenda-se que haja monitorização, controlo e melhoria contínua (Shari et al., 2000).

De forma a orientar a gestão ágil é essencial fornecer uma ferramenta importante para apoiar todos os projetos (produto, serviço, sistema produto-serviço, solução) desenvolvidos pela empresa, e por isso é que a gestão ágil de projetos é obrigatória (Martinsuo, 2019; Midler et al., 2019).

A gestão ágil de projetos deve ser baseada nos princípios ágeis (Estrutura Ágil) que determina uma conduta ágil entre os colaboradores, essencialmente a equipa ágil (Práticas Ágeis), que incita a empresa a desenvolver Produtos/Serviços de acordo com o Valor do Cliente, de forma pró-ativa (Valores Ágeis) e baseada num ambiente transformador que cria Oportunidades e Novos Produtos. Essa estrutura generalizada da gestão ágil de projetos é apresentada na Figura 9 (Loiro et al., 2019).

Figura 9 - Estrutura geral de uma gestão ágil de projetos. Adaptado de Loiro et al. (2019)

No que diz respeito às fases relativamente à gestão de projetos, estas são necessárias para a criação de uma visão clara na evolução do projeto. Sustentado em princípios ágeis, o projeto deve ser orientado de forma sistemática, e não dividido por fases, uma vez que todas as fases devem ser tratadas de forma simultânea. Assim, são identificados cinco momentos numa gestão ágil de projetos (Loiro et al., 2019):

- Análise de requisitos - projetos, metas e objetivos são definidos de acordo com a necessidade do cliente e das metas da empresa;
- Planeamento - equipa ágil é criada e as tarefas são distribuídas. É realizada a discussão e formulação dos requisitos iniciais;
- *Design* - equipa ágil trabalha diariamente de acordo com os requisitos que são precisos e fornece um feedback contínuo do seu progresso;
- Implementação/Desenvolvimento - equipa ágil e equipa de apoio negoceiam o trabalho a ser feito, reveem os testes de qualidade do produto, a documentação de desenvolvimento e a versão final da iteração para entrar em produção;
- Operação e Manutenção: O produto é entregue ao cliente e os serviços pós-venda são prestados de forma contínua. O feedback do cliente é considerado e mantém-se importante no processo de melhoria contínua.

2.3.1. Gestão ágil

A agilidade permite que as empresas prosperem em ambientes dinâmicos e turbulentos. Dessa forma, as organizações são forçadas a liderar eficazmente com ambientes imprevisíveis, dinâmicos e em constante mudança. (P'shewchuk, 1998).

Ser ágil integra todos os colaboradores e ferramentas de informação que participam em todo o sistema produtivo. O principal objetivo da gestão ágil é apresentar uma solução para as necessidades do cliente, não apenas um produto, envolvendo-o mais de perto no processo de desenvolvimento de um produto (Gunasekaran, 1998). Para implementar um sistema de

manufatura ágil adequado, as organizações devem estar informadas sobre aquilo que o cliente precisa no presente e o que provavelmente irá precisar no futuro (Plonka, 1997).

A gestão ágil surgiu como uma estratégia de resposta e adaptação à complexidade e ambientes incertos (Sherehiy et al., 2007). Esta estratégia não envolve apenas uma empresa, mas depende da cooperação e interação entre várias empresas, sendo importante o estabelecimento de relacionamentos apropriados entre elas (Loiro et al., 2019). O estabelecimento de relacionamentos adequados entre as empresas torna-se uma vantagem competitiva (Sanchez & Nagi, 2001), e dessa forma, a cooperação pode ser a chave para relacionamentos possivelmente complementares. Existem diversas habilidades estratégias do conceito de agilidade que podem ser aplicadas a vários aspetos da empresa como: a flexibilidade, a capacidade de resposta, a velocidade, a cultura de mudança, a integração, a baixa complexidade, a alta qualidade e produtos customizados e a associação de competências fundamentais (Sherehiy et al., 2007).

Para adotar a agilidade numa organização, é necessário analisar detalhadamente todos os dados da empresa (sejam internos, externos, informações sobre a concorrência ou consumidor) e somente quando toda a empresa entender a importância desses mesmos dados, é a altura de adotar a agilidade (Alexander M., 2018). Portanto, os dados são o elemento-chave para alcançar um ambiente de gestão ágil. Pelo facto de a gestão ágil ser usada em muitas empresas de *software*, essa informação pode ser considerada um aspeto essencial (Loiro et al., 2019).

Atualmente, o conceito ágil continua a ser uma prioridade para muitas empresas. De acordo com 89% dos entrevistados para o relatório anual do estado ágil, as equipas ágeis de alto desempenho têm os valores centrados nas pessoas, cultura clara, ferramentas e habilidade de liderança. Com os valores anteriormente mencionados, os resultados não poderiam ser diferentes de: um lugar melhor para trabalhar, as pessoas trabalham mais e com maior entreajuda, têm maior visibilidade do seu trabalho e isso leva a um melhor alinhamento com as necessidades de negócios. Se o conceito ágil for aplicado com sucesso, o benefício não será só para os indivíduos envolvidos, mas para toda a organização, pelo que é razão suficiente para as empresas investirem, sendo que as organizações continuam a fazê-lo. O pensamento principal das empresas é em alcançar três aspetos: aumentar a receita, reduzir os custos ou reduzir o custo (State of Agile, 2022).

Com a implementação das práticas ágeis, mais de metade dos entrevistados (52%) afirma que é para acelerar o tempo de lançamento no mercado, 44% afirma que é para previsibilidade de entrega e outros 31% afirma que é para diminuir o risco. Ser capaz de se mover rapidamente e ainda ser previsível é uma vantagem importante que se destaca entre uma série de benefícios (State of Agile, 2022).

As vantagens de uma abordagem ágil são nítidas, por isso cada vez mais é implementado por uma grande parte das organizações. Ainda foi possível aferir, através do estudo realizado pelo *State of Agile,* que as organizações não estão só interessadas apenas no conceito ágil para desenvolvimento de *software*, uma vez que 49% dos entrevistados, quase metade, aplicam práticas ágeis em todo o ciclo de vida de entrega. O resultado mais comum de satisfação com as práticas ágeis nas organizações, é o aumento da colaboração (69%), uma vez que é mais fácil fazer o trabalho com a ajuda de terceiros. O segundo resultado mais comum, sendo pouco mais de metade, é o melhor alinhamento às necessidades do negócio (54%). Depois disso, os próximos dois resultados mais comuns que os deixam satisfeitos são um ambiente de trabalho melhor (39%) e uma maior visibilidade no ciclo de vida do desenvolvimento do aplicativo. Este último resultado é um benefício

claro de um contínuo foco em ferramentas e metodologias ágeis para o próprio desenvolvimento de *software* (State of Agile, 2022).

2.3.2. Metodologias ágeis

Por largos anos, ágil tem sido um conceito apresentado como uma metodologia para planeamento e execução de projetos que aborda muitas das falhas do processo tradicional de planeamento em cascata (Serrador & Pinto, 2015). As principais características da agilidade são as equipas auto-organizadas, execução rápida, orientada para o valor e orientada para os negócios (Mkoba & Marnewick, 2020). A prática ágil promove o conceito de equipas auto-organizadas que têm o poder de organizar o seu trabalho de forma independente (Hoda et al., 2008; Stankovic et al., 2013).

A forma ágil é frequentemente mal interpretada. É uma filosofia e não um processo ou metodologia específica. A agilidade é a capacidade de se adaptar rapidamente às mudanças, quer sejam elas no ambiente, nas necessidades do usuário ou nas restrições do projeto. Quanto mais agilidade a empresa possuir, maior a sua competitividade. Um ciclo ágil de trabalho inclui planeamento, análise de requisitos, *design*, codificação e testes, e é realizado em curtos períodos.

A principal vantagem que o método tradicional apresenta é o facto de permitir uma fácil transição de equipas permanentes para equipas distribuídas, pois fornece uma estrutura clara para organizar e controlar as atividades durante todo o processo de desenvolvimento. Cada fase, desde a definição dos requisitos até à entrega ao cliente, possui entradas e saídas claras que são documentadas e consideradas finais quando uma fase é concluída. Assim que a fase de projeto é concluída, a documentação completa do projeto fica disponível e os grupos de desenvolvimento podem ser designados para implementar diferentes partes neste. Mesmo quando os membros do grupo estão localizados em países diferentes, o projeto documentado permite uma atribuição direta de tarefas a membros que se encontram dispersos. As dependências entre as funções são identificadas e documentadas e podem ser trabalhadas de forma independente até que a etapa de integração seja iniciada. Pelo facto de as equipas e membros terem a oportunidade de trabalhar de forma relativamente autónoma, minimiza-se os problemas e riscos associados à comunicação à distância e à partilha de informação, uma vez que a sobrecarga de comunicação entre eles é mínima (Papadopoulos, 2015).

Já para projetos que seguem metodologias ágeis, passar de equipas permanentes para equipas grandes e distribuídas não é tão simples quanto no método tradicional. As primeiras tentativas ao implementar as metodologias ágeis foram feitas em equipas pequenas e fixas, mas de acordo com estudos realizados, os feedbacks de milhares de projetos mostraram que mais de 50% dos projetos ágeis têm pelo menos uma pessoa a trabalhar remotamente, o que permite que as equipas distribuídas sejam a norma hoje, e não uma exceção. Quanto mais as empresas querem escalar os projetos para projetos maiores, estes só podem ser atendidos a partir de grupos maiores e daí surge a necessidade inevitável das equipas distribuídas (Papadopoulos, 2015).

Na Figura 10, pode-se observar a diferença entre o processo de metodologia tradicional e a metodologia ágil.

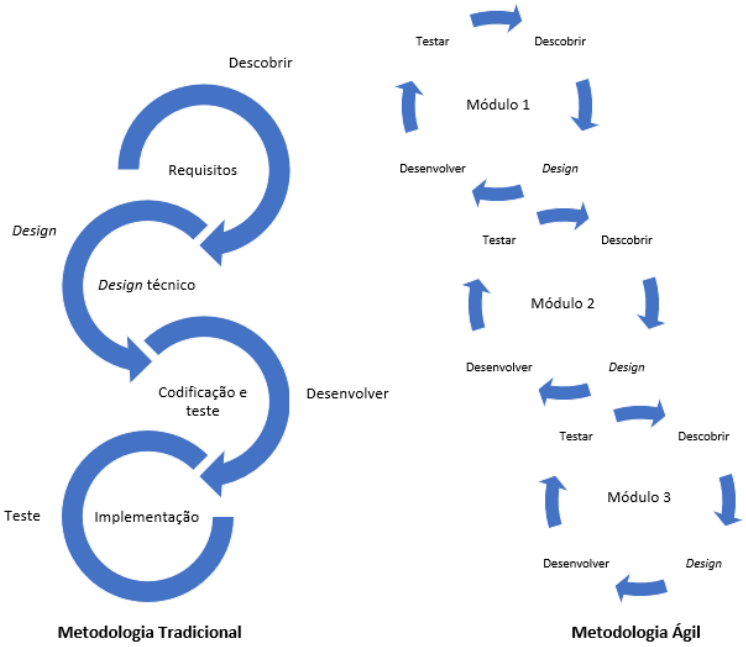


Figura 10 - Metodologia Tradicional vs Metodologia Ágil. Adaptado de Shaikh & Abro (2019)

O foco principal ao adotar as metodologias ágeis em projetos grandes e distribuídas é alcançar a satisfação do cliente e isso só pode ser alcançado seguindo os valores centrais e os princípios orientados das metodologias. O desafio extra de abordar o aumento da complexidade como resultado da distribuição e dimensionamento requer uma avaliação cuidadosa dos fatores organizacionais e práticas ágeis para que a flexibilidade e a capacidade de respostas às mudanças que as metodologias ágeis oferecem permaneçam evidentes (Papadopoulos, 2015).

Os métodos ágeis são processos que suportam a filosofia ágil, ou seja, valores e princípios ágeis. Cada método difere entre si pela escolha do seu conjunto apropriado de terminologia e práticas (Elbanna & Sarker, 2016) e são ainda projetados para usar um mínimo de documentação a fim de facilitar a flexibilidade e a capacidade de resposta a condições de mudança, o que implica que menos planeamento e mais flexibilidade sejam usados em projetos ágeis do que aquando da gestão de projetos tradicionais. Estes tornaram-se cada vez mais comuns em projetos de tecnologia desde o seu desenvolvimento (Lindvall et al., 2002), porque abordam diretamente os desafios frequentemente enfrentados ao lidar com projetos dinâmicos em ambientes em constante mudança (Serrador & Pinto, 2015).

Uma metodologia verdadeiramente ágil deve ser (Jiménez et al., 2020):

- Iterativa (vários ciclos para ser concluída);
- Incremental (não entrega o produto de uma só vez);
- Auto-organizada (equipas determinam a melhor maneira de lidar com o trabalho);
- Emergente (processos, princípios e estruturas de trabalho são reconhecidos durante o projeto e não predeterminados).

Há uma grande variedade de diferentes tipos de metodologias ágeis. Alguns dos métodos mais conhecidos são o *Test-Driven Development* (TDD), o *Feature Driven Development* (FDD), o método *Extreme Programming* (XP), o método *Lean Software Development* (LSD), o método *Scrum*, o método *Dynamic system Development model* (DSDM), os métodos *Crystal* e o método *Kanban* (Al-Saqqa et al., 2020; Mkoba & Marnewick, 2020).

O estudo realizado no State of Agile Report, aferiu que quase 9 em cada 10 entrevistados utilizam o método *Scrum* e mais de metade utiliza o método *Kanban*. Nos últimos 3 anos de pesquisa, o método *Scrum* continua a liderar, passando de 58%, em 2020, para 87% na pesquisa atual. O método *Kanban* explodiu de 7%, na pesquisa realizada em 2020, para 56% no levantamento atual. Na Figura 11, pode-se observar, em suma, as respetivas percentagens da utilização das metodologias ágeis (State of Agile, 2022).

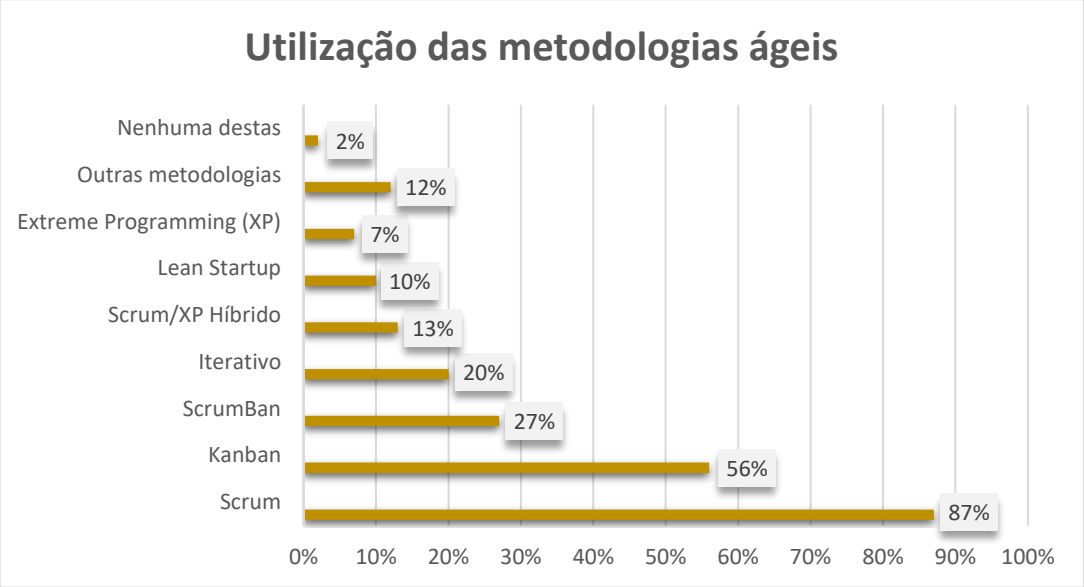


Figura 11 - Percentagem de utilização das metodologias ágeis por empresas dos setores de TI e não-TI. Adaptado de State of Agile (2022)

O método *Scrum* é uma implementação concreta de um *framework* ágil que foi proposto por Schwaber & Beedle (2002) para uma gestão de projetos para o processo iterativo de desenvolvimento de *software*. Este método foca-se na entrega do maior valor possível no menor tempo. É uma metodologia ágil orientada a equipas que especifica um determinado papel, estabelece uma iteração de curto prazo chamada *sprints*, em que o sistema é desenvolvido de forma incremental e produz um artefacto diferente que coordena o seu trabalho (Schmidt, 2016). A popularidade desde método deve-se à sua simplicidade e ao foco nas questões de gestão de software, e não nas práticas técnicas de desenvolvimento de software, o que o torna aplicável a qualquer domínio (Al-Saqqa et al., 2020).

No que diz respeito ao ciclo de vida do *Scrum*, existem três fases principais que são ilustradas na Figura 12.

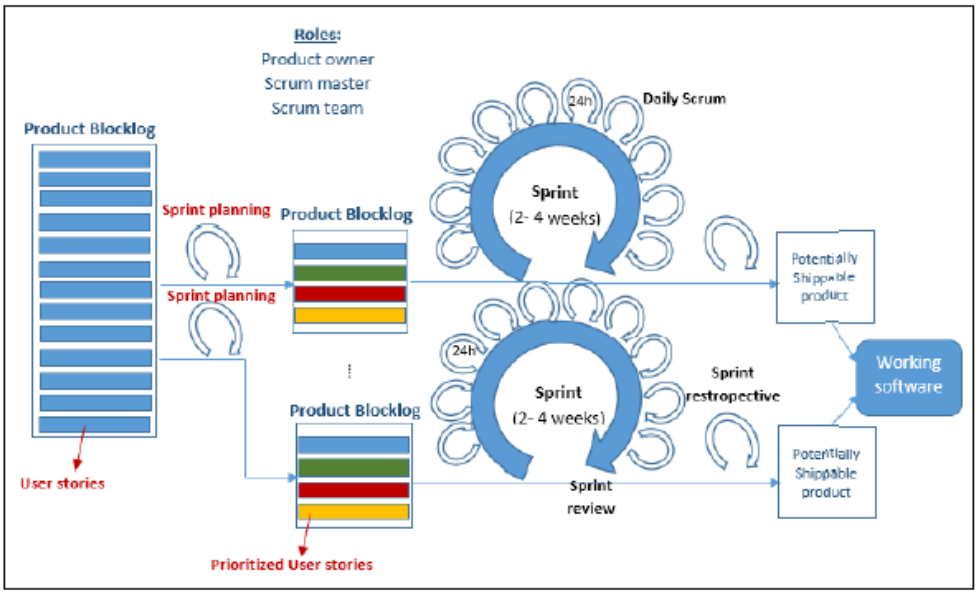


Figura 12 - Ciclo de vida da metodologia Scrum. Adaptado de Al-Saqqa et al. (2020)

Os projetos são divididos em ciclos chamados *sprints*. Um *sprint* representa uma caixa de tempo dentro da qual um conjunto de atividades deve se executado. As funcionalidades a serem implementadas num projeto são mantidas numa lista conhecida como *product backlog*. No início de cada *sprint* é realizada uma reunião de planeamento do *sprint*, ou seja, uma reunião de planeamento na qual o *product owner* prioriza os itens do *blacklog* do produto e a equipa seleciona as atividades que poderá implementar durante o *sprint* inicial. As tarefas alocadas num *sprint* são transferidas do *blocklog* do produto para o *blocklog* do sprint (Jiménez et al., 2020).

Num *sprint* puro, a cada dia, a equipa realiza uma breve reunião denominada por *Daily Scrum*. O objetivo principal dessas reuniões é disseminar o conhecimento sobre o que foi feito no dia anterior, identificar impedimentos e priorizar o trabalho do que deve ser iniciado. Ao final de cada *sprint*, a equipa apresenta os recursos implementados numa reunião de revisão do *sprint* e recebe o respetivo *feedback*. Caso o sprint seja aprovado, a equipa é encaminhada para o próximo *sprint* e assim se recomeça o ciclo (Schwaber, 1997; Senabre Hidalgo, 2019).

Na metodologia *Scrum*, todas as partes interessadas, incluindo clientes e utilizadores finais, devem assumir o compromisso com o projeto e fornecer *feedback* sobre como o funcionamento das entregas para identificar se o produto ou o serviço atende às suas necessidades em cada instante de tempo (Torrecilla-Salinas et al., 2015).

As metodologias ágeis têm como objetivo definir os seguintes conceitos: PM, ciclo de vida do projeto, gestão de equipa, engenharia e entrega. A implementação da metodologia inclui as seguintes etapas: identificação da metodologia apropriada, identificação dos requisitos específicos da empresa, adaptação e implementação da metodologia (Rasnacis & Berzisa, 2016).

A implementação de metodologias ágeis de gestão de projetos está relacionada com a melhoria do processo de desenvolvimento: menos *bugs*, entrega mais rápida, comunicação mais eficaz e em maior quantidade, melhor qualidade, melhor análise de risco e menos sobrecustos. De várias pesquisas que já foram feitas, verifica-se que os problemas na equipa de projeto podem afetar a aplicação da metodologia ágil de gestão de projetos e a implementação bem-sucedida do projeto (Rasnacis & Berzisa, 2016).

Highsmith (2009) na sua visão sobre as práticas ágeis apresentou os resultados da pesquisa segundo os quais 89% dos entrevistadores relataram sobre o aumento de produtividade, 84% sobre a diminuição do número de erros e defeitos, 82% sobre a diminuição da duração dos projetos e 66% sobre a redução dos custos dos projetos. Mah (2008) realizou dois estudos de caso e averiguou que uma empresa que se baseie na gestão ágil, teve uma redução de custos em 61%, economia de tempo em 24% e redução do número de defeitos em 83%. Na segunda empresa, encontrou redução de custos em 5%, redução da duração do projeto em 59% e redução do número de defeitos em 11%.

Apesar de, na maioria das pesquisas realizadas se averiguarem aspetos positivos acerca da implementação das metodologias ágeis, existem também pesquisas em que se destaca que a gestão ágil de projetos leva a riscos adicionais, diminuição da controlabilidade e, como resultado, desempenho desfavorável. Pelo que, há a necessidade de um maior número de pesquisas sobre a aplicação das metodologias ágeis para se poder analisar às consequências desta mesma introdução Highsmith (2009).

Em contraste com as pesquisas anteriormente abordadas, os resultados obtidos pela pesquisa realizada por Sergei Suetin, Elena, Sergei Nikitin e Irina (2016), mostraram que a implementação das metodologias ágeis nas empresas de engenharia de *software* levou a resultados ambíguos. No projeto desenvolvido, esta mesma implementação provocou a decadência dos índices de desempenho dos custos e prazos. Por outro lado, as métricas de desempenho de qualidade aumentaram após a introdução de ferramentas e técnicas ágeis. Através desta pesquisa, pode-se verificar que é necessário que os gerentes de projeto sejam cautelosos nas suas iniciativas ágeis e levem em consideração as possíveis desvantagens para o desempenho de tempo e custo do projeto. No entanto, nos projetos em que os clientes enfatizam a sua prioridade no que diz respeito à qualidade e podem pagar por isso com orçamentos adicionais e aumento na duração do projeto, a implementação das metodologias ágeis pode ser efetuada com sucesso (Suetin et al., 2016).

2.4. Gestão ágil de projeto aberto (*Open Design*) orientado a comunidades

Conceitos como “comunidade” e “participação comunitária” têm sido intensamente problematizados nas últimas décadas em países desenvolvidos e em desenvolvimento. Os contextos são de facto diferentes e variados. A palavra “comunidade” é um termo abrangente que é definido e aplicado de várias maneiras. Envolver a comunidade num projeto leva a melhores resultados, e, para além disso, a própria comunidade pode preferir uma abordagem participativa em vez de uma abordagem exclusivamente orientada por especialistas no projeto (Mahdavinejad & Abedi, 2011). Uma abordagem orientada para a comunidade envolve um processo que pertence a uma comunidade de indivíduos que têm propriedade sobre o projeto, independentemente da sua afiliação a uma organização específica (Raasch et al., 2009).

A gestão ágil é uma metodologia que prioriza a flexibilidade e a colaboração no processo de desenvolvimento, podendo ser aplicada também na fase de conceção de um projeto, concentrando-se na adaptação rápida às mudanças no mercado. No contexto do *Open Design*, a gestão ágil significa envolver a comunidade desde o início e frequentemente no processo de design. Os membros da comunidade podem fornecer feedback e sugestões durante todo o processo de *design*, e não apenas no final, quando o produto final é apresentado. Essa abordagem resulta em *designs* mais centrados no usuário e que atendem às necessidades da comunidade pretendida.

O *Open Design* é um tipo de processo de design inclusivo, colaborativo e transparente. É uma forma de envolver uma ampla gama de partes interessadas e especialistas no processo de *design* e também incentiva a partilha de ideias, conhecimentos e materiais, permitindo uma maior transparência e confiança e envolvimento com a comunidade. Essa abordagem leva a designs com maior probabilidade de serem adotados e apoiados pela comunidade. O *Open Design* orientado às comunidades significa que o processo de *design* é aberto aos membros da comunidade que podem dar feedback e sugestões, permitindo que o design seja moldado pelas necessidades da comunidade. Essa abordagem é particularmente útil para projetar bens ou serviços públicos destinados a atender às necessidades de uma determinada comunidade.

A gestão ágil de projeto aberto orientado a comunidades permite que a equipa de projeto se adapte rapidamente às mudanças no projeto, enquanto ainda se mantém alinhada com as necessidades e prioridades da comunidade. Isso é possível através da incorporação do pensamento ágil, que se concentra na flexibilidade, na entrega incremental e no trabalho em equipa, para fazer face às mudanças e desafios. Isso permite que a equipa faça as adaptações necessárias no projeto, enquanto mantém o compromisso com a comunidade.

O uso da gestão ágil em projetos abertos (*Open Design*) voltado para comunidades também promove a sustentabilidade no processo de *design*. Sustentabilidade não é apenas uma questão de impacto ambiental, mas também de impacto social e econômico. Ao envolver a comunidade no processo de projeto, o projeto final terá mais chances de ser adotado, usado e mantido pela comunidade, evitando o desperdício de recursos e energia.

O conceito *Open Community Manufacturing* (OCM) foi desenvolvido como uma organização independente e não comercial que faz a ponte entre o mercado de capitais e os grupos sociais em desenvolvimento. As Pequenas e Médias Empresas ainda sofrem com a escassez de habilidades (*know-how*), tecnologia apropriada e ausência de um sistema de apoio coletivo. Isto leva à criação ineficiente de valor, ao desperdício de alto grau de energia (recursos) nos processos produtivos e à poluição assustadora (Rebensdorf et al., 2015).

A capacidade de co-criar valor de forma independente, mesmo em regiões social e geograficamente isoladas, em conjunto com melhores capacidades de ensino, pode criar inúmeras possibilidades para os países em desenvolvimento. Estes desenvolvimentos levam ao lançamento de programas que visam apoiar as Pequenas e Médias Empresas (PME´S) de forma a melhorar a situação económica do país. Assim, o conceito de *Open Community Manufacturing* é o seguinte: *Development Challenge* visa utilizar a estratégia de *Open Design* juntando especialistas e voluntários para criar desenvolvimentos sustentáveis por meio de inovações sociais (Rebensdorf et al., 2015).

3. AVALIAÇÃO DAS COMUNIDADES *OPEN SOURCE*

Este capítulo concentra-se na avaliação criteriosa das comunidades de código aberto, uma vertente crucial no panorama atual de desenvolvimento de *software* e *hardware*. Tendo em vista os objetivos do estudo, que visam estabelecer um conjunto robusto de métricas para avaliação da gestão de projetos em plataformas de código aberto, a metodologia aqui delineada propõe uma abordagem sistemática e detalhada. Esta abordagem não só facilitará a compreensão dos critérios que definem o sucesso de um projeto, mas também fornecerá *insights* pertinentes sobre a dinâmica das comunidades *Open Source*, em especial plataformas como o GitHub.

O propósito central desta dissertação é explorar e entender as métricas fundamentais que influenciam o sucesso de projetos em plataformas de design aberto. Em virtude da natureza exploratória desta investigação e com o intuito de identificar novas perspetivas e padrões, a abordagem metodológica adotada é de caráter indutivo. Esta não se baseia rigidamente em teorias já estabelecidas, mas sim na recolha e avaliação de dados, permitindo a identificação de tendências e padrões emergentes. Devido ao foco na obtenção e análise de métricas específicas e mensuráveis, a investigação adota uma vertente quantitativa. Tal escolha metodológica assegura uma avaliação objetiva e fundamentada dos projetos, sustentada por dados precisos e quantificáveis.

3.1. Métricas de Avaliação do Sucesso em Plataformas de Desenvolvimento de Código Aberto

Grande parte das investigações foca-se na análise do comportamento dos membros em comunidades virtuais, contudo, existe ainda um número limitado de estudos sobre a gestão dessas comunidades, apesar de o controlo na gestão ser vital para o êxito das mesmas. Avaliar o sucesso torna-se um elemento-chave para a direção eficiente de qualquer comunidade *online*. O papel do gestor da comunidade vai além de simplesmente entender as necessidades dos membros, envolve também a deteção de padrões e evoluções significativas para manter a comunidade – o seu crescimento, atividade e senso da comunidade – e a capacidade de antecipar e corrigir tendências negativas antes que se transformem em problemas.

As métricas para quantificar o sucesso das comunidades online são bastante diversificadas. Os responsáveis pela gestão das comunidades devem reunir uma quantidade de dados de maneira contínua, consistente e precisa durante toda a existência da comunidade. Contudo, a recolha regular desses dados detalhados pode não ser uma tarefa viável para a maioria dos gestores de comunidade.

Os critérios de sucesso são fundamentais em todos os projetos, uma vez que auxiliam as comunidades a estabelecer métodos eficientes de ação coletiva e autogestão, bem como a criar e obter valor de forma efetiva. Estes critérios também contribuem na formação de uma identidade coesa e um conjunto de práticas amplamente reconhecidas. Além disso, as métricas têm a capacidade de avaliar práticas e elementos determinantes para o sucesso (Antoniou et al., 2022).

Para compreender o sucesso de projetos em comunidades abertas deve-se responder às seguintes perguntas:

- Que características e práticas estão presentes em projetos OS bem-sucedidos?

- Quais as métricas que podem ser usadas para medir o sucesso em projetos OS?

Os gestores de comunidade devem atentar a indicadores tais como:

- **Crescimento da Comunidade**: Isto abrange o monitoramento do número de inscritos no projeto (membros), bem como os que estão ativamente envolvidos (membros ativos), como os que contribuem para o projeto com código ou reportam *bugs*. Inclui também a observação das contribuições em diferentes períodos a nível temporal. Um elevado número de inscritos, em relação ao número de participantes, pode revelar uma baixa taxa de conversão, sinalizando a existência de barreiras à participação, como um ambiente desencorajador para novos contribuidores ou a complexidade inerente ao projeto.
- **Exibição de Atividade e Envolvimento**: A avaliação da atividade e do compromisso pode ser feita através do acompanhamento de métricas como o número de *commits*, *pull requests*, *issues* abertas e fechadas, e o número de participações em discussões em cada mês. O tempo médio para um *pull request* receber uma resposta ou ser *merged* e a frequência de visitas de cada membro ativo também são indicadores relevantes.
- **Melhoria da Comunidade, Identificação de Desafios e Validação do que é eficaz**: A monitorização de *commits* bem-sucedidos, a deteção de problemas recorrentes nas *issues*, e a identificação dos aspetos do projeto que atraem o maior envolvimento dos membros podem ajudar na melhoria contínua da comunidade e do projeto. Por exemplo, se uma funcionalidade específica está a originar várias *issues*, isso pode sinalizar um problema que necessita de solução.
- **Relatório de Progresso e Demonstração do Valor da Comunidade aos *Stakeholders***: Os gestores podem utilizar os dados para reconhecer os tipos de contribuições e atividades que fomentam maior envolvimento e identificar os comportamentos dos membros que tendem a impulsionar o aumento da atividade na comunidade.

Para reunir informações acerca da comunidade, diversos aspetos devem ser medidos e monitorizados. O número de contribuidores do projeto, o número de utilizadores inscritos na lista de discussão, a quantidade de solicitações, *feedback* ou perguntas recebidas, assim como o número de acessos à plataforma e o número de acessos que o projeto recebe e qual é a quantidade de atividade na visibilidade do projeto nas redes sociais, como o Twitter. A distribuição geográfica dos membros da comunidade pode também ser observada.

Informações diretas sobre a atividade na comunidade podem ser obtidas através do número de pessoas que participam em eventos e reuniões do projeto e do número de publicações científicas que fazem referência à comunidade. Do ponto de vista técnico, o número de *downloads*, a quantidade de *bugs* reportados, os recursos solicitados e o volume e o impacto das contribuições recebidas fornecem *insights* significativos sobre o desenvolvimento realizado fora do núcleo da comunidade. Outro aspeto interessante a ser analisado é o número de projetos desenvolvidos na plataforma (Kilamo et al., 2012).

Para avaliar o uso, visibilidade e sucesso consideram-se duas métricas: os *downloads* e as estrelas. Essas métricas refletem duas dimensões distintas do uso dos projetos: o número de vezes que são descarregados e a quantidade de *feedback* social positivo que recebem (estrelas). As duas métricas destacam diferentes aspetos do uso. Os *downloads* apresentam uma medida de uso mais técnica. As estrelas são mais sugestivas de visibilidade e, juntamente com outras formas de *feedback* social, incluindo seguidores e patrocinadores, desempenham um papel significativo como indicadores de

reconhecimento dentro da comunidade (Jarczyk et al., 2014). Contudo, a métrica estrela não indica o uso do *software* ou *hardware* (Cosentino et al., 2017; Schueller et al., 2022).

Os *forks* tendem a ser mais representativos da dimensão da comunidade do que as avaliações positivas (estrelas). Dado que os *forks* são frequentemente utilizados no processo de contribuição, esta métrica é um indicador relevante para a comunidade de *developers* (Link & Vargas, 2022).

Os responsáveis pela gestão da comunidade devem recorrer a um gráfico de *forks*, permitindo aos líderes do projeto visualizar a frequência com que o projeto foi copiado num dia específico, com uma distinção entre o número total de *forks* e os *cloners* únicos.

Já Kolassa et al. concentraram-se na frequência de *commits* nos projetos, identificando cada *commit* como unidade fundamental de trabalho e uma contribuição vital ao código. Esta perspetiva destaca a importância da atividade constante e da contribuição regular como indicadores de um projeto saudável.

Existem várias métricas que podem servir como indicadores fiáveis da popularidade. Para além do número de *forks* e avaliações positivas (estrelas), destacam-se também o número de seguidores e visualizações (Cosentino et al., 2017).

Projetos que beneficiam de uma manutenção constante e ativa tendem a ter uma maior viabilidade futura em comparação com aqueles que apenas registam contribuições esporádicas. Além disso, uma interação eficaz com o conjunto de utilizadores, bem como a existência de documentação adequada, estão associadas à continuidade e sucesso a longo prazo do projeto.

A sobrevivência dos projetos pode também ser influenciada pela habilidade de atrair e manter colaboradores. Fatores como processos de contribuição descomplicados, disponibilidade de documentação, complexidade e popularidade do projeto estão frequentemente ligados à capacidade de atrair novos contribuidores. É comum que os projetos sejam mais eficazes em manter os colaboradores existentes do que em atrair novos.

A popularidade de um projeto desempenha um papel crucial na atração de novos programadores, uma vez que um projeto popular é frequentemente percebido como de elevada qualidade e, portanto, digno de investimento. Esta popularidade, juntamente com uma manutenção ativa e uma boa interação com a base de utilizadores, pode ser vista como um indicador vital para a sobrevivência futura do projeto, reforçando a noção de que a gestão cuidadosa e a atenção aos detalhes são fundamentais para o sucesso a longo prazo (Cosentino et al., 2017).

Além disso, as métricas de sucesso mais comuns nos projetos de código aberto incluem o número de contribuidores e o crescimento contínuo da base de contribuidores (McDonald & Goggins, 2013). Estes fatores, quando combinados com a frequência de *commits* e a resposta da equipa, não só refletem a qualidade do projeto, mas também a sua capacidade de atrair e reter colaboradores.

Uma métrica adicional que os responsáveis pelo projeto devem monitorizar é a quantidade de visitantes à documentação da *Application Programming Interface* (API). A utilização da documentação é, frequentemente, um indicativo de que o código está a ser utilizado (McDonald & Goggins, 2013).

Pode também ser útil compreender quem está a integrar a comunidade. No entanto, a recolha completa de dados demográficos durante o processo de registo pode não ser recomendável, pois pode constituir um obstáculo à participação. Alternativamente, os membros que completam o seu

perfil podem ser vistos como uma medida de envolvimento e senso de comunidade, ou seja, uma medida de envolvimento inicial bem-sucedida. Os membros que não preenchem o seu perfil representam uma oportunidade para os responsáveis pela comunidade entrarem em contacto e obterem feedback valioso sobre o motivo pelo qual não concluíram o perfil. Essa situação pode ser por um problema de usabilidade ou o membro precisa de incentivo ou orientação para se sentir integrado na comunidade online (Young, 2013).

Adicionalmente, é possível avaliar o intervalo de tempo existente entre as várias fases do processo de contribuição, tais como:

- A duração média durante a qual um *issue* permanece em aberto;
- A verificação se os *issues* são resolvidos através de *pull requests* (PRs);
- A identificação e encerramento de *issues* desatualizados;
- O tempo médio necessário para realizar um *merge* de uma *pull request*.

Outros aspetos da comunidade, por exemplo, a quantidade de comunicação ou o nível de comprometimento, podem ser igualmente monitorados (Kilamo et al., 2012).

A realização periódica de medições representativas dentro da comunidade pode ser útil. Por exemplo, o conhecimento acerca do número de contribuições feitas por cada membro ativo num período específico pode demonstrar se a atividade é compartilhada entre muitos ou apenas alguns membros, permitindo identificar os líderes de cada repositório e reconhecer aqueles que contribuem com menor frequência. É igualmente importante monitorizar a rapidez da resposta, a quantidade e a natureza das mesmas. Uma resposta mais ágil pode ser indicativa de uma maior atividade no repositório em questão.

É vital compreender e monitorizar as várias métricas abordadas que refletem a saúde, o crescimento e o envolvimento da comunidade. Estas métricas fornecem *insights* sobre a eficácia das estratégias de envolvimento, a satisfação dos membros da comunidade e a sustentabilidade geral do projeto. A Tabela 6 resume as principais métricas diárias, mensais e anuais que podem ser avaliadas, juntamente com perguntas-chaves que ajudam a interpretar o que essas métricas podem significar para o projeto. Apesar da abrangência das métricas referidas, poderá não ser sensato proceder à medição de todas, e, de igual modo, poderá ser necessário adicionar uma métrica que se revele pertinente.

Tabela 6 – Métricas de Avaliação e Perguntas-Chave para Projetos Open Source.

Medições (diárias, mensais e/ou anuais)	Perguntas-Chave
Número de Contribuidores (Iriberri & Leroy, 2009; Kaur et al., 2022; Kilamo et al., 2012; Şeker et al., 2021; Young, 2013)	Este valor está em crescimento? Que estratégias podem ser implementadas para encorajar mais contribuidores?
Número de Utilizadores (Iriberri & Leroy, 2009; Kaur et al., 2022; Kilamo et al., 2012; Young, 2013)	A quantidade de utilizadores está a aumentar de forma consistente? Como se pode atrair mais utilizadores?
Número de Empresas Envolvidas (Salo et al., 2015)	Existem empresas a participar? Como se pode melhorar a colaboração corporativa?
Número de Contribuições ou C*ommits* (Iriberri & Leroy, 2009; Kaur et al., 2022; Padhye et al., 2014; Şeker et al., 2021; Young, 2013)	Os membros da comunidade estão a contribuir ativamente? Quais são os fatores que motivam ou impedem as contribuições?
Diversidade de Contribuidores (Padhye et al., 2014)	A comunidade está diversificada em termos de localização geográfica, setores, competências? Como se pode promover uma maior diversidade?
Número de *Downloads* (Antoniou et al., 2022; Kilamo et al., 2012; Schueller et al., 2022)	O projeto está a ser utilizado? Existem estratégias para aumentar o número de downloads?
Número de Estrelas (Jarczyk et al., 2014; Schueller et al., 2022)	Qual é o nível de popularidade do projeto? Como se pode aumentar a visibilidade do projeto?
Número de *Forks* (Padhye et al., 2014)	Quantos utilizadores estão a adaptar o projeto às suas necessidades? Como se pode facilitar este processo?
Número de *Issues* (Şeker et al., 2021; Young, 2013)	Estão a surgir muitos problemas? Como se pode melhorar a eficiência na resolução de problemas?
Tempo de Resposta a P*ull Requests* (Cosentino et al., 2017; Jarczyk et al., 2014; Şeker et al., 2021)	As *pull requests* são aceites a uma taxa satisfatória? Como se pode otimizar a taxa de aceitação?
Número de Dependências do Projeto	Quantos outros projetos dependem deste? Este projeto é essencial para outros projetos?
Atividade de C*ommit* (Kilamo et al., 2012; Xia et al., 2022; Young, 2013)	Qual é o nível de atividade da comunidade em termos de contribuições? Como se pode incentivar uma maior atividade?
Empenho da Comunidade (Iriberri & Leroy, 2009; Young, 2013)	A comunidade está comprometida e investida no projeto? Como se pode aumentar o envolvimento?
Número de Parcerias/*Sponsors*	Quantas outras plataformas ou empresas estão envolvidas? Como se pode expandir as parcerias?
Nível de satisfação da comunidade (Antoniou et al., 2022; Iriberri & Leroy, 2009; Kilamo et al., 2012)	Os membros da comunidade estão satisfeitos? Existem formas de aumentar a satisfação?
Número de Utilizadores Ativos (Young, 2013)	Quantos utilizadores estão ativamente envolvidos? Como se pode aumentar a atividade dos utilizadores?
Número de Contribuidores Ativos (Young, 2013)	Quantos contribuidores estão ativamente envolvidos? Como se pode aumentar a atividade dos contribuidores?
Número de R*eleases*	O projeto é atualizado regularmente? Como se pode melhorar a frequência e a qualidade das atualizações?

Tabela 6 - Métricas de Avaliação e Perguntas-Chave para Projetos Open Source.

Medições (diárias, mensais e/ou anuais)	Perguntas-Chave
Nível de Adoção de Comunidades ou Projetos (Externa)	Outras comunidades estão a utilizar este projeto? Como se pode promover uma maior adoção?
Atividade no Fórum ou Canal de Discussão (Young, 2013)	A comunicação está a fluir eficientemente? Como se pode aumentar a atividade nos canais de comunicação?
Taxa de Retenção de Membros	Os membros permanecem na comunidade? Como se pode aumentar a retenção de membros?
Tempo Médio de Resposta às I*ssues*	O tempo de resposta aos problemas é adequado? Como se pode melhorar o tempo de resposta?
Número de *Pull Requests*	Estão a ser feitas contribuições ativas e melhorias ao projeto? Como se podem incentivar mais contribuições?
Nível de Documentação (Antoniou et al., 2022)	A documentação é suficiente e eficaz? Como se pode aprimorar a documentação?
Número de Visitas à Plataforma e Página do Projeto (Antoniou et al., 2022; Young, 2013)	O projeto está a atrair visitantes? Como se pode aumentar o número de visitantes?

Esta tabela serve como um guia abrangente para os gestores de comunidade, auxiliando-os a concentrar-se em áreas críticas que possam requerer atenção ou aprimoramento. Cada métrica e questão associada é concebida para proporcionar uma perspetiva nítida do estado atual do projeto e direcionar decisões futuras. Por exemplo, o número de contribuidores ativos e a taxa de retenção de membros podem denotar o grau de envolvimento dentro da comunidade, ao passo que o número de downloads e estrelas podem espelhar a popularidade e a aceitação do projeto no contexto de mercado mais vasto.

Ao monitorizar estes dados de forma regular, consistente e rigorosa ao longo da existência da comunidade, os responsáveis pela gestão podem tomar decisões bem fundamentadas que favorecem o sucesso contínuo da comunidade e do projeto em questão. Sendo que é importante ter em conta que utilizar métricas é positivo como forma de informar a tendência, mas usá-las apenas como o único método de sucesso pode levar a problemas.

Após a análise das métricas gerais que podem ser aplicadas ao monitoramento e avaliação de projetos de código aberto, é igualmente importante considerar as diferentes componentes envolvidas dentro do ecossistema digital. A Tabela 7 oferece uma visão mais detalhada e segmentada das métricas consideradas, categorizando-as de acordo com diferentes componentes, como a Plataforma, Contribuidores, Proprietários, *Maintainers*, Utilizadores, Empresas e Projetos. Por exemplo, enquanto o número de contribuidores é uma métrica relevante para a maioria dos componentes, o número de *releases* pode ser de particular interesse para a plataforma e os projetos em si.

Tabela 7 - Métricas de Avaliação e os seus Componentes associados em Projetos e Plataformas de *Open Source*.

	Componentes						
Métricas	Plataforma	Contribuidores	Proprietários	*Maintainers*	Utilizadores	Empresas	Projetos
Número de Contribuidores (Iriberri & Leroy, 2009; Kaur et al., 2022; Kilamo et al., 2012; Şeker et al., 2021; Young, 2013)	✓	✓	✓	✓		✓	✓
Número de Utilizadores (Iriberri & Leroy, 2009; Kaur et al., 2022; Kilamo et al., 2012; Young, 2013)	✓				✓		
Número de Empresas Envolvidas (Salo et al., 2015)						✓	✓
Número de Contribuições ou C*ommits* (Iriberri & Leroy, 2009; Kaur et al., 2022; Padhye et al., 2014; Şeker et al., 2021; Young, 2013)		✓	✓	✓		✓	✓
Diversidade de Contribuidores (Padhye et al., 2014)		✓	✓	✓		✓	✓
Número de *Downloads* (Antoniou et al., 2022; Kilamo et al., 2012; Schueller et al., 2022)					✓		✓
Número de Estrelas (Jarczyk et al., 2014; Schueller et al., 2022)					✓		✓
Número de *Forks* (Padhye et al., 2014)					✓		✓
Número de *Issues* (Şeker et al., 2021; Young, 2013)		✓		✓	✓		✓
Tempo de Resposta a P*ull Requests* (Cosentino et al., 2017; Jarczyk et al., 2014; Şeker et al., 2021)		✓	✓	✓			✓
Número de Dependências do Projeto							✓
Atividade de C*ommit* (Kilamo et al., 2012; Xia et al., 2022; Young, 2013)		✓		✓			✓
Empenho da Comunidade (Iriberri & Leroy, 2009; Young, 2013)	✓				✓		✓
Número de Parcerias/*Sponsors*	✓		✓			✓	
Nível de satisfação da comunidade (Antoniou et al., 2022; Iriberri & Leroy, 2009; Kilamo et al., 2012)	✓				✓		
Número de Utilizadores Ativos (Young, 2013)					✓		✓

Tabela 7 - Métricas de Avaliação e os seus Componentes associados em Projetos e Plataformas de *Open Source*.

	Componentes						
Métricas	Plataforma	Contribuidores	Proprietários	*Maintainers*	Utilizadores	Empresas	Projetos
Número de Contribuidores Ativos (Young, 2013)		✓	✓	✓		✓	✓
Número de *Releases*							✓
Nível de Adoção de Comunidades ou Projetos (Externa)	✓				✓		✓
Atividade no Fórum ou Canal de Discussão (Young, 2013)					✓		✓
Taxa de Retenção de Membros	✓	✓	✓	✓	✓	✓	✓
Tempo Médio de Resposta às I*ssues*		✓	✓	✓			✓
Número de *Pull Requests*							✓
Nível de Documentação							✓
Número de Visitas à Plataforma e Página do Projeto (Iriberri & Leroy, 2009; Kilamo et al., 2012; Young, 2013)	✓				✓		✓

3.2. Modelo *Onion* e Estrutura das Comunidades *Open Source*

Para uma compreensão aprofundada da evolução em projetos código aberto, é imperativo analisar tanto o sistema e a comunidade, que é a força motriz da evolução. O processo de fomentar uma comunidade próspera é complexo e é necessário ter em conta vários aspetos que devem ser considerados. A comunidade de código aberto geralmente é modelada através de uma estrutura hierárquica em camadas (Christian & Vu, 2020).

Uma comunidade de código aberto é dividida em camadas de acordo com o nível de envolvimento com o desenvolvimento e o papel no projeto em geral.

Assim, para análise de projetos de Open Source (OS), é essencial focar nos papéis que os membros da comunidade desempenham e a estrutura da comunidade medida pelas interações colaborativas entre esses distintos papéis. Um modelo amplamente citado que ilustra uma estrutura de uma comunidade de código aberto é proposto por (Nakakoji et al., 2002). O modelo descreve a comunidade em camadas, denominado por Modelo *Onion*, ou Modelo Núcleo-Periferia, com os papéis mais preponderantes no núcleo, e cada camada subsequente composta por papéis de decrescente influência, conforme se ilustra na Figura 13 .

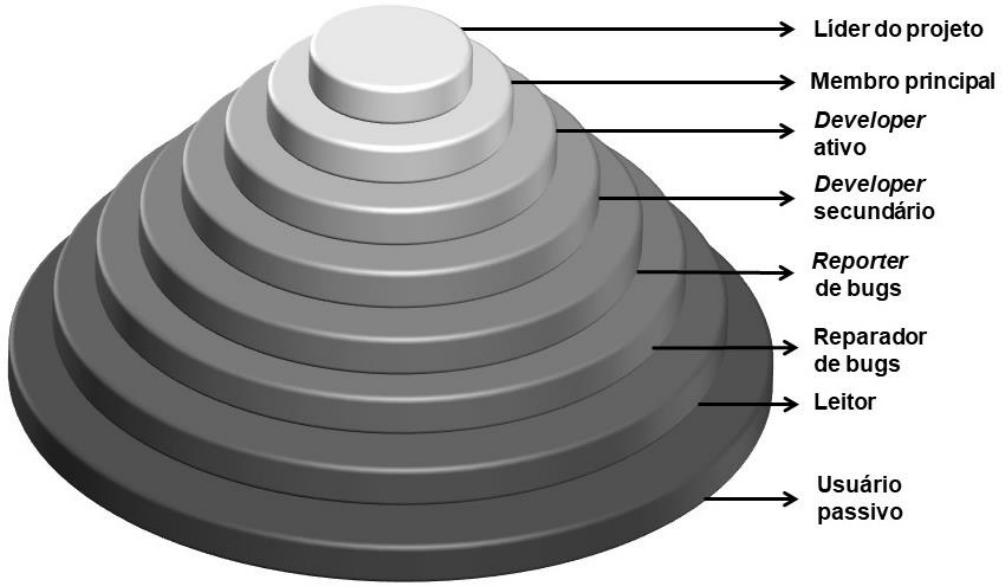


Figura 13 - Modelo *Onion* de comunidades *Open Source*. Adaptado de Kilamo et al., 2012; Nakakoji et al., 2002

Esta configuração da comunidade é de tal importância que se acredita que qualquer contribuição técnica não só altera o código, mas também reconfigura o papel dos membros participantes, modificando assim a dinâmica social da comunidade OSS. O modelo é uma estrutura orientada para tarefas frequentemente adotada, sublinhando diferenças na participação, competência e relevância entre subgrupos responsáveis por tarefas específicas. Através deste modelo, os estudos têm sido capazes de explicar e entender a dinâmica da comunidade no desenvolvimento de código aberto (Christian & Vu, 2020; Middleton et al., 2018).

No núcleo encontra-se o líder do projeto, que normalmente é a figura que deu origem ao projeto e que se destaca, ou destacou, como o *developer* mais ativo. Um típico projeto de código aberto tem

um pequeno grupo de membros principais que participam há muito tempo do projeto. Estes coordenam o projeto e contribuem, em muito, com o código. Conforme o estudo de Gloor (2006) , observa-se que um núcleo típico começa com três a sete membros e cresce até dez a 15 membros, quando a comunidade é estabelecida. Os utilizadores passivos, situados na camada mais externa, geralmente constituem a maioria e limitam-se a utilizar o *software* ou *hardware*. Contudo, mesmo os utilizadores e leitores passivos, que se encontram nas camadas externas da estrutura, são cruciais para uma comunidade sustentável.

Os membros dos projetos OS assumem determinados papéis de acordo com o seu interesse pessoal no projeto. Evidentemente, este modelo de comunidade de código aberto não representa todo o ecossistema. Existem parceiros comerciais e industriais e grupos de interesse semelhantes que, embora não estejam diretamente contemplados no modelo, desempenham um papel vital no ecossistema de código aberto (Kilamo et al., 2012).

A harmonia entre as camadas em grandes comunidades OSS é fundamental para o desenvolvimento saudável dos projetos. No entanto, o Modelo *Onion*, por si só, pode não refletir adequadamente a mobilidade dos *developers* entre os diferentes níveis. Ainda é reconhecido que a maioria das colaborações tende a ser efémera, e até os membros mais centrais podem deslocar-se para posições mais periféricas. Esta estrutura é frequentemente usada para entender as dinâmicas dos projetos, reconhecendo ao mesmo tempo a fluidez na posição dos contribuidores ao longo do tempo (Middleton et al., 2018).

A lista de tarefas é normalmente reconhecida ao nível do projeto e engloba tarefas tanto de discussão quanto de implementação. A ordem em que estas tarefas estão organizadas fundamenta-se em diversos critérios, tais como a complexidade técnica associada, dimensão proporcional do grupo que realiza essa tarefa e o impacto do grupo no projeto.

Os principais projetos de OS são comunidades altamente hierárquicas e meritocráticas. Cinco diferentes status são geralmente distinguidos nesses projetos, de acordo com os direitos e poderes distintos dos participantes. Alguns participantes podem modificar o código e participar diretamente e nas decisões sobre o software ou hardware:

- O líder do projeto;
- A equipa principal ou proprietários, encarregados de preservar a base do código e a documentação associada;
- Os *developers* ou colaboradores, que contribuem para o desenvolvimento e manutenção de segmentos específicos do projeto;
- Os utilizadores. Podem ser chamados de utilizadores ativos se participarem das discussões, auxiliam novos membros, relatam ou corrigem *bugs* como *patches* e propõem novos componentes. Esses utilizadores ativos num determinado projeto podem ser *developers* noutro projeto. Por outro lado, existem os utilizadores passivos, que se limitam ao uso do software ou hardware, ou a consultar os fóruns de discussão e a documentação do projeto.

A Tabela 8 fornece uma lista dessas atividades, a sua posição estrutural subjacente e o seu âmbito de ação.

Tabela 8 - Papéis orientados a tarefas no desenvolvimento de *Open Source. Adaptado de* Barcellini et al., 2014; Christian & Vu, 2020; Jensen & Scacchi, 2007; Jergensen et al., 2011; Nakakoji et al., 2002

Posição Estrutural	**Função**	**Descrição**	**Principal Área de Interação**
Núcleo	Líder do projeto	Inclui proprietários de projetos, administradores da comunidade e gestores de projetos	Canais de comunicação privados
	Membro Principal	Orientação e coordenação das atividades de desenvolvimento	Canais de comunicação privados
	Developers ativos	Contribuidores regulares para o projeto através de novas funcionalidades e correções de erros	Repositório
Periferia	*Developers* secundários	Contribuidores esporádicos que contribuem com novas funcionalidades e correções de erros	Repositório
	Reporter de *bugs*	Especialistas em identificar e reportar correções de erros	Ferramentas de monitorização de erros
	Reparador de *bugs*	Especialistas em correções de erros	Repositório e ferramentas de monitorização de erros
	Utilizadores ativos	Utilizadores do *software* que podem contribuir em tarefas menos técnicas, como documentação e assistência de utilizador a utilizador	Redes sociais, fóruns (oficiais e não oficiais) e *wikis*
	Utilizadores passivos	Utilizadores do *software* que não oferecem contribuições diretas ao projeto	Lista de leitura, blogs, redes sociais, site oficial do projeto

É evidente que a progressão nos estatutos dentro de um projeto de código aberto é alcançada através da demonstração e validação de competências técnicas e da capacidade de participação ativa em discussões *online*. Estes estatutos não são estáticos; emergem e são moldados ativamente pela comunidade. O papel desempenhado por cada membro reflete o seu comportamento e envolvimento efetivo no projeto. Em determinadas situações, as atividades desempenhadas podem alinhar-se com as expectativas associadas ao seu estatuto.

Os pilares de qualquer projeto de código aberto são a sua comunidade e o repositório. Ferramentas e tecnologias que promovam a comunicação entre os membros, a coordenação do projeto e a gestão eficiente são essenciais. Exemplo destas ferramentas é o fórum de discussão. A importância da comunidade e a complexidade da gestão de projetos em plataformas online levaram à criação da função de gestor de comunidade. Enquanto o líder do projeto tem uma abordagem mais técnica, focando no código e na colaboração com os programadores, o gestor de comunidade assegura a saúde e dinâmica da comunidade, interagindo por exemplo, com os utilizadores e *developers*, e facilitando a organização do projeto (Kilamo et al., 2012; Michlmayr, 2009).

Uma das principais funções do gestor da comunidade é assegurar que todos possam contribuir eficazmente. Para maximizar a eficiência dos *developers* principais e fomentar uma comunidade crescente e saudável, é imperativo que exista alguém dedicado a esta gestão. Os gestores de

comunidade devem liderar, mas também estar atentos às necessidades da comunidade. Neste contexto aberto, os membros do projeto têm limitada autoridade formal sobre os *developers* e vice-versa (Tsay et al., 2014).

Um estudo de (Middleton et al., 2018) identificou três fatores principais que influenciam a aceitação de novos membros em equipas de código aberto:

- A frequência de *pull requests* de um *developer*, sendo este o fator mais significativo;
- As atividades e qualidades independentes do *developer*, como o seu histórico com projetos pessoais e outras equipas;
- As características da equipa.

Estes *insights* sublinham a relevância de uma contribuição ativa, uma reputação sólida na comunidade e uma cultura de equipa inclusiva. As plataformas de código aberto podem capitalizar estas observações, promovendo recursos que facilitem a participação ativa, mantendo a transparência sobre o envolvimento dos *developers* e fomentando uma cultura colaborativa. Ao adotar tais práticas, as plataformas podem potenciar a formação de equipas eficientes e o crescimento sustentável das comunidades de *hardware* e *software* (Middleton et al., 2018).

Antes de detalhar os recursos destas plataformas, é crucial identificar os seus principais componentes:

- **Repositório/Projeto:** Um repositório contém todos os ficheiros de código-fonte de um projeto. Este pode ser visto como uma estrutura para um repositório, onde se destaca uma página específica do projeto que exibe problema, discussões, histórico de *commits*, entre outras informações relevantes.
- **Questões/Problemas**: Estes podem abranger desde *bugs*, pedidos de funcionalidades, melhorias, entre outro tipo de relatórios relacionados a um repositório.
- **Colaboradores e Contribuidores:** Os colaboradores são integrantes da equipa do projeto com permissões de leitura e escrita, enquanto os contribuidores podem, por exemplo, reportar problemas e enviar *pull requests*, discutindo as suas perspetivas com os principais *developers* (colaboradores), mas não têm permissão de escrita. Além disso, o proprietário (autor) do projeto/repositório, que por padrão é o criador do mesmo, tem privilégios totais para o gerir e aceder, podendo também definir o nível de permissões para os colaboradores do projeto.

A maioria das plataformas de código aberto está equipada com uma série de recursos que promovem o envolvimento da comunidade, facilitando a colaboração entre *developers* e otimizando o desenvolvimento de software e hardware aberto. Estes recursos podem ser agrupados nas seguintes categorias (Alamer & Alyahya, 2017):

- **Gestão de Repositórios e Projetos:** Plataformas como o GitHub, SourceForge e GitLab, oferecem soluções robustas de gestão. Estas permitem a formação de diversas equipas, cada uma com permissões específicas de acesso ao repositório. Assim, em vez de atribuir permissões a cada membro, criar equipas com permissões diferentes e convidar membros para essas equipas é uma abordagem de gestão disponível. O proprietário de um projeto pode criar um número ilimitado de equipas, nomeá-las e atribuir-lhes determinadas permissões de acesso. Adicionalmente, é essencial que todas as tarefas, incluindo *issues* e *pull requests*, sejam devidamente organizadas. Algumas plataformas, como o GitHub, disponibilizam ferramentas de gestão que permitem aos programadores estruturar o seu trabalho de forma eficiente. As equipas, de forma simples, podem ser organizadas e geridas da seguinte forma:
    - Equipa de *Developers*: Com permissões de leitura e escrita;
    - Equipa de *Maintainers*: Com permissões de leitura;
    - Equipa do Proprietário: Com acesso total a todos os repositórios.

- **Funcionalidades Sociais:** Estas têm revolucionado o desenvolvimento colaborativo. Plataformas como GitHub e Bitbucket introduziram recursos sociais que facilitam a interação entre *developers*. Estes incluem a criação de perfis pessoais com informações pessoais básicas, a capacidade de seguir e ser seguido por outros *developers*, receber notificações sobre atividades relevantes e destacar projetos de interesse. A quantidade de seguidores pode influenciar a reputação de um *developer* na comunidade. A rede de seguidores de um desenvolvedor pode refletir e influenciar a sua reputação na comunidade. De facto, todas as plataformas oferecem aos seus utilizadores a oportunidade de criar e personalizar um perfil. No entanto, a profundidade e o tipo de informação que um perfil exibe variam consideravelmente de uma plataforma para outra. Um perfil bem estruturado pode revelar rapidamente o nível de atividade de um desenvolvedor e os seus campos de interesse. Uma característica distintiva em algumas plataformas, como o GitHub, BitBucket e OHWR, é a presença de gráficos de contribuição. Estes gráficos proporcionam uma visão consolidada das atividades e contribuições do desenvolvedor desde a criação do perfil. Em contraste, outras plataformas optam por apresentar as atividades mais recentes em formato de fluxo, sem a utilização de gráficos. Há ainda plataformas que optam por uma abordagem mais minimalista, mostrando apenas informações básicas e omitindo detalhes sobre as atividades recentes do desenvolvedor.
- **Revisão de Código**: Este recurso é vital para assegurar a qualidade do código, o que é perdido em algumas plataformas, como é o caso do SourceForge. Esta prática possibilita que tanto os principais desenvolvedores quanto os externos partilhem *feedback* sobre o código. Ao iniciar uma revisão, cria-se um espaço para discussão, onde os revisores podem partilhar sugestões e observações. Com base neste feedback, os principais *developers* têm a responsabilidade de decidir se integram ou não as sugestões e modificações propostas.

- ***Pull Requests***: Estas solicitações são frequentemente usadas para contribuições de código e revisões. Existem, essencialmente, dois modelos principais de *pull requests*. No primeiro, os proprietários e colaboradores do projeto propõem alterações e solicitam *feedback*, permitindo uma revisão abrangente do código. No segundo modelo, um contribuidor externo, com acesso limitado ao repositório, sugere melhorias após criar um *fork* do projeto original. Os *maintainers* do projeto avaliam estas sugestões, decidindo integrá-las ou não. Em ambos os modelos, a discussão é fundamental, enriquecendo o projeto com diferentes perspetivas e experiências. Estas práticas são características distintivas de plataformas de software de código aberto reconhecidas, como o GitHub, e são facilitadas por interfaces gráficas intuitivas.
- **Monitorização de Problemas**: A gestão eficaz de problemas é essencial nas plataformas de desenvolvimento, permitindo uma colaboração fluida entre contribuidores e colaboradores. A correção de bugs é uma das atividades mais relevantes, frequentes, demoradas e dispendiosas no desenvolvimento de código. Os chamados "bugs" em muitas comunidades de OS são os defeitos e falhas do código. Esta gestão abrange desde *bugs* e defeitos até solicitações de recursos e tarefas. Estas plataformas facilitam a pesquisa, filtragem e atribuição de problemas, bem como a discussão em torno deles. Os colaboradores têm a capacidade de atualizar o estado dos problemas, fechá-los ou reabri-los conforme necessário. Enquanto algumas plataformas focam principalmente na monitorização de *bugs*, a maioria oferece ferramentas abrangentes que rastreiam uma variedade de problemas (Ramírez-Mora et al., 2021).

Como resultado da pesquisa e observação de certas plataformas, como o GitHub, BitBucket, Open Hardware Repository e GitLab, conforme mostrado na Tabela 9, é demonstrada a disponibilidade de vários tipos de recursos nessas plataformas.

Tabela 9 - Funcionalidades detalhadas das plataformas online OS. Adaptado de Alamer & Alyahya, (2017)

✓ - Indica que a plataforma tem a *feature* completa ± - Indica que a plataforma tem parte da *feature* Células destacadas a azul indicam que a *feature* pode ser feita ou mostrada aos visitantes. Não é necessário registo ou início de sessão. **FUNCIONALIDADES**			**GitHub**	**GitLab**	**OHWR**	**SourceForge**	**BitBucket**
Repositório/Gestão de Projetos		Criar um projeto/repositório	✓	✓	O projeto tem de ser aprovado primeiro	✓	✓
		Eliminar um projeto/repositório	✓	✓		✓	✓
		Carregar o *Source Code* para um repositório:	✓	✓		✓	✓
	1	Através de ficheiros de arrastar e largar	✓				
	2	Criar e editar código online diretamente no browser	✓	✓		✓	✓
	3	Importar um repositório de outras plataformas de código aberto	✓	✓		✓	✓
	4	Através de uma linha de comandos	✓	✓		✓	✓
		Procurar repositórios de outros utilizadores na comunidade código aberto	✓	✓	✓	✓	✓

Tabela 9 - Funcionalidades detalhadas das plataformas *online* OS. Adaptado de Alamer & Alyahya (2017)

✓ - Indica que a plataforma tem a *feature* completa ± - Indica que a plataforma tem parte da *feature* Células destacadas a azul indicam que a *feature* pode ser feita ou mostrada aos visitantes. Não é necessário registo ou início de sessão. **FUNCIONALIDADES**		**GitHub**	**GitLab**	**OHWR**	**SourceForge**	**BitBucket**
Repositório/Gestão de Projetos	Filtrar repositórios/projetos por vários critérios	✓	✓	✓	✓	✓
	Convidar um colaborador para um repositório/projeto	✓	✓		✓	✓
	Definir permissões de acesso ao repositório/projeto	✓	✓	✓	✓	✓
	Subscrever um projeto - receber notificações	✓	✓		✓	✓
	Permitir repositórios/projetos privado	✓	✓		✓	
	Aceder e transferir o *source code* do repositório/projeto	✓	✓	✓	✓	✓
	Mostrar o histórico de todos os *commits* de um repositório através da interface do utilizador	✓	✓	✓	✓	✓
	Criar vários grupos/equipas para um projeto	✓	✓		✓	
	Convidar membros para grupos/equipas do projeto	✓	✓		✓	✓
	Definir permissões de acesso a projetos para cada grupo/equipa	✓	✓		✓	✓
	Organizar itens de trabalho em fases	✓	✓			
	Rever projetos					
***Features* Sociais**	Perfil pessoal	✓		✓		✓
	Seguir outros *developers* na comunidade OS	✓				✓
	Seguir projetos/repositórios					
	Avaliar projetos/repositórios (com estrelas)	✓	✓	✓		
	Mostrar *developers* populares	✓				✓
	Mostrar os projetos em alta (projetos/repositórios populares)	✓	✓	✓	✓	✓
	Avaliar projetos				✓	
Code Review	Solicitar um código de revisão – rever uma parte do código antes de ser incluído	✓	✓	✓		✓
	Selecionar *reviewer*	✓	✓	✓		✓
	Discutir o código que está a ser revisto	✓	✓	✓		✓

Tabela 9 - Funcionalidades detalhadas das plataformas *online* OS. Adaptado de Alamer & Alyahya (2017)

✓ - Indica que a plataforma tem a *feature* completa ± - Indica que a plataforma tem parte da *feature* Células destacadas a azul indicam que a *feature* pode ser feita ou mostrada aos visitantes. Não é necessário registo ou início de sessão. **FUNCIONALIDADES**		**GitHub**	**GitLab**	**OHWR**	**SourceForge**	**BitBucket**
Code Review	Inclusão do código revisto ao ramo principal ou mestre através da interface do utilizador	✓	✓	✓		✓
Pull Requests	Criar uma solicitação de pull (*merge request*) em suporte à interface do utilizador	✓	✓	✓		✓
	Mostrar o número de *forks* para cada repositório/projeto	✓	✓		✓	✓
	Aceitar ou recusar uma *pull request* através da interface do utilizador	✓	✓	✓		✓
	Inclusão de uma *pull request* para o ramo principal através da interface do utilizador	✓	✓	✓		✓
	Discutir pedidos *pull* na secção de pedidos *pull* – comentários	✓	✓	✓		✓
	Filtrar pedidos *pull* (abertos, fechados, *merged*)	✓	✓	✓		✓
	Atribuir pedidos *pull* a colaboradores	✓	✓	✓		✓
Issue/bug Tracking	Usa o rastreador de problemas (*bug*, solicitação de recurso)	✓	✓		✓	✓
	Usa o rastreador de erros					
	Alterar o estado dos problemas/*bugs* por um proprietário ou colaboradores	✓	✓		✓	✓
	Filtrar problemas pelos seus tipos (*bug*, pedido de funcionalidade)	✓	✓		✓	✓
	Filtrar problemas/*bugs* por outros critérios (fechado, aberto)	✓	✓	✓	✓	✓
	Discutir problemas/*bugs* na secção de problemas/*bugs* através de comentários	✓	✓	✓	✓	✓
	Mencionar os utilizadores num comentário/envolver o utilizador numa discussão	✓	✓	✓	✓	✓
	Priorizar problemas/*bugs*	✓	✓		✓	✓
	Classificar ou votar em problemas/*bugs*	✓	✓		✓	
	Atribuir questões específicas a colaboradores	✓	✓		✓	✓
	Solicitação de funcionalidades através do envio de formulários e arquivos de *patches*					

Tabela 9 - Funcionalidades detalhadas das plataformas *online* OS. Adaptado de Alamer & Alyahya (2017)

✓ - Indica que a plataforma tem a *feature* completa ± - Indica que a plataforma tem parte da *feature* Células destacadas a azul indicam que a *feature* pode ser feita ou mostrada aos visitantes. Não é necessário registo ou início de sessão. **FUNCIONALIDADES**		**GitHub**	**GitLab**	**OHWR**	**SourceForge**	**BitBucket**
Issue/bug Tracking	Gerenciar dependências entre itens de trabalho (por exemplo, problemas, *merge requests*)	±	±			±
Outros	Gráfico das Contribuições e Atividades do Utilizador	✓	✓	✓		
	Fluxo de contribuições/atividades	✓	✓	✓	✓	✓
	Painel de controlo para a conta pessoal do *developer*	✓	✓	✓	✓	✓
	Wikis	✓	✓	✓	✓	✓
	Fóruns de discussão			✓	✓	

A partir da análise pormenorizada da Tabela 9 , podem-se extrair as seguintes conclusões acerca de algumas das plataformas de desenvolvimento Open Source:

- GitHub e GitLab sobressaem devido à sua vasta gama de funcionalidades, abrangendo desde a gestão de repositórios até aos recursos sociais e processos de revisão de código;
- No domínio da representação visual da atividade dos *developers*, somente as plataformas GitHub, GitLab e OHWR disponibilizam um gráfico de contribuições, o que pode ser indicativo da sua orientação para uma experiência do utilizador mais interativa e informativa;
- A plataforma OHWR parece adotar um critério mais rigoroso para a criação de novos projetos, indicando uma possível preferência por projetos mais especializados ou específicos;
- Plataformas como o GitHub e o SourceForge manifestam uma abordagem mais aberta, permitindo aos visitantes acederem a diversas funcionalidades sem a necessidade de registo ou autenticação;
- Recursos que promovem a interatividade e a socialização, como seguir outros *developers* ou avaliar projetos, são especialmente marcantes no GitHub, enquanto outras plataformas apresentam prioridades distintas;
- Todas as plataformas parecem possuir mecanismos sofisticados de rastreamento de problemas e revisão de código, sublinhando a sua adequação para projetos colaborativos de grande escala;
- Enquanto a inclusão de *wikis* é uma constante na maioria das plataformas, a presença de fóruns de discussão e listas de correio varia, o que pode impactar a comunicação interna em certos projetos;
- Uma característica comum a todas as plataformas é a permissão concedida aos proprietários de projetos de integrar colaboradores nos seus respetivos projetos *Open Source*;

- Uma constante transversal a todas as plataformas é a opção dada aos *developers* de subscreverem um projeto, o que lhes habilita a receber notificações relativas a todas as atualizações associadas ao mesmo;
- A revisão de código e a gestão de *pull requests* são fundamentais na maior parte das plataformas, evidenciando um compromisso com o desenvolvimento colaborativo e a manutenção da integridade do código;
- Em relação ao rastreamento de problemas e *bugs*, a vasta maioria das plataformas disponibiliza ferramentas robustas, favorecendo uma gestão eficiente de problemas e a implementação de um código de alta qualidade.

Estas observações refletem não só uma análise comparativa das funcionalidades disponíveis nas diferentes plataformas, mas também evidenciam tendências gerais no campo do desenvolvimento *Open Source* e as prioridades distintas de cada plataforma.

3.3. A Comunidade GitHub

O GitHub revolucionou o desenvolvimento de código aberto desde que se estabeleceu como a primeira plataforma a adotar o modelo de colaboração baseado em *pull*. Neste modelo, um contribuidor efetua um *fork* de um repositório com o intuito de fazer alterações de forma autónoma e, posteriormente, submete uma *pull request* ao repositório original para revisão (Xia et al., 2022).

Adicionalmente, o GitHub constitui-se como um serviço de alojamento de projetos fundamentados no *Git*. Sistemas de controlo de versão descentralizados, tal como o *Git*, possibilitam que os utilizadores efetuem *forks* de repositórios na sua totalidade, concedendo a cada utilizador permissões de escrita nos seus próprios *forks* locais. Esta caraterística simplifica a gestão da participação orientada pela comunidade. O GitHub é especialmente adequado para permitir o crescimento de comunidade, uma vez que expõe publicamente, em toda a plataforma, as identidades dos utilizadores, os componentes do projeto e as ações exercidas sobre os laços sociais da comunidade e contribui para o sucesso dos projetos (Padhye et al., 2014; Wagstrom et al., 2012).

Cada utilizador do GitHub assume a responsabilidade de gerir o seu tempo pessoal e as suas competências profissionais. Por essa razão, o processo de formação de equipas no GitHub é descentralizado e pode ser considerado como auto-organizado. Este processo ocorre de forma orgânica e descentralizada, com base nas necessidades do projeto e nas contribuições dos membros da comunidade. Os proprietários dos repositórios ou os principais colaboradores podem convidar outros para se juntarem à equipa, ou os interessados podem solicitar para contribuir. Em outras palavras, não há um mecanismo central que oriente a formação das equipas de código aberto (Jarczyk et al., 2014).

Nos últimos anos, o GitHub tem assistido a uma popularidade crescente e notável. De facto, após a sua inauguração em 2008, o número de projetos alojados e de utilizadores aumentou exponencialmente, passando de 135 mil projetos em 2009 para mais de 35 milhões de projetos em 2015, num intervalo de apenas 6 anos. Em 2022, o GitHub acomodou mais de 94 milhões de utilizadores e mais de 413 milhões de contribuições de código aberto foram registadas. Adicionalmente, 20.5 milhões de novos *developers* aderiram ao GitHub e foram contabilizadas mais

de 3.5 mil milhões de contribuições totais para todos os projetos no GitHub, incluindo *commits*, *pull requests*, discussões, *gists*, *push* e revisões de *pull requests* (Carroll, 2022; Gehring, 2022).

O GitHub tem vindo a incluir cada vez mais suporte para desenvolvimento de código aberto. Para além de uma interface baseada na *web* sobre o sistema de controlo de versão *git*, dispõe de um rastreador de problemas, mecanismos para gerir colaboradores do projeto, um fórum de discussão para cada repositório, facilidades para gerir *pull requests* ou até mesmo submeter novos PRs diretamente através da interface do GitHub, ferramentas para criar lançamentos de projetos, capacidade de criar e alojar *websites* de projetos, funcionalidades para planear e monitorizar projetos, suporte para análise de dependências desatualizadas e deteção de vulnerabilidades de segurança, bem como métricas e visualizações que fornecem informações sobre a evolução do projeto e da sua comunidade.

3.3.1. A Estrutura da Comunidade GitHub

Identificam-se dois tipos de perfis no GitHub: contas individuais e contas de entidades organizacionais. A principal diferença reside no facto de que as contas individuais pertencem a um único utilizador, enquanto as contas organizacionais são contas compartilhadas, onde grupos de pessoas podem colaborar em diversos projetos ao mesmo tempo. Estas últimas, destinadas a organizações, possuem mecanismos avançados para gerir o acesso dos membros aos dados e projetos da organização com segurança sofisticada e recursos administrativos (Zöller et al., 2020).

Um histórico de projeto típico inicia-se com um utilizador que desenvolveu e disponibilizou um software. Posteriormente, com o tempo, outros *developers* adotam o software e contribuem no seu desenvolvimento, mas o controlo sobre o repositório permanece com o *maintainer* original (Zöller et al., 2020).

Um repositório, frequentemente designado como “repo”, é o elemento fundamental do GitHub. Pode ser visualizado como uma pasta de projeto digital que contém todos os arquivos associados a um projeto, incluindo a sua documentação. Além disso, um repositório armazena o histórico completo de revisões de cada arquivo, permitindo monitorizar e revisitar versões anteriores de um projeto. Cada repositório, por padrão, possuiu um *wiki* para a documentação; um rastreador de problemas para a gestão de *bugs*, e um sistema para gerir *pull requests*. As *pull requests* são propostas de alterações submetidas por utilizadores que desejam contribuir para o projeto.

Ao criar um repositório, identifica-se um diretório específico para monitorização. O conteúdo desse diretório pode ser selecionado para a lista de ficheiros a monitorizar. Subsequentemente, quaisquer modificações aos ficheiros monitorizados, após serem validadas, são registadas como uma nova revisão, identificando de forma inequívoca as alterações em todos os ficheiros modificados (Perez-Riverol et al., 2016).

Para melhor compreensão da dinâmica intrínseca de um projeto no GitHub, a Figura 14 apresenta uma representação visual abrangente de todos os componentes e interações que constituem um projeto, dividida em diferentes *clusters*. Estes clusters evidenciam desde a estrutura básica da equipa e gestão do projeto, mas também detalha processos específicos, como a realização de um *commit* por um membro e a ação de *fork* de um repositório.

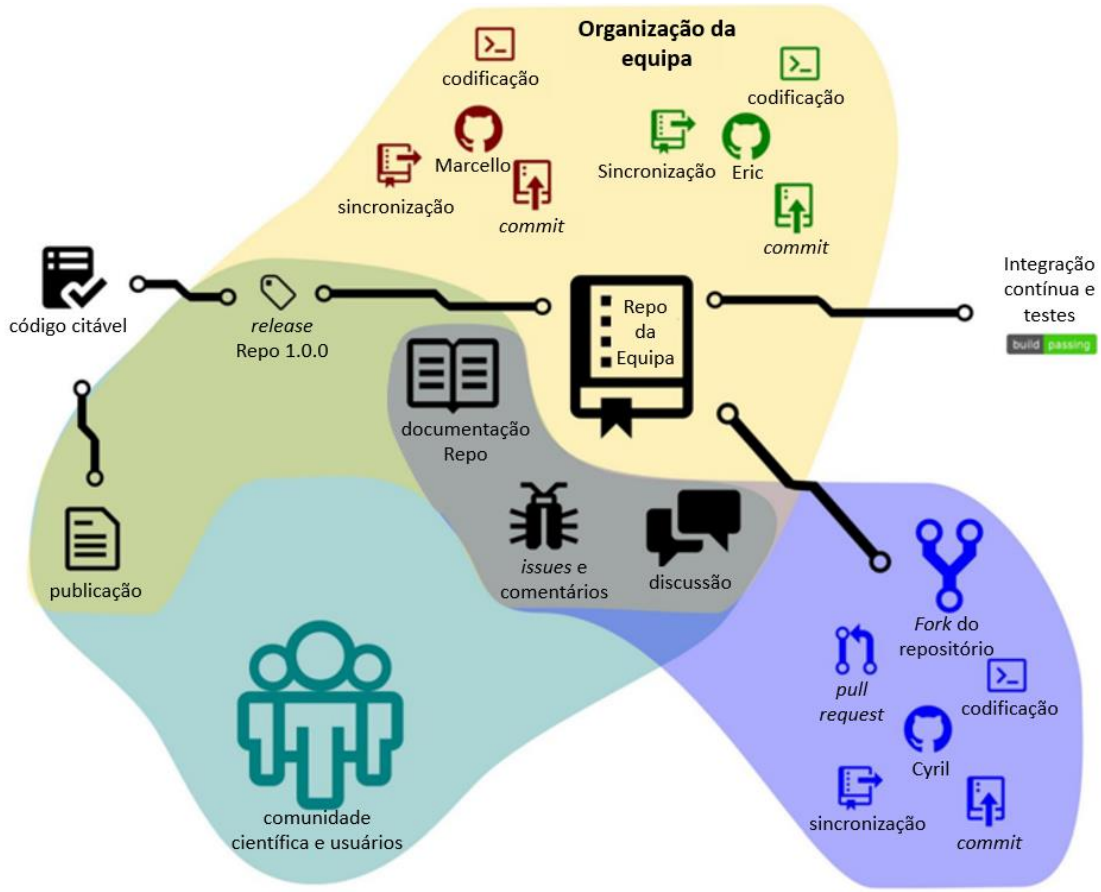


Figura 14 - Fluxo de Trabalho e Interações em Projetos GitHub. Adaptado de Perez-Riverol et al. (2016)

Através da Figura 14, é possível observar desde a organização inicial da equipa, passando pela integração contínua, testes, *releases* e até à interação com a comunidade científica e utilizadores. Esta representação destaca a dinâmica de colaboração e serve como um guia visual para entender as várias etapas e interações que ocorrem dentro de um projeto na plataforma.

Além disso, a figura demonstra como o GitHub, enquanto plataforma, é mais do que apenas um local para armazenar código. É um ecossistema interconectado onde a colaboração, a revisão e a partilha de ideias acontecem de forma contínua. Esta imagem enfatiza a importância de compreender o fluxo de trabalho e as interações no GitHub para maximizar a eficiência e a produtividade no desenvolvimento de software.

Os utilizadores são o núcleo do GitHub, tal como em qualquer outra plataforma. Cada utilizador tem um perfil que lista os seus projetos e atividades na plataforma. Este perfil pode ser complementado com detalhes pessoais, tais como nome, endereço de email, imagem e página *web*. Para se manter informado sobre as atividades de outros utilizadores, é possível seguir outras contas.

Os projetos públicos no GitHub são acessíveis para todos. No entanto, a permissão de alterar diretamente o conteúdo de um repositório, conhecida como permissão de gravação, deve ser atribuída de forma explícita. Enquanto proprietário de um repositório, é possível conceder esse direito a outros utilizadores do GitHub. A colaboração pode ser facilitada simplesmente ao adicionar um colaborador de confiança, concedendo-lhe assim direitos de gravação.

O desenvolvimento em grandes projetos geralmente é conduzido por equipas de pessoas dentro de uma organização mais ampla. As organizações do GitHub representam uma ferramenta valiosa para gerir permissões de acesso baseadas em equipa, para projetos individuais de instituições, laboratórios de pesquisa e grandes iniciativas de código aberto que requerem vários proprietários e administradores. Os proprietários têm acesso total à organização, situando-se no nível mais alto da hierarquia de permissões. Relativamente à gestão organizacional, detêm a capacidade de convidar e excluir membros, criar e excluir equipas, bem criar e excluir repositórios. Adicionalmente, têm autoridade para gerir os níveis de permissão atribuídos a todos os membros e repositórios, assim como podem editar as definições da organização.

Assumir o papel de membro é, geralmente, a função padrão aquando da integração de um novo indivíduo na organização. No âmbito das suas competências mínimas, um membro tem a capacidade de estabelecer uma nova equipa e integrar membros e repositórios já existentes à mesma. Um membro também pode ser promovido a *maintainer* da equipa para uma equipa específica. Com esse nível de acesso, têm a autoridade para adicionar e excluir membros da equipa. Este papel pode ser percecionado como uma responsabilidade adicional, intrínseca à posição de membro. É de notar que qualquer indivíduo que cria uma equipa, é automaticamente atribuído com a função de *maintainer* da equipa para essa equipa em questão.

Existem quatro categorias de permissões atribuíveis a um repositório: Proprietário, Administrador, Gravação e Leitura. A categoria Proprietário é o nível de permissão universal superior reservado aos proprietários da organização. Com o acesso Administrador, o membro adquire privilégios de proprietário, porém apenas para o repositório específico. Esta autorização permite, entre outras ações, a submissão, eliminação, inclusão ou exclusão de equipas, modificação do grau de permissão de uma equipa e adição de colaboradores externos. Esta permissão assemelha-se à criação de um repositório no espaço pessoal do utilizador. A permissão Gravação habilita o utilizador a submeter informações ao repositório, enquanto a Leitura lhe confere a permissão de realizar um *fork* e extrair dados.

O perfil de um membro do GitHub fornece informações importantes acerca das suas contribuições e o seu envolvimento na comunidade de Código Aberto. Proporciona uma visão geral abrangente da sua atividade, incluindo repositórios, contribuições, seguidores e organizações às quais estão associados. A análise aprofundada do perfil de um membro é crucial para obter uma compreensão mais profunda da sua experiência, interesses e áreas de foco. Ao explorar o perfil de um membro do GitHub, é possível identificar os projetos e linguagens de programação que domina, facilitando a identificação de potenciais colaboradores e contribuidores que partilhem interesses e metas similares. A avaliação da atividade e as contribuições de um membro na plataforma permite inferir sobre a sua influência, confiabilidade e grau de habilidades, fatores preponderantes quando são consideradas colaborações num projeto ou para avaliar a qualidade do seu trabalho.

Adicionalmente, o perfil de um membro proporciona dados sobre o total de *issues* e *pull requests* abertos. Tais métricas elucidam sobre a interação do membro com a comunidade, incluindo a sua participação em discussões, resolução de problemas e revisão de código. Estas interações demonstram a capacidade do membro no que diz respeito ao trabalho em equipa, assim como a capacidade de fornecer feedback construtivo e contribuir para a melhoria contínua dos projetos.

Para além disso, um perfil do GitHub funciona como uma ferramenta de rede, permitindo estabelecer conexões dentro da comunidade *Open Source*. A ação de seguir um membro e

envolver-se nas suas atividades promove a criação de ligações, a participação em debates e troca de conhecimentos. Estas ligações promovem um ambiente de colaboração, incentivando a aprendizagem partilhada e a troca de ideias inovadoras.

A vantagem de incorporar este tipo de ferramenta de perfil no GitHub reside na capacidade de fornecer uma plataforma centralizada para monitorizar e reconhecer as contribuições dos membros da comunidade *Open Source*. Esta visibilidade aumenta a transparência e a colaboração dentro do ecossistema, permitindo que outros membros identifiquem e se conectem com pessoas dotadas e experientes. A visualização da atividade de um membro serve ainda como meio para promover o reconhecimento e a valorização do empenho e das realizações dos indivíduos na comunidade.

A identificação dos membros de uma equipa pode ser realizada através da interface do GitHub. A plataforma define e apresenta o seu conjunto próprio de funções de equipa. A função de um *developer* individual numa equipa é visível sempre que o *developer* participa de uma discussão no GitHub, seja em questões ou *pull requests*. Contudo, os developers do GitHub têm a opção de tornar o seu estatuto de associação como privado, tornando-o visível apenas para outros membros do projeto (Middleton et al., 2018b).

Na Figura 15, destaca-se a organização e colaboração dentro de um projeto, evidenciando os diferentes papéis desempenhados pelos membros da equipa. É possível identificar o nome da equipa, o nome do *developer* e a respetiva função na equipa, elementos fundamentais para compreender a dinâmica e estrutura de colaboração neste ambiente.

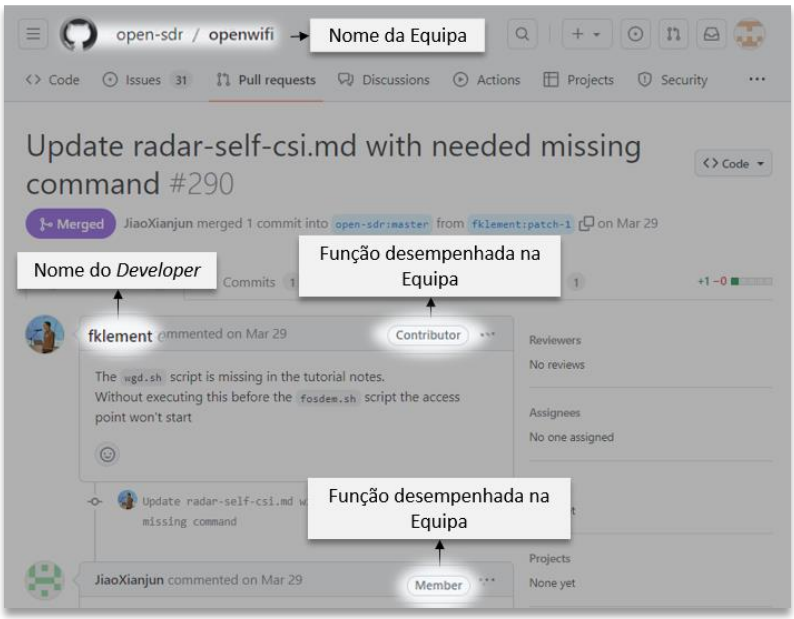


Figura 15 - Estrutura de Colaboração no GitHub: Papéis e Participação dos Membros da Equipa num Projeto.

A maioria dos projetos de código aberto apresentam uma estrutura definida, e os projetos de maior renome têm a estrutura e a governança do projeto claramente descritas no *website* do projeto ou na respetiva documentação. Embora a estrutura possa variar muito entre os diferentes projetos, há funções em comum:

- **Líde**r: No mínimo, deve existir alguém responsável pela tomada de decisões finais sobre funcionalidades, lançamentos e outras atividades. Em determinados contextos, esta função é desempenhada por um único indivíduo. Noutros projetos, pode haver um ou mais responsáveis por distintos aspetos de um projeto.
- ***Maintainers*:** A maioria dos líderes delega algumas das decisões a membros responsáveis pela manutenção de segmentos específicos do projeto. Em grandes projetos, esses *maintainers* podem, por sua vez, delegar a indivíduos responsáveis por subcomponentes específicos.
- ***Committers*:** Alguns projetos contam com grupos de indivíduos que contribuíram para o projeto e são considerados confiáveis e responsáveis o suficiente para serem autorizados a comprometer-se diretamente com todas ou algumas partes do projeto, sem a necessidade de submissão a um *maintainer* para revisão. Contudo, as contribuições dos *committers* permanecem passíveis de revisão pelos *maintainers* ou líderes do projeto, e podem ser revertidas caso suscitem alguma preocupação.
- **Contribuidores:** Muitos indivíduos contribuem para projetos de código aberto, seja através de código, documentação e outras melhorias. Tais contribuições, em regra, estão sujeitas a uma revisão de um *committer* e/ou *maintainer* experiente antes da sua efetiva incorporação.
- **Utilizadores:** Este grupo, de importância incontestável para a expansão e maturação de um projeto de código aberto, confere propósito e ajudam na evolução de um projeto. Estes membros da comunidade têm a capacidade de oferecer feedback sobre funcionalidades, reportar *bugs* e contribuir de diversas outras maneiras.

3.3.2. Ferramentas e Recursos do GitHub

A plataforma GitHub propõe duas arquiteturas de colaboração sociotécnica. A primeira assemelha-se aos padrões tradicionais de desenvolvimento de código aberto, onde apenas indivíduos confiáveis para se comprometer com um projeto, recebem acesso direto para adicionar código ao repositório central. Em contraste, a segunda arquitetura, tira proveito dos padrões de desenvolvimento distribuídos habilitados pelo *git* e pela infraestrutura fornecida pelo GitHub. Nesta, os *developers* que desejam contribuir com o código realizam *fork* do repositório original, implementam as suas alterações e, posteriormente, sugerem essas modificações ao projeto original através de *pull requests*.

O GitHub também disponibiliza uma API REST e GraphQL para consultar e recuperar dados do GitHub ou para integrar repositórios GitHub com ferramentas externas. No final de 2022, o GitHub introduziu uma variedade de novos recursos, incluindo:

- github.dev, um editor de código baseado na *web* que opera integralmente no navegador, permitindo navegar, editar e confirmar alterações de código diretamente no navegador;
- GitHub CodeSpaces, um ambiente de desenvolvimento mais completo e alojado na *cloud*;
- Pacotes GitHub, que permitem criar, publicar, visualizar e instalar novos pacotes diretamente do repositório de código;
- GitHub CoPilot, uma ferramenta baseada em Inteligência Artificial que proporciona sugestões automáticas de código;
- GitHub Actions, uma ferramenta de automação de fluxo de trabalho totalmente integrada ao GitHub.

O GitHub promove a criação de repositórios públicos, permitindo a qualquer utilizador realizar um *fork* e adaptar o código conforme as suas necessidades ou propor alterações via *pull requests*. Os forks são feitos por dois motivos principais: primeiro, para usar o código de alguma forma variada; e segundo, como um precursor para contribuir de volta ao projeto original através de uma solicitação pull, que pode ser mesclada por aqueles com acesso. Todas essas atividades são publicadas num fluxo aberto e visível no GitHub e o conteúdo e a discussão associados a *issues*, *commits* e *pull requests* também são públicos (Blincoe et al., 2016; McDonald & Goggins, 2013).

O termo *fork* refere-se a uma cópia pessoal de um repositório de outro usuário que reside na conta do *forker*. Esta funcionalidade permite que os *developers* experimentem e modifiquem um projeto sem afetar o repositório original. Existem três categorias distintas de *forks* no GitHub:

- ***Forks* de Contribuição:** Destinados a terminar com *pull requests* para o projeto original, visando integrar as mudanças.
- ***Forks* desenvolvidos independentemente:** *Forks* que não enviam *pull requests* para o projeto original.
- ***Forks* inativos:** Não enviam nem recebem *pull requests* nem possuem *commits* internos.

As duas últimas categorias são as mais comuns, já que a maioria dos *forks* não recupera novas atualizações dos projetos originais. Em particular, os *forks* são usados principalmente para corrigir *bugs* e adicionar novos recursos e, com menos frequência, para adicionar documentação. Um *fork* é considerado positivo por vários motivos, como preservar projetos abandonados, melhorar a qualidade de um projeto, dar aos developers controlo sobre a base de código, bem como enviar solicitações pull e fazer contribuições para os repositórios originais.

A contribuição para projetos no GitHub pode ser realizada de duas maneiras principais: diretamente, através de *push commits*, ou indiretamente, através de *pull request*. Os Enquanto os *push commits* são realizados exclusivamente por *developers* com permissões de gravação no repositório, enquanto as *pull requests* podem ser enviadas por qualquer membro da comunidade. As *pull requests* são propostas de alterações que, após revisão, podem ser integradas ao projeto original (Bleiel, 2016).

No ecossistema do GitHub, os "*bots* de desenvolvimento" desempenham um papel fundamental. Estes *bots*, vistos como desenvolvedores artificiais, são especializados em atividades específicas, como assistência CI/CD, gestão de problemas, revisão de código, entre outras. Na prática, interagem com a plataforma de forma semelhante a um *developer* humano, possuindo contas no GitHub, confirmando código e participando ativamente em discussões. Um exemplo proeminente é o Dependabot, que se tornou uma ferramenta essencial para manter as dependências de projetos atualizadas. Estes *bots*, em alguns casos, têm a capacidade de operar com total autonomia, tomando decisões sobre a integração de alterações em *pull requests*.

Além dos bots, o GitHub introduziu o "GitHub Actions", um serviço de automação lançado em 2019. Este serviço permite aos *maintainers* definir fluxos de trabalho automatizados para diversas atividades, desde revisões de código até monitorização de vulnerabilidades. Em pouco tempo, o GitHub Actions estabeleceu-se como o serviço de CI/CD mais dominante na plataforma (Wessel et al., 2023).

O GitHub, como plataforma de desenvolvimento colaborativo, oferece uma série de mecanismos que promovem a transparência, a comunicação e a colaboração entre os *developers*. Um desses

mecanismos é o sistema de acompanhamento, que permite aos utilizadores receber notificações sobre as atividades dos outros, seja sobre o que estão a trabalhar ou com quem estão a interagir. Esta funcionalidade, além de promover a sensibilização, é uma ferramenta valiosa para descobrir novas tendências, projetos e oportunidades de aprendizagem e socialização.

Todas as atividades, desde alterações a discussões, são visíveis a todos os utilizadores. Esta visibilidade tem implicações diretas na popularidade e atividade de um projeto. Por exemplo, projetos com um elevado número de *forks* tendem a ser mais ativos, embora o número de *forks* não esteja necessariamente correlacionado com o número de *pull requests* aceites. Transparência também significa que os utilizadores podem ver quais são as intenções dos outros e podem discutir as mudanças antes que aconteçam.

A transparência fornecida pelo GitHub é uma ferramenta valiosa para compreender a evolução do ecossistema ao longo do tempo. Em particular, a quantidade de *commits*, *branches*, *forks*, bem como *pull requests* abertos e fechados e problemas, são feitos para refletir a atividade e o tamanho do projeto e como o projeto é gerido. Por exemplo, um elevado número de pull requests abertos e ignorados podem sinalizar que as contribuições externas não são levadas em consideração, uma vez que cada *pull request* aberto indica uma oferta de código que está a ser ignorada em vez de aceite, rejeitada ou comentada (Cosentino et al., 2017).

O GitHub não só permite a monitorização de problemas, *pull requests*, mas também promove a colaboração através das *pull requests*. Estas são o principal meio de contribuição para um projeto, exigindo que contribuidor realize um *fork* primeiro do projeto e, uma vez concluídas as alterações, solicite a integração dessas alterações no projeto original. O GitHub oferece também suporte a recursos sociais como seguidores e visualizadores (Cosentino et al., 2017).

O GitHub também valoriza a individualidade e a identidade dos seus utilizadores. Permite que os *developers* criem perfis detalhados, incluindo informações como nome, endereço de email, organização, localização e página web. Estes perfis têm uma dupla função: social e técnica. Socialmente, promovem a interação entre os utilizadores, permitindo que sigam uns aos outros e acompanhem projetos de interesse. Tecnicamente, servem como um portfólio das contribuições e atividades do *developer* na plataforma.

A comunicação é outro pilar fundamental do GitHub. Os *developers* podem interagir através de comentários em *commits*, *issues* ou *pull requests*. Estes problemas, em particular, são uma ferramenta versátil para rastrear bugs, tarefas, solicitações de recursos e melhorias. Cada problema é dinâmico, com uma formatação clara e espaço para feedback. Como exemplo prático, um *issue* no repositório permitiu a interação de 17 *developers* e atraiu 40 comentários. Na Figura 16, é apresentada a interação mencionada, destacando a colaboração e o envolvimento dos *developers* no contexto do problema discutido.

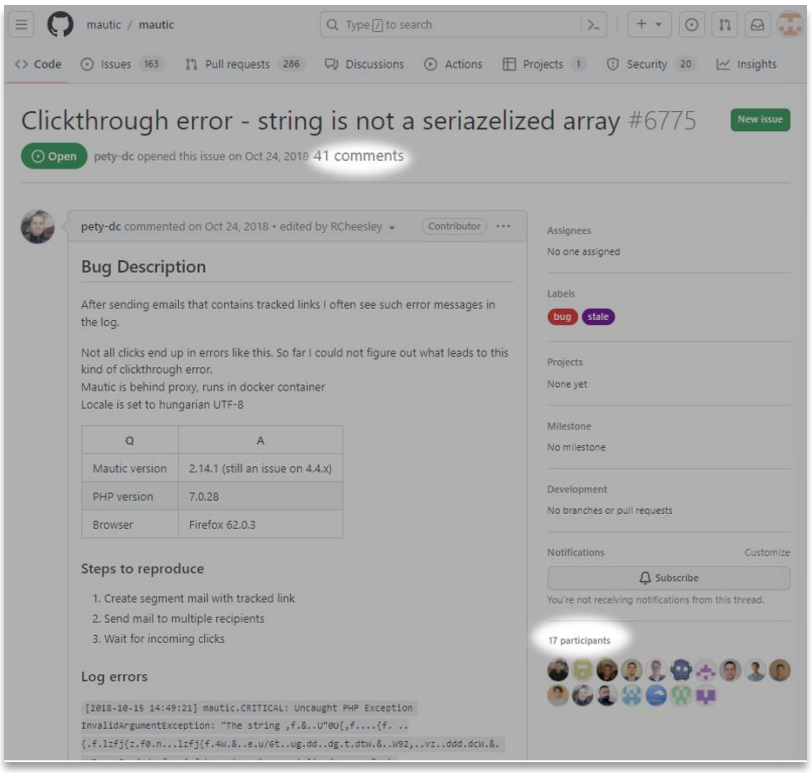


Figura 16 - Interação de *developers* num problema de um repositório no GitHub.

Os problemas do GitHub são uma ótima maneira de acompanhar *bugs*, tarefas, solicitações de recursos e aprimoramentos. Embora os rastreadores de problemas clássicos sejam usados principalmente como rastreadores de *bugs*, os rastreadores de problemas do GitHub seguem uma filosofia diferente: cada rastreador tem sua própria seção em cada repositório e pode ser usado para rastrear *bugs*, novas ideias e aprimoramentos usando um poderoso sistema de marcação. O principal objetivo dos problemas no GitHub é promover a colaboração e fornecer contexto usando referências cruzadas.

A plataforma também introduziu conceitos como *Watch* e *Star*, que permitem aos utilizadores expressar interesse, avaliar e contribuir para projetos, respetivamente. No GitHub, o ato de *Watch* é uma forma dos utilizadores se manterem atualizados sobre as alterações num repositório. Permite que os utilizadores recebam atualizações do estado de um projeto, como por exemplo quando o código foi confirmado. Isso é considerado um ato passivo, pois significa envolvimento na comunidade, mas nenhuma ação direta para contribuir (Peterson, 2013).

Enquanto marcar um projeto com uma estrela é uma forma dos utilizadores indicarem que o projeto é interessante para eles, para indicar se o projeto vale a pena ser usado e, portanto, são um sinal da saúde dos projetos de código aberto (Pipinellis, 2015).

O botão *Watch* permite aos utilizadores gerir o nível de inscrição num repositório, recebendo notificações por email sempre que uma ação ocorre num repositório que o usuário segue e, ao mesmo tempo, lista-as na área de notificações, onde pode marcá-las posteriormente como lidas. Existem quatro níveis de assinatura, variando desde notificações mínimas até notificações completas, dependendo do envolvimento e preferência do utilizador. A interface do GitHub destaca essas métricas, assim como o número de *forks* e o número de estrelas, que são visíveis na página principal do projeto, conforme é mostrado na Figura 17.

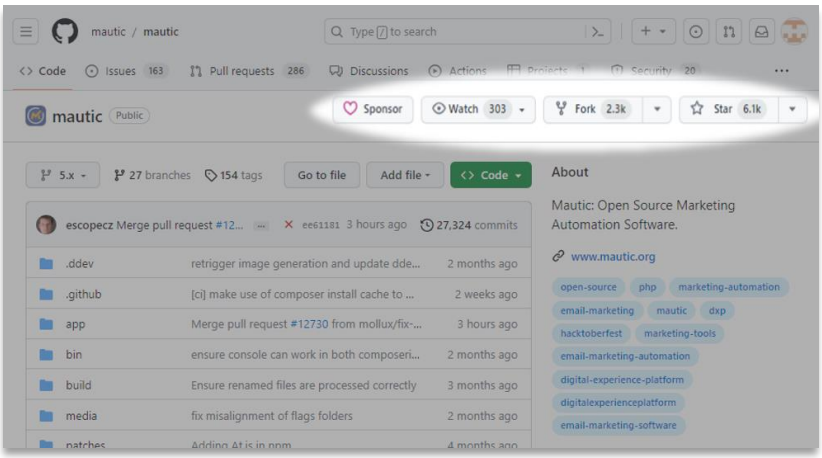


Figura 17 - Interface do GitHub com as opções de inscrição do Botão *Watch* e métricas do projeto: *Stars* e *Forks*.

A plataforma fornece recursos que ajudam a identificar os conhecimentos dos utilizadores, os seus interesses e a sua relevância na comunidade, dando assim suporte a impressões mais precisas dos colaboradores. A experiência de um membro é frequentemente avaliada pela diversidade e profundidade dos projetos em que participa e pela linguagem de programação que utiliza.

Apesar das métricas robustas oferecidas pelo GitHub, é importante notar que a plataforma não fornece dados objetivos sobre a atividade de manutenção do projeto. No entanto, os utilizadores podem inferir a saúde e a manutenção de um projeto com base em métricas como o número de *commits*, estrelas, *forks* e *watchers*. Estas métricas podem ser indicativas cruciais para determinar a viabilidade e a relevância de um projeto. No entanto, com base nos valores destas métricas, pode-se julgar se um projeto está ou não em manutenção, e, por conseguinte, se vale a pena utilizá-lo (Coelho et al., 2020).

A Figura 18 ilustra a estrutura abrangente da plataforma GitHub, destacando as métricas essenciais disponibilizadas aos utilizadores e a organização hierárquica dos seus componentes.

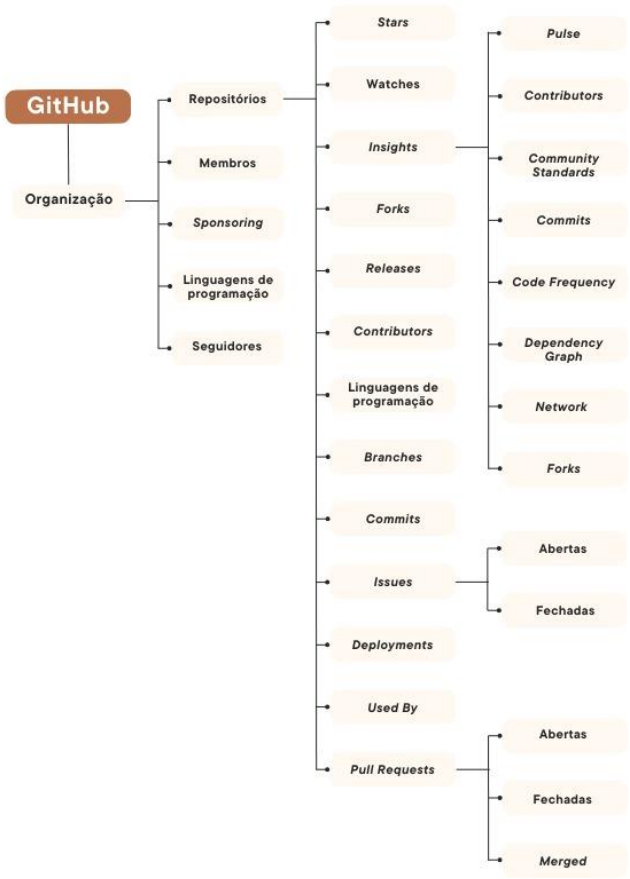


Figura 18 - Estrutura e Métricas Fornecidas pela Plataforma GitHub.

O GitHub foi projetado para facilitar a colaboração, a monitorização e a gestão de projetos no contexto do código aberto. A plataforma proporciona uma visão detalhada das atividades e contribuições de organizações ou membros individuais, bem como dos repositórios que as compõem. Tal estruturação é primordial para uma análise criteriosa e uma compreensão do envolvimento e impacto de um conjunto de membros ou de uma entidade específica na vasta comunidade de código aberto.

Dentro da plataforma, uma organização representa uma estratégia eficaz de agrupar diversos repositórios e colaboradores sob um designação unificada. Ao proceder-se à análise de uma organização no GitHub, torna-se possível observar:

- **Seguidores:** Reflete o número de seguidores que demonstram interesse na organização, seguindo-a para se manterem atualizados sobre as suas atividades;
- **Membros:** Indica o número de membros integrantes da organização;
- ***Sponsoring*:** O número de entidades ou projetos patrocinados pela organização;
- **Linguagens de programação:** Mostra as linguagens de programação preponderantes nos repositórios da organização;
- **Repositórios:** Reflete o número total de repositórios que a organização possui.

A fim de proporcionar uma compreensão prática e visual sobre a estrutura e métricas fundamentais de uma Organização no GitHub, a Figura 19 exemplifica uma organização concreta na plataforma, realçando os elementos anteriormente referidos.

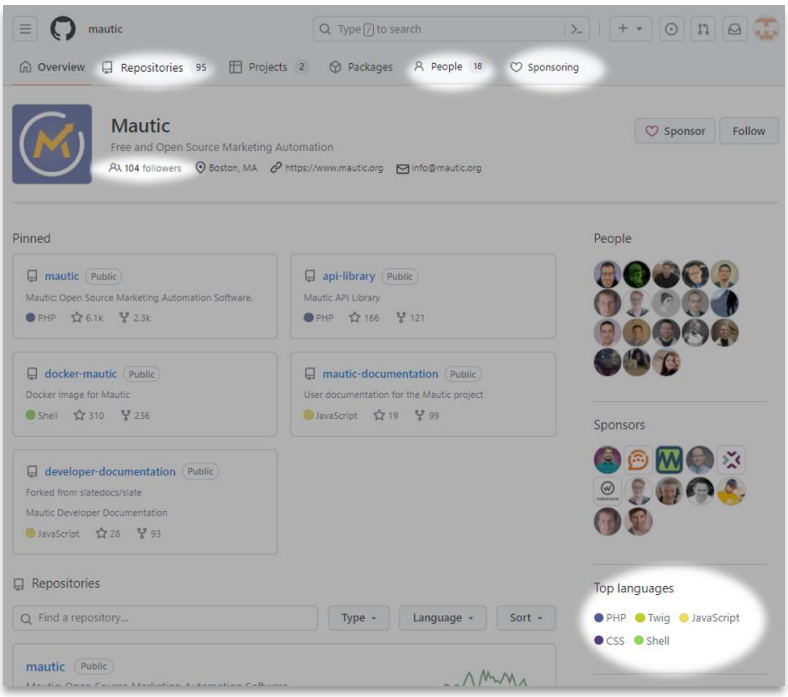


Figura 19 - Visão Geral de uma Organização no GitHub.

Ao explorar-se no GitHub, é imperativo considerar, além das organizações, a atividade e evolução de um projeto específico, representado por um repositório. Na interface principal de um repositório, é possível observar-se informações vitais sobre a atividade e reconhecimento do projeto:

- ***Stars:*** Indica o número de estrelas atribuídas ao repositório, evidenciando a apreciação e valorização do projeto pela comunidade;
- ***Watch:*** Indica o número de utilizadores que "observam" (*watching*) o repositório, demonstrando um interesse contínuo nas atualizações do projeto;
- ***Forks:*** Apresenta o número de *forks*, que reflete quantas vezes outros utilizadores replicaram o repositório, evidenciando a sua popularidade e adaptabilidade;
- ***Releases:*** Apresenta o número de *releases*, indicando as versões lançadas do projeto, permitindo perceber a sua evolução ao longo do tempo;
- ***Contributors:*** Quantifica o número de colaboradores, que contribuíram ou contribuem para o projeto, demonstrando a colaboração e envolvimento da comunidade;
- **Linguagens de Programação:** Destaca as linguagens de programação mais utilizadas no projeto e a respetiva percentagem de uso, permitindo perceber as tecnologias subjacentes;
- ***Branches:*** Enumera o número de *branches*, sendo estas as ramificações criadas a partir do projeto principal, incluindo tanto os *branches ativos* e *inativos*, indicando a diversidade e variantes do código;
- ***Commits:*** Evidencia o número total de *commits* realizados, refletindo a quantidade de alterações efetuadas no código e a atividade, ao longo da existência do projeto;
- ***Deployments:*** Na página principal do repositório, esta métrica indica as implementações ativas do projeto. Ao aceder à secção específica de "Deployments", é possível observar todas as implementações, ativas e inativas, permitindo uma compreensão mais detalhada sobre em que ambientes e configurações o projeto foi ou está a ser utilizado;

- ***Used By:*** Esta métrica, visível na página principal do repositório, indica o número de projetos que utilizam o repositório em questão como dependência. Ao explorar a secção "Used by", é possível visualizar uma lista detalhada de todos os repositórios e *packages* dependentes, proporcionando uma perspetiva sobre a influência e a relevância do projeto na comunidade de desenvolvimento.

Para uma visualização direta e contextualizada das métrica discutidas, apresenta-se nas Figura 20 e Figura 21, a página principal de um repositório no GitHub, onde é possível visualizar diretamente as métricas cruciais que refletem a atividade e o desenvolvimento de um projeto.

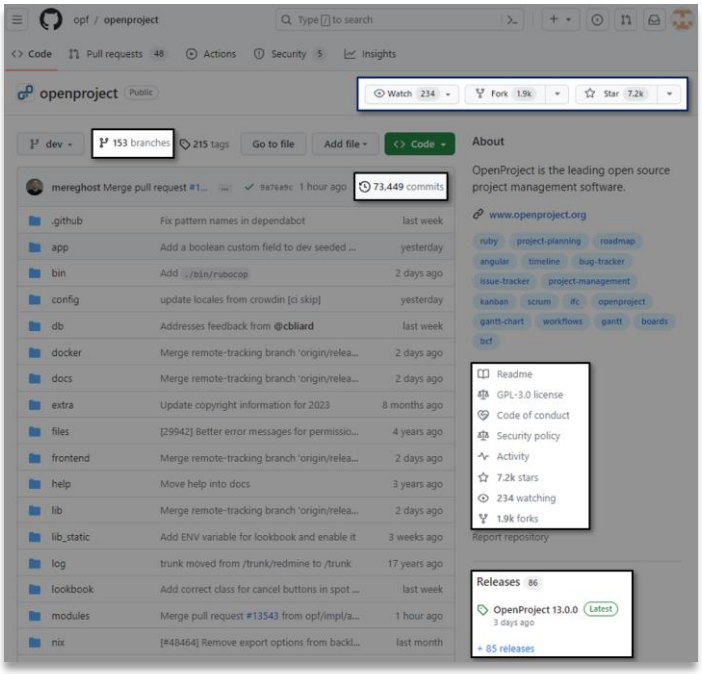


Figura 20 - Página Principal de um Repositório no GitHub: Métricas de Atividade e Reconhecimento do Projeto.

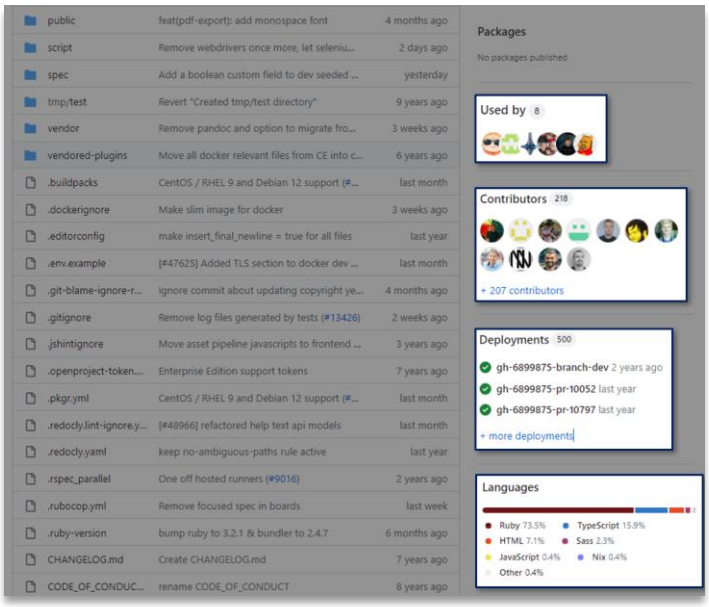


Figura 21 - Página Principal de um Repositório no GitHub: Métricas de Atividade e Reconhecimento do Projeto.

Dentro da estrutura de um repositório, duas páginas emergem como fundamentais para a compreensão da dinâmica e evolução de projetos: a página de *Issues* e a página de *Pull Requests*. Na página de *Issues*, é possível discernir o número total de *issues*, categorizando-as entre as que permanecem abertas e as já resolvidas, denominadas por fechadas. A segunda, por sua vez, oferece uma perspetiva sobre as propostas de alterações no código, revelando o número total de *pull requests*, diferenciando-os entre os estados de abertos e fechados, e os que foram efetivamente integrados (*merged*) no projeto é só filtrar como *"is:pr is:merged"*. Para ilustrar de forma clara estas seções, apresentam-se na Figura 22, Figura 23 e Figura 24, as visualizações destas páginas, evidenciando as métricas e funcionalidades mencionadas.

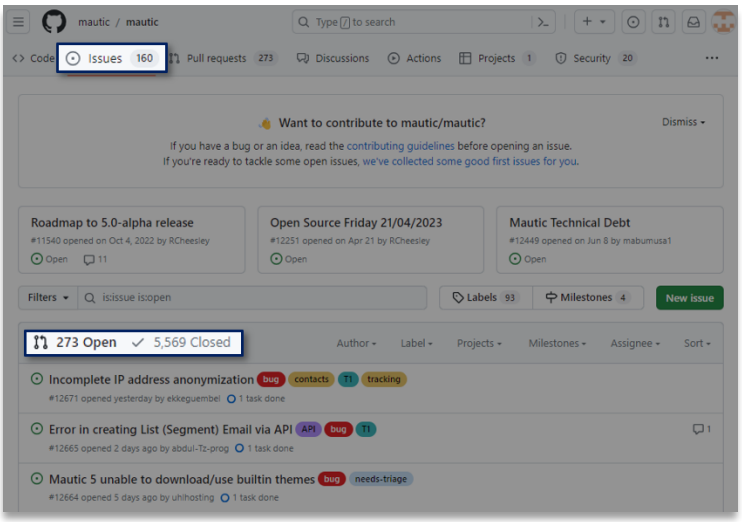


Figura 22 - Página de *Issues* de um Repositório no GitHub: Visão Geral dos *Issues* Abertos e Fechados.

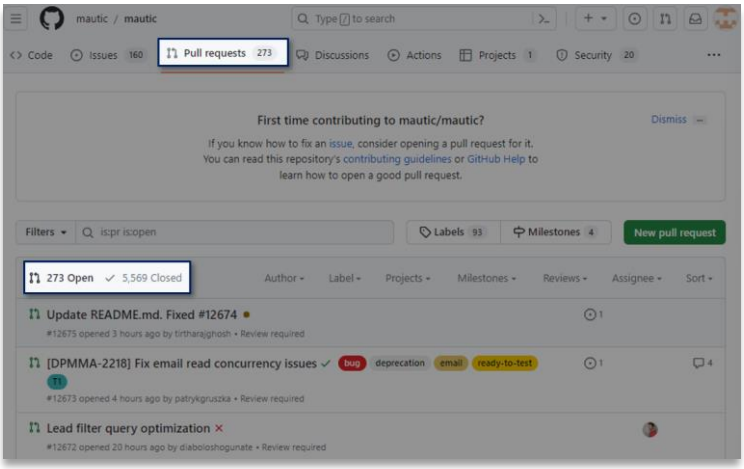


Figura 23 - Página de *Pull Requests* de um Repositório no GitHub: Visão Geral dos *Pull Requests* Abertos e Fechados.

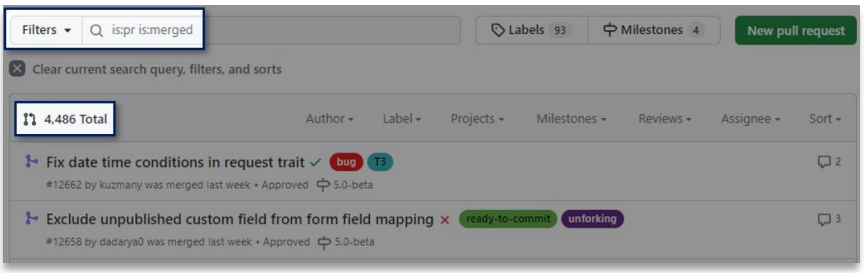


Figura 24 - Filtragem de *Pull Requests* no GitHub: Visualização dos *Merged Pull Requests.*

Aprofundando a secção *Insights* do repositório, é evidente que esta secção foi concebida para proporcionar uma análise profunda da atividade e contribuições ao projeto. Neste contexto, é possível distinguir as seguinte subsecções:

- *Pulse;*
- *Contributors;*
- *Community Standards;*
- *Commits;*
- *Code Frequency;*
- *Dependency Graph;*
- *Network;*
- *Forks.*

O GitHub, através da interface *Pulse*, oferece uma visão abrangente da atividade do repositório. Inclui uma lista de *pull requests*, tanto os que estão abertos quanto os que foram *merged*, bem como *issues* abertas e fechadas. Adicionalmente, apresenta um gráfico que destaca a atividade de *commit* dos 15 principais utilizadores que contribuíram para o *branch* padrão do projeto no período selecionado. É relevante mencionar que coautores de *comits* também são incluídos, caso se encontrem entre os 15 principais utilizadores. A Figura 25 exemplifica a interface *Pulse* no GitHub, permitindo demonstrar a sua funcionalidade e a sua análise.

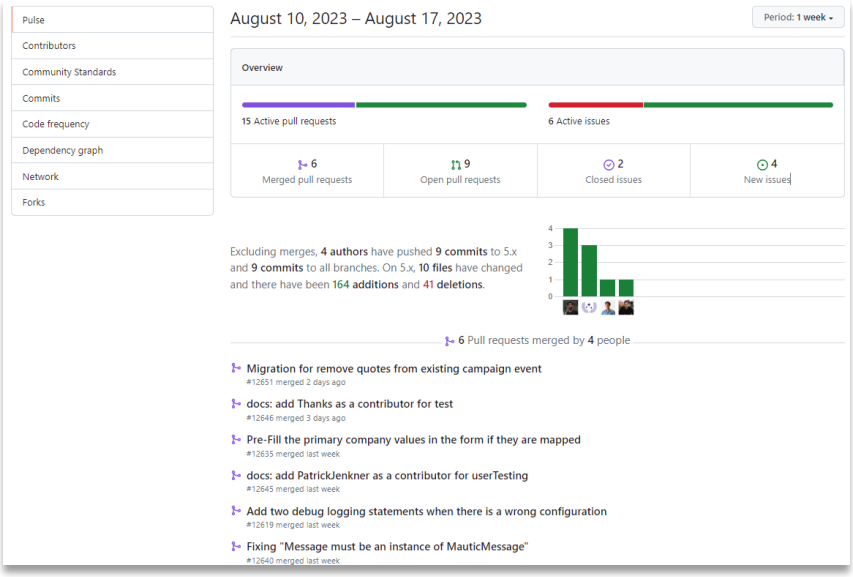


Figura 25 - Visão Geral da Atividade do Repositório através do *Pulse*.

A página dos *Contributors* é dedicada a evidenciar os 100 principais contribuidores do repositórios. Importa salientar que os *merged commits* e *commits* vazios não são contabilizados para o gráfico exibido nesta página. Além disso, a página proporciona uma visão temporal dos *commits* desde a conceção do projeto até ao presente, bem como um gráfico de *commits* dos 100 principais contribuidores. Há ainda a possibilidade de visualizar os *commits* num intervalo de tempo específico, mediante a seleção do período desejado. De forma a ilustrar de maneira clara e objetiva as características e funcionalidades descritas acerca da secção *Contributors*, apresenta-se a Figura 26, destacando a visualização dos contribuidores mais ativos, assim como a trajetória dos *commits* ao longo do tempo.

Figura 26 - Secção *Contributors* no GitHub: Representação Gráfica dos Principais Contribuidores e Evolução Temporal dos *Commits*.

Além da comunicação e transparência, o GitHub enfatiza a importância da documentação e licenciamento. Todo os repositórios devem, idealmente, conter três arquivos essenciais: LICENSE, CODE_OF_CONDUCT, README, CONTRIBUTING, servindo como um indicador da adesão às melhores práticas. Estes arquivos garantem que os direitos, descrições, instruções e créditos do projeto sejam claramente comunicados à comunidade (Perez-Riverol et al., 2016).

A importância atribuída a uma documentação clara e abrangente manifesta-se na *checklist* de perfil da comunidade disponibilizada pelo GitHub para cada repositório. Além disso, o GitHub estabelece diretrizes específicas para os *maintainers*, incentivando-os a adotar padrões recomendados que fortalecem a comunidade. A Figura 27 ilustra um exemplo desta *checklist*, extraída da subsecção *Community Standards* na secção *Insights* de um determinado repositório.

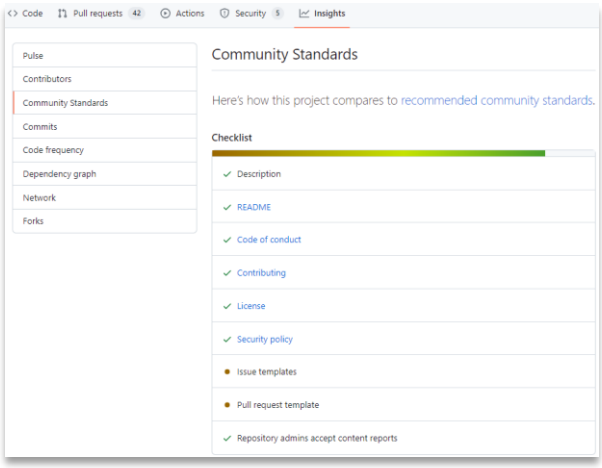


Figura 27 - *Checklist* de Perfil da Comunidade de um projeto no GitHub.

Na subsecção denominada *Commits*, observar-se o registo de todos os *commits* realizados no repositório, correspondentes ao último ano de atividade desde o momento em que o repositório é consultado, excluindo *merged commits*. Esta métrica é visualizada através de um gráfico de barras, onde cada barra representa o número de *commits* efetuados numa semana específica, ao longo do ano. Ao selecionar uma semana em particular, é possível aprofundar ainda mais a análise, visualizando um gráfico complementar que detalha o número de *commits* efetuados em cada dia da semana em questão. A Figura 28 proporciona uma representação visual, ilustrando tanto a distribuição anual dos *commits* como a distribuição destes ao longo de uma semana específica, evidenciando assim a dinâmica e frequência das contribuições por *commits* no repositório em análise.

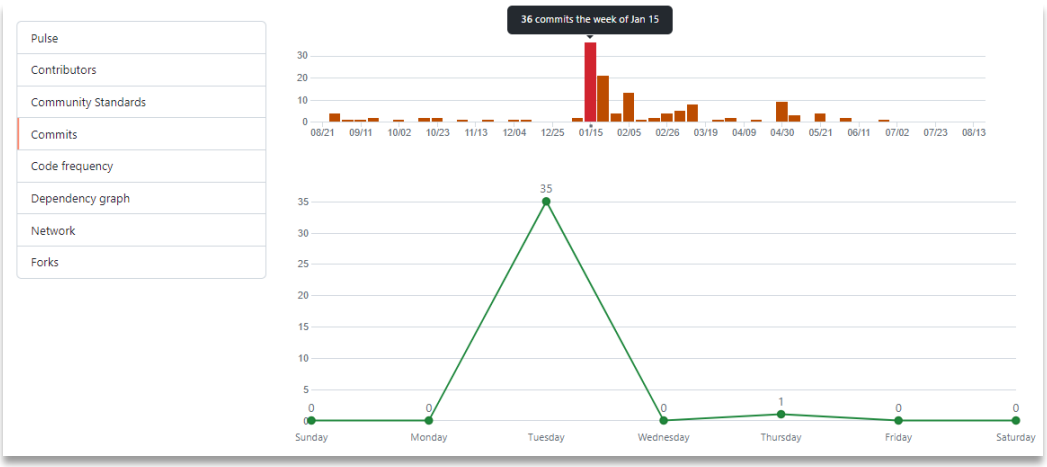


Figura 28 - Distribuição de *Commits* no Repositório.

O *Code Frequency* serve como uma ferramenta analítica no GitHub, permitindo a visualização da frequência de código ao longo do tempo, através da representação de adições e exclusões de conteúdo de forma semanal no histórico de um repositório. Este gráfico é usado para avaliar a atividade e manutenção de repositórios, auxiliando na tomada de decisão sobre a sua incorporação em projeto específicos. A consistência e a frequência das contribuições são observadas como indicativos cruciais da manutenção contínua de um projeto. A comunidade GitHub recorre a este gráfico para variados propósitos, desde a avaliação da manutenção de bibliotecas de código aberto até a reflexão sobre o próprio desempenho de codificação. No entanto, como qualquer ferramenta,

existem áreas de melhoria. A comunidade ativa e envolvida, frequentemente fornece feedback para tornar esses recursos mais úteis e mais intuitivos. Entre as sugestões propostas, destacam-se a implementação de uma funcionalidade que permita a seleção de intervalos de datas específicos, a possibilidade de visualizar dados numéricos exatos ao interagir com o gráfico e a opção de excluir determinados *commits*, especialmente aqueles de grande magnitude inicial, que possam introduzir distorções na análise.

Para uma compreensão mais tangível da subsecção *Code Frequency* de um repositório e da sua relevância na avaliação da atividade e manutenção de repositórios, a Figura 29 apresenta uma representação visual deste gráfico, demonstrando as adições e exclusões de conteúdo ao longo do tempo de um repositório específico.

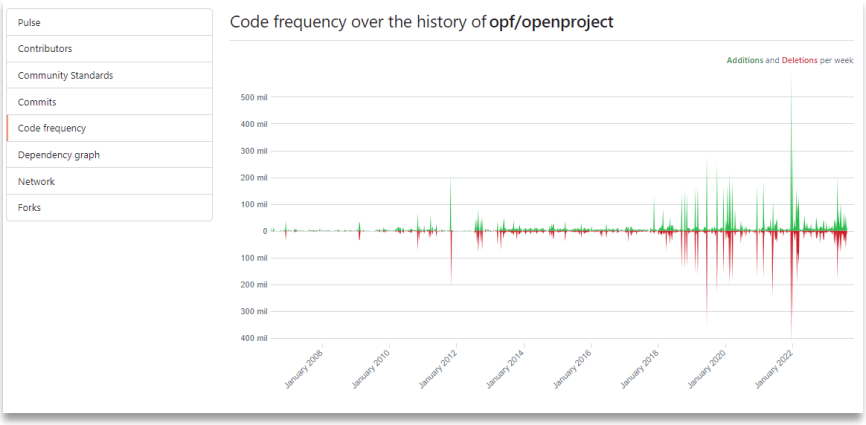


Figura 29 - Gráfico de *Code Frequency*: Adições e Exclusões Semanais no Repositório.

A visualização proporcionada pela subsecção *Dependecy Graph* destaca de forma clara as dependências e os dependentes do projeto. Esta ferramenta fornece um resumo dos arquivos de manifesto e de bloqueio armazenados num repositório, elucidando tanto as dependências -os ecossistemas e os pacotes dos quais o projeto se sustenta – quanto os dependentes, que compreendem os repositórios e os pacotes que dependem dele. Adicionalmente, para cada dependência, é possível aceder às informações de licença e a gravidade da vulnerabilidade.

O *Dependency Graph* desempenha um papel crucial ao auxiliar os *developers* a identificar as dependências, tanto diretas quanto indiretas, ou seja, permitindo uma compreensão ampla do impacto e influência de um determinado projeto ou biblioteca na comunidade. Ao fornecer uma visão clara das dependências e dependentes, os *developers* podem tomar decisões informadas acerca da inclusão, atualização ou depreciação de pacotes, garantindo a integridade, segurança e relevância do código.

Ao recorrer ao *Dependency graph*, pode-se explorar o código do qual o repositório depende. É imperativo salientar que todo o software ou hardware depende de código desenvolvido e mantido por terceiros, geralmente conhecido como *pipeline* de dependências. Estas dependências, intrínsecas ao código podem ser suscetíveis a bugs ou vulnerabilidades que, por sua vez, podem comprometer o projeto principal. Assim, torna-se essencial a revisão contínua e a manutenção destas dependências. As Figura 30 e Figura 31 ilustram esta funcionalidade, evidenciando a complexa rede de dependências e dependentes associados a um repositório específico.

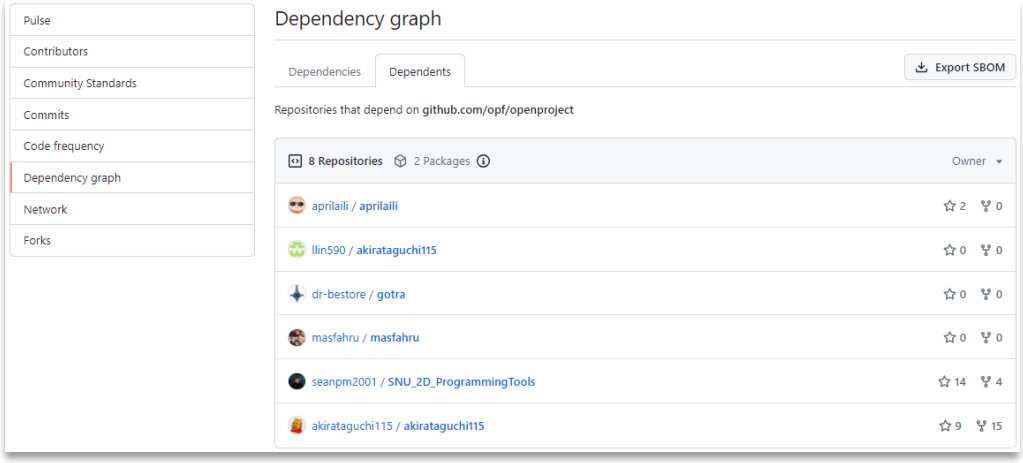


Figura 30 - Visualização da subsecção *Dependency Graph* de um repositório no GitHub: Dependentes.

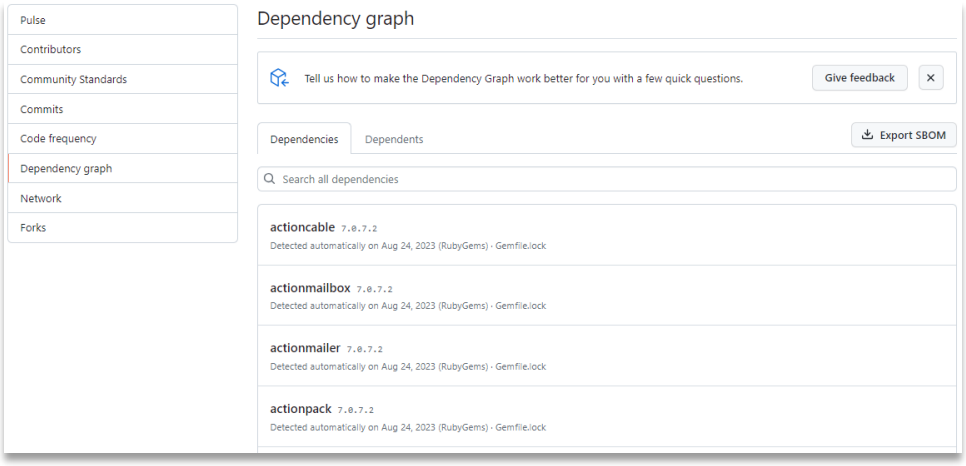


Figura 31 - Visualização da subsecção *Dependency Graph* de um repositório no GitHub: Dependências.

O *Network Graph* constituiu uma ferramenta essencial que retrata o histórico de *branches* de toda a rede do repositório, incluindo os *forks* derivados do mesmo. Esta representação é uma linha do tempo dos *commits* mais recentes, exibindo até um total de 100 das *branches* mais recentemente submetidas. A primeira linha faz referência à data, enquanto a primeira coluna faz referência ao proprietário da respetiva *branche*. Tal visualização é indispensável para entender a evolução e colaboração intrínseca no projeto. A Figura 32 oferece uma ilustração desta representação.

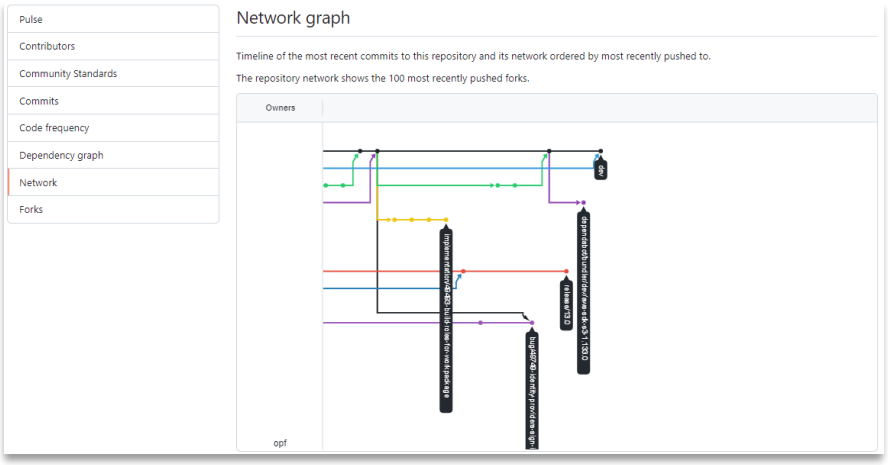


Figura 32 - Visualização do *Network Graph*: Histórico e Colaboração das *Branches* num Repositório do GitHub.

Na página específica para *forks,* o GitHub lista todos os *forks* associados a um repositório, fornecendo informações detalhadas sobre cada um. Para cada *fork*, é possível ver quantos vezes o foi marcado com uma estrela, o número de *forks* diretos, o número de *issues* em aberto, o número de *pull requests* abertos, bem como quando este foi atualizado pela última vez, ou seja, o último *push* para qualquer *branch*, e quando foi criado. Adicionalmente, oferece a funcionalidade de filtrar a lista de *forks* para exibir os ativos, inativos, com estrela ou arquivados, ou para exibir apenas os que foram atualizados dentro de um período específico, até um período de cinco anos. Para visualizar os mais úteis ou mais ativos, pode-se classificar a lista de *forks* pela maioria de *forks* com estrela ou atualizados mais recentemente, ou pelo número de *issues* abertos ou *pull requests* abertas. A Figura 33 ilustra esta subsecção, evidenciado os principais elementos e funcionalidades associados à visualização e gestão dos *forks* num repositório específico.

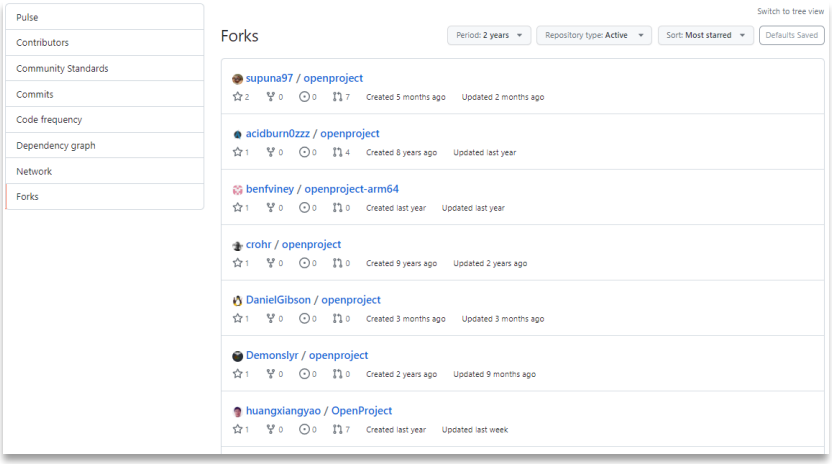


Figura 33 - Visualização e Gestão dos *Forks* num Repositório no GitHub.

Os proprietários do projeto ou membros da equipa dispõem de métricas detalhadas que facilitam a monitorização da atividade e do envolvimento no respetivo repositório. Uma dessas métricas é denominada Tráfego, que fornece uma perspetiva sobre a frequência com que o repositório é objetivo de *Git Clones,* distinguindo entre o total de clones e os clones únicos realizados por diferentes utilizadores. Esta Esta métrica, não acessível a um utilizador comum, pode ser analisada temporalmente, permitindo aos proprietários visualizar o número de clonagens diárias e semanais.

Adicionalmente, a secção Tráfego proporciona dados sobre o volume total de visualizações que o repositório registou nas últimas duas semanas, diferenciando o total de visualizações das visitas únicas, permitindo assim uma compreensão mais aprofundada da interação dos utilizadores com o repositório. A Figura 34 apresenta a interface de Tráfego no GitHub, sublinhando as métricas supracitadas e os gráficos correspondentes tanto para os *Git Clones* como para os visitantes.

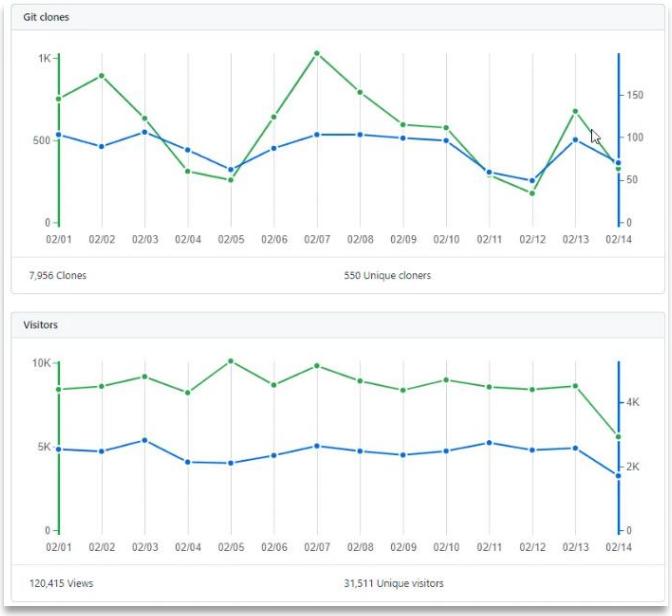


Figura 34 - Análise de Tráfego de um Repositório no GitHub: *Git Clones* e Visitantes.

Para além das organizações e repositórios, cada utilizador individual possui um perfil. Este perfil constitui uma representação digital da atividade e contribuições do utilizador na plataforma. Através deste, é viável analisar a trajetória de atividade do membro desde o ano da sua integração à plataforma até ao ano atual. Ao aceder ao perfil de um membro, é possível observar os seguintes elementos:

- **Visão Geral da Atividade**: As organizações retratadas na visão geral da atividade são priorizadas de acordo com a forma como o membro está ativo na organização. Adicionalmente, apresenta-se um gráfico elucidativo que discrimina a percentagem de diferentes tipos de atividades realizadas pelo membro, nomeadamente *code review*, *issues*, *pull requests* e *commits*.
- **Calendário de Contribuições**: A plataforma GitHub categoriza e destaca uma série de ações específicas como contribuições no perfil de um membro. Todas estas contribuições são destacadas no perfil do membro, através de um calendário visual que mostra a atividade de contribuição do membro ao longo do tempo, permitindo uma rápida visão geral da sua atividade. Estas ações incluem:
    - Realização de *commits* no *branch* padrão de um repositório;
    - Criação de *branches;*
    - Abertura de *issues* e discussões;
    - Reposta a discussões;
    - Proposta de *pull requests*;
    - Envio de revisões de *pull requests*.

- **Atividade de Contribuição**: Esta secção engloba uma linha do tempo detalhada do trabalho do membro, incluindo *commits*, *pull requests* propostas e *issues* abertas. Momentos importantes, como a data de adesão a uma organização ou a primeira *pull request*, são destacados.
- ***Pinned***: Nesta secção, são exibidos até seis dos repositórios públicos ou *gists* do membro, elucidando detalhes pertinentes de cada um;
- **Repositórios**: O perfil evidencia repositórios e *gists* que o membro possui ou aos quais contribuiu;
- ***Stars***: Esta secção revela os repositórios que o membro marcou com uma estrela, podendo refletir sobre os interesses e preferências do membro;
- **Rede de Conexões:** Esta componente do perfil mostra o número de seguidores, bem como o número de utilizadores seguidos pelo membro;
- **Organizações:** São listadas as organizações às quais o membro pertence, facilitando uma rápida navegação para esses grupos coletivos.

A Figura 35 apresenta o perfil de um membro específico de uma organização. Esta representação visual evidencia os diversos componentes e métricas associados à trajetória e envolvimento do membro da plataforma, desde a sua adesão até ao momento atual.

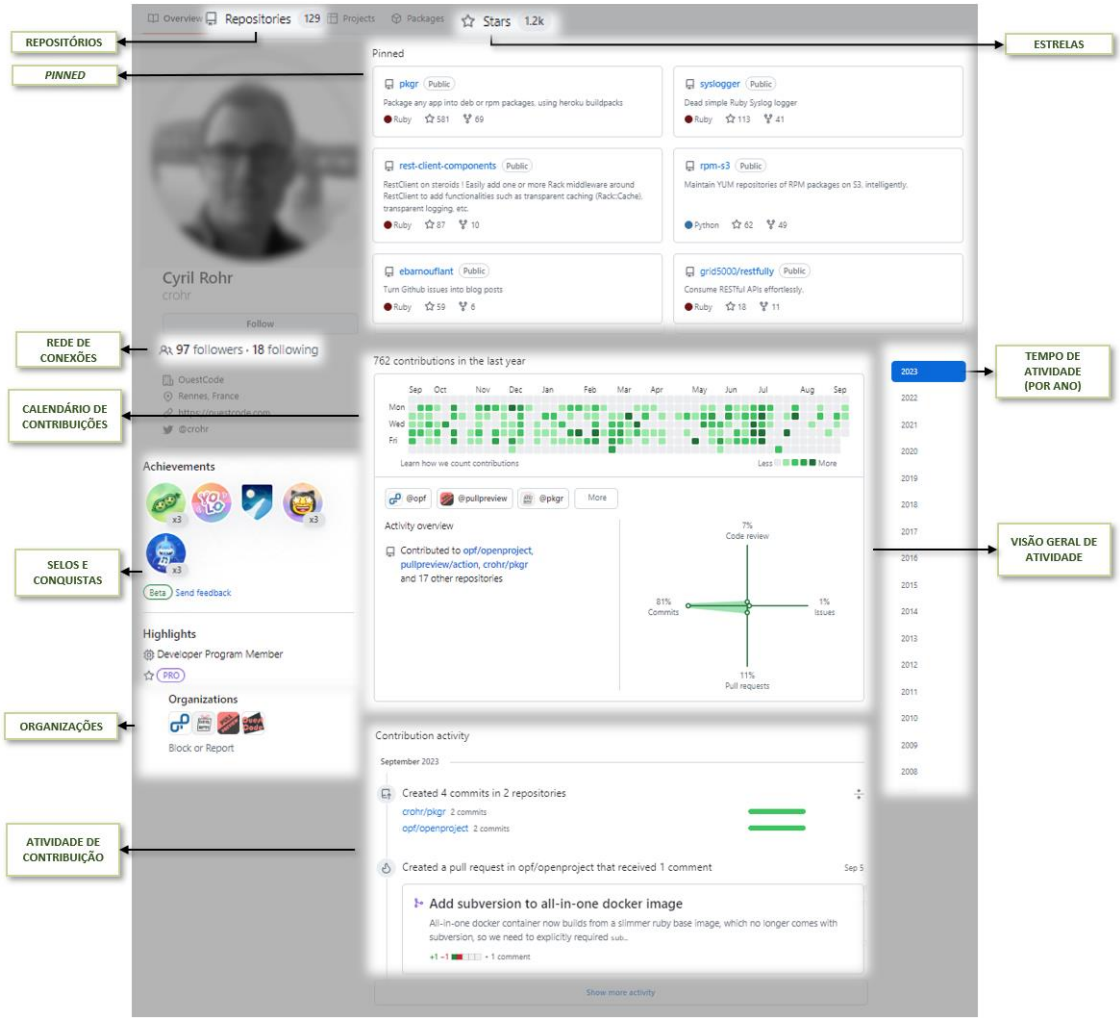


Figura 35 - Perfil de Membro no GitHub: Visão Geral e Métricas de Atividade.

3.3.3. Relações Externas do GitHub (Twitter e StackOverFlow)

Muitos *developers* recorrem a diversas plataformas para obter feedback da comunidade. O Twitter, por exemplo, é uma ferramenta valiosa para os *developers*. Eles utilizam esta plataforma não apenas para se manterem informados sobre as mudanças do setor, mas também para aprender, construir relacionamentos e identificar projetos do GitHub que possam ser interessantes para *fork* e possam beneficiar de colaboração. De acordo com Cosentino et al. (2017), a presença de discussões sobre um projeto no Twitter, seja em tom positivo ou negativo, é um indicativo da relevância desse projeto no cenário atual.

Além do Twitter, o StackOverflow é outra plataforma amplamente utilizada pelos *developers*, especialmente como canal de comunicação para a divulgação de projetos. Existe uma interação notável entre as atividades dos utilizadores no GitHub e no StackOverflow. Os colaboradores ativos do GitHub tendem a fornecer mais respostas no StackOverflow, enquanto os respondentes ativos desta plataforma tendem a fazer mais *commits* no GitHub.

O que é particularmente interessante é o modo como plataformas como o GitHub estão aproveitando estas comunidades externas. Através da integração e da promoção de discussões externas, o GitHub não apenas expande a sua presença, mas também fomenta uma cultura de colaboração aberta. Estas plataformas externas, seja o Twitter ou o StackOverflow, não são meramente suplementares ao GitHub; elas acrescentam valor de maneira significativa.

Há ainda um aspeto financeiro a ser considerado. Comunidades como o StackOverflow e o Twitter oferecem oportunidades para financiamento e patrocínio de projetos. As discussões nestas plataformas podem chamar a atenção de investidores ou empresas interessadas em colaborar ou financiar determinados projetos hospedados no GitHub.

3.4. Open Hardware Repository

O presente capítulo apresenta uma exploração aprofundada do Open Harware Repository (OHWR), uma inovação crucial no panorama do hardware aberto. Através de uma introdução detalhada ao OHWR (subcapítulo 3.4.1), este capítulo busca entender como o OHWR opera como plataforma e qual é a sua dinâmica interna. O foco então se desloca para um caso de estudo centrado numa reunião com os gestores do OHWR (subcapítulo 3.4.2). Esta interação é crucial para desvendar como os projetos são geridos dentro da plataforma, oferecendo insights valiosos sobre os desafios, estratégias e visões que moldam o OHWR e, por extensão, o campo do hardware aberto.

3.4.1. Introdução ao Open Hardware Repository

O GitHub, reconhecido como uma das principais plataformas para hospedagem de código-fonte e colaboração, estabeleceu um padrão para plataformas subsequentes que aspiram a oferecer funcionalidades avançadas e recursos de colaboração. O GitLab é uma dessas plataformas emergentes que se destaca neste contexto. O GitLab é uma plataforma completa de DevOps, oferecendo um conjunto abrangente de ferramentas desde o planeamento de projetos até a integração e entrega contínuas. Enquanto o GitHub foi pioneiro em muitos aspetos da colaboração em código aberto, o GitLab expandiu esse paradigma ao integrar uma variedade de ferramentas de desenvolvimento num único ambiente. O OHWR, por sua vez, optou por adotar a estrutura do

GitLab para as suas necessidades de gestão de projetos e repositórios, demonstrando a flexibilidade e capacidade desta plataforma. Embora o OHWR e o GitHub compartilhem certas semelhanças, é importante notar que o OHWR se baseia especificamente na infraestrutura oferecida pelo GitLab.

Open Hardware Repository (OHWR) é uma plataforma administrada por membros do CERN, uma organização europeia de pesquisa nuclear reconhecida mundialmente pelos seus avanços em física de partículas. Javier Serrano, membro integrante do CERN, é responsável pela gestão dos membros e projetos do OHWR. Para integrar esta plataforma, é necessário contactar o gestor da plataforma, que criará uma conta que permite o acesso para a contribuição nos projetos. Este mesmo procedimento também se aplica à criação de novos projetos, embora essa tarefa seja geralmente desempenhada por Erik Van Der Bij, outro membro que é gestor da plataforma.

A Figura 36 ilustra a estrutura abrangente da plataforma OHWR, destacando as métricas essenciais disponibilizadas aos utilizadores e a organização hierárquica dos seus componentes.

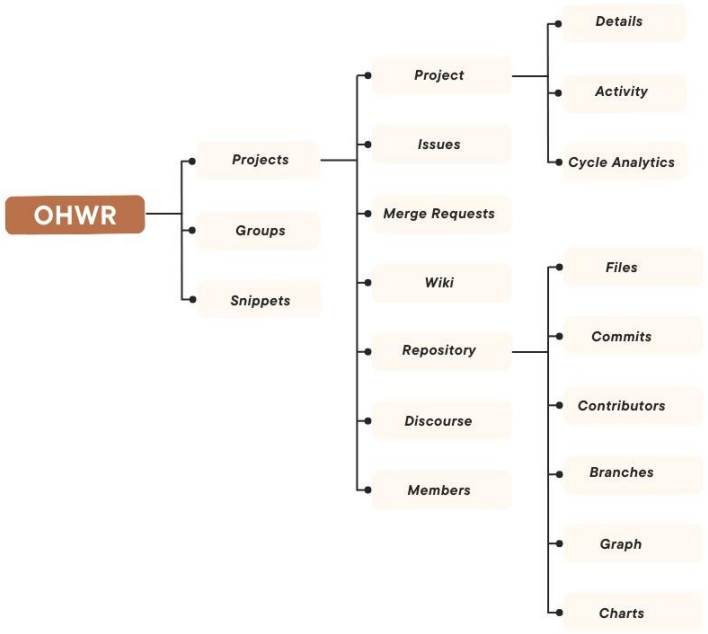


Figura 36 - Estrutura e Métricas Fornecidas pela Plataforma OHWR.

Dentro da plataforma OHWR, na interface principal, os utilizadores podem navegar por diversas categorias, nomeadamente:

- ***Projects*:** Esta secção destaca-se pela apresentação de projetos em destaque, bem como outros projetos de relevância hospedados na plataforma;
- ***Groups*:** Os utilizadores encontram uma lista de grupos públicos, aos quais podem solicitar adesão para contribuir ativamente. Estes grupos podem representar coletivos de projetos ou colaborações específicas;
- ***Snippets:*** Esta categoria, por sua vez, funciona como um repositório dedicado a trechos de código ou anotações de menor envergadura. São conteúdos que, embora não justifiquem a criação de um projeto autónomo, são considerados valiosos para partilha e consulta na comunidade.

Ao explorar um projeto específico no OHWR, identificam-se várias secções principais:

- *Project: Details, Activity e Cycle Analytics;*
- *Repository: Files, Commits, Contributors, Branches, Graph e Charts;*
- *Issues;*
- *Merge Requests;*
- *Wiki;*
- *Discourse;*
- *Members.*

A secção *Project* serve como uma visão geral do projeto, englobando diversas subsecções: *Details*, *Activity*, *Cycle Analytics*. A subsecção "*Details*", concentra diversas informações cruciais sobre o projeto. Nela, é possível identificar o número de estrelas atribuídas ao projeto e a componente *Readme*, que fornece uma descrição tanto introdutória quanto detalhada, abordando o propósito, configuração, utilização e formas de contribuição. Esta mesma subsecção destaca o número de *commits*, detalhando as alterações no código-fonte, o autor, a mensagem descritiva e a data da alteração. Adicionalmente, apresenta o número de *branches*, elucidando as diferentes ramificações do projeto, bem como as linguagens de programação utilizadas, juntamente com a sua percentagem de uso.

A Figura 37 exemplifica a subsecção *Details*, previamente descrita, destacando métricas fundamentais como o número de estrelas, *commits* e *branches* associados a um projeto.

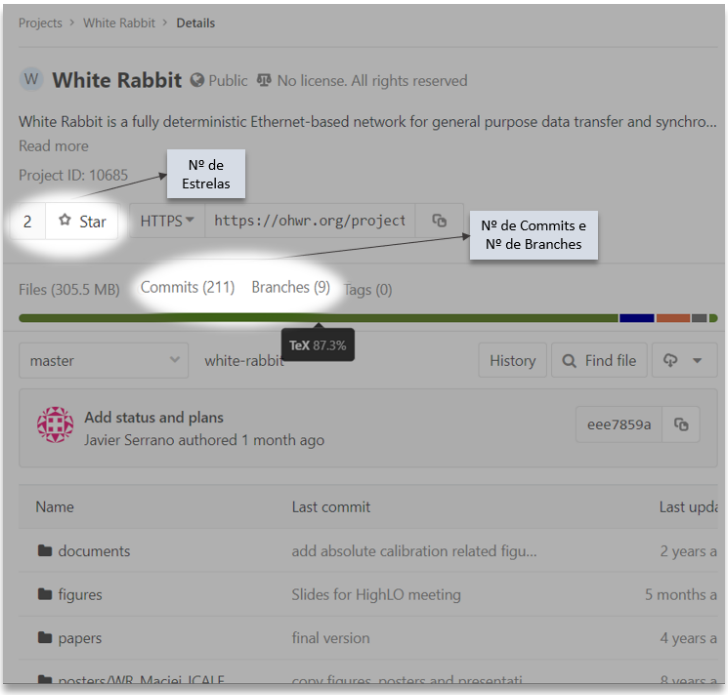


Figura 37 - Visualização da Subsecção *Details* no OHWR: Estrelas, *Commits* e *Branches*

Prosseguindo na exploração, a subsecção *Activity*, permite aos utilizadores observar os eventos recentes relacionados ao projeto, incluindo *push events*, *merge events*, *issue events*, comentários e atividades ou alterações relacionadas à equipa do projeto. A plataforma, atenta às necessidades de atualização dos seus utilizadores, disponibiliza uma opção de subscrição para notificações destes eventos.

A subsecção *Cycle Analytics*, emerge como uma ferramenta valiosa, fornecendo insights sobre o tempo necessário para transformar uma ideia em produção. O processo é meticulosamente dividido em vários estágios, desde a identificação do problema até a sua efetiva produção. Cada estágio é acompanhado de uma descrição detalhada e métricas associadas, permitindo uma análise aprofundada. A finalidade primordial desta ferramenta é otimizar a tomada de decisão e potenciar a eficiência colaborativa, auxiliando as equipes a identificar áreas de melhoria e a prever lançamentos com maior precisão. A Figura 38 ilustra de forma exemplificativa a representação de um *Cycle Analytics* referente a um projeto.

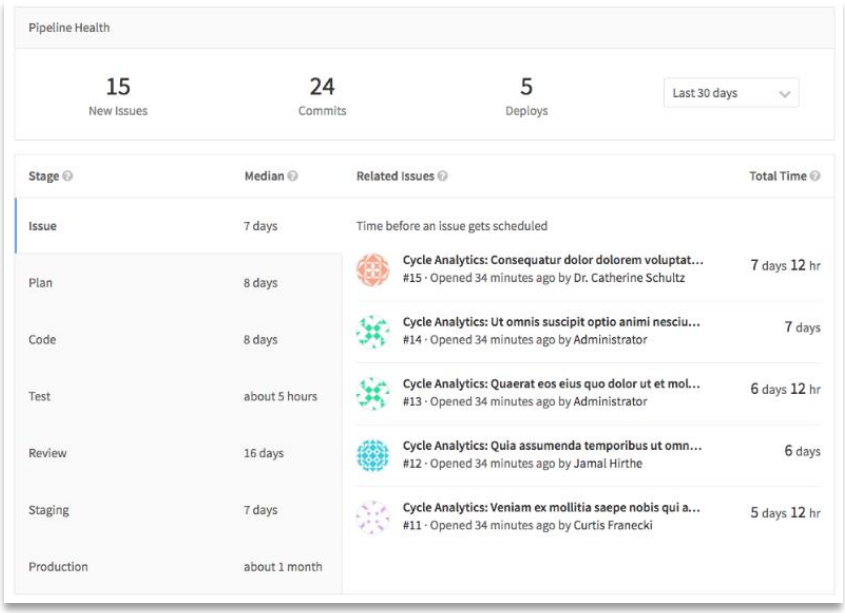


Figura 38 - Representação do *Cycle Analytics* de um projeto no OHWR.

Já a secção *Repository*, são estruturadas as diversas subsecções que proporcionam uma visão abrangente do repositório em questão. A subsecção *Files* enumera todos os arquivos e diretórios presentes no repositório. Esta enumeração não se limita apenas ao código-fonte, mas também abrange documentação, imagens, scripts, entre outros componentes essenciais que compõem o projeto.

A subsecção *Commits* desempenha um papel crucial ao manter um registo cronológico de todas as alterações realizadas no repositório. Para aqueles que desejam acompanhar de perto estas alterações, a plataforma oferece a possibilidade de subscrição ao *feed* de *commits*. Cada *commit*, por sua vez, é detalhadamente caracterizado, mencionando o autor, uma mensagem descritiva, a data da alteração e um identificador único. Esta organização meticulosa facilita o acompanhamento da evolução do código e, se necessário, permite regressar a versões anteriores.

Prosseguindo na estrutura, a subsecção *Branches* oferece uma visão clara do número de *branches* ativos e inativos, permitindo aos utilizadores compreender as diferentes ramificações do projeto.

Por sua vez, a subsecção *Contributors* destaca-se ao apresentar um gráfico que retrata o histórico de *commits* totais do projeto. Os utilizadores têm a liberdade de especificar o período temporal que desejam analisar, proporcionando uma análise adaptada às suas necessidades. É pertinente mencionar que, para garantir uma visualização otimizada, a representação gráfica tem um limite de 6000 *commits*, excluindo os *mescled commits*. Além disso, cada contribuidor tem a oportunidade

de visualizar um gráfico individual, que ilustra a distribuição temporal dos seus *commits*. A Figura 39 exemplifica esta representação gráfica, ilustrando os gráficos referidos.

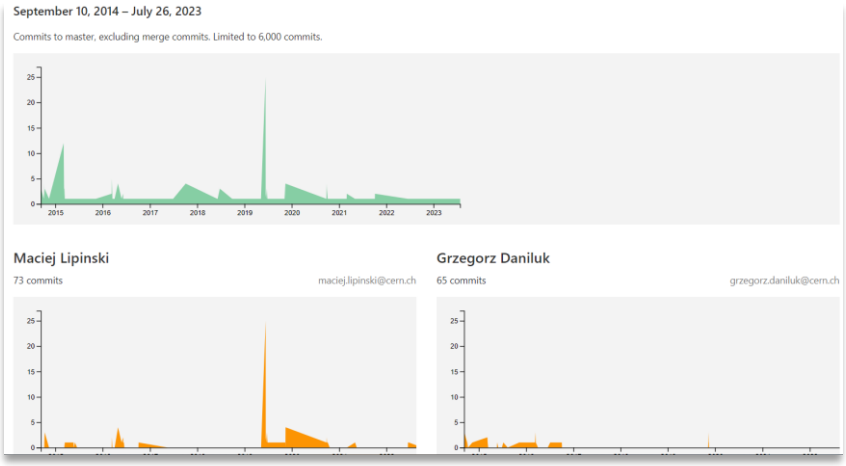


Figura 39 - Representação gráfica do histórico de *commits* e atividade individual dos contribuidores.

Concluindo a secção *Repository*, a subsecção *Charts* enriquece a análise ao proporcionar uma visão detalhada das linguagens de programação utilizadas no repositório. Esta análise é complementada com gráficos que representam as estatísticas de *commits* em diferentes intervalos temporais. Estes gráficos ilustram, entre outros, a distribuição de commits ao longo dos dias do mês, a frequência de *commits* pelos dias da semana e a distribuição horária dos *commits* ao longo do dia, oferecendo uma perspetiva abrangente e detalhada da atividade no repositório. Nas Figura 40 e Figura 41, é possível observar o que se visualiza nesta subsecção, exibindo os gráficos já mencionados.

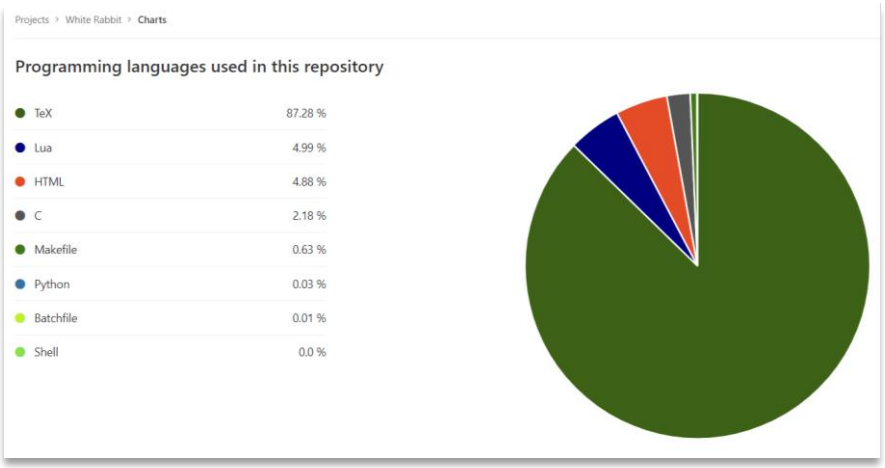


Figura 40 - Visualização gráfica das linguagens de programação no repositório.

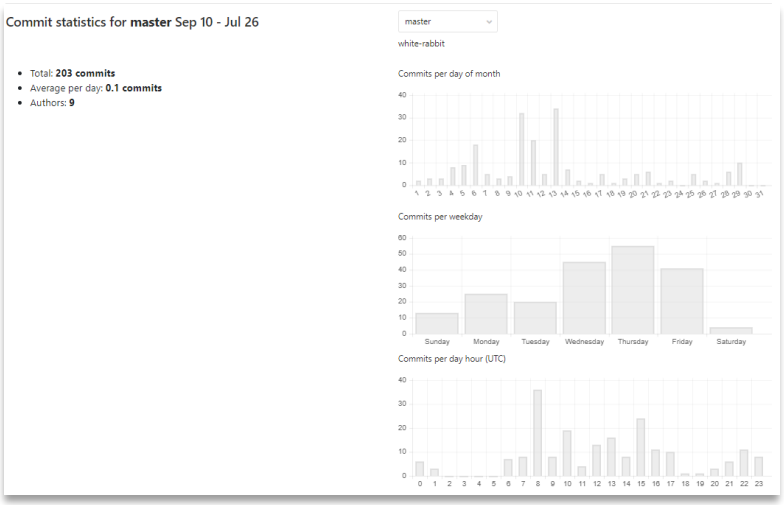


Figura 41 - Visualização gráfica das estatísticas de *commits* no repositório.

Dando continuidade à análise das secções apresentadas num projeto do OHWR, a secção *Issues* é uma ferramenta vital para a gestão de problemas e tarefas associadas ao projeto. Esta secção não só permite aos utilizadores uma visão quantitativa dos *issues* - sejam eles abertos ou já concluídos - mas também proporciona detalhes sobre os *commits*, as datas de atualização, a identificação dos responsáveis por sua abertura e o volume de comentários gerados por cada *issue.* Além disso, para aqueles que desejam manter-se atualizados sobre as questões mais recentes, a plataforma oferece a possibilidade de subscrever tanto o *feed* RSS como o calendário de issues. Importa salientar que cada *issue* não é apenas um mero registo, podendo ser enriquecida com discussões, etiquetas, marcos temporais e até mesmo ter um responsável designado para a sua resolução. A Figura 42 ilustra de forma clara a secção *Issues*, destacando os elementos supracitados.

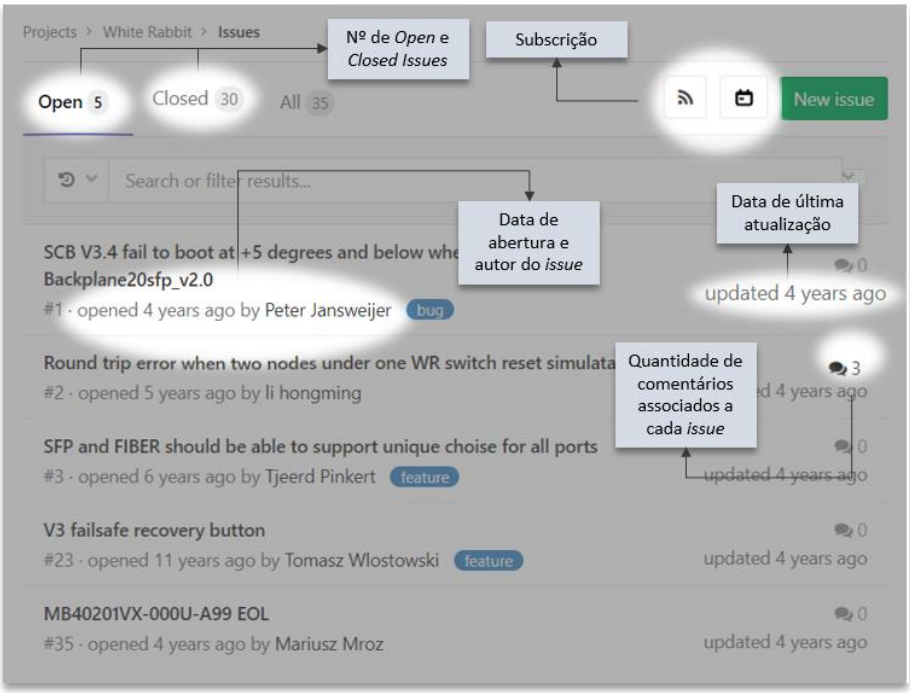


Figura 42 - Representação da secção *Issues* de um repositório: *commits*, estado dos *issues* e interações associadas.

Na sequência, a secção *Merge Requests* é uma área crucial para a integração de alterações no código principal. Os utilizadores têm a capacidade de monitorizar o estado das solicitações de *merge*, sejam elas abertas, fechadas ou já integradas. É possível discernir a data de atualização de cada *pull request*, identificar quem a abriu e quando, bem como o volume de comentários gerados por cada uma. Adicionalmente, é destacada a atribuição de determinadas *pull requests* a colaboradores específicos, indicando quem é responsável pela sua revisão e subsequente ação, conforme exemplificado pelo "*assigned to*" na Figura 43. A plataforma, ciente da necessidade de uma análise rigorosa, promove um ambiente propício para a revisão do código, incentivando a interação e discussão entre os colaboradores antes de qualquer fusão. A Figura 43 apresenta uma representação visual da secção *Merge Requests* de um repositório no OHWR.

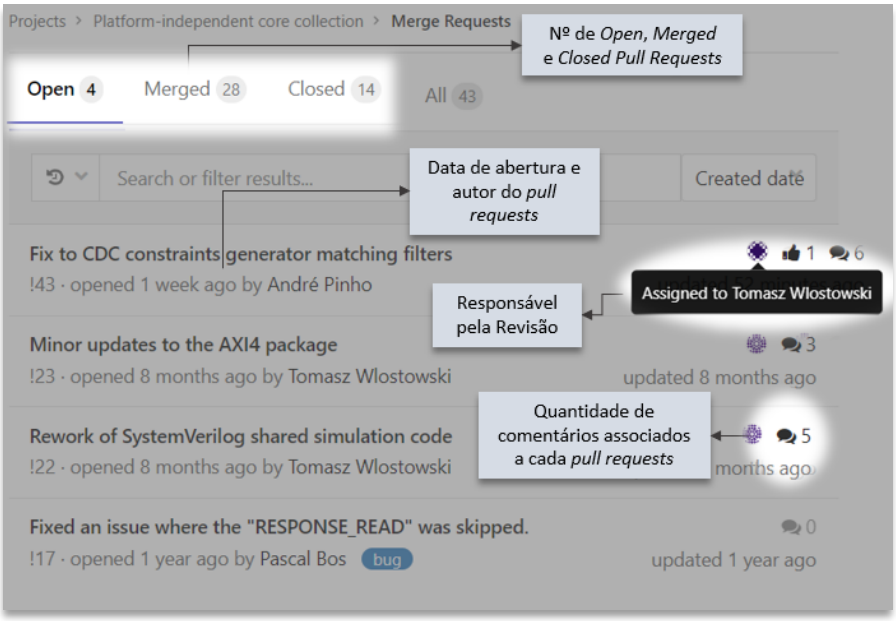


Figura 43 - Visão Geral da Secção *Merge Requests* de um Repositório no OHWR: Detalhes, Responsáveis e Comentários.

No que concerne à documentação do projeto, a secção *Wiki* assume-se como um pilar fundamental. Esta área funciona como um repositório de conhecimento, onde os colaboradores têm a liberdade de criar e editar páginas que se relacionem com o projeto, garantindo assim que toda a informação relevante esteja acessível e organizada.

Na secção *Discourse*, estabelece uma conexão direta com fóruns ou discussões vinculadas ao projeto. Este espaço é essencial para a comunidade, uma vez que proporciona um ambiente onde os membros podem debater, formular questões e disseminar conhecimento de maneira organizada e colaborativa.

Por fim, a secção *Members* oferece uma visão clara e compreensiva dos colaboradores e membros da equipa associados ao projeto. Esta lista não só identifica os participantes, mas também detalha os papéis e permissões que cada um possui, garantindo transparência e organização na gestão do projeto. A Figura 44 apresenta uma representação visual desta secção, evidenciando a data em que cada membro recebeu acesso ao projeto e a respetiva função que desempenha na equipa.

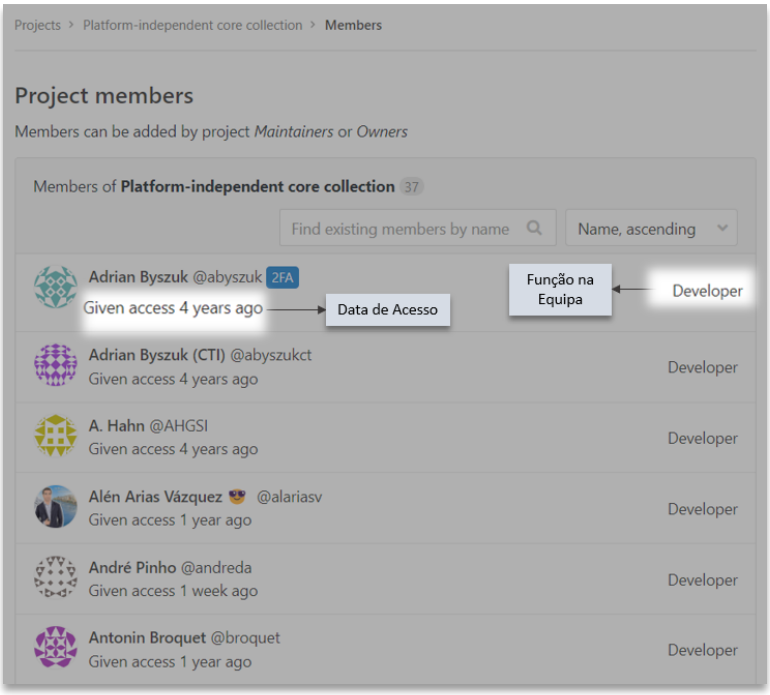


Figura 44 - Visão Geral da Secção *Members* - Acesso e Funções dos Colaboradores num Repositório.

Para além dos projetos, a plataforma OHWR proporciona uma visão detalhada sobre os perfis dos membros que contribuem ativamente para o ecossistema. Estes perfis são estruturados em várias secções, refletindo não apenas a atividade individual, mas também a interação e colaboração dentro da comunidade.

Para uma compreensão mais clara e organizada, as secções que compõem o perfil de um membro no OHWR são detalhadas da seguinte forma:

- Visão Geral;
- Atividade;
- Grupos;
- Projetos contribuídos;
- Projetos Pessoais;
- *Snippets.*

A secção de visão geral oferece uma representação visual das contribuições do membro ao longo do último ano, permitindo uma análise quantitativa da sua atividade. Se o membro tiver projetos pessoais, estes são destacados nesta secção. Adicionalmente, é possível observar uma lista cronológica das mais recentes contribuições do membro, bem como o ano em que começou a contribuir para o OHWR. Uma funcionalidade notável nesta secção é a opção de subscrever o membro, permitindo aos utilizadores seguir de perto as atividades e atualizações desse membro específico.

Na secção Atividade, um gráfico de contribuições do último ano destaca-se, ilustrando as atividades do membro, como *issues*, *merge requests*, *push events* e comentários. Este gráfico permite uma visão diária das contribuições, enquanto uma lista cronológica subsequente detalha as atividades mais recentes, oferecendo uma descrição pormenorizada.

A secção Grupos permite identificar os grupos ou organizações aos quais o membro pertence. Esta informação é crucial para entender as colaborações e redes de trabalho do membro dentro da plataforma.

A secção Projetos Contribuídos evidencia a amplitude da colaboração do membro, destacando os projetos aos quais contribuiu e indicando o número total de projetos em que esteve envolvido. Por outro lado, a secção Projetos Pessoais oferece uma visão das iniciativas individuais do membro, revelando os projetos que foram concebidos e geridos por ele.

Por último, a secção *Snippets* apresenta fragmentos de código partilhados pelo membro, refletindo as suas competências técnicas e a vontade de partilhar soluções com a comunidade.

Estas secções, na sua totalidade, fornecem um panorama aprofundado da atuação e contribuições de um membro no OHWR, sublinhando o papel vital de cada participante no ecossistema colaborativo da plataforma. A Figura 45 ilustra um perfil típico de membro no OHWR.

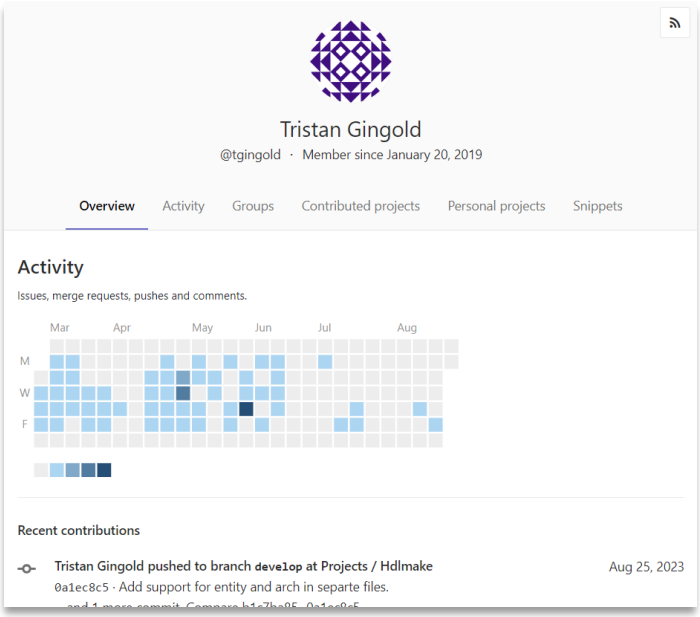


Figura 45 - Representação visual do perfil de um membro no OHWR.

Após uma análise abrangente da plataforma OHWR, que incluiu a exploração dos projetos, a estrutura da plataforma e o perfil detalhado dos membros, é pertinente expandir a perspetiva para compreender a dimensão coletiva e institucional da mesma. Os projetos hospedados no OHWR são limitados a desenvolvimentos que se alinham com os interesses dos laboratórios de física. Até ao momento, a plataforma conta com a participação de 617 membros ativos. Destes, 49 destes estão associados a 24 empresas distintas, conforme ilustrado na Tabela 10.

Essas 24 empresas, presentes na plataforma, não se limitar a utilizar o OHWR como um mero meio de desenvolvimento. Elas também têm a possibilidade de serem renumeradas pelos desenvolvimentos que realizam, abrangendo áreas tão variadas como hardware, software e drivers abertos. Contudo, é crucial salientar que os dados apresentados na Tabela 10 refletem unicamente as atividades destas empresas no âmbito do OHWR, podendo existir outras vertentes e áreas de atuação não detalhadas neste contexto.

Tabela 10 - Empresas Associadas ao OHWR: Atividades, Origem e Contribuições. Adaptado de Open Hardware Repository (2023)

Nome	País	Desenvolvimento de *Hardware*	Comercialização de *Hardware*	Desenvolvimento de Linguagem de Descrição de *Hardware* (HDL)	Desenvolvimento de *Software*	Projetos	Membros
Aivon	Finlândia	×	✓	×	×	2	0
Cosylab	Eslovênia	✓	×	✓	×	4	4
Creotech	Polónia	✓	✓	✓	✓	32	13
Digicom Electronics	USA	×	✓	×	×	1	0
ELMA	Suiça	✓	×	×	×	1	1
GL-Research	Espanha	✓	✓	✓	✓	3	1
Gnudd	Itália	×	×	×	✓	18	2
HLP Technologies	França	✓	✓	✓	✓	1	1
Igalia	Espanha	×	×	×	✓	2	3
INCAA Computers	Amsterdão	✓	✓	×	✓	6	2
Institute of Electronic Systems	Polónia	✓	×	✓	✓	2	4
Integrasys	Espanha	×	×	✓	✓	2	3
ISD	Grécia	×	✓	×	×	1	0
Janz Tec	Alemanha	×	✓	✓	✓	3	0
KAYA Instruments	Israel	✓	✓	✓	✓	1	1
MagentaSys	Suiça	✓	×	×	×	2	0
Milky Mist	França	×	×	✓	×	1	1
NetTimeLogic	Suiça	×	×	✓	✓	2	1
OPNT	Amesterdão	✓	✓	✓	×	2	2
ORSoC	Suécia	✓	×	×	×	1	0
Seven Solutions	Espanha	✓	✓	✓	✓	17	6
Splendeo Innovación	Espanha	×	×	×	✓	1	1
Sundance Technology	Reino Unido	✓	✓	✓	✓	3	2
SyncTechnology	China	✓	✓	✓	×	4	1

A fim de destacar o papel ativo das empresas no Open Hardware Repository (OHWR), foi elaborado um gráfico que demonstra tanto o número de projetos em que cada empresa está envolvida quanto o número de membros associados a cada projeto. Este gráfico, apresentado na Figura 46, oferece uma perspetiva detalhada e clara da participação e contribuição das empresas na plataforma OHWR.

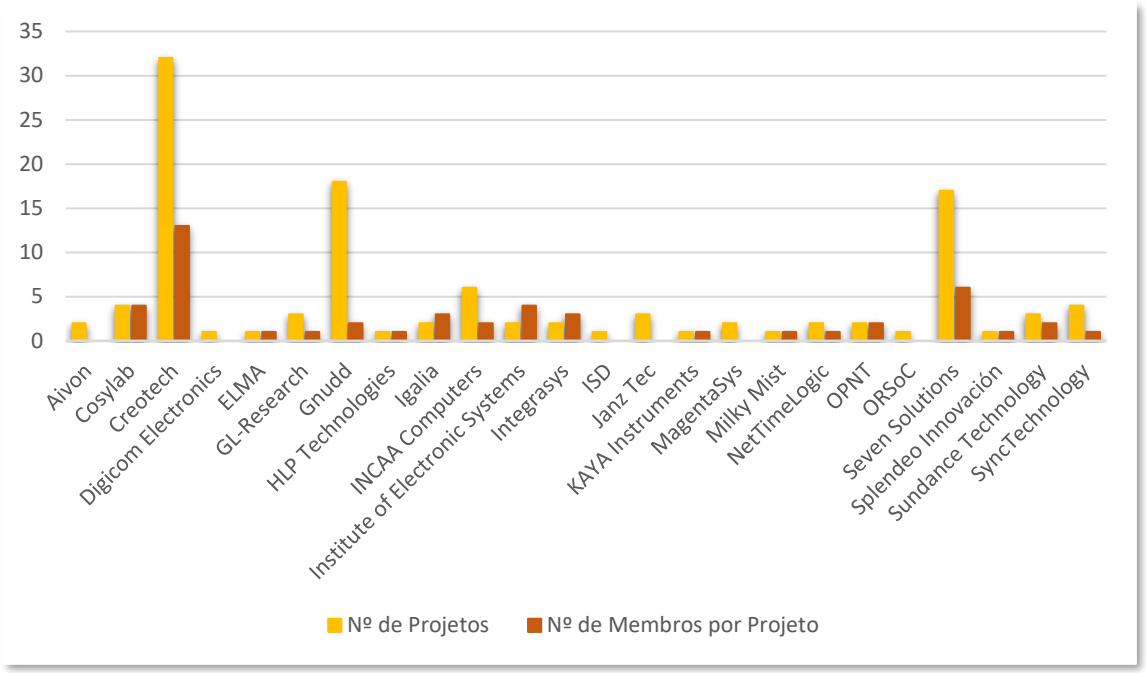


Figura 46 - Envolvimento das Empresas no OHWR: Projetos e Membros por Projeto.

O gráfico revela uma distribuição variada de projetos e membros entre as empresas, sendo interessante notar estas variações, pois fornecem perspetivas sobre a dinâmica e estratégias de cada empresa na plataforma. A empresa Creotech é a que mais se destaca, tanto em projetos como em membros, sugerindo uma forte presença e atividade na plataforma. Por outro lado, as empresas Gnudd e Seven Solutions também se destacam a nível de projetos, embora possuam uma menor quantidade de membros em comparação com a Creotech, o que pode indicar uma maior concentração em projetos específicos ou uma equipa mais especializada.

Prosseguindo com a análise da distribuição dos membros na plataforma OHWR, é pertinente observar a proporção entre membros individuais e aqueles associados a empresas. A Figura 47 ilustra esta distribuição, contrastando o número total de membros ativos com o número de membros associados a empresas listadas, permitindo uma melhor compreensão da participação empresarial em relação ao total de membros na plataforma.

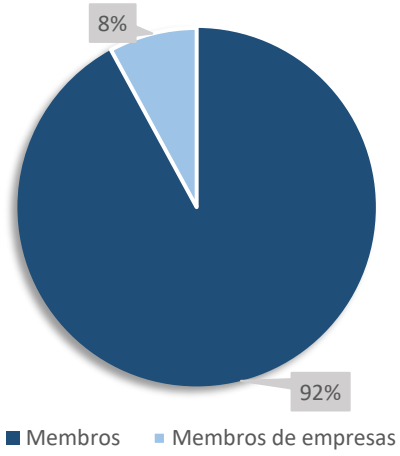


Figura 47 - Distribuição de Membros na Plataforma OHWR.

A estrutura do OHWR é baseada no GitLab, contando com 391 repositórios. A colaboração entre os membros é facilitada por ferramentas que permitem a identificação de problemas e a criação de ramificações para o desenvolvimento de novos recursos. Quando um recurso é finalizado, procede-se à elaboração de uma *merge request*. Esta solicitação é então submetida para revisão pelos proprietários do projeto, sendo posteriormente testada e, se aprovada, integrada ao repositório principal. Dentro desta estrutura, os membros unem esforços no desenvolvimento de *design* eletrônico, *gateware*, *firmware* e *software*.

O OHWR oferece uma vasta gama de informações cuja acessibilidade aos dados varia consoante o nível de privilégio do utilizador:

- O número total de membros, que atualmente é de 617. Contudo, é importante salientar que esta lista é de acesso restrito, estando disponível apenas para administradores;
- Os *commits*, que são uma característica intrínseca do GitLab, estão disponíveis para consulta individual de cada projeto;
- Informações detalhadas sobre o desenvolvimento e histórico de cada projeto, bem como a sua wiki, podem ser acedidas diretamente no projeto.

A plataforma destaca-se não apenas pela sua capacidade de gestão e organização de projetos, mas também pela clareza com que define os níveis de acesso e intervenção de cada membro. Esta definição clara de papéis e responsabilidades é crucial para garantir a eficiência e a transparência na colaboração entre os diversos intervenientes. As funções do OHWR são categorizadas da seguinte forma:

- Administrador – Esta função representa o nível mais elevado de autoridade num projeto na plataforma. O administrador tem a capacidade para intervir em todas as ações na plataforma, garantindo a integridade e a segurança dos dados e projetos;
- *Maintainer* (de um projeto) – Apesar de possuir privilégios semelhantes ao do administrador, o seu controlo é restrito ao projeto específico em que colabora. Esta restrição assegura uma gestão focada e especializada para cada projeto
- *Developer* (de um projeto) – O *developer* pode realizar ações semelhantes ao *maintainer*, exceto em áreas como gestão de documentos, criação de fóruns, gestão de páginas *wiki* notícias;
- *Reporter* (de um projeto) – Esta função é essencialmente orientada para a criação e gestão de *issue*. Embora tenha direitos de leitura abrangentes, a sua capacidade de intervenção é mais restrita, focando-se na identificação e reporte de problemas ou necessidades;
- Convidado – A função de convidado é mais passiva, permitindo apenas a visualização das informações do projeto. No contexto do OHWR, é importante salientar que esta função não é aplicada. A razão para tal reside no facto de a plataforma promover a transparência e o acesso livre a todos os projetos, eliminando a necessidade de autenticação para visualização.

3.4.2. Caso de Estudo: Reunião com OHWR

O CERN, reconhecido pelo seu papel fundamental na investigação, tem consistentemente expandido as fronteiras da tecnologia, cultivando uma cultura de inovação que frequentemente resulta em benefícios tangíveis para a sociedade. Um dos exemplos mais emblemáticos desta

inovação é a *World Wide Web*, que, embora inicialmente criada para permitir que cientistas partilhassem informações, revolucionou a comunicação global.

No seguimento da Web, o CERN continuou a promover a abertura dos seus desenvolvimentos de *software* ao público. Esta abertura não se limitou apenas ao *software*, mas também se estendeu ao *hardware*. Reconhecendo os benefícios que uma filosofia de código aberto poderia trazer para o desenvolvimento e disseminação de *hardware*, em 2009, um grupo de *designers* de eletrónica do Departamento de *Beams* do CERN estabeleceu o Open Hardware Repository. Este repositório, conforme o seu manifesto, serve como um espaço para *designers* de eletrónica de instalações de física experimental colaborarem em projetos de *hardware* aberto, alinhando-se com a filosofia do movimento de *software* livre.

A necessidade de um quadro jurídico sólido para a distribuição de *hardware* aberto tornou-se evidente. O CERN, ao explorar diferentes opções, percebeu que as licenças de *software* ou documentação existentes não eram adequadas para o propósito de *hardware*. Assim, foi criada a Licença de *Hardware* Aberto pelo CERN, que regula a utilização, cópia, modificação e distribuição da documentação de *design* de *hardware*, bem como o fabrico e a distribuição de produtos baseados nessa documentação. Esta licença garante que qualquer pessoa possa exercer estes direitos, desde que os novos desenvolvimentos sejam publicados sob os mesmos termos, permitindo que melhorias feitas pela comunidade de *hardware* aberto sejam acessíveis a todos.

O Open Hardware Repository (OHWR) é uma plataforma gerida pela secção BE-CEM-EDL do CERN (Beams Department > Controls Electronics & Mechatronics group > Electronics Design & Low Level Software section), com o apoio da secção BE-CSS-ISA (Beams Department > Controls Software & Services group > Infrastructure & System Administration section). A liderança administrativa do OHWR está a cargo de Javier Serrano e Erik Van Der Bij. Importa salientar que Javier Serrano, além de ser o chefe da secção BE-CEM-EDL, desempenha um papel crucial na gestão dos membros e projetos da OHWR.

No âmbito deste estudo, e com o objetivo de aprofundar o entendimento sobre a gestão e funcionamento do OHWR, foi estabelecido contacto com vários membros do CERN associados ao OHWR. Esta interação contou com a participação de 1 gestor (Javier Serrano) e 3 associados à plataforma (Vasco Guita, Juan David Gonzales Cobas e Dimitris Lapridis). Ao longo da reunião, foram abordados diversos tópicos, visando esclarecer aspetos desde a génese dos projetos até à avaliação da contribuição dos membros. As questões colocadas foram:

- Como surgem os projetos atuais e futuros? Surgem exclusivamente das necessidades do CERN, dos seus parceiros/empresas ou da comunidade em geral?
- Existe uma métrica de desempenho para os membros na plataforma? Em caso afirmativo, quais são os indicadores utilizados para identificar os membros mais ativos e que estratégias são adotadas para incentivar a sua atividade?
- Quais são os critérios de avaliação da contribuição de um potencial membro para um projeto específico?
- Dos membros registados, excluindo os 49 associados a empresas, pertencem ao CERN ou são independentes?
- É possível fornecer detalhes sobre os projetos em que cada empresa está envolvida e o número de contribuições de cada uma?
- Quem assume a liderança de cada projeto e existe uma liderança dinâmica?

- Quais são os critérios de seleção para a inclusão de perfis específicos de membros num determinado projeto?

As respostas obtidas nesta reunião foram elucidativas e revelaram alguns dos processos e práticas adotadas no OHWR. Durante a reunião, foi possível entender o processo de admissão de novos projetos. Quando um projeto é proposto, é detalhado o objetivo do projeto, a licença desejada e é sugerido um nome provisório. Essas solicitações são enviadas para o responsável pela gestão da plataforma, que determina se o projeto se enquadra nos critérios para do OHWR.

Durante a reunião, salientou-se:

"Limitamos os projetos que são de potencial interesse para laboratórios de física".

O gestor da plataforma prosseguiu, enfatizando que, devido à restrição de recursos e falta de tempo para investir ainda mais na plataforma, a equipa do OHWR optou, desde o início, por focar exclusivamente em projetos que se alinham a este critério específico. A solicitação do projeto é encaminhada para Erik Van Der Bij, outro membro responsável pela gestão da plataforma, que criará o projeto e fornecerá orientações inicias ao proponente.

Relativamente à monitorização do desempenho dos membros na plataforma, verificou-se que não existe um sistema formalizado. Embora o GitLab disponibilize algumas métricas de atividade, como por exemplo o número de contribuições de um membro, o OHWR não realiza um acompanhamento ativo do desempenho dos membros. Neste contexto, uma observação do gestor da plataforma foi particularmente esclarecedora:

"O objetivo não é avaliar as pessoas, queremos que a plataforma seja útil… Há sempre uma noção de como está o estado de um projeto".

Esta declaração sublinha a filosofia central do OHWR em relação à sua comunidade de membros. No entanto, é reconhecido pelos gestores da plataforma a importância de manter os projetos ativos e mencionarem que, caso um projeto pareça estar inativo, a equipa pode tomar a iniciativa de entrar em contacto para verificar o estado e incentivar o seu desenvolvimento contínuo. No decorrer da discussão sobre a contribuição dos membros à plataforma, o gestor do OHWR observou:

"Só fazemos isso para os nossos próprios projetos, há muitos projetos que os membros do CERN não fazem parte. Apenas hospedamos a plataforma e não interferimos com os procedimentos e desenvolvimentos no resto da comunidade".

Contudo, foi mencionado que a equipa do OHWR aspira adotar um processo de desenvolvimento que se assemelhe mais ao adotado no mundo do software, englobando solicitações de integração, diálogo e tomada de decisões. É relevante mencionar que a maioria dos membros não pertence ao CERN, sendo externos que solicitam a criação de uma conta. O processo de adesão não é muito restritivo, e os membros têm alguns direitos fundamentais, como a capacidade de sinalizar problemas e criar *issues*. Além disso, o gestor de comunidade esclareceu o papel da equipa do CERN, afirmando:

"A equipa do CERN só se envolve no início de um projeto para criar o projeto e também para atribuir um *maintainer*. E então esse *maintainer* tem o direito de promover qualquer outra pessoa a *maintainer* ou *developer* e assim por diante."

Deste modo, a equipa do OHWR não desempenha nenhum papel na estrutura interna dos projetos que não são originários do CERN. A estrutura de liderança é variável, dependendo do projeto, contudo, a maioria mantém uma liderança estável. O gestor da plataforma sublinhou a importância de atribuir funções apropriadas no repositório, realçando:

"Se sou apenas um *reporter*, isso é uma função relativamente simples de ser atribuída e não somos muito exigentes em atribuir essa função num projeto, mas ser um *developer* significa que pode realmente escrever no repositório git, então promover alguém a *developer* num determinado projeto, é algo que deve ser cuidadosamente considerado".

A análise da reunião com os representantes da comunidade OHWR proporcionou uma visão mais aprofundada do funcionamento desta plataforma, assim como o compromisso do OHWR com o Open Source. Ficou evidente que existem diversas áreas que requerem desenvolvimento e aprimoramento na plataforma. Uma das principais observações foi a possibilidade de incorporar práticas consolidadas de outras plataformas, como o GitHub, que disponibilizam recursos avançados para revisão e avaliação de projetos.

Destacou-se a ausência da funcionalidade de *forks*, percebida como uma lacuna que limita a colaboração descentralizada. A interface do usuário, embora funcional, poderia beneficiar-se de aprimoramentos para torná-la mais intuitiva. Além disso, a atual burocracia associada à criação de projetos, que exige uma série de etapas manuais, poderia ser simplificada. A introdução de um sistema de registo automático e a possibilidade de criar projetos sem a necessidade de aprovação manual poderiam tornar a plataforma mais acessível, expandindo-a além dos domínios tradicionais dos Laboratórios de Física.

O ambiente aberto do OHWR possui vantagens inerentes. A revisão por pares, por exemplo, promove a reutilização de *designs* e a colaboração com a indústria, com expectativas de resultar em produtos de qualidade superior. Este compromisso com a abertura é evidenciado pelo "Suporte OHR", um projeto que atua como recurso para os utilizadores e ponto de notificação de problemas.

A automação de processos foi destacada como uma área crucial para desenvolvimento no OHWR. A atual dependência de tarefas manuais pode comprometer a eficiência e escalabilidade da plataforma. Através da implementação de ferramentas e fluxos de trabalho automatizados, é possível otimizar a criação de projetos, a atribuição de funções e a gestão de permissões. Tal automação minimizará a necessidade de intervenções manuais, permitindo que a equipa do OHWR se concentre em atividades de maior valor.

No que diz respeito à comunicação e colaboração entre membros e projetos na plataforma também podem ser aprimoradas. Embora existam direitos e permissões distintos para diferentes níveis de envolvimento, a introdução de recursos que facilitem a interação, como solicitações de integração e discussões estruturadas, pode fortalecer a colaboração e cultivar um ambiente mais inclusivo e participativo.

Outro aspeto notável foi a diversidade dos membros na plataforma. Apesar da já existente variedade de membros no OHWR, é essencial expandir essa diversidade, abrindo a plataforma para áreas além das necessidades específicas dos Laboratórios de Física, atraindo assim membros de diferentes campos e especializações.

A falta de uma API dedicada foi outra área identificada para melhoria. Uma API robusta permitira integrações inovadoras e uma interação mais rica com a plataforma. Para fomentar a participação

contínua, o OHWR poderia introduzir um sistema de reconhecimento para os contribuidores mais ativos, incentivando a colaboração e o envolvimento.

A reunião ainda ressaltou a necessidade de monitorar o desempenho dos membros na plataforma. A implementação de métricas para monitorar a atividade dos membros pode fomentar uma colaboração contínua e fornecer informações necessárias e importantes para avaliar a eficácia dos projetos.

No entanto, foi informado que estão em desenvolvimento, planos para uma nova versão do OHWR, liderada por dois representantes do CERN, Vasco Guita e Erik Van der Bij, que transformará a plataforma num catálogo de projetos. Os projetos, por sua vez, serão armazenados ou publicados numa plataforma Git à escolha dos *maintainers*, como por exemplo, o GitHub. Esta abordagem visa aproximar ainda mais os projetos da comunidade, tornando-os mais acessíveis para novos membros e contribuições. Adicionalmente, espera-se uma redução nos custos de manutenção e gestão, uma vez que não será necessário suportar a atual plataforma GitLab, cuja versão está desatualizada, nem gerenciar utilizadores. Até que esta nova versão esteja em produção, a responsabilidade pela plataforma atual continuar a recair sobre os mesmos gestores da plataforma.

Dessa forma, o OHWR, com a sua filosofia de ambiente aberto e uma série de módulos úteis, apresenta um potencial significativo para crescimento e evolução. Ao adotar melhores práticas de supervisão, monitoramento de desempenho, automação de processos, comunicação efetiva, diversidade de membros e recursos específicos, a plataforma pode fortalecer o seu papel como um ambiente colaborativo para projetos de Open Hardware e Software.

A Tabela 11 resume as principais conclusões e recomendações derivadas da reunião, delineando áreas específicas de foco e sugerindo medidas para otimizar a eficiência, funcionalidade e abrangência da plataforma OHWR.

Tabela 11 - Conclusões da Reunião com a Comunidade OHWR.

Área de Foco	Conclusões e Recomendações
Automação de Processos	É imperativo implementar ferramentas e fluxos de trabalho automatizados para otimizar a criação de projetos, atribuição de funções e gestão de permissões, minimizando a necessidade de intervenções manuais e potencializando a eficiência da plataforma.
Comunicação e Colaboração	Recomenda-se a introdução de recursos avançados que facilitem a interação entre membros, promovendo discussões estruturadas e fortalecendo a colaboração, cultivando assim um ambiente mais inclusivo e participativo.
Diversidade de Membros	A plataforma deveria expandir sua diversidade, abrindo-se para áreas além dos Laboratórios de Física, incentivando a participação de membros de diferentes campos e especializações.
Monitoramento de Desempenho	É essencial implementar métricas robustas para monitorar a atividade e contribuição dos membros, fornecendo insights valiosos para avaliar a eficácia dos projetos e incentivar a colaboração contínua.
Incorporação de Práticas Estabelecidas	Sugere-se adotar e integrar práticas e recursos avançados de plataformas consolidadas, como o GitHub, para aprimorar a revisão, avaliação e gestão de projetos.
Potencial de Crescimento	Fortalecer o papel do OHWR como ambiente colaborativo para projetos de open hardware e software.

3.5. Dados Extraídos das Plataformas

Antes de explorar as técnicas automatizadas de extração de dados, é imperativo compreender o que pode ser identificado manualmente nas plataformas. A extração manual, embora mais morosa e menos abrangente do que os métodos automatizados, oferece uma visão única e detalhada dos dados. Esta abordagem permite uma análise mais contextualizada, que muitas vezes é essencial para compreender as particularidades de cada projeto.

Mediante uma análise observacional dos projetos em distintas plataformas, como o GitHub e OHWR, é possível extrair um conjunto de métricas que proporcionam perspetivas valiosas sobre a atividade, popularidade e colaboração associadas a cada projeto. Esta abordagem observacional serve como um referencial para entender o tipo de informação acessível e estabelecer um parâmetro comparativo quando se recorre a métodos automatizados, tais como scripts em Python, para a extração de dados em grande magnitude. A Tabela apresenta uma comparação de métricas extraídas manualmente de repositórios em diferentes plataformas.

Tabela 12 - Comparação de Métricas extraídas de Repositórios nas plataformas GitHub e OHWR.

	GitHub			**OHWR**		
Métricas/ Indicadores	Projetos					
	google / opength	mautic /mautic	Home-assistant/core	Hdlmake	White Rabbit	Platform-independent core collection
Nº de Contribuidores	46	312	3337	23	38	37
Nº de *Commits*	805[9]	18669[10]	47780[11]	1393	211	1083
Nº de *Branches*	25	27	160	22	9	67
Nº de Estrelas	437	6000	62889	9	2	5
Nº de *Forks*	211	2200	24676	×	×	×
Nº de *Releases*	14	141	1153	×	×	×
Nº de *Watches*	46	302	1356	×	×	×
Nº de linguagens de programação usadas	5	6	1	8	7	12
Nº de *Open Issues*	104	155	2125	14	5	10

[9] No GitHub, apenas é possível observar o número de *commits* dos contribuidores principais. Neste projeto, foi possível auferir acerca dos 45 principais contribuidores.
[10] No GitHub, apenas é possível observar o número de *commits* dos contribuidores principais. Neste projeto, foi possível auferir acerca dos 99 principais contribuidores.
[11] No GitHub, apenas é possível observar o número de *commits* dos contribuidores principais. Neste projeto, foi possível auferir acerca dos 44 principais contribuidores.

Tabela 12 - Comparação de Métricas extraídas de Repositórios nas plataformas GitHub e OHWR.

	GitHub			**OHWR**		
Métricas/ Indicadores	Projetos					
	google / opength	mautic /mautic	Home-assistant/core	Hdlmake	White Rabbit	Platform-independent core collection
Nº de *Closed Issues*	316	6638	39368	104	30	30
Nº de *Open Pull Requests*	39	267	515	2	0	4
Nº *de Closed Pull Requests*	645	5567	57706	3	0	14
Nº de *Merged Pull Requests*	523	4486	49726	21	0	25
Nº de Repositórios Dependentes do Projeto	13	8	1813	×	×	×

Ao analisar-se as métricas disponibilizadas nas plataformas GitHub e OHWR, destaca-se uma diferença notável na abrangência e detalhe de informações proporcionadas por cada uma.

O GitHub fornece uma visão abrangente dos projetos, permitindo aos utilizadores e *developers* avaliar a atividade do projeto, evidenciada pelo número de *commits*, *branches* e *releases*. Adicionalmente, é possível discernir o grau de envolvimento da comunidade através de indicadores como o número de *estrelas*, *forks* e *watches*. O número de repositórios que dependem de um determinado projeto serve como um indicativo do seu impacto e relevância na comunidade alargada.

Em contrapartida, o OHWR apresenta um leque de métricas mais circunscrito. Embora forneça informações como o número de estrelas, carece de métricas como o número de *releases*, número de *forks* e número de *watches*, o que pode limitar a capacidade de avaliar de forma integral a evolução ou a popularidade de um projeto.

Enquanto no GitHub a ênfase recai sobre o número total de contribuidores, o OHWR, a distribuição funcional dos contribuidores é evidente. No projeto Hdlmake, os 23 contribuidores dividem-se em: 2 *Owners*, 2 *Maintainers*, 18 *Developers* e 1 *Reporter*. O projeto White Rabbit conta com 38 contribuidores, sendo: 1 *Owner*, 5 *Maintainers* e 31 *Developers*, além de 1 *Reporter*. Já o projeto Platform-independent core collection, composto por 37 contribuidores, apresenta 2 *Owners*, 6 *Maintainers* e 29 *Developers*, sem a presença de *Reporters*.

Esta perspetiva permite uma compreensão mais aprofundada da estrutura e hierarquia da comunidade de contribuidores. Contudo, deve-se salientar que a recolha de dados foi efetuada com permissões ao nível de utilizador, no GitHub, o que pode ter ocultado algumas métricas específicas relacionadas às funções dos contribuidores.

Contudo, ambas as plataformas fornecem métricas sobre *issues* e *pull requests*, permitindo aos utilizadores avaliar a atividade em termos de gestão de problemas e contribuições.

3.6. Recolha de Dados

Neste capítulo delineia-se o processo metodológico subjacente à recolha da informação que fundamenta o presente estudo. Com um compromisso firme com a transparência e o rigor, delinear-se-ão os critérios adotados e as ferramentas selecionadas que moldaram o processo de aquisição de dados. Os subcapítulos subsequentes aprofundarão a descrição deste procedimento, elucidando em detalhe cada métrica recolhida. Assim, pretende-se proporcionar ao leitor uma visão integral e aprofundada da base empírica que sustenta a análise.

3.6.1. Descrição do Processo

Tendo o GitHub como fonte de dados, é necessário definir como extrair os dados e trazê-los para o ambiente de trabalho. A extração de dados de plataformas como o GitHub, constitui uma tarefa complexa que requer uma combinação de ferramentas e técnicas para garantir a precisão e a integridade dos dados coletados. O GitHub, sendo uma das principais plataformas de alojamento de código-fonte, disponibiliza uma vasta quantidade de informações que são de grande valor para investigadores e *developers*. No entanto, a extração destes dados apresenta vários desafios.

Inicialmente, é imperativo mencionar a existência de ferramentas meticulosamente desenvolvidas com o propósito específico de extrair dados do GitHub. O GHTorrent serve para fornecer um espelho offline dos dados do GitHub. Para além do GHTorrent, existe o GHArchive, sendo que estes dois são mais apropriados para recuperar dados históricos. Por outro lado, o GitHub REST API é mais indicado para aceder a dados mais recentes e atualizados (Xia et al., 2022).

No entanto, a API do GitHub, apesar de ser uma fonte valiosa de informações atualizadas, tem as suas limitações na versão gratuita aberta ao público. Um das principais é o seu limite de solicitações, que atua como uma barreira para obter dados do GitHub. As limitações autoimpostas do GitHub ao uso da sua API dificultam o processo de coleta de dados necessário para realizar a recolha de dados em larga escala. Uma estratégia frequentemente usada para contornar esta limitação é a utilização de de *tokens* de acesso de diferentes utilizadores para coletar as informações necessárias. Um *token* de acesso permite interações automatizadas com a API, ao mesmo tempo que protege os dados e controla o acesso. Contudo, para estudos de grande dimensão, a necessidade de *tokens* é ampliada. Uma solução mais completa seria o GitHub rever as suas políticas de limitação, aumentando o limite de solicitação de API para projetos, se necessário, uma aprovação prévia, ou facilitar a partilha de *tokens* para fins académicos (Cosentino et al., 2017).

Além disso, a API de eventos do GitHub é uma ferramenta valiosa que define vários tipos de eventos, abrangendo uma ampla gama de atividades dos utilizadores na plataforma, desde realizar um *fork* de um projeto para colaborar em *issues* ou pull *requests*. No entanto, embora da API de eventos do GitHub seja bem documentada e projetada, o limite de taxa da API, que é de 5 mil solicitações por hora, é uma restrição que dificulta a realização de explorações em grande escala (Xia et al., 2022).

Apesar de 5 mil solicitações por hora, possa parecer generoso, é uma restrição que dificulta a extração de dados em grande escala. Por exemplo, o repositório "kicad-source-mirror" no GitHub, que é o espelho do código-fonte do KiCad. Este repositório conta com mais de 1.000 estrelas, cerca de 500 forks e mais de 200 contribuidores. Ao tentar extrair informações detalhadas sobre cada *fork* ou sobre a atividade de cada contribuidor, o número de solicitações à API pode acumular-se rapidamente. Em repositórios desta amplitude, torna-se evidente que o limite de 5 mil solicitações por hora pode ser um desafio, especialmente quando se pretende realizar uma análise abrangente em curtos períodos.

A API do GitHub e o GHArchive, são utilizados nesta dissertação como fontes primordiais de dados a serem minerados. A constituição da base de dados através da API Rest realiza-se pela extração e subsequente transferência das informações para uma base de dados. A API Rest foi acedida através de Python, ao passo que o GHArchive disponibiliza uma opção para consultar uma versão online da base alojada no Google BigQuery. Neste trabalho, optou-se pela utilização da versão da base no Google BigQuery, que alberga pormenores sobre os projetos.

Ao recorrer à API Rest oficial do GitHub, é possível aceder a informações que englobam todo o histórico da plataforma. Contudo, essa abrangência não se verifica com o GitHub Archive. O GHArchive regista eventos desde 12/02/2011. Assim, apesar de conter uma vasta quantidade de dados, não abarca todo o histórico de eventos do GitHub. Em síntese, o GitHub Archive visa armazenar uma fração dos dados disponíveis no GitHub, facilitando o acesso a estes sem a imposição de um limite de consultas.

O GitHub Archive organiza os dados de eventos ocorridos no GitHub em ficheiros segmentados por hora, consoante o momento em que se deram. Deste modo, ao utilizar o GitHub Archive, é exequível analisar a evolução dos dados e replicar um estudo com o mesmo conjunto de informações.

Para uma compreensão mais clara e comparativa das fontes de dados utilizadas na dissertação, apresenta-se a Tabela 13 que contrasta as características e especificidades da GitHub REST API V3, do GHArchive e do GHTorrent. Esta tabela permite uma visão rápida e concisa das diferenças e semelhanças entre estas fontes, facilitando a compreensão das escolhas feitas ao longo desta dissertação.

Tabela 13 - Análise do GitHub REST API V3, GitHub Archive e GHTorrent. Adaptado de (Mombach & Valente, 2018)

	GitHub REST API V3	**GitHub Archive**	**GHTorrent**
Dados Desde	-	12/02/2011	2012
Origem dos Dados	-	GitHub REST API Events	GitHub REST API
Tipo de Dados	Dados do GitHub	Eventos do GitHub	Dados Estruturados do GitHub
Frequência	Em tempo real	Hora a hora	Mensal

3.6.2. Recolha dos Dados das Métricas

Nesta dissertação, a metodologia de recolha de dados foi criteriosamente delineada para assegurar a integridade e a relevância dos dados obtidos. A fundamentação para a recolha baseou-se nas métricas discutidas no Capítulo 3.1, onde se realizou uma avaliação minuciosa para identificar quais

métricas eram passíveis de extração através da API do GitHub e do GHArchive. Este critério de seleção foi imprescindível para estabelecer o âmbito dos dados, garantindo assim a pertinência dos mesmos para a análise proposta.

No desenvolvimento subsequente da metodologia, o processo de recolha de dados foi estruturado em diversas fases essenciais:

- **Recolha de Dados**: Após a identificação das métricas relevantes, procedeu-se à fase de extração dos dados, recorrendo-se às ferramentas já mencionadas: Python e BigQuery.
- **Análise**: Uma vez obtidos os dados, realizou-se um exame aprofundado para discernir possíveis inter-relações entre as métricas, identificando assim quais poderiam ser analisadas em conjunto ou de forma comparativa.
- **Preparação para Análises Futuras**: Os dados, uma vez recolhidos e examinados, foram organizados e preparados para análises subsequentes, estabelecendo o caminho para avaliações mais detalhadas nos capítulos subsequentes da dissertação.

No contexto da presente dissertação, foi imperativo compreender a dinâmica e as particularidades dos repositórios *Open Source* no GitHub. Nesta perspetiva, procedeu-se à seleção criteriosa de dois repositórios distintos: um representante de *Open Source Software* e outro de *Open Source Hardware*, sendo estes o "home-assistant/core" e o "fossasia/pslab-hardware", respetivamente. Esta escolha não foi aleatória e responde à necessidade de explorar duas facetas complementares do universo *Open Source*.

Por um lado, o repositório de *Open Source Software* foi selecionado para analisar o cenário digital e *online*. O *software*, pela sua natureza intangível, proporciona *insights* sobre colaborações, modificações e distribuições num ambiente estritamente digital. A análise deste repositório permitiu uma análise aprofundada das métricas que caracterizam a colaboração *online*, as contribuições dos *developers* e a dinâmica do ecossistema digital em que o *Open Source Software* prospera.

Em contrapartida, a escolha de um projeto de *Open Source Hardware* alcança uma dimensão tangível. Ao contrário do *software*, o *hardware* tem uma presença física, onde o *design*, os esquemas e as implementações físicas têm implicações diretas no mundo real. Investigar um projeto deste tipo oferece uma perspetiva sobre como as ideias são transpostas para objetos físicos e como a comunidade colabora para melhorar, adaptar e distribuir *designs* de *hardware*.

Esta dualidade proporciona uma visão holística dos desafios, oportunidades e particularidades dos movimentos de código aberto, tanto no plano digital quanto no mundo real. Através desta abordagem, procura-se compreender não apenas as métricas que definem o sucesso e a atividade desses projetos, mas também as características intrínsecas que diferenciam o *Open Source Software* do *Open Source Hardware*.

Ao analisar as métricas de sucesso nas comunidades de código aberto, conforme detalhado no Capítulo 3.1, constata-se uma diversidade de indicadores que podem ser considerados. Para a recolha destas métricas, foram empregues duas metodologias distintas: a utilização da API do GitHub com recurso ao Python e o recurso ao GHArchive com o auxílio da ferramenta BigQuery.

Através da API do GitHub e recorrendo ao Python, conseguiu-se extrair a nível mensal:

- Número de *Releases*;
- Número de *Forks*.

Por sua vez, com o GHArchive, através da ferramenta BigQuery, obteve-se, também a nível mensal:

- Número de *Commits* e *Pushes*;
- Número de Estrelas;
- Número de Issues (Totais, Abertas e Fechadas);
- Número de Pull Requests (Totais, Abertas, Fechadas e *Merged*);
- Tempo Médio de Resposta/Fecho *Issue*;
- Tempo Médio *Pull Request*.

Para facilitar a compreensão da diversidade e abrangência das métricas extraídas, apresenta-se na Figura 48, uma representação esquemática que sintetiza as métricas que foram tidas em conta na extração de dados das métricas através das ferramentas utilizadas, dos respetivos. Esta representação visa elucidar visualmente as categorias de dados recolhidos.

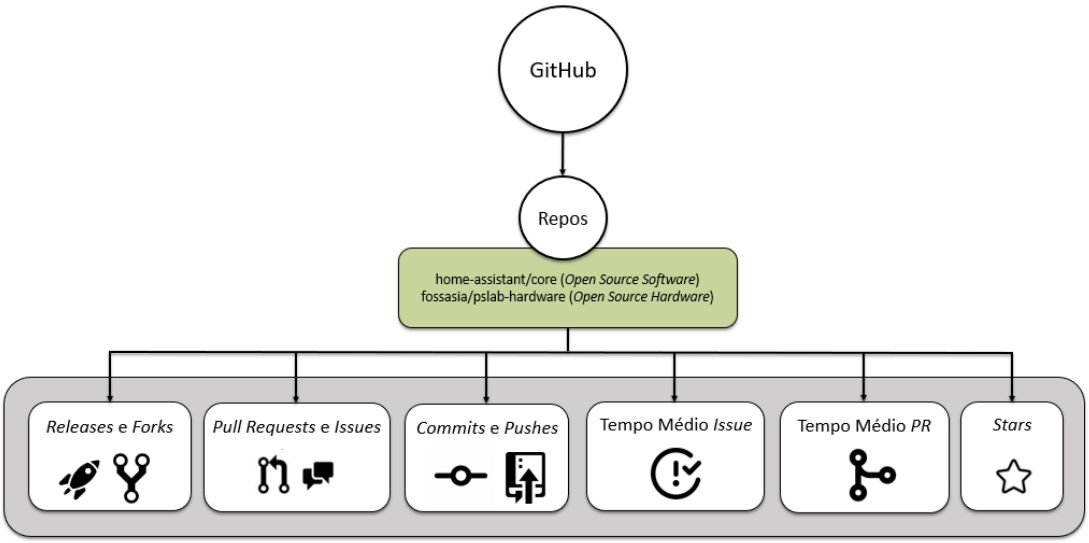


Figura 48 - Representação Esquemática das Métricas para Análise dos Projetos Selecionados do GitHub.

A utilização da API do Github na extração de métricas requer uma abordagem estratégica e cuidadosa.

Salienta-se a imperatividade de um *token* de acesso pessoal para efetuar requisições à API do GitHub. Este *token* não só assegura que as requisições sejam autenticadas, como também proporciona um limite de acesso ampliado. Deste modo, torna-se viável extrair uma quantidade substancial de dados num intervalo temporal reduzido. Para além disso, o recurso ao *token* aprimora a segurança e personalização do acesso aos dados.

Os *scripts* foram criteriosamente concebidos visando a automação e a capacidade de repetir o processo de extração com fiabilidade. Uma característica distintiva reside na sua versatilidade. Com ajustes pontuais, os *scripts* podem ser utilizados para recolher dados de variados projetos no GitHub, assegurando, deste modo, uma metodologia uniforme e replicável de recolha de métricas.

A construção modular do código possibilita que a análise de diferentes repositórios ocorra mediante a simples modificação da variável "repo", garantindo uma abordagem sistemática na recolha de informação.

O mesmo se aplica para a construção das consultas SQL. Estas são adaptáveis, bastando modificar o ID do repositório para obter dos dados das métricas em análise para qualquer projeto hospedado no GitHub.

Os dados, de ambas as ferramentas, foram extraídos desde a data de criação dos repositórios até 31-08-2023.

- **Métrica: Número de *Releases***

Na sequência da recolha de dados, abordou-se a métrica relativa à frequência de lançamentos (*releases*) dos projetos no GitHub. A frequência de lançamentos proporciona uma visão sobre a regularidade das atualizações, a rapidez com que os *bugs* são corrigidos e a atitude geral da equipa de desenvolvimento em relação à manutenção do projeto.

O objetivo desta métrica é avaliar a frequência e a consistência das publicações de *releases*. Uma elevada frequência pode indicar uma equipa ativa e uma comunidade envolvida, enquanto longos intervalos entre *releases* podem sugerir uma evolução mais lenta ou períodos de inatividade.

No *script* desenvolvido em Python, recorre-se à função ´get_releases` para extrair todos os lançamentos associados ao repositório. Esta função contempla a utilização da paginação da API, tendo em conta que o GitHub estabelece restrições quanto ao número de resultados por requisição. Assim, são realizadas múltiplas solicitações, cada uma direcionada para uma página distinta de resultados, até que a totalidade das páginas seja processada.

Após a obtenção dos *releases*, os dados são organizados num *DataFrame* e categorizados por mês. Os resultados são armazenados, posteriormente, num ficheiro Excel.

A Tabela 14 elucida os principais componentes do *script* utilizado para extrair e analisar a métrica de *releases* do repositório em estudo. Cada componente desempenha uma função específica, assegurando que a extração, processamento e exportação dos dados sejam realizados de forma metódica e eficaz.

Tabela 14 - Processo de Extração e Análise dos *Releases* do GitHub.

Componente	Descrição
Função de Extração	A função get_releases estabelece a interação com a API do GitHub, recuperando todos os *releases* dos projetos.
Segmentação Temporal	O *script* segmenta os *releases* ao longo dos meses, facilitando a interpretação.
Exportação de Dados	Os dados dos *releases* são armazenados num ficheiro Excel para análise posterior.

No contexto da extração de métricas do GitHub, é essencial compreender como os dados são adquiridos. A função ´get_releases`, apresentada na Figura 49, exemplifica a abordagem utilizada para interagir com a API do GitHub e recolher as informações sobre os *releases* dos repositórios em estudo. Esta função é a base do processo de análise, garantindo que os dados necessários sejam recuperados de forma eficaz e completa.

```python
def get_releases(repo):
    releases = []
    page = 1
    while True:
        url = f"https://api.github.com/repos/{repo}/releases?per_page=100&page={page}"
        response = requests.get(url, headers=HEADERS)
        data = response.json()
        if not data:
            break
        releases.extend(data)
        page += 1
    return releases
```

Figura 49 - Função ´get_releases` para extração de *releases* via API do GitHub.

No âmbito da avaliação de projetos hospedados no GitHub, a Tabela 15 sumariza a abordagem adotada para avaliar a métrica de *releases*. Esta detalha os objetivos dessa métrica, os agregadores utilizados para uma análise mais profunda e as potenciais visualizações que podem ser derivadas a partir dos dados coletados.

Tabela 15 - Descrição do Processo de Avaliação da Métrica de *Releases*

Métricas	Descrição
Objetivos	Avaliação da frequência de *releases*, permitindo compreender a regularidade das atualizações.
Agregadores	Categorização dos *releases* a nível mensal, permitindo uma análise temporal mais estruturada.
Visualizações	Gráficos temporais do número de lançamentos, proporcionado uma visão detalhada do comportamento dessa métrica ao longo do tempo.

- **Métrica: Número de *Forks* a Nível Temporal**

Esta métrica tem como propósito principal discernir o envolvimento e adesão da comunidade ao projeto.

Através do *script* desenvolvido em Python, utilizou-se a função ´get_forks` para extrair todos os *forks* dos repositórios em análise. Esta função incorpora a paginação da API do GitHub, dado que existem restrições quanto ao volume de resultados por pedido. Desta forma, são efetuadas várias solicitações, cada qual dirigida a uma página distinta, até que todo o conjunto de dados seja obtido.

Posteriormente à recolha dos *forks*, os dados são estruturados num DataFrame, sendo depois categorizados mensalmente. A conclusão deste processo culmina no armazenamento dos resultados num ficheiro Excel.

A Tabela 16 descreve, de forma sucinta, os principais componentes do *script* empregue para a extração e análise da métrica de *forks* dos repositórios selecionados.

Tabela 16 - Procedimento de Extração e Análise dos *Forks* do repositório em estudo.

Componente	Descrição
Função de Extração	A função ´get_forks` permite a recolha dos *forks* do projeto a partir da API do GitHub.
Segmentação Temporal	Os dados são organizados por mês, facilitando a sua posterior interpretação e análise.
Exportação de Dados	A informação é armazenada num ficheiro Excel para futuras consultas e análises.

No contexto da extração de métricas do GitHub, é crucial entender a metodologia de aquisição dos dados. A função ´get_forks`, ilustrada na Figura 50 , demonstra a técnica utilizada para aceder à API do GitHub e recolher os *forks* dos repositórios selecionado.

```
def get_forks(repo):
    forks = []
    page = 1

    while True:
        url = f"https://api.github.com/repos/{repo}/forks?per_page=100&page={page}"
        response = requests.get(url, headers=HEADERS)
        data = response.json()
        if not data:
            break
        forks.extend(data)
        page += 1
    return forks

repo = "home-assistant/core"
```

Figura 50 - Função ´get_forks` empregue na recolha de forks via API do GitHub.

Ao analisar projetos no GitHub, a Tabela 17 sintetiza a metodologia utilizada para aferir a métrica de *forks*, detalhando os seus objetivos, os mecanismos de agregação dos dados e as potenciais representações gráficas derivadas.

Tabela 17 - Descrição do Procedimento de Análise da Métrica de Forks.

Métricas	**Descrição**
Objetivos	Quantificar os *forks* para perceber o envolvimento da comunidade.
Agregadores	Os *forks* são organizados mensalmente, permitindo uma análise ao longo do tempo.
Visualizações	Representações gráficas que exibem a evolução temporal do número de *forks* podem ser elaboradas com base nos dados recolhidos.

- **Métrica: Número de *Commits* e *Pushes***

Na continuidade da análise das métricas dos repositórios em estudo, procedeu-se à extração e avaliação da métrica de número *pushes* e *commits* em intervalos temporais específicos, empregando o BigQuery como ferramenta de análise de dados. Esta métrica oferece uma visão crucial sobre a regularidade das contribuições e a intensidade da atividade de desenvolvimento ao longo do tempo. O principal objetivo é avaliar como a comunidade de *developers* está a contribuir para o projeto, bem como compreender qualquer tendência de mudança nesse comportamento.

O processo de extração de dados pode ser resumido da seguinte forma:

1) Subconsultas YearDat e DayData: Utilizou-se duas subconsultas para extrair os dados de interesse. A subconsulta YearData recolhe informações de *commits* e *pushes* a nível mensal, abrangendo todos os anos disponíveis. Enquanto isso, a subconsulta DayData foca-se especificamente no ano de 2023, com um filtro temporal definido até agosto.
2) Consulta Principal: Os resultados das subconsultas YearData e DayData são combinados numa consulta principal que agrega os dados por mês. Esta consulta final oferece uma visão completa da métrica de *commits* e *pushes* ao longo do tempo.

Os resultados da consulta SQL são visualizados numa tabela, no BigQuery, que contém informações detalhadas sobre os dados obtidos da métrica ao longo do tempo. Posteriormente, os dados são exportados para um arquivo CSV e analisados num arquivo Excel

Na Tabela 18, apresenta-se uma visão geral das etapas envolvidas no processo de extração das métricas de *commits* e *pushes* do projeto. Cada componente desempenha um papel específico na obtenção e processamento dos dados para uma análise detalhada.

Tabela 18 - Processo de Extração e Análise de *Commits* e *Pushes* dos repositórios em estudo.

Componente	Descrição
Subconsulta YearData	A subconsulta YearData extrai dados mensais de *commits* e *pushes* abrangendo todos os anos disponíveis.
Subconsulta DayData	A subconsulta DayData extrai dados mensais de *commits* e *pushes* para o ano de 2023.
Consulta Principal	Os resultados das subconsultas são unidos e agregados por mês.

A Figura 51, ilustra a função de extração de dados utilizada para obter as informações necessárias para a análise da métrica. A função de extração representada é a base do processo de análise, garantindo que os dados necessários são recuperados de forma eficaz e completa.

```
1  WITH YearData AS (
2      SELECT
3          FORMAT_TIMESTAMP('%Y-%m-01', TIMESTAMP(created_at)) as event_month,
4          COUNT(*) AS pushes,
5          SUM(ARRAY_LENGTH(JSON_EXTRACT_ARRAY(payload, '$.commits'))) AS commits
6      FROM `githubarchive.year.*`
7      WHERE repo.id = 86472815 and type = 'PushEvent'
8      GROUP BY event_month
9  ),
10 DayData AS (
11     SELECT
12         FORMAT_TIMESTAMP('%Y-%m-01', TIMESTAMP(created_at)) as event_month,
13         COUNT(*) AS pushes,
14         SUM(ARRAY_LENGTH(JSON_EXTRACT_ARRAY(payload, '$.commits'))) AS commits
15     FROM `githubarchive.day.2023*`
16     WHERE repo.id = 86472815 and type = 'PushEvent'
17     AND created_at <= '2023-08-31'
18     GROUP BY event_month
```

Figura 51 - Consulta SQL para Extração de Métricas de *Commits* e *Pushes*.

A Tabela 19 fornece informações sobre a avaliação da métrica no contexto da análise temporal. Apresentam-se os objetivos, agregadores e visualizações relacionados a essa métrica, destacando a sua importância na compreensão do projeto.

Tabela 19 - Descrição do Processo de Avaliação da Métrica de *Commits* e *Pushes.*

Métricas	Descrição
Objetivos	Avaliação da frequência de *commits* e *pushes* para compreender a regularidade das contribuições.
Agregadores	Categorização das contribuições a nível mensal, permitindo uma análise temporal mais detalhada.
Visualizações	Gráficos temporais do número de *commits* e *pushes*, proporcionando uma visão detalhada da evolução dessas métricas ao longo do tempo.

- **Métrica: Número de *Issues***

No seguimento da análise das métricas, foca-se agora na métrica que contempla o número total de *issues*, bem como as *issues* abertas e fechadas em intervalos temporais específicos, utilizando o BigQuery como instrumento de análise de dados. Esta métrica fornece uma perspetiva aprofundada acerca da gestão e resolução de problemas no projeto, servindo como um indicador fundamental da saúde e eficácia do desenvolvimento. O objetivo principal é perceber a frequência com que novos problemas são identificados, bem como a eficácia com que são resolvidos, e ainda obter uma visão geral do volume total de *issues*.

O procedimento de extração dos dados desenvolveu-se da seguinte forma:

1) União dos Dados: A consulta combina informações de tabelas anuais e diárias do GHArchive, abrangendo todos os anos disponíveis até 31 de agosto de 2023;
2) Extração de Informação: Através das funções EXTRACT e JSON_EXTRACT_SCALAR, são retiradas informações essenciais como o tipo de evento, ação associada à *issue*, e a data de criação;
3) Filtragem: São considerados apenas eventos do tipo 'IssuesEvent' para os repositórios;
4) Agregação: Os dados são agrupados por mês, fornecendo uma análise temporal clara e organizada.

Após a execução da consulta, os resultados são apresentados numa tabela no BigQuery. A informação é, posteriormente, exportada para um ficheiro CSV para análise mais detalhada no Excel.

Na Tabela 20, descreve-se o processo envolvido na extração das métricas de *issues* do projeto. Cada componente tem um papel determinante na recolha e tratamento dos dados para uma análise minuciosa.

Tabela 20 - Procedimento de Extração e Análise de *Issues* dos repositórios em análise.

Componente	Descrição
Dados Anuais e Diários	Recolha de informações sobre *issues* em tabelas anuais e diárias.
Extração e Filtragem	Uso de funções específicas para extrair e filtrar os dados relevantes.
Agregação	Organização dos dados por mês, proporcionando uma análise temporal.

A Figura 52 ilustra a consulta SQL desenvolvida com o objetivo de extrair informações sobre as *issues* do repositório alvo. Esta consulta permite discernir, de forma mensal e anual, a quantidade de *issues* criadas, as que permanecem abertas e as que foram encerradas. A análise deste tipo de métrica oferece uma visão clara da atividade e dinâmica dos repositórios em questão, bem como da interação da comunidade com os mesmos.

```sql
SELECT
  EXTRACT(YEAR FROM created_at) as year,
  EXTRACT(MONTH FROM created_at) as month,
  COUNT(DISTINCT CASE WHEN event_type='IssuesEvent' THEN issue_number END) AS total_issues,
  COUNTIF(action='opened') AS open_issues,
  COUNTIF(action='closed') AS closed_issues,
FROM (
  SELECT
    type AS event_type,
    JSON_EXTRACT_SCALAR(payload, '$.action') AS action,
    actor.id AS actor_id,
    JSON_EXTRACT_SCALAR(payload, '$.issue.user.id') AS user_id,
    JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS issue_number,
    created_at
  FROM `githubarchive.year.*`
  WHERE repo.id = 12888993
  AND type IN ('IssuesEvent')
```

Figura 52 - Segmento da Consulta SQL para Extração do Número de Issues (Total, Abertas e Fechadas).

A Tabela 21 apresenta a metodologia adotada para avaliar esta métrica num contexto temporal. Enumeram-se os objetivos, agregadores e visualizações associados a esta métrica, sublinhando a sua relevância na avaliação dos repositórios.

Tabela 21 - Descrição do Procedimento de Avaliação da Métrica de *Issues* Totais, Abertas e Fechadas.

Métricas	**Descrição**
Objetivos	Avaliação da frequência e volume de *issues* para compreender a gestão e resolução de problemas no projeto.
Agregadores	Categorização das *issues* ao nível mensal, possibilitando uma análise temporal mais precisa.
Visualizações	Representações gráficas temporais que ilustram a evolução do número total, abertas e fechadas de *issues* ao longo do tempo.

- **Métrica: Número de *Pull Requests***

No âmbito da análise das métricas do projeto, surge agora a necessidade de compreender a atividade em torno dos *pull requests*. O recurso ao BigQuery proporciona uma visão detalhada sobre a criação, fecho e *merged* das *pull requests*.

As características principais da extração de dados através da consulta SQL, pode ser resumida da seguinte forma:

1) Extração de Dados: A consulta congrega dados das tabelas anuais e diárias do GHArchive, considerando todos os anos em registo até 31 de agosto de 2023;
2) Funções de Extração: Utilizam-se funções como EXTRACT e JSON_EXTRACT_SCALAR para colher informações essenciais, como o tipo de evento, a ação associada ao *pull request*, e os dados temporais;
3) Filtragem de Dados: A consulta centra-se em eventos do tipo 'PullRequestEvent' e para um repositório em estudo;
4) Agregação de Dados: Os resultados são agrupados por mês, garantindo uma análise estruturada e temporal.

Os dados resultantes da consulta, são os seguintes:

– Número total de *pull requests* em cada mês.

- Número de *pull requests* abertos em cada mês.
- Número de *pull requests* fechados em cada mês;
- Total de *pull requests merged* em cada mês.

Na Tabela 22 , descreve-se o procedimento adotado para a extração e análise da métrica associada aos *pull requests*.

Tabela 22 - Procedimento de Extração e Análise da Atividade Relacionada com *Pull Requests*.

Componente	Descrição
Extração de Dados	Recolha de informações das tabelas anuais e diárias.
Funções de Extração	Aplicação de funções específicas para a recolha de dados relevantes.
Filtragem de Dados	Foco em eventos associados a *pull requests para os* repositórios especificados.
Agregação de Dados	Organização dos resultados por mês, permitindo uma análise temporal estruturada.

A Figura 53 destaca uma seção vital da consulta SQL, focando nos contadores que resumem as ações realizadas nos *pull requests*. Estes contadores proporcionam uma visão clara da dinâmica dos contribuintes no processo de revisão e fusão de código no repositório.

```
27
28  SELECT
29    year,
30    month,
31    COUNT(*) AS total_prs,
32    COUNTIF(action='opened') AS open_prs,
33    COUNTIF(action='closed') AS closed_prs,
34    COUNTIF(merged='true') AS merged_prs,
35  FROM all_data
36  GROUP BY year, month
37  ORDER BY year, month;
```

Figura 53 - Segmento da consulta SQL: Resumo da atividade dos *pull requests*, incluindo o total, os abertos, os fechados e os *merged*.

A Tabela 23 apresenta a metodologia estabelecida para avaliar esta métrica em contexto temporal.

Tabela 23 - Descrição do Procedimento de Avaliação da Métrica de *Pull Requests*.

Métricas	Descrição
Objetivos	Compreensão da atividade e gestão dos *pull requests*, incluindo a sua criação, fecho e *merged*.
Agregadores	Classificação dos *pull requests* ao nível mensal, possibilitando uma análise temporal precisa.
Visualizações	Representações gráficas temporais que ilustram a atividade dos *pull requests* ao longo do tempo, destacando tendências e padrões de comportamento.

- **Métrica: Tempo Médio *Pull Request***

No prosseguimento da avaliação das métricas associadas aos repositórios, destaca-se a análise do tempo médio despendido desde a criação até um *pull request* ser *merged*. Este indicador fornece *insights* sobre a eficiência do processo de revisão e integração de código, sendo uma métrica crucial para avaliar a agilidade da equipa de desenvolvimento.

A implementação desta consulta SQL foi estruturada em várias etapas essenciais, cada uma desempenhando um papel específico no processamento e análise dos dados. As etapas fundamentais são as seguintes:

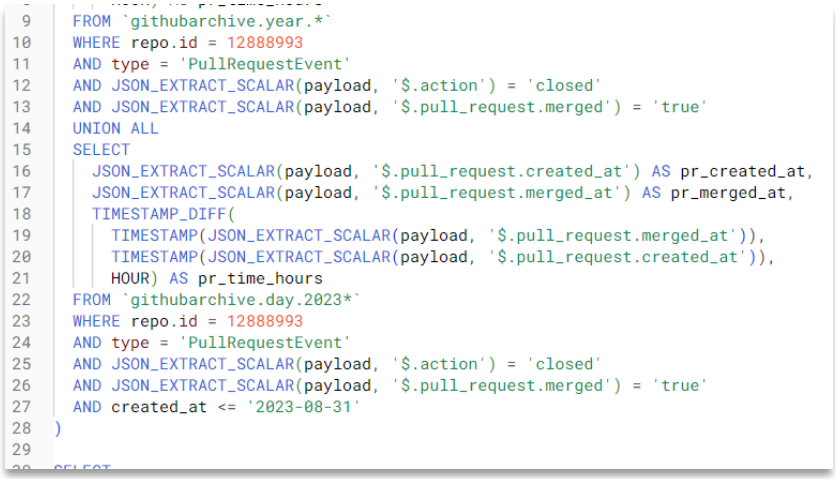
1) Recolha de Dados: A consulta visa reunir informações das tabelas anuais e diárias do GHArchive, abrangendo todos os anos documentados até ao final de agosto de 2023.
2) Filtragem de Eventos: A análise incide sobre eventos do tipo 'PullRequestEvent' que foram fechados e cujos *pull requests* foram *merged*.
3) Cálculo de Duração: Com recurso às funções TIMESTAMP_DIFF e TIMESTAMP, calcula-se a diferença temporal entre a criação e a integração do *pull request*, resultando no tempo despendido em horas.
4) Agregação Temporal: Os dados são, posteriormente, agregados por mês, resultando no cálculo do tempo médio despendido em *pull requests* para cada período.

Na Tabela 24, é delineado o procedimento adotado para a extração e análise da métrica associada ao tempo de *pull requests*.

Tabela 24 - Abordagem de Extração e Análise do Tempo Médio de *Pull Requests*.

Componente	Descrição
Recolha de Dados	Extração de dados das tabelas anuais e diárias.
Filtragem de Eventos	Concentração em eventos específicos relacionados com a conclusão e *merge* de *pull requests*.
Cálculo de Duração	Estimativa do tempo despendido em cada *pull request* desde a sua criação até à sua fusão.
Agregação Temporal	Organização dos dados por mês, fornecendo uma visão média do tempo despendido por período.

Na Figura 54, é ilustrado um segmento crucial da consulta SQL, onde se destaca o cálculo do intervalo temporal entre a criação e a fusão de um pull request. Esta métrica é vital para entender a eficiência com que as contribuições são revistas e integradas ao repositório principal.



```
  FROM `githubarchive.year.*`
  WHERE repo.id = 12888993
  AND type = 'PullRequestEvent'
  AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'closed'
  AND JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged') = 'true'
  UNION ALL
  SELECT
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.created_at') AS pr_created_at,
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged_at') AS pr_merged_at,
    TIMESTAMP_DIFF(
      TIMESTAMP(JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged_at')),
      TIMESTAMP(JSON_EXTRACT_SCALAR(payload, '$.pull_request.created_at')),
      HOUR) AS pr_time_hours
  FROM `githubarchive.day.2023*`
  WHERE repo.id = 12888993
  AND type = 'PullRequestEvent'
  AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'closed'
  AND JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged') = 'true'
  AND created_at <= '2023-08-31'
)
```

Figura 54 - Segmento da Consulta SQL: Representação do cálculo da diferença de tempo entre a criação e a fusão de pull requests.

A Tabela 25 detalha a metodologia estabelecida para avaliar esta métrica em contexto temporal.

Tabela 25 - Descrição da Avaliação da Métrica do Tempo Médio de *Pull Requests*.

Métricas	**Descrição**
Objetivos	Analisar o tempo médio necessário desde a submissão até à integração de um *pull request*.
Agregadores	Estruturação dos dados ao nível mensal, proporcionando uma perspetiva temporal do tempo médio despendido.
Visualizações	Representações gráficas temporais que elucidam o tempo médio despendido em *pull requests* ao longo dos meses.

- **Métrica: Número de Estrelas**

Dentro das métricas analisadas relativas ao projeto, destaca-se agora a métrica relacionada com o número de estrelas que o repositório recebeu. Esta métrica tem uma importância intrínseca, pois reflete o grau de popularidade e reconhecimento que o projeto alcançou na comunidade.

Para a execução desta consulta SQL, delineou-se uma metodologia composta por diversas fases, cada uma com sua peculiaridade e contribuição no tratamento e interpretação dos dados. As fases cruciais compreendem:

1) Compilação de Dados: A consulta agrega informações oriundas de tabelas anuais e diárias do GHArchive, considerando todos os anos em arquivo até ao final de agosto de 2023.
2) Eventos Observados: A análise incide sobre o tipo de evento 'WatchEvent', que, no contexto do GitHub, representa a ação de um utilizador dar uma estrela a um repositório.
3) Funções Utilizadas: Recorre-se à função ROW_NUMBER() para atribuir um número sequencial a cada entrada, particionado pelo identificador dos repositórios e pelo login do *actor*. A função FORMAT_TIMESTAMP é usada para formatar a data, facilitando a posterior agregação por mês.
4) Agregação dos Dados: Os dados são, inicialmente, agregados por mês, resultando no total de estrelas para cada mês.

Na Tabela 26, é detalhadamente apresentada a estratégia metodológica empregue na recolha e tratamento dos dados referentes à métrica do número de estrelas. Esta tabela destaca cada componente chave deste processo, desde a origem dos dados até à sua agregação final.

Tabela 26 - Procedimento de Extração e Análise do Número de Estrelas.

Componente	Descrição
Compilação de Dados	Reunião de informações das tabelas anuais e diárias, abrangendo todos os anos disponíveis até ao final de agosto de 2023.
Eventos Observados	Foco nos eventos de tipo 'WatchEvent', representativos da atribuição de estrelas.
Funções Utilizadas	Aplicação de funções específicas para processamento e formatação dos dados.
Agregação dos Dados	Organização dos dados por mês, proporcionando uma visão temporal.

Na Figura 55 é apresentado um excerto da consulta SQL desenvolvida, destacando as operações chave para a extração e processamento dos dados relativos à métrica de estrelas.

```
    FROM `githubarchive.year.*`
    WHERE
        type = 'WatchEvent'
        AND repo.id IN (12888993)
    UNION ALL
    SELECT
        repo.id AS repo_id,
        created_at,
        ROW_NUMBER() OVER (PARTITION BY repo.id, actor.login) AS row_num
    FROM `githubarchive.day.2023*`
    WHERE
        type = 'WatchEvent'
        AND repo.id IN (12888993)
        AND created_at <= '2023-08-31'
),
stars_per_month AS (
    SELECT
        repo_id,
        FORMAT_TIMESTAMP('%Y-%m-01', TIMESTAMP(created_at)) AS t_month,
        COUNT(*) AS total
    FROM stars
    WHERE
        row_num = 1
    GROUP BY
        repo_id, t_month
```

Figura 55 - Segmento da Consulta SQL para Extração e Contabilização de Estrelas no Repositório.

Para uma análise aprofundada da métrica de estrelas, é fundamental especificar cada fase da sua avaliação. A Tabela 27 detalha a abordagem adotada para avaliar esta métrica em contexto temporal.

Tabela 27 - Descrição do Procedimento de Avaliação da Métrica de Estrelas.

Métricas	Descrição
Objetivos	Entendimento da popularidade e reconhecimento do projeto, medido pelo número de estrelas.
Agregadores	Classificação dos dados ao nível mensal, proporcionando uma visão temporal.
Visualizações	Representações gráficas que demonstram a trajetória e crescimento do reconhecimento do projeto ao longo do tempo.

- **Métrica: Tempo Médio de Resposta/Fecho *Issue***

No contexto da análise de métricas relativas ao projeto em questão, o tempo médio até a primeira resposta ou interação com *issues* destaca-se como uma métrica relevante. Esta métrica permite avaliar a eficiência e reatividade da equipa de desenvolvimento e da comunidade em geral, face às questões ou problemas levantados.

Para a elaboração desta consulta SQL, foi concebido um procedimento meticuloso composto por várias etapas chave. Estas etapas, essenciais para a interpretação e manipulação dos dados, englobam:

1) Identificação de Respostas: Foca-se em identificar o primeiro momento de resposta ou interação para cada *issue*, seja ela um comentário ou a resolução da própria *issue*.
2) Recolha de Dados: A consulta extrai informações das tabelas anuais e diárias do GHArchive, incorporando todos os registos disponíveis até agosto de 2023.
3) Cálculo Temporal: Utilizam-se funções específicas, como UNIX_SECONDS e TIMESTAMP, para calcular a diferença temporal entre a abertura da *issue* e a sua primeira resposta ou interação.
4) Agregação Mensal: Os dados são posteriormente organizados por mês, resultando no cálculo do tempo médio de resposta a *issues* para cada período.

Na Tabela 28, é apresentado o método adotado para a extração e análise da métrica do tempo médio de resposta a *issues*.

Tabela 28 - Estratégia de Extração e Análise do Tempo Médio até a Primeira Resposta ou Interação com *Issues*.

Componente	Descrição
Identificação de Respostas	Identificação do primeiro ponto de resposta ou interação para cada *issue*.
Recolha de Dados	Captura de informações das tabelas anuais e diárias, enfocando particularmente no intervalo especificado.
Cálculo Temporal	Dedução do intervalo temporal entre a criação da *issue* e a sua primeira interação.
Agregação Mensal	Síntese dos resultados por mês, dando uma visão do tempo médio até a primeira resposta ou interação.

A Figura 56 ilustra um segmento da consulta SQL elaborada, evidenciando os componentes principais da mesma.

```
37
38      SELECT
39          JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS number,
40          created_at AS opened_at
41      FROM `githubarchive.day.2023*`
42      WHERE repo.id = 86472815
43      AND type = 'IssuesEvent' AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'opened'
44      AND created_at <= '2023-09-17'
45  ),
46  tdiff AS (
47      SELECT
48          t_month,
49          AVG(UNIX_SECONDS(TIMESTAMP(iwfr.first_responded_at)) - UNIX_SECONDS(TIMESTAMP(iwo.opened_at))) / 3600 AS
    avg_diff_hours
50      FROM
```

Figura 56 - Consulta SQL para Cálculo do Tempo Médio até primeira resposta ou fecho em *Issues*.

Após a visualização da consulta SQL, torna-se pertinente detalhar e esclarecer os componentes principais que compõem a métrica de tempo até a primeira resposta ou interação com *issues*. A Tabela 29 apresenta os elementos-chave dessa métrica, elucidando os objetivos, os processos de agregação dos dados e as potenciais visualizações. Estes elementos são cruciais para uma compreensão abrangente e para garantir que a métrica é interpretada e utilizada de forma apropriada no contexto da análise do projeto.

Tabela 29 - Descrição da Avaliação da Métrica de Tempo de Resposta a *Issues*.

Métricas	**Descrição**
Objetivos	Avaliar a rapidez e eficácia com que a equipa e a comunidade interagem inicialmente com as issues criadas no repositório.
Agregadores	Síntese dos resultados ao nível mensal, proporcionando uma visão consolidada do tempo médio até a primeira resposta ou interação por período.
Visualizações	Representações gráficas temporais que ilustram o tempo médio até a primeira resposta ou interação com issues, destacando a evolução dessa métrica ao longo dos meses.

4. ANÁLISE DOS DADOS OBTIDOS E DISCUSSÃO

Neste capítulo, procedeu-se à análise aprofundada dos dados obtidos através do Python e BigQuery, já abordados no Capítulo 3, bem como à discussão abrangente dos resultados. Durante a análise deste estudo, foram aplicadas várias métricas de avaliação aos projetos *Open Source* selecionados, com o objetivo de avaliar o seu sucesso e sustentabilidade. Esta análise é fundamental para compreender o panorama completo desses projetos e para identificar correlações significativas entre as diferentes métricas utilizadas. Para além disso, será viável inferir sobre as especificidades inerentes a cada projeto, particularmente no que concerne às vertentes de *Open Source Software* e *Open Source Hardware*.

4.1. Apresentação dos Dados Obtidos

A análise dos dados será conduzida de forma sistemática, seguindo uma estrutura que permitirá uma avaliação detalhada de cada métrica e a sua relevância para os objetivos deste estudo. Além disso, serão identificados padrões, tendências e relações interdependentes entre as métricas, fornecendo *insights* fundamentais sobre a dinâmica dos projetos *Open Source*.

Todas as variáveis de dados consistem em séries temporais que se estendem desde a data de criação do projeto, até 31 de agosto de 2023. No âmbito do projeto de *Open Source Hardware*, o intervalo compreende 78 meses desde a sua conceção até a data referida, enquanto para o projeto de *Open Source Software*, o período engloba 120 meses.

Inicialmente, cada métrica foi analisada individualmente, sendo utilizadas visualizações adaptadas aos dados para facilitar a respetiva avaliação.

4.1.1. Análise das Métricas

- **Número de *forks***

O número de *forks* de um repositório é uma métrica fundamental em projetos de código aberto, indicando quantas vezes o repositório original foi replicado por diferentes utilizadores. Esta métrica reflete o interesse em contribuir, adaptar, explorar ou utilizar o projeto. Frequentemente, a comunidade replica um projeto para propor alterações via *pull request* ou adaptá-lo às suas necessidades específicas.

Ao observar esta métrica ao longo dos meses dos repositórios em estudo, podem-se identificar padrões e tendências que proporcionam informação valiosa acerca da recetividade e interação da comunidade com os repositórios.

A Figura 57 ilustra a evolução temporal do número de *forks* associados ao projeto de desenvolvimento de *Open Source Hardware* (OSH) em análise. Através deste gráfico, é possível compreender a progressão e o grau de interesse e adoção que o projeto tem suscitado na comunidade ao longo do seu desenvolvimento.

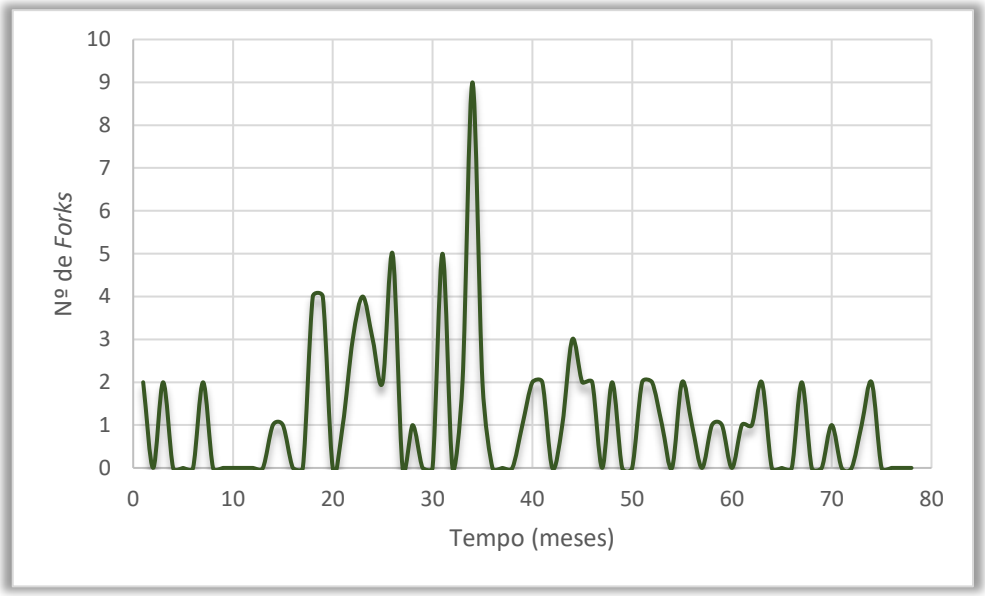


Figura 57 – Representação Gráfica da Variação Mensal do Número de *Forks* no Projeto OSH.

A visualização da evolução dos *forks* no projeto apresenta uma trajetória peculiar. Ao longo dos 78 meses analisados, 38 meses não evidenciaram qualquer *fork*. Estes intervalos de inatividade podem refletir diversas características associadas ao desenvolvimento de *Open Source Hardware*, nomeadamente a exigência de recursos materiais para testes e a complexidade associada à replicação de *designs* de *hardware*.

Nos meses restantes, identifica-se uma modesta atividade de *forks*, com a predominância de valores entre 1 e 3 *forks*. Contudo, emergem picos específicos que alcançam máximos de 4, 5 e 9 *forks*. Tais variações podem ser motivadas por atualizações pontuais no projeto, promoções em eventos ou plataformas, ou avanços relevantes que atraíram a atenção da comunidade. Assim, infere-se do gráfico uma dinâmica de envolvimento da comunidade intermitente no contexto do projeto no desenvolvimento de *Open Source Hardware*.

A natureza física do hardware pode fazer com que menos indivíduos estejam preparados ou motivados a criar um *fork* e contribuir ativamente para o projeto.

Adicionalmente, a contribuição em projetos de desenvolvimento de *hardware* pode enfrentar obstáculos inerentes à sua natureza. A ação de efetuar um *fork* pode transcender a mera replicação de código, abrangendo a interpretação de esquemas, identificação de componentes e, em determinados contextos, a produção efetiva do *hardware*. Tais desafios podem condicionar a frequência de *forks* observados.

O desenvolvimento de hardware pode ser marcado por ciclos temporais alargados, o que significa que os *forks* podem surgir em reação a atualizações substanciais ou lançamentos inovadores, eventos estes que ocorrem de forma não recorrente. É relevante ainda considerar que projetos nesta área exigem frequentemente um saber técnico distinto, facto que pode restringir o leque de potenciais contribuidores.

A Figura 58 proporciona uma visualização cumulativa da evolução do número de *forks* associados ao projeto de desenvolvimento de OSH em análise. Ao contrário de um gráfico de linhas convencional que ilustra variações mensais, esta representação oferece uma perspetiva agregada,

permitindo avaliar o crescimento total e a acumulação de interesse e adoção pelo projeto desde o seu início até ao momento mais recente.

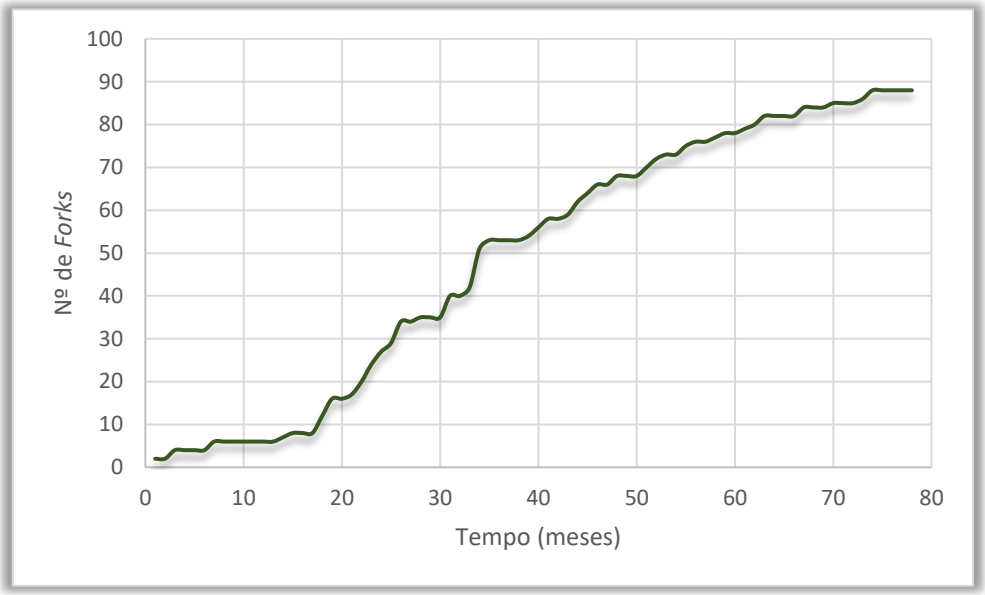


Figura 58 - Tendência Acumulada do Número de *Forks* no Projeto OSH.

O gráfico mostra uma tendência de crescimento, mas uma ascensão inconstante, salientando períodos de estagnação intercalados com crescimentos graduais. Esta progressão reforça a ideia de que, apesar do interesse contínuo no projeto, as contribuições e adesões acontecem de forma esporádica e ponderada. A natureza cumulativa desta visualização enfatiza os 38 meses sem qualquer adição de *forks*, ilustrando os intervalos de inatividade e os momentos de retoma.

Os crescimentos moderados no número acumulado de *forks* podem ser interpretados como fases em que o projeto alcançou certos marcos ou inovações, atraindo, assim, a atenção da comunidade. Estes aumentos, no contexto cumulativo, evidenciam a adição contínua ao total de *forks*, destacando a progressão lenta, mas constante, do projeto ao longo do tempo.

A natureza física do hardware, com os desafios e barreiras previamente mencionados, é evidenciada na inconstância desta trajetória acumulativa. A ação de efetuar um *fork* em projetos de hardware é, indubitavelmente mais complexa, resultando numa dinâmica de contribuição mais esporádica e menos previsível.

Analisada a métrica para o projeto OSH, procede-se à análise da métrica para o projeto OSS. A Figura 59 exibe a evolução temporal do número de *forks* associados ao projeto *Open Source Software*. Este gráfico possibilita a compreensão da dinâmica de interesse e participação da comunidade neste tipo específico de projeto. Ao contrário dos projetos de *Open Source Hardware*, que têm implicações tangíveis e complexidades próprias, os projetos de *software* apresentam características e desafios distintos, que podem refletir-se de forma diferente no número de *forks* e na interação da comunidade.

Dada a natureza intrínseca do software, é esperado que o padrão de contribuição e adoção seja diferente do observado em hardware. A facilidade de acesso, a ausência de barreiras físicas e a potencial rapidez de desenvolvimento e teste podem influenciar significativamente a dinâmica de *forks* neste contexto.

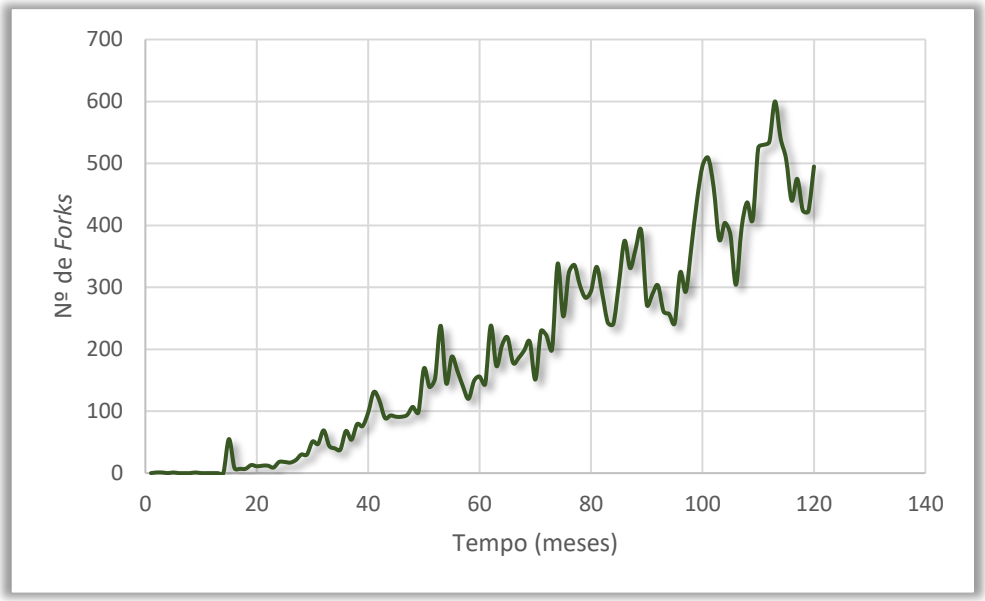


Figura 59 - Representação Gráfica da Variação Mensal do Número de *Forks* no Projeto OSS.

Nos primeiros 14 meses, a atividade mostrou-se relativamente baixa, com um número restrito de *forks*. Após este período, observou-se uma atividade mais regular e crescente, no entanto, esta atividade é caracterizada por flutuações, observando-se vales e picos no gráfico. Os picos representam momentos em que houve um aumento súbito no interesse ou na necessidade do projeto. Estes picos podem ser influenciados por:

- **Lançamentos de Novas Funcionalidades:** Quando novas e importantes funcionalidades são adicionadas, pode haver um aumento no número de *forks*, pois mais utilizadores podem encontrar valor no projeto.
- **Divulgação em Plataformas:** Uma menção ou destaque em plataformas reconhecidas ou em redes sociais pode resultar num aumento súbito de visibilidade.
- **Recomendações:** Se o projeto for adotado ou recomendado por projetos influentes, isso pode levar a um aumento no número de *forks*.

Os vales representam os períodos de menor atividade. Estes podem surgir devido a:

- **Estabilidade do Projeto:** Um projeto estável com poucas mudanças pode não incentivar a novos *forks*.
- **Competição:** O surgimento de projetos alternativos ou soluções mais eficientes pode levar a uma diminuição no número de *forks*.
- **Falta de Atualizações:** Caso o projeto não receba atualizações regulares pode resultar numa diminuição no interesse por parte da comunidade.

Destaca-se a inexistência de uma regressão significativa no projeto. Apesar das flutuações observadas, após os meses iniciais, o projeto nunca voltou a registar um total de 0 *forks*. A tendência predominante indica progresso e crescimento. Além disso, ao observar-se a distribuição ao longo do tempo, é evidente que os meses com um aumento no número de *forks* superam aqueles com diminuições, reforçando a perspetiva de uma trajetória ascendente do projeto.

O gráfico ilustra uma clara tendência ascendente no número de *forks* ao longo do tempo, indicando um aumento contínuo no envolvimento e interesse da comunidade. Esta tendência é uma indicação positiva da saúde e viabilidade do projeto, bem como da sua ressonância contínua na comunidade.

Para uma perspetiva a longo prazo, o gráfico cumulativo apresenta-se como uma ferramenta crucial. Enquanto o gráfico de linhas representa a atividade mensal, o gráfico cumulativo mostra o crescimento total do projeto ao longo do tempo. Este encontra-se representado na Figura 60.

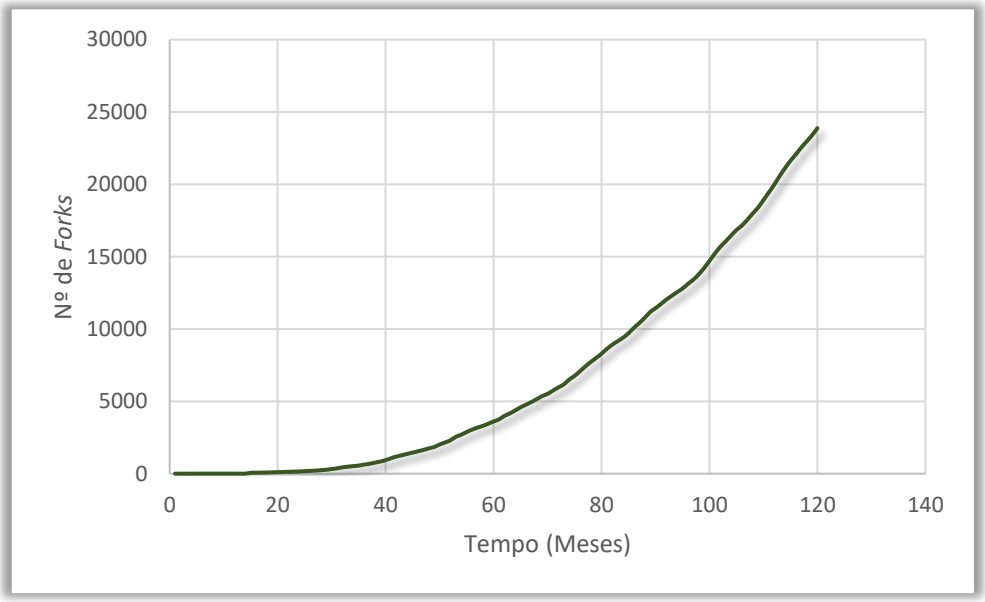


Figura 60 - Tendência Acumulada do Número de *Forks* no Projeto OSS.

O gráfico cumulativo do número de *forks* revela uma trajetória de crescimento contínuo, indicando que o projeto manteve, consistentemente, a sua capacidade de atração. Nos meses iniciais, o gráfico de linhas acumulativas corroborou os dados observados no gráfico de linhas. No entanto, a partir do mês 15, começou-se a registar uma atividade de *forks* no projeto, e um crescimento mais acentuado foi claramente evidenciado a nível cumulativo a partir do mês 40, ou seja, ao fim de 3 anos e 4 meses.

Esta evolução ascendente reflete a contínua relevância e apelo do projeto para a comunidade. O crescimento sustentado indica que, regularmente, novos membros reconhecem valor no projeto, seja no âmbito da contribuição, adaptação ou simples utilização. Dentro deste contexto, emergem aspetos fundamentais a considerar: o crescimento é consistente e, mesmo face a eventuais variações no panorama mensal, a trajetória no gráfico cumulativo mantém-se firme, sublinhando a pertinência e funcionalidade do projeto. Este aumento contínuo nos *forks* constitui uma sólida evidência do valor que a comunidade confere ao projeto, atestando a sua capacidade em responder a uma necessidade ou colmatar um vazio no ecossistema. Além disso, a tendência positiva antecipa uma base sólida para o projeto, sugerindo um potencial evolutivo caso seja devidamente gerido e atualizado.

- **Número de *Releases***

Releases são versões estáveis que são disponibilizadas para os utilizadores. O número de *releases* pode indicar a frequência de atualizações e a rapidez com que o projeto responde às necessidades ou problemas da comunidade.

A Figura 61 ilustra a evolução mensal do número de *releases* associados ao projeto *Open Source Hardware*. Esta representação gráfica proporciona uma visão pormenorizada sobre a cadência e frequência com que o projeto lança novas versões ou atualizações. Os *releases*, em projetos de *hardware* de código aberto, não indicam apenas a introdução de novas características ou melhorias, mas também podem refletir ajustes em *design*, otimizações ou correções de problemas identificados.

No contexto de *Open Source Hardware*, a periodicidade e a natureza dos *releases* podem ser distintas quando comparadas a projetos de *software*, dada a complexidade inerente à materialização física de *designs* e à necessidade de testes tangíveis. Assim, a análise deste gráfico procura compreender a dinâmica de desenvolvimento, inovação e resposta às necessidades e *feedback* da comunidade.

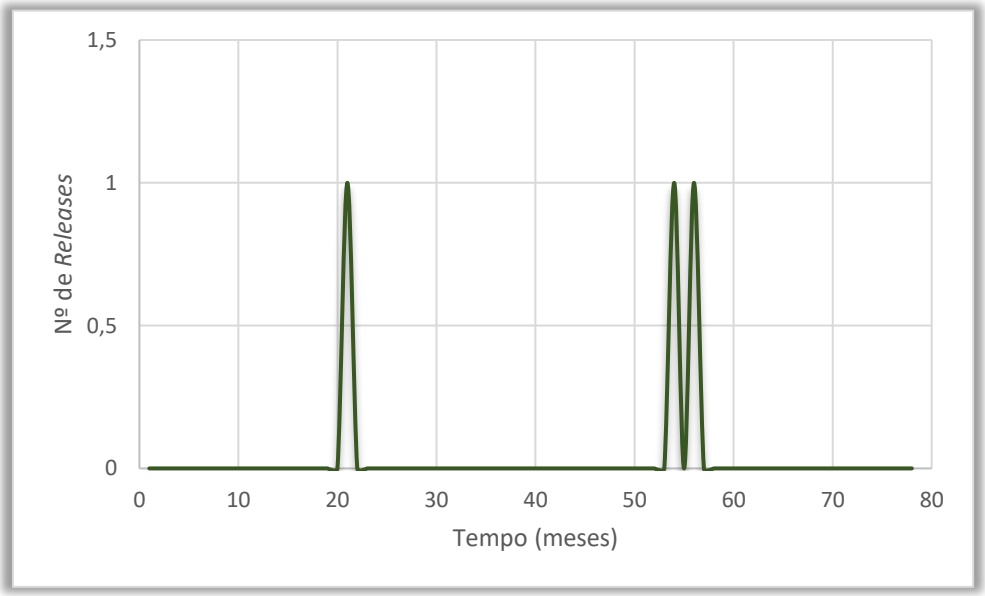


Figura 61 - Representação Gráfica da Variação Mensal do Número de *Releases* no Projeto OSH.

Ao longo da sua existência, o projeto apresenta uma predominância de períodos sem registo de novos *releases*. Esta dinâmica, similar ao observado na métrica do número de *forks*, sugere *releases* de natureza esporádica. Tal comportamento pode ser indicativo de um ciclo de desenvolvimento prolongado, em que os lançamentos ocorrem após extensas fases de desenvolvimento, testes e validação. No entanto, ao longo de aproximadamente seis anos de evolução do projeto, destacam-se apenas três *releases*, com 75 meses desprovidos de qualquer lançamento.

As especificidades dos projetos de *hardware* introduzem complexidades que podem culminar em ciclos de lançamento mais dilatados. Após etapas de *design*, é frequentemente requerida a realização de prototipagem, testes físicos e ajustes antes de se proceder a um *release* final.

A recorrência de meses sem lançamentos pode insinuar fases de inatividade do projeto ou a possibilidade de que as modificações implementadas não justificassem a emissão de um novo

release. Adicionalmente, pode existir um processo meticuloso de recolha de feedback e iteração no design, resultando em intervalos alargados entre *releases* sucessivos.

Para uma perspetiva a longo prazo, o gráfico cumulativo apresenta-se como uma ferramenta crucial. Enquanto o gráfico de linhas representa a atividade mensal, o gráfico cumulativo mostra o crescimento total do projeto ao longo do tempo. Este encontra-se representado na Figura 62Figura 60.

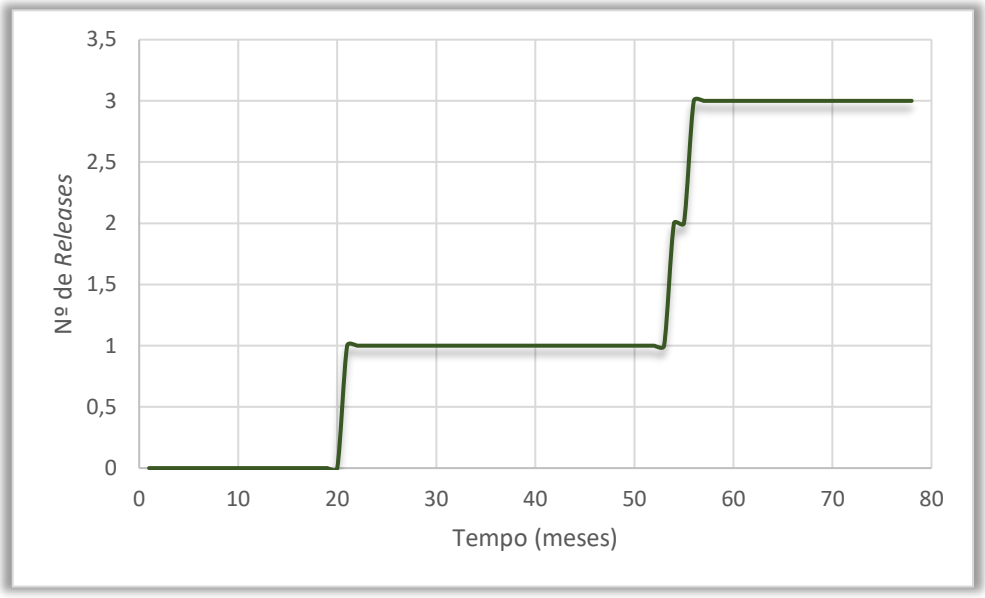


Figura 62 - Tendência Acumulada do Número de *Releases* no Projeto OSH.

A trajetória acumulativa dos *releases* do projeto *Open Source Hardware* destaca-se pela sua intermitência. A linha acumulativa, embora demonstre um aumento global ao longo dos meses, apresenta períodos distintos de estabilidade horizontal. Estas fases refletem os intervalos temporais onde o projeto não lançou novas versões.

Esta representação acumulativa proporciona uma visão global do ritmo de lançamentos. Percebe-se que, mesmo quando o projeto introduziu novos *releases*, estes não foram suficientes para impulsionar uma ascensão acentuada na curva acumulativa. Tal fenómeno sugere que, quando ocorrem, os lançamentos tendem a ser isolados e não parte de uma sequência consecutiva.

O desenvolvimento de *hardware*, pela sua natureza, pode ter fases mais prolongadas de pesquisa, desenvolvimento e validação, o que pode resultar em lançamentos menos frequentes e mais espaçados no tempo. Esta dinâmica acumulativa pode ser interpretada como uma manifestação desta característica intrínseca ao desenvolvimento de *hardware*.

Após a análise da métrica de *releases* no contexto do projeto *Open Source Hardware*, agora direciona-se a atenção para os padrões de lançamento observados no projeto de *software* de código aberto selecionado.

Ao longo da evolução do projeto *Open Source Software* sob análise, identificou-se uma série temporal referente ao número de *releases* publicados ao longo de 120 meses. Como se pode observar na Figura 63, esta métrica, ao ser examinada, fornece informações valiosas sobre o ritmo de desenvolvimento, lançamentos e possíveis fases de estabilização ou reestruturação do projeto.

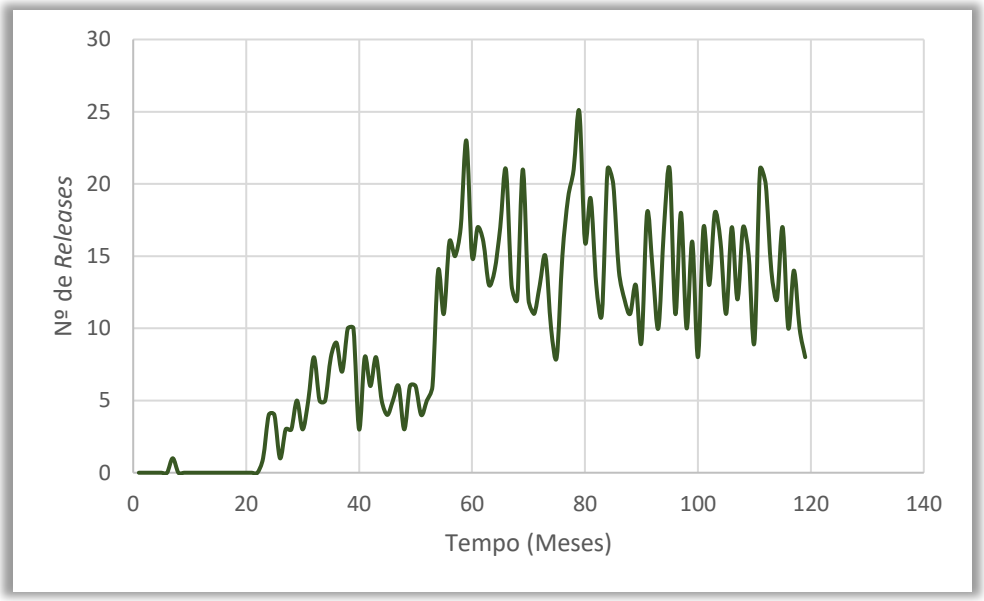


Figura 63 - Representação Gráfica da Variação Mensal do Número de *Releases* no Projeto OSS.

Durante os primeiros 22 meses, é evidente que não houve grande atividade no número de lançamentos. Entre os meses 23 e 53, ou aproximadamente entre 2015 e 2018, observa-se uma mudança. O número de lançamentos começa a aumentar, mas com uma cadência aproximadamente constante. Este período pode ser interpretado como uma fase de estabilização, onde o projeto alcançou um estado de maturidade suficiente para garantir lançamentos mais frequentes. No entanto, os valores dos lançamentos ainda se mantêm relativamente baixos.

Após 53 meses, aproximadamente 4 anos após a criação do projeto, verifica-se uma mudança significativa. O número de lançamentos começa a aumentar de forma mais acentuada, mas com picos e vales notáveis. Esse aumento súbito pode ser resultado de uma reestruturação, a conclusão de um ciclo de desenvolvimento significativo ou a resposta a *feedbacks* e a procura dos utilizadores da plataforma. Este padrão sugere uma fase de maturidade no desenvolvimento do *software*. O aumento na frequência das *releases* pode ser interpretado de diversas formas:

- **Resposta Ativa ao *Feedback***: A equipa pode lançar correções e melhorias em função das opiniões e necessidades dos utilizadores.
- **Ciclos de Desenvolvimento Consistentes:** Esta fase pode também sugerir a existência de ciclos de desenvolvimento bem definidos, que orientam a regularidade dos lançamentos.
- **Evolução Contínua:** O *software*, nesta fase, demonstra sinais claros de evolução contínua, com a incorporação de novas funcionalidades e o aprimoramento das já existentes.

Contudo, o projeto também teve períodos caracterizados pela diminuição de *releases*, evidenciado no gráfico pelos vales. Estas diminuições podem indicar:

- **Planeamento e Execução:** Há momentos em que a prioridade recai sobre o planeamento e a execução de funcionalidades mais elaboradas ou correções que exigem um maior tempo de desenvolvimento.
- **Fases de Teste e Validação:** Antes de oficializar um lançamento, é imperativo que todas as novas funcionalidades sejam meticulosamente testadas e validadas. Este processo, dependendo do volume de mudanças, pode ser moroso.

- ***Feedback* da Comunidade:** Em certas ocasiões, pode ser estratégico aguardar um retorno da comunidade acerca de uma *release* anterior antes de avançar com a próxima.

É crucial entender que meses com menos lançamentos não denota, necessariamente, um retrocesso. A ênfase deve ser colocada na qualidade e na robustez dos lançamentos, bem como na capacidade de resposta da equipa aos pedidos e *feedbacks* da comunidade. O facto de que em todos os meses houve *releases*, nunca retornando aos valores iniciais, reflete um comprometimento contínuo com o projeto, indicando que ele está não só ativo, mas em constante aprimoramento.

Para complementar a análise anterior e oferecer uma perspetiva mais abrangente sobre o crescimento total do projeto, recorreu-se à visualização acumulativa do número de *releases*. Na Figura 64, é possível observar a trajetória acumulativa destes lançamentos ao longo dos 120 meses. Esta representação gráfica evidencia de forma clara o acumular consistente de lançamentos, permitindo uma compreensão ampla da evolução e do compromisso da equipa de desenvolvimento ao longo do tempo.

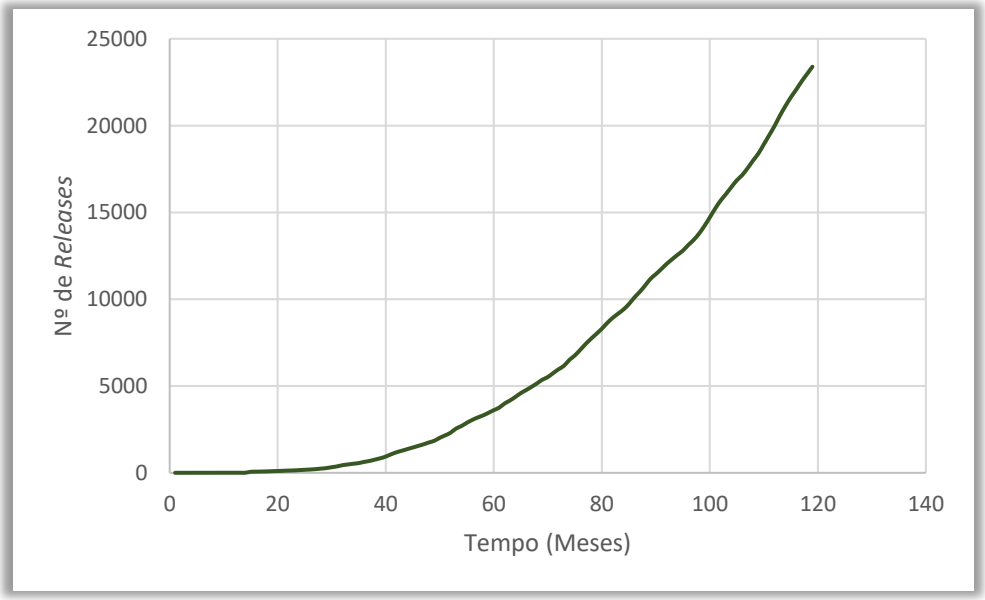


Figura 64 - Tendência Acumulada do Número de *Releases* no Projeto OSS.

Nos primeiros 22 meses, a inclinação quase inalterada da curva reforça a ideia, já evidenciada pelo gráfico de linhas, de uma atividade de lançamento reduzida. Contudo, após esse período, assiste-se a uma clara inflexão na curva, refletindo um ritmo mais intensificado de lançamentos. O que é notável é que, mesmo durante os vales observados no gráfico de linhas, o gráfico acumulativo continua a crescer. Isso sugere que, embora haja períodos de menor atividade, o projeto nunca sofreu estagnação ou retrocesso no que respeita ao total de lançamentos.

Em determinadas fases do gráfico acumulativo, destaca-se um incremento mais pronunciado no número cumulativo de lançamentos. Estas elevações podem ser associadas a lançamentos de versões de maior relevância ou a momentos de particular dinamismo por parte da equipa de desenvolvimento. A comparação com o gráfico de linhas, que apresenta a atividade de lançamento de forma mensal, permite inferir que, apesar das flutuações evidentes neste último, o gráfico acumulativo oferece uma visão mais harmonizada e elucidativa da trajetória global do projeto. O carácter contínuo e ascendente deste gráfico sublinha que, independentemente das variações

mensais, o projeto persistiu na sua expansão. Afinal, lançamentos frequentes e sistemáticos são, muitas vezes, uma resposta ao *feedback* e às contribuições da comunidade.

O gráfico de linhas acumulativas para o número de *releases* reafirma a saúde e a atividade contínua do projeto. Este serve como uma prova do comprometimento da equipa em fornecer atualizações, melhorias e correções de forma regular. Esta regularidade não só garante a relevância do *software*, mas também destaca a sua adaptabilidade e evolução ao longo do tempo.

Prosseguindo com a análise após a observação da evolução do número de *stars* no projeto *Open Source Hardware*, a Figura 65 apresenta a variação mensal do número de *stars* para o projeto *Open Source Software*. Esta representação proporciona uma compreensão da dinâmica de interesse e reconhecimento da comunidade em ambientes distintos de desenvolvimento.

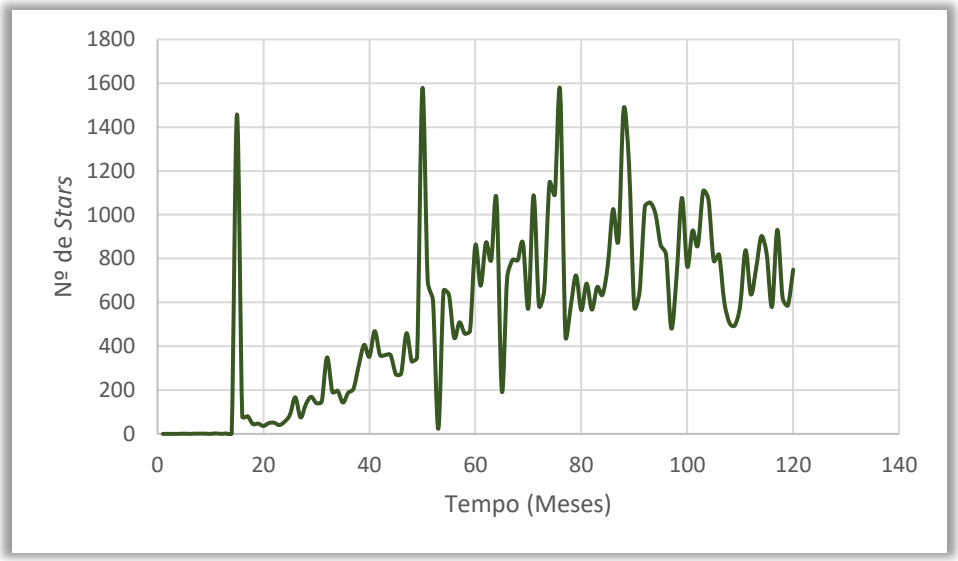


Figura 65 - Representação Gráfica da Variação Mensal do Número de *Stars* no Projeto OSS.

Durante os primeiros 14 meses do projeto, observa-se uma atividade mínima em termos de atribuição de *stars*, na qual o projeto recebeu poucas ou nenhumas estrelas.

No mês 15, verifica-se um pico notável no número de *stars*, indicando um marco significativo na trajetória do projeto. Tal pico pode ser o resultado de um lançamento significativo, de uma promoção bem-sucedida ou de uma menção em plataformas de grande visibilidade. A partir desse momento, o gráfico revela uma tendência geral crescente, ainda que com picos e vales intercalados. Os picos podem indicar momentos de inovação, lançamentos de funcionalidades esperadas pela comunidade, ou até mesmo períodos onde o projeto foi amplamente divulgado ou discutido em fóruns e conferências relevantes.

Contrariamente aos períodos de crescimento, existem momentos em que a evolução no número de estrelas desacelera ou até mesmo evidencia uma retração. Estas desacelerações podem decorrer de uma breve saturação após uma fase intensa de crescimento, em que grande parte dos interessados já terá atribuído uma estrela ao projeto. Outro facto que pode contribuir para esta desaceleração é o aparecimento de projetos concorrentes ou a introdução de soluções alternativas que captam a atenção da comunidade. Adicionalmente, se o projeto atravessa uma fase sem introduzir atualizações significativas ou novidades, é natural que se verifique uma diminuição no interesse manifestado pela comunidade.

A visualização acumulativa proporciona uma perspetiva global da trajetória de crescimento do projeto de desenvolvimento de *Open Source Software*, como se pode observar na Figura 66.

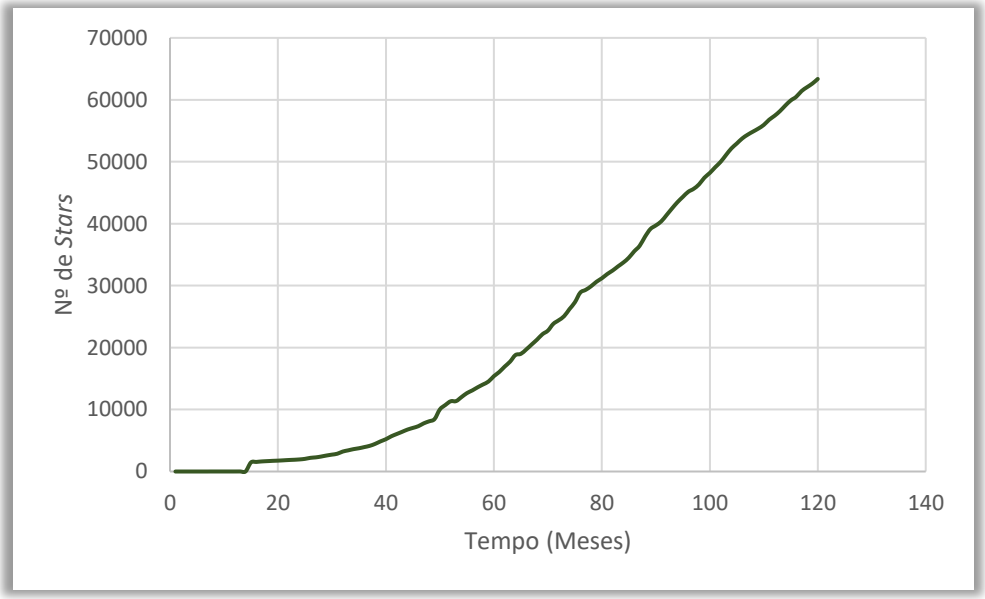


Figura 66 - Tendência Acumulada do Número de *Stars* no Projeto OSS.

Uma das observações mais proeminentes no gráfico acumulativo é o crescimento quase ininterrupto. A tendência continuamente ascendente sinaliza o aumento contínuo no reconhecimento e interesse pela comunidade. Mesmo considerando os períodos de vales no gráfico de linhas, no contexto acumulativo, a evolução é sempre positiva, evidenciando a resiliência e relevância contínua do projeto.

A inclinação da curva acumulativa oferece *insights* sobre o ritmo de crescimento. O crescimento inicial mais suave, seguido de uma inclinação mais acentuada, corrobora a observação anterior de um aumento de interesse após 15 meses de desenvolvimento do projeto.

A natureza ascendente do gráfico acumulativo sugere que, mesmo com as flutuações mensais, a base de utilizadores que marca o projeto com uma "estrela" está em constante crescimento. Este é um indicativo positivo para o futuro, sugerindo que enquanto o projeto continuar a inovar e servir à comunidade, o seu reconhecimento e influência só tendem a crescer. Esta métrica reflete a capacidade do projeto de reter e atrair continuamente novos contribuidores e utilizadores, solidificando a sua posição como um projeto valioso e relevante.

- **Número de *Commits* e *Pushes***

Tanto o número de *commits* quanto o de *pushes* num repositório do GitHub são métricas essenciais para avaliar a atividade e colaboração do desenvolvimento de um projeto. *Commits* frequentes indicam uma cadência ativa de desenvolvimento, refletindo as alterações e ajustes individuais feitos no código. Por outro lado, os *pushes* representam a sincronização dessas alterações com o repositório central, indicando momentos em que as contribuições locais são compartilhadas com a equipa.

Na Figura 67 apresenta-se a evolução mensal dos *commits* e *pushes* associados ao projeto *Open Source Hardware*. Esta representação gráfica proporciona *insights* sobre o ritmo e a intensidade das contribuições ao longo do tempo, servindo como um indicador da atividade de

desenvolvimento e da dinâmica de atualizações do projeto. A interpretação que se segue visa elucidar os padrões observados e os possíveis fatores que influenciam estes movimentos no contexto específico do *hardware* de código aberto.

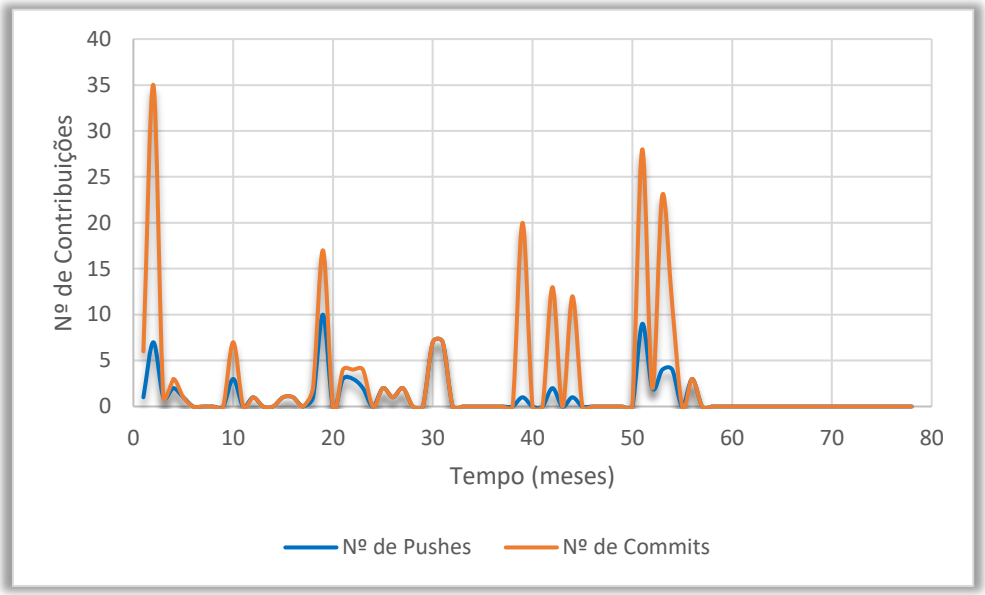


Figura 67 - Representação Gráfica da Variação Mensal do Número de *Pushes* e *Commits* no Projeto OSH.

Ao longo dos 78 meses em análise, constata-se que a frequência mensal de *commits* supera a de *pushes*. Tal observação alinha-se com a expectativa de que múltiplos *commits* podem ser agrupados num único *push*. Salienta-se um período de inatividade distinto, que se estende do mês 57 ao 78, durante o qual não se registou qualquer *commit* ou *push*. Esta inatividade é ainda reforçada pela presença de diversos meses, dispersos na linha temporal, desprovidos de contribuições.

Das contribuições efetuadas ao projeto, estas manifestam-se de forma intermitente, com apenas 27 dos 78 meses a evidenciarem atividade de *commit* ou *push*. Este padrão sugere que, apesar de existir interesse e atividade, estes não se materializam de forma uniforme. A superioridade numérica dos *commits* em relação aos *pushes* indica um desenvolvimento estruturado em fases, onde múltiplos *commits* ilustram avanços incrementais que, posteriormente, são consolidados num *push* para o repositório principal.

Dando continuidade ao que foi mencionado anteriormente, a natureza tangível do *Open Source Hardware* pode ditar a cadência e periodicidade das contribuições. Os ciclos de desenvolvimento prolongados, que abrangem desde o design até testes físicos, podem ser a causa dos períodos de inatividade observados. A esporadicidade das contribuições pode assim refletir os desafios únicos associados ao desenvolvimento de *hardware* em código aberto.

A representação acumulativa das métricas de *pushes* e *commits* para o projeto Open Source Hardware, exposta na Figura 68, oferece uma visão consolidada da evolução destas contribuições ao longo do tempo.

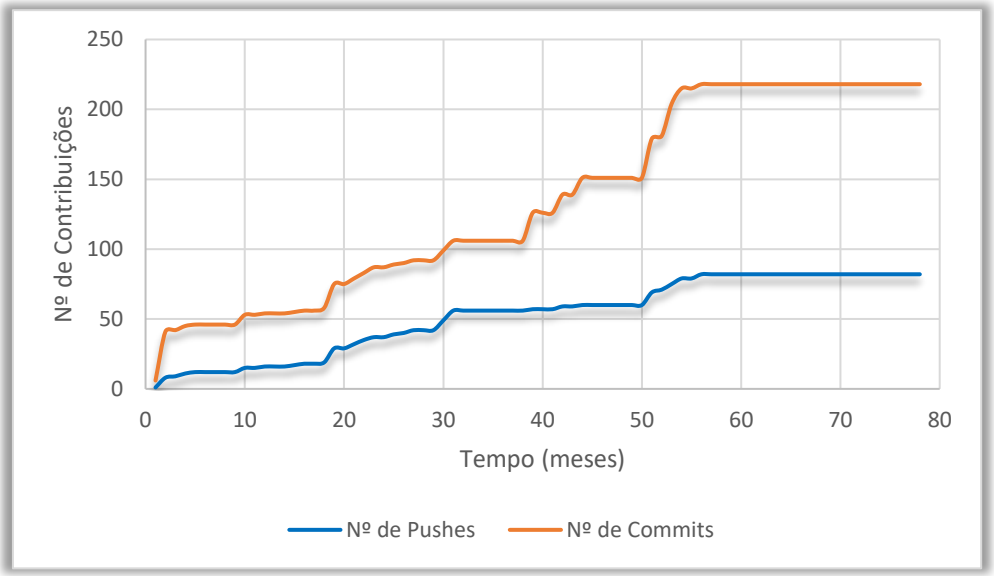


Figura 68 - Tendência Acumulada do Número de *Pushes* e *Commits* no Projeto OSH.

A evolução acumulativa das métricas de *pushes* e *commits*, ao longo do período em análise, denota um crescimento ténue e segmentado. Esta progressão, embora ascendente, é frequentemente interrompida por patamares de estabilidade, traduzidos por segmentos de declive quase horizontal no gráfico. As variações que surgem, quando presentes, não são expressivas, sublinhando o avanço gradual e contido.

Reiterando observações anteriores, destaca-se um período específico, do mês 57 ao 78, no qual não se registou qualquer *commit* ou *push*. Esta ausência prolongada de atividade corrobora com a perceção de fases intermitentes de desenvolvimento e possíveis pausas.

As especificidades do desenvolvimento de *hardware* conduzem a ciclos alargados. Uma vez atingida uma etapa de maturação no projeto, as intervenções podem tornar-se menos frequentes até que novos requisitos ou desafios emergem. Adicionalmente, a concretização de avanços em projetos de hardware pode ser condicionada por fatores como financiamento, aquisição de materiais ou desafios logísticos. Posteriormente a um ciclo de desenvolvimento, é comum a dedicação a fases de revisão, teste e validação, durante as quais o *design* se mantém inalterado.

A intermitência nas contribuições reflete, por sua vez, a dinâmica da comunidade envolvida. Certos meses podem ser marcados por um maior envolvimento de colaboradores, enquanto outros evidenciam um envolvimento mais comedido.

Prosseguindo-se para a análise da métrica para o projeto de desenvolvimento de *Open Source Software*, o gráfico apresentado na Figura 69 demonstra a evolução mensal do número de *commits* e *pushes*. Esta representação permite identificar padrões de atividade e possíveis flutuações no ritmo de desenvolvimento. Para além disso, permite identificar não apenas os padrões individuais de cada métrica, mas também a relação entre elas, dando *insights* sobre a frequência de alterações e a cadência de partilha dessas alterações.

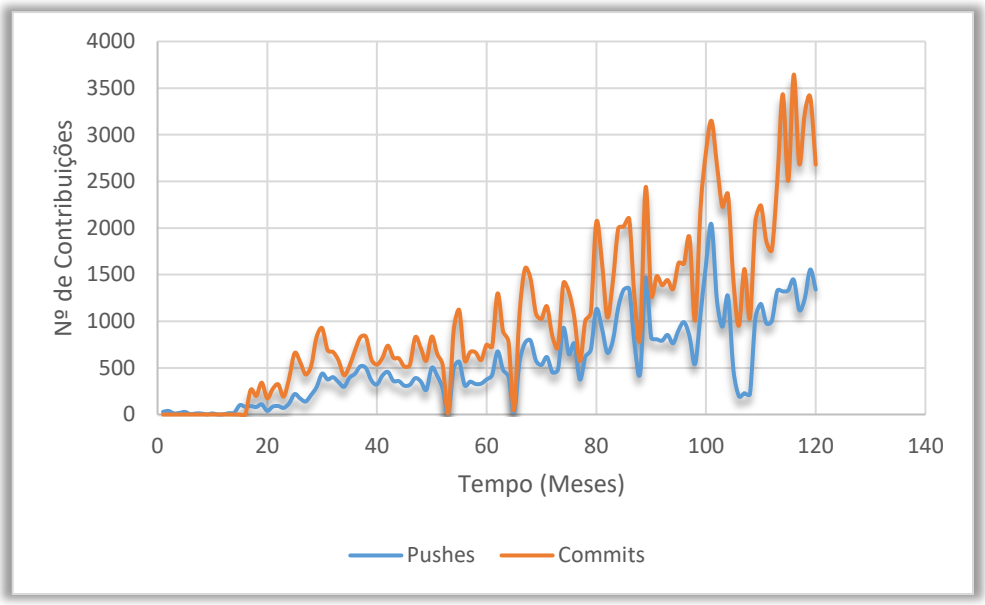


Figura 69 - Representação Gráfica da Variação Mensal do Número de *Pushes* e *Commits* no Projeto OSS.

Nos estágios iniciais do projeto, observa-se um período de inatividade, destacando-se a ausência de *commits* durante os primeiros 16 meses.

Este aumento sugere um renovado vigor ou, possivelmente, a entrada de um maior número contribuidores no projeto. A presença de *commits* e *pushes* são um indicativo de um projeto ativo e em progresso contínuo.

Quando se analisa a relação entre *pushes* e *commits*, percebe-se que, desde o momento em que os *commits* começaram a ser consistentemente registados, o seu volume frequentemente excedeu o de *pushes*. Esta dinâmica está alinhada com muitos fluxos de trabalho em desenvolvimento de código aberto, onde múltiplos *commits* são frequentemente agrupados e sincronizados com o repositório principal através de um único *push*.

Ao detalhar as tendências e comportamentos associados a estas métricas, identificam-se picos e vales recorrentes. Estas flutuações podem ser interpretadas de diversas maneiras. Por exemplo, a natureza iterativa do desenvolvimento de *software* muitas vezes leva a ciclos de intensa atividade focada no desenvolvimento e implementação de novas funcionalidades, seguida por fases de atividade mais reduzida, onde a ênfase recai sobre testes, revisão de código e planeamento. Adicionalmente, os picos mais evidentes podem estar alinhados com fases que antecedem lançamentos ou a concretização de etapas cruciais no projeto, momentos estes que demandam intensificação no desenvolvimento e resolução de falhas. Por outro lado, os intervalos caracterizados por menor atividade podem refletir fases nas quais a equipa concentra esforços na análise, aprimoramento e otimização do código preexistente, em vez da incorporação de novos elementos ou funcionalidades.

No contexto mais amplo do projeto, *pushes* e *commits* têm significados e implicações distintas. *Pushes* quantificam a frequência com que o código é sincronizado com o repositório principal, e o aumento deste número pode sinalizar fases de intensa colaboração ou a integração de novos membros à equipa. Por outro lado, *commits* retratam as modificações individuais no código. A proporção elevada de *commits* em relação aos *pushes* sugere que os *developers* estão a

implementar e registar alterações de forma incremental, uma abordagem alinhada com as melhores práticas em desenvolvimento de *software*.

Na Figura 70, apresenta-se o gráfico de linhas acumulativo relativo ao número de *pushes* e *commits* do projeto, proporcionando uma compreensão agregada da atividade de desenvolvimento.

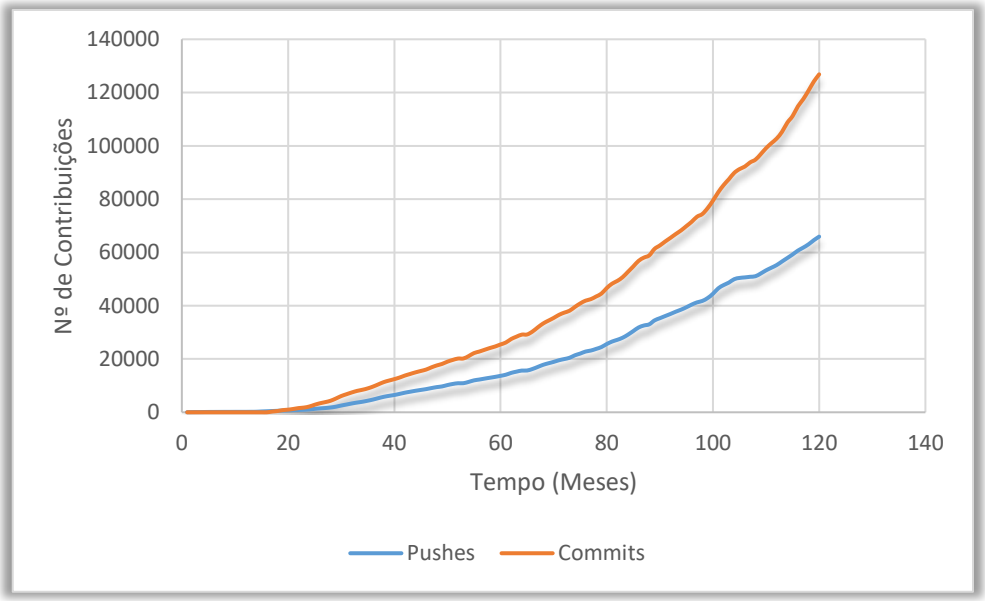


Figura 70 - Tendência Acumulada do Número de *Pushes* e *Commits* no Projeto OSS.

Na análise do gráfico, verifica-se que durante os primeiros 16 meses não houve registo de atividade, quer em *pushes* quer em *commits*. No gráfico acumulativo, esta fase inicial é evidenciada pela estagnação representada pela linha horizontal.

Após 16 meses da data de criação do projeto, assiste-se a uma retoma da atividade, tanto de *pushes* como de *commits*. Neste contexto, a subida contínua reflete a acumulação do trabalho e o esforço investido. Enquanto o gráfico de linhas evidenciou fases de elevada atividade intercaladas com pausas, o gráfico acumulativo salienta o impacto cumulativo desses esforços ao longo do desenvolvimento do projeto.

O gráfico acumulativo revela uma trajetória ascendente e estável no total de *pushes* e *commits*. Tal demonstra que, mesmo considerando as flutuações mensais, o envolvimento no projeto manteve-se persistente. As zonas em que a curva se torna mais íngreme estão associadas aos meses com maior número de *pushes* e *commits*.

Esta representação gráfica acumulativa sublinha claramente a progressão e o empenho contínuo investidos no projeto. Mesmo em meses com registo reduzido de *commits* ou *pushes*, a curva acumulativa prossegue a sua ascensão, enfatizando a evolução sustentada.

A análise da métrica de *commits* ao longo dos meses revela um padrão de crescimento e envolvimento no projeto em estudo. O aumento da atividade de *commits* pode ser um indicador da maturidade do projeto, do envolvimento da comunidade e da capacidade de resposta às necessidades e *feedback* dos utilizadores. Esta métrica, comparada a outras, permite entender melhor o ciclo de vida e a saúde geral do projeto.

- **Número de *Pull Requests***

Pull Requests (PRs) são uma parte essencial do desenvolvimento colaborativo em projetos hospedados em plataformas como o GitHub. Eles indicam contribuições propostas para o código principal e passam por fases de revisão antes de serem incorporados ao repositório. Analisar a evolução e o estado das *pull requests* no projeto, pode oferecer *insights* sobre o envolvimento da comunidade, a agilidade da equipa de manutenção e a saúde geral de um projeto.

A análise da evolução dos *Pull Requests* (PRs), observada na Figura 71, no âmbito do projeto de *Open Source Hardware* proporciona uma visão aprofundada dos padrões de contribuição e dos mecanismos subjacentes de revisão e integração. Esta métrica permite discernir a dinâmica de colaboração, a recetividade do projeto a novas propostas e o ritmo de aprimoramento e inovação ao longo do tempo.

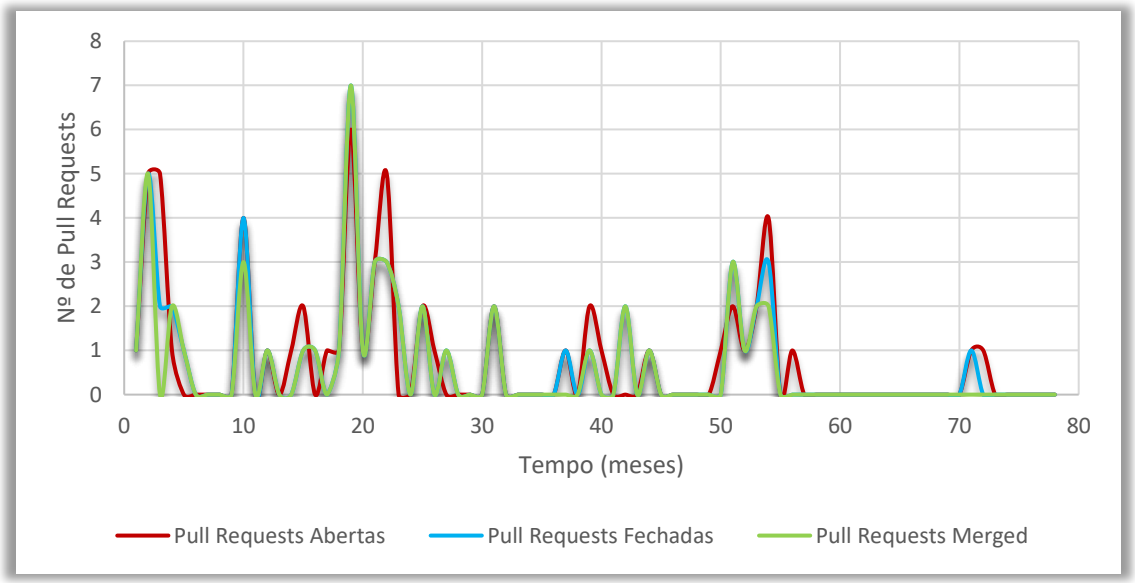


Figura 71 - Representação Gráfica da Variação Mensal do Número de *Pull Requests* no Projeto OSH.

A análise da evolução dos *Pull Requests* (PRs) do projeto evidencia uma dinâmica singular das contribuições. Em grande parte da duração do projeto, a atividade de PRs é caracteristicamente esporádica, refletindo as particularidades associadas ao desenvolvimento de *hardware*. Estas particularidades abrangem desde a necessidade de prototipagem meticulosa até à realização de testes físicos e validações subsequentes.

Nos primeiros meses, observa-se um aumento na atividade, o que pode ser reflexo do arranque entusiástico do projeto. Ao longo do tempo, é visível uma correlação próxima entre PRs abertas, fechadas e integradas ao projeto. Este padrão sugere que a comunidade está ativa e que a maioria das sugestões e contribuições são levadas em consideração e integradas após um processo de revisão detalhado.

No âmbito de um projeto de hardware de código aberto, é imperativo que a integridade e funcionalidade do design sejam mantidas. Assim, PRs que não cumpram com os padrões exigidos ou que apresentem soluções redundantes podem não ser integradas. A necessidade de escrutínio técnico, viabilidade de fabrico e considerações de custo acrescenta uma camada adicional de análise, podendo justificar a natureza pontual dos PRs.

A recorrência de PRs, ainda que intermitente, é indicativa do compromisso da comunidade, evidenciando o interesse contínuo em aprimorar e adaptar o projeto. Este comprometimento é ainda mais relevante considerando-se os desafios inerentes à contribuição em *hardware*.

O intervalo sem contribuições entre os meses 55 e 78 é notável. Esta pausa pode ser fruto de uma fase de maturidade do projeto ou de desafios externos, como mudanças no cenário de *hardware*. Contudo, embora o número de PRs não seja elevado, cada contribuição indica um compromisso significativo. No mundo do *hardware*, isso traduz-se em horas dedicadas a *design*, prototipagem e testes.

A Figura 72 oferece uma representação acumulativa dos *Pull Requests* (PRs) ao longo do tempo, permitindo uma compreensão mais aprofundada da trajetória das contribuições ao projeto de desenvolvimento de *Open Source Hardware*. Esta visualização proporciona uma panorâmica do envolvimento da comunidade e das colaborações estabelecidas ao longo da existência do projeto.

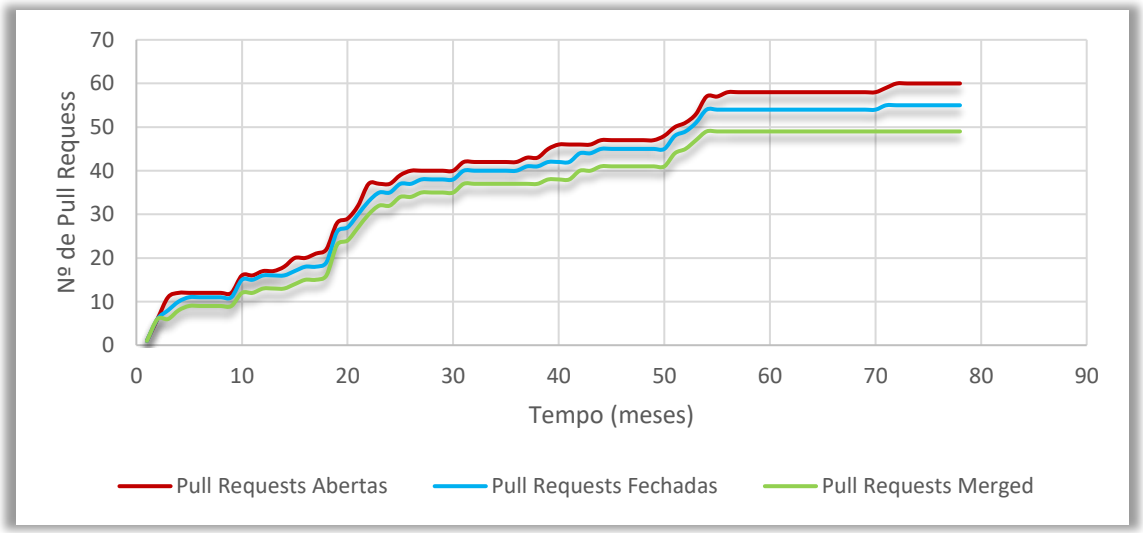


Figura 72 - Tendência Acumulada do Número de *Pull Requests* no Projeto OSH.

As trajetórias quase paralelas das PRs – abertas, fechadas e *merged* – sublinham uma comunidade envolvida e um processo de revisão diligente. Contudo, após 55 meses, identifica-se um patamar de estabilidade, que sinaliza um período notável de inércia no projeto.

Destacam-se os segmentos horizontais prolongados no gráfico, que denotam fases de inatividade ou contribuições reduzidas. Estes segmentos, particularmente acentuados a partir dos 20 meses após o início do projeto, evidenciam momentos em que o fluxo de contribuições foi limitado ou inexistente.

Os incrementos visíveis no gráfico, embora positivos, manifestam-se de forma contida e com inclinações suaves. Estes crescimentos pontuais refletem a meticulosidade intrínseca às contribuições em contextos de hardware. Cada PR submetido neste âmbito pode requerer uma avaliação rigorosa, dada a necessidade de ponderar implicações técnicas, funcionais e até económicas.

Dessa forma, a convergência das linhas indica uma gestão e integração coesas das contribuições, enquanto os períodos de estagnação reforçam a complexidade inerente a projetos desta natureza.

A análise subsequente foca-se no projeto de *Open Source Software*. Na Figura 73, observa-se uma visualização detalhada da evolução das *Pull Requests* (PRs) ao longo do tempo, nas categorias de PRs abertas, fechadas e *merged*.

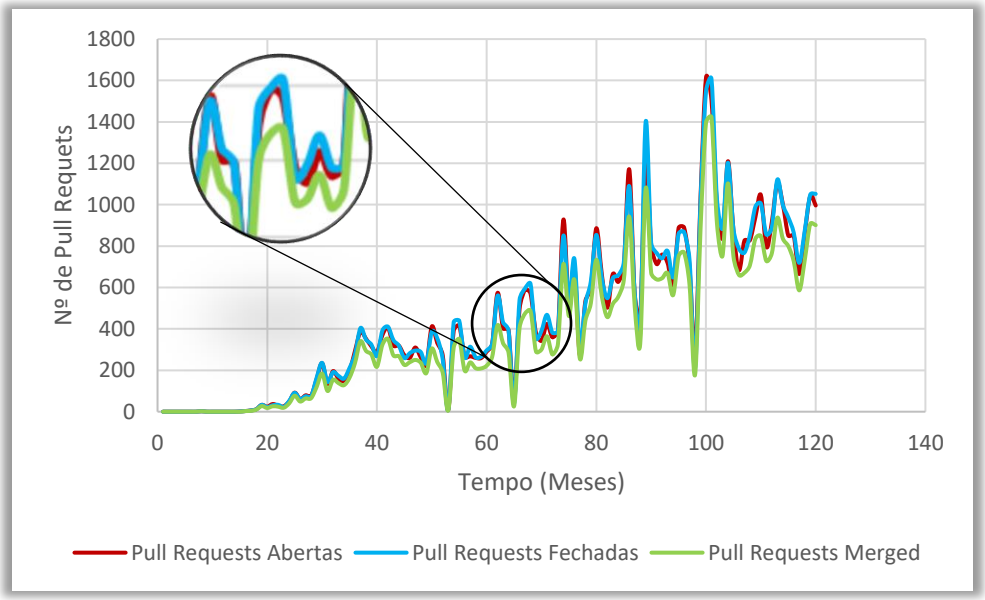


Figura 73 - Representação Gráfica da Variação Mensal do Número de *Pull Requests* no Projeto OSS.

Nos primeiros 15 meses, observa-se que pouco ou nenhum *pull request* foi criado. Este período de inatividade pode estar relacionado com uma fase de planeamento, configuração, ou até mesmo um alinhamento estratégico. Esta observação é consistente com as análise das métricas anteriores.

Após esse período inicial, nota-se um aumento na atividade, indicando o início efetivo do desenvolvimento e colaboração no repositório. O número de PRs abertos, fechados e *merged*, parecem seguir um padrão semelhante. Isto sugere que a maioria dos PRs abertos foi revisada e incorporada ao projeto num ritmo semelhante, o que é um bom indicativo de eficiência no processo de revisão.

Através da análise do gráfico, observa-se ainda que as métricas de *pull requests* abertas e fechadas acompanham-se de perto. Esta correlação é um indicador positivo para o projeto. Significa que, para cada PR aberto, há uma resposta rápida em termos de revisão e decisão, seja ela para fundir o PR ou para fechá-lo sem fusão. Uma proporção equilibrada entre estes dois valores indica um processo de revisão eficaz, onde as contribuições são ativamente gerenciadas.

O aumento do número de PRs ao longo do tempo é um indicativo de atividade crescente e envolvimento da comunidade. Mais importante ainda, o facto de que os PRs abertos e fechados andam de mãos dadas sugere que os gestores do projeto lidam eficazmente com essa atividade. As métricas de PRs, portanto, não indicam apenas a saúde e atividade do projeto, mas também refletem a eficácia dos processos e da gestão em prática. É essencial para qualquer projeto manter um equilíbrio saudável entre as PRs abertos e fechados, garantindo que as contribuições sejam revisadas e incorporadas em tempo hábil.

O gráfico de linhas acumulativas, apresentado na Figura 74, proporciona uma visão agregada da atividade de *Pull Requests* (PRs) ao longo do tempo, revelando tendências e comportamentos que podem não ser imediatamente óbvios numa visão mensal.

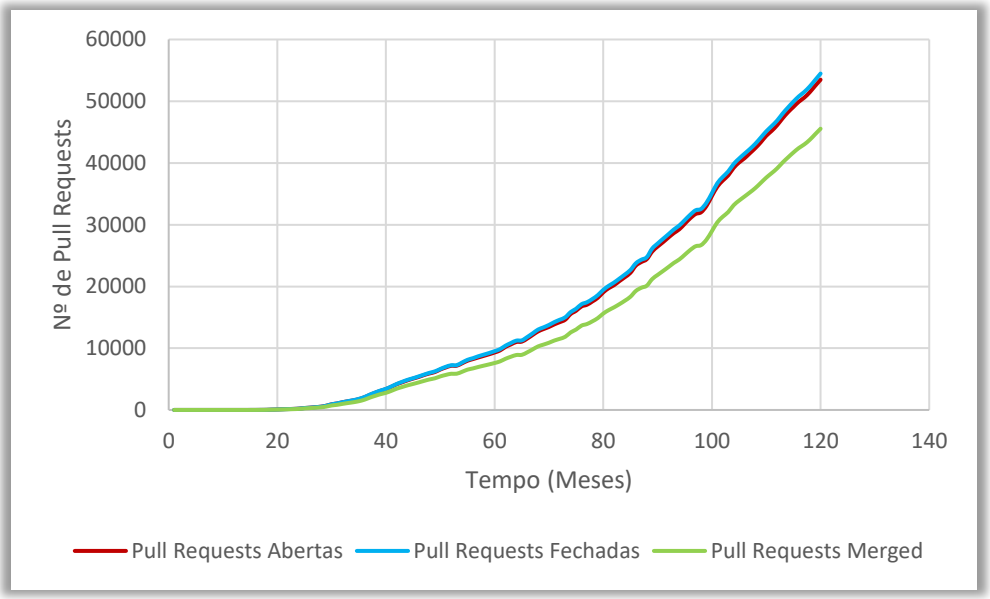


Figura 74 - Tendência Acumulada do Número de *Pull Requests* no Projeto OSS.

É notável que as linhas representativas de PRs abertos e fechados quase se sobrepõem, com a linha de PRs fechados ligeiramente acima da linha de PRs abertos. Este comportamento sugere que a maioria dos PRs abertos é prontamente atendida, seja aprovando e fundindo ou decidindo fechá-los sem efetuar o *merge* da *pull request*. O facto de a linha de PRs fechados estar ligeiramente acima, indica que alguns PRs foram fechados sem serem *merged*, o que é uma prática comum em projetos de código aberto, onde algumas contribuições podem não ser aceites por várias razões.

A proximidade das linhas de PRs abertos e fechados indica uma gestão ativa e responsiva das contribuições. Esta atividade sugere que o projeto está efetivamente a gerir a sua comunidade e a garantir que todas as contribuições são revistas em tempo hábil.

A linha representativa dos PRs *merged* encontra-se ligeiramente abaixo das linhas de PRs abertos e fechados. Esta disposição é esperada, uma vez que nem todos os PRs abertos são eventualmente *merged*. Alguns PRs podem ser fechados devido a problemas de qualidade, redundância, falta de alinhamento com os objetivos do projeto ou outros critérios definidos pela equipa de revisão.

O facto de a linha de PRs *merged* ser menor que as linhas de PRs abertos e fechados reflete um processo de revisão criterioso, onde apenas as contribuições que atendem a determinados padrões são incorporadas.

A análise do gráfico de linhas acumulativas de PRs revela uma gestão eficiente e criteriosa das contribuições ao projeto. A proximidade das linhas de PRs abertos e fechados representa uma resposta rápida às contribuições, enquanto a posição da linha de PRs *merged* reflete um processo de revisão rigoroso. Tais observações são indicativos de um projeto bem gerido, com foco na qualidade e na colaboração ativa com a comunidade.

- **Tempo Médio de *Pull Request***

O tempo médio despendido desde a criação de um *pull request* até à sua integração constitui uma métrica essencial para avaliar a eficácia do processo de revisão e incorporação de código no projeto. Um período médio reduzido pode refletir um procedimento de revisão ágil e eficiente, ao passo que um tempo mais alargado pode indicar potenciais constrangimentos ou a demanda de mais

recursos durante a revisão. Esta métrica representa, em horas, o intervalo médio entre a submissão e a integração de um *pull request*.

A Figura 75 ilustra a variação mensal do tempo médio despendido desde a criação até à integração de um *pull request* no projeto de *Open Source Hardware*. Esta representação permite uma análise pormenorizada da eficácia e agilidade do processo de revisão ao longo dos meses, possibilitando a identificação de eventuais padrões, eficiências e desafios inerentes ao desenvolvimento de hardware em código aberto.

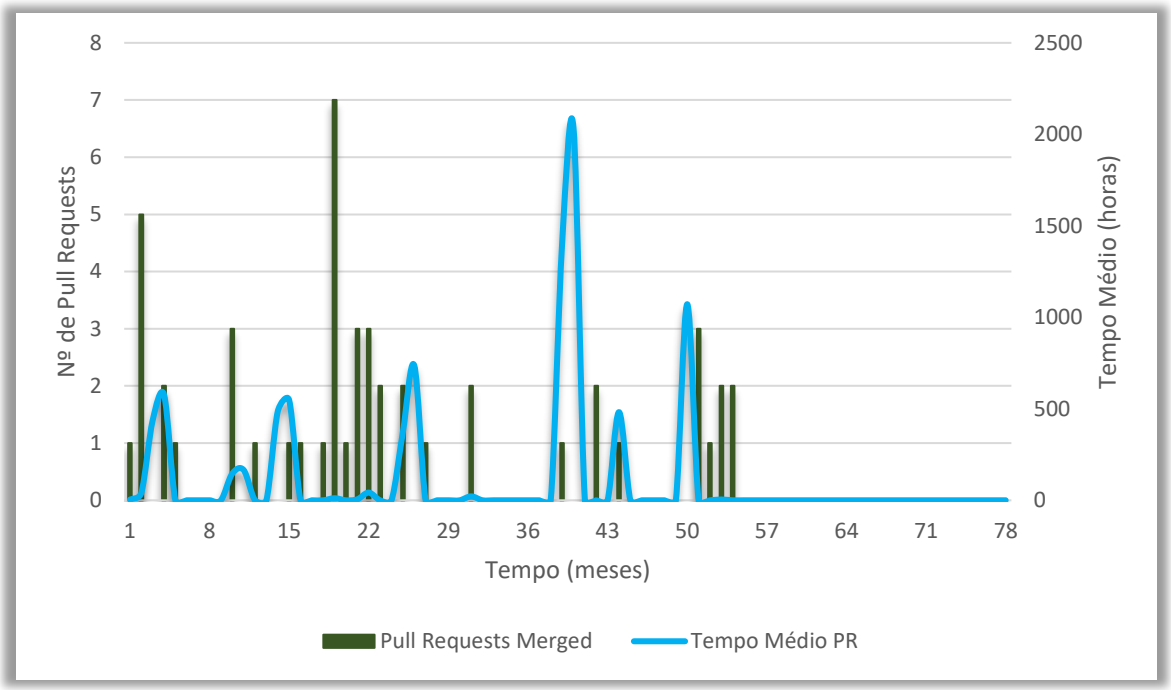


Figura 75 - Representação Gráfica da Variação Mensal do Tempo Médio de *Pull Request* no Projeto OSH.

A análise do tempo médio que uma *pull request* demora a ser integrada revela oscilações distintas ao longo do tempo. Existem fases em que o período médio se estende, podendo refletir PRs de maior complexidade ou que necessitam de revisões meticulosas antes de serem aprovadas. Em contraste, há momentos caracterizados por uma integração mais eficaz das contribuições.

Em certos intervalos temporais, um incremento na quantidade de PRs integradas coincide com um prolongamento do tempo médio de PR. Esta tendência pode insinuar que a magnitude das contribuições está intrinsecamente ligada à sua complexidade. Assim, fases com uma maior integração de contribuições podem coincidir com PRs mais complexas que visam um processo de revisão intensificado.

Dado o carácter tangível do *hardware*, as PRs neste âmbito podem abordar não apenas questões de código, mas também *designs*, esquemas e outros elementos intrínsecos ao *hardware*. Estas oscilações no tempo médio podem ser indicativas da heterogeneidade e profundidade das contribuições. Por exemplo, uma PR que sugira uma modificação num componente específico pode requerer um debate mais aprofundado e extenso comparativamente a uma simples atualização documental.

Para os colaboradores, variações prolongadas no tempo médio de revisão podem ser desmotivadoras, sugerindo uma demora na análise e integração das PRs. Torna-se, assim,

imperativo que a equipa central do projeto mantenha um diálogo fluido e transparente com os colaboradores, elucidando eventuais demoras e assegurando um tratamento adequado a todas as PRs.

Por outro lado, em certos meses, apesar do baixo volume de PRs integradas, o tempo médio ascende. Tal sugere que as contribuições durante esses períodos possam ser de natureza particularmente complexa, exigindo uma avaliação mais criteriosa. Em contraste, fases com um elevado número de PRs integradas, mas com um tempo médio reduzido, evidenciam uma eficiência notável no processo de revisão.

A dialética entre rigor e eficácia é uma constante em projetos de código aberto. Este padrão reflete a capacidade do projeto em equilibrar meticulosidade e rapidez. No contexto específico do *Open Source Hardware*, este equilíbrio é ainda mais preponderante, dada a vasta gama de contribuições possíveis, desde *software* a modificações no *design* do *hardware*, justificando, assim, as variações observadas.

Prosseguindo-se com a análise da métrica para o projeto de desenvolvimento de *Open Source Software*, observa-se na Figura 76 e Figura 77 visualizações detalhadas do tempo médio com a visualização da trajetória das *pull requests* em duas fases distintas do projeto: os primeiros 60 meses e os subsequentes 60 meses, respetivamente.

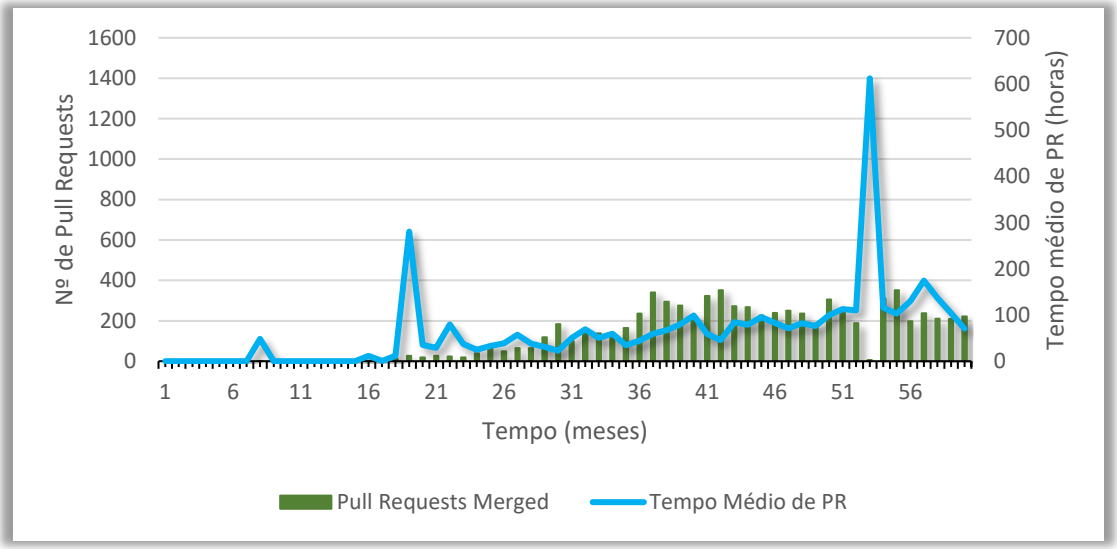


Figura 76 - Representação Gráfica da Variação Mensal do Tempo Médio de *Pull Request* no Projeto OSS.

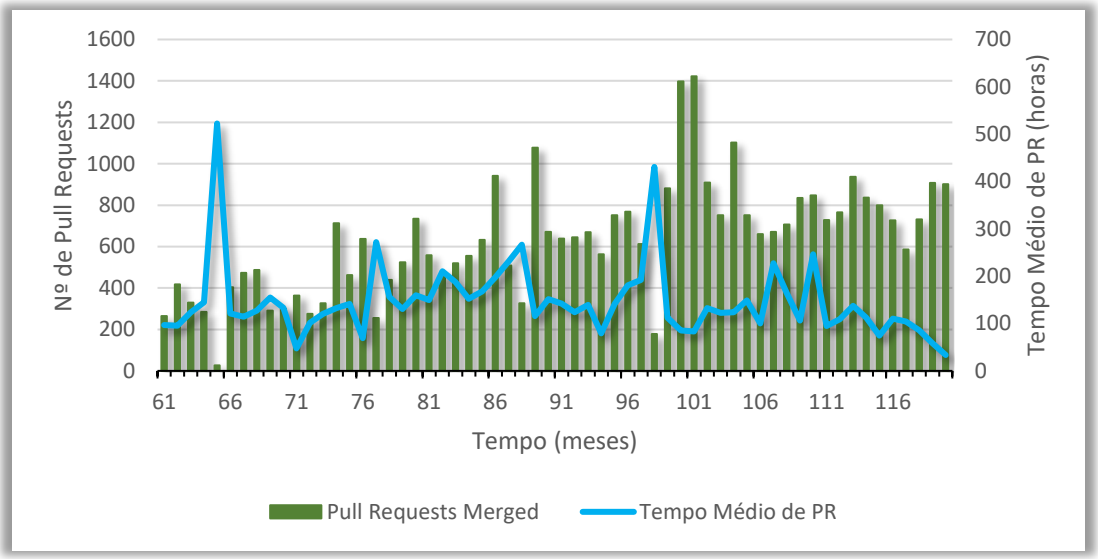


Figura 77 - Representação Gráfica da Variação Mensal do Tempo Médio de *Pull Request* no Projeto OSS.

Ao longo dos meses, observam-se variações significativas no tempo médio necessário para as *pull* requests serem *merged*. Em determinados períodos, o tempo médio apresenta-se reduzido, o que sugere uma revisão e incorporação eficaz dos PRs. Porém, noutros períodos, observa-se um aumento no tempo médio, o que pode ser causado por uma variedade de razões, como PRs mais complexos, necessidade de revisões adicionais ou períodos de alta carga de trabalho para a equipa.

Uma observação intrigante é que, em meses com menos PRs abertas, o tempo médio para realizar o *merge* de um PR tende a ser maior. Isso pode ser devido a uma variedade de razões, como a complexidade das contribuições, a necessidade de revisões mais detalhadas ou uma disponibilidade reduzida da equipa. Por outro lado, em meses com maior atividade de PRs, o tempo médio é reduzido, o que sugere uma operação eficiente e possivelmente uma dedicação maior da equipa de revisão para lidar com o volume. Esta eficiência na gestão de PRs, mesmo durante picos de atividade, é um sinal positivo de que os gestores do projeto estão a responder adequadamente às demandas.

Por outro lado, em meses de maior atividade e submissão de PRs, o tempo médio de revisão e integração é mais curto. Este padrão sugere que, perante um aumento de demanda, existe uma otimização nos processos de revisão, permitindo acomodar o maior volume de contribuições. Este comportamento é indicativo de uma gestão eficaz, demonstrando a capacidade da equipa de adaptar-se e responder a diferentes intensidades de trabalho. O fluxo mais rápido pode também ser um indicativo de PRs menos complexos ou de um processo de revisão mais otimizado durante picos de atividade.

Estas flutuações no tempo médio de revisão e de um pull ser *merged*, e a sua aparente correlação com o volume de PRs, ilustram a capacidade do projeto de ajustar os seus processos de acordo com as necessidades momentâneas. Em momentos de menor pressão, a equipa adota uma postura mais reflexiva, enquanto em fases de maior atividade se foca em manter a eficiência.

O monitoramento e análise do tempo médio de ciclo de PRs são essenciais para compreender a saúde operacional de um projeto. A capacidade de ajustar o tempo de revisão com base na demanda e complexidade dos PRs é um indicativo da maturidade e flexibilidade do projeto. Ao

equilibrar a qualidade e eficiência, o projeto demonstra a sua capacidade de gerir eficazmente as contribuições e garantir que o código integrado mantenha um alto padrão de qualidade.

- **Número de *Issues***

As *issues*, frequentemente utilizadas em plataformas como o GitHub, servem como indicadores das necessidades, problemas ou sugestões propostas pela comunidade ou pela própria equipa de desenvolvimento. A relação entre *issues* abertas e fechadas fornece *insights* sobre a eficiência da gestão do projeto, a rapidez de resposta da equipa e o envolvimento da comunidade.

No contexto do desenvolvimento de *hardware* em código aberto, as *Issues* podem variar desde problemas de *design*, incompatibilidades de componentes, desafios de fabricação até *feedback* sobre a funcionalidade e usabilidade do *hardware*. A Figura 78 ilustra a evolução do número de *issues* (tanto abertas quanto fechadas) ao longo do tempo para o projeto de *Open Source Hardware*.

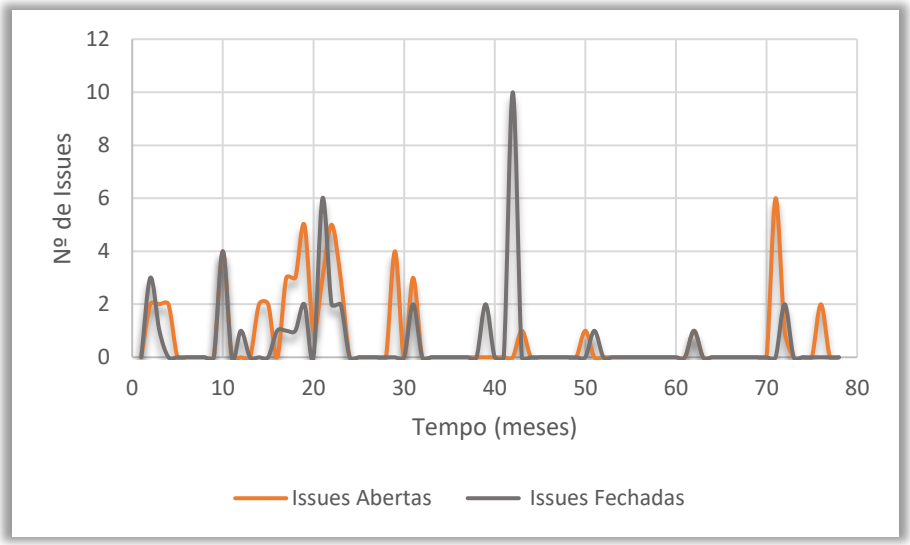


Figura 78 - Representação Gráfica da Variação Mensal do Número de *Issues* no Projeto OSH.

Ao longo dos 78 meses de evolução do projeto, observa-se uma flutuação no padrão de atividade relacionado às *issues*. Há meses caracterizados por um volume considerável de *issues*, contrastando com outros que evidenciam escassa ou nenhuma dinâmica neste aspeto.

Nos meses que apresentam atividade, não se evidencia um padrão de intensa dinâmica. Existem meses mais ativos, com picos de *issues* abertas, seguidos por vales, refletindo uma intermitência na receção de feedback ou identificação de problemas. A proximidade entre as linhas de *issues* abertas e fechadas sugere um eficiente mecanismo de gestão e resolução. Contudo, há ocasiões em que se destaca uma maior quantidade de *issues* abertas em relação às fechadas. Tal padrão sugere fases de receção intensificada de *feedback* ou identificação de desafios, cuja resolução poderá prosseguir nos meses subsequentes.

Notoriamente, em 57 dos 78 meses, não se registou a abertura de novas *issues*, e em 61 meses, não houve *issues* encerradas. Estes intervalos de aparente inatividade podem sinalizar etapas de consolidação do projeto ou momentos de foco intenso em desenvolvimento interno.

Os períodos de inatividade podem estar associados a distintas razões, como etapas em que o desenvolvimento se centra na implementação e testes, colocando a gestão de feedback em

segundo plano. Adicionalmente, a natureza tangível do *hardware* em código aberto implica uma abordagem diferenciada, frequentemente condicionada por fatores físicos e logísticos.

A Figura 79 apresenta a visualização da trajetória cumulativa de *issues* abertas, e fechadas. Esta representação oferece uma perspetiva longitudinal do envolvimento da comunidade e da resposta do projeto a esse envolvimento.

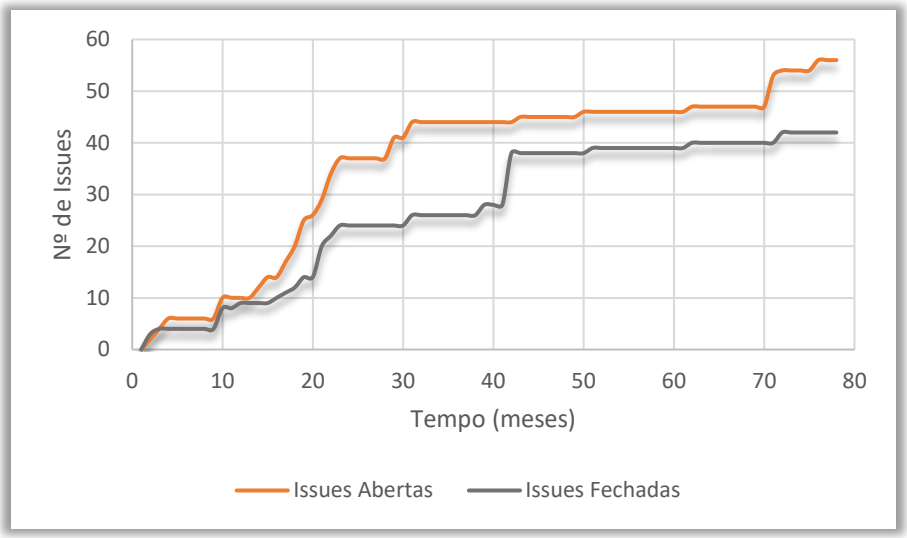


Figura 79 - Tendência Acumulada do Número de *Issues* no Projeto OSH.

No gráfico, destaca-se a progressão contida dos declives, ausentando-se de incrementos acentuados. Esta característica pode inferir que, mesmo nos períodos de dinamismo elevado, a quantificação de *issues* introduzidas ou solucionadas num determinado mês não é expressiva.

Adicionalmente, é notória a dominância de segmentos horizontais, especialmente após o vigésimo mês, sinalizando intervalos sem registo de novas *issues* ou resolução das existentes. Estes patamares no gráfico acumulativo podem aludir a etapas de consolidação no projeto, podendo sugerir que o projeto alcançou um grau de maturidade, com os desafios preponderantes já catalogados e abordados, ou momentos em que a equipa concentrou-se noutras vertentes do desenvolvimento.

Um pormenor relevante é o posicionamento da curva acumulativa de *issues* abertas face à de *issues* fechadas. A trajetória das *issues* abertas mantém-se invariavelmente acima da de *issues* fechadas, indicando um constante saldo de *issues* pendentes. Contudo, a estreita distância entre as curvas reflete um empenho contínuo na resolução das *issues*, apesar de nem todas serem prontamente solucionadas.

Assim, a tendência acumulativa sugere que o projeto que, embora receba feedbacks e identifique desafios de maneira faseada, demonstra uma resposta assídua e comprometida. Os segmentos estáveis e a proximidade entre as trajetórias de *issues* abertas e fechadas evidenciam a gestão proativa e a dedicação que dirigem este projeto no domínio do desenvolvimento de *Open Source Hardware*.

No seguimento da análise da métrica, agora, para o projeto de *Open Source Software*, apresenta-se a evolução mensal do número de *issues* abertas e fechadas na Figura 80. Uma análise detalhada deste gráfico pode revelar períodos de maior atividade, fases de realinhamento ou momentos em

que foram enfrentados desafios mais complexos. Esta visualização é, assim, crucial para compreender o compromisso da equipa de desenvolvimento e a interação com a sua comunidade ao longo do tempo.

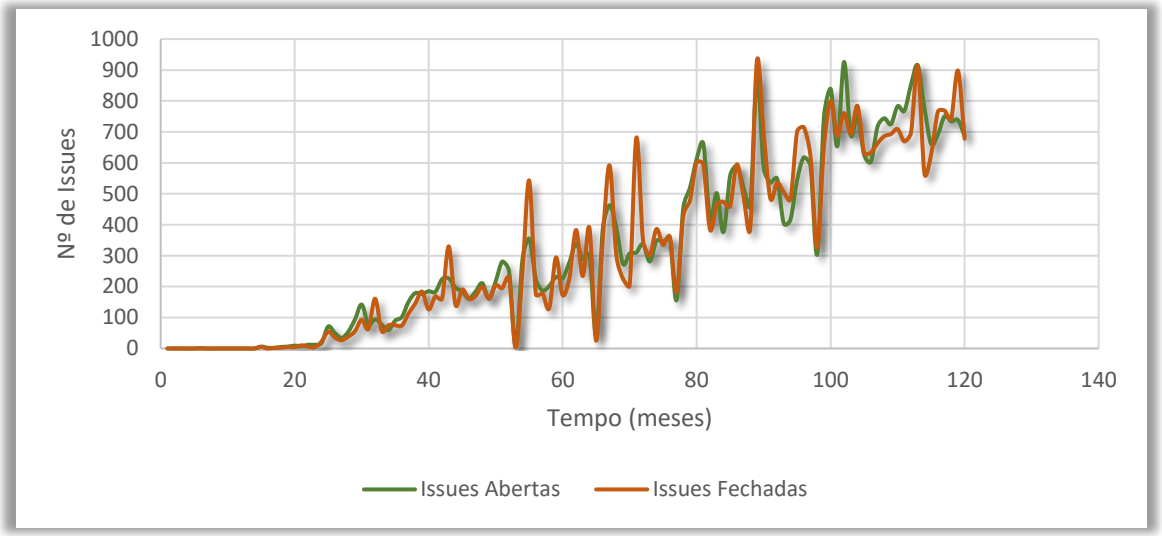


Figura 80 - Representação Gráfica da Variação Mensal do Número de *Issues* no Projeto OSS.

Nos primeiros 16 meses, identifica-se uma atividade quase inexistente, o que pode representar a fase de arranque do projeto, englobando etapas como definição de escopo, planeamento e preparação para a implementação. Esta etapa inicial é crucial, pois estabelece as bases para as fases seguintes do projeto e a sua adaptação à plataforma.

Subsequentemente, nota-se um crescimento contínuo na atividade, pautado por certos picos e vales. Apesar destas flutuações mensais, a tendência global demonstra uma gestão consistente das *issues*. Em diversos momentos, o número de *issues* abertas excede o de *issues* fechadas. Esta discrepância pode resultar de um influxo significativo de feedback da comunidade, introdução de novas funcionalidades que acarretam desafios ou uma fase de reajuste do projeto. Embora a prevalência de *issues* abertas possa suscitar preocupações iniciais, evidencia o envolvimento ativo da comunidade. Contudo, é imperativo que a equipa do projeto responda atempadamente a estas *issues* para assegurar a continuidade e sucesso do projeto.

A proximidade entre as linhas de *issues* abertas e *issues* fechadas é um indicador positivo. Sinaliza que a equipa está a abordar e resolver as *issues* de forma eficiente, quase à mesma velocidade com que são reportadas. Esta prática é sintomática de uma gestão proactiva e um comprometimento com a satisfação da comunidade. Além disso, o projeto evidencia resiliência, adaptabilidade e foco na resolução de problemas. As fases em que as linhas se aproximam indicam períodos de intensa atividade resolutiva, podendo coincidir com lançamentos, atualizações ou outros eventos marcantes no ciclo de desenvolvimento.

Globalmente, a resposta da equipa às *issues* reflete uma abordagem ativa e consistente face aos *feedbacks* da comunidade. A diferença entre issues abertas e fechadas pode ser influenciada pela complexidade das mesmas ou por um crescimento na atividade comunitária, impondo desafios acrescidos à equipa de desenvolvimento.

A Figura 81 , fornece uma perspetiva de longo prazo sobre a gestão de *issues*, permitindo aos *stakeholders* avaliar o compromisso da equipa ao longo do tempo e discernir a evolução da gestão

de *issues*. Esta perspetiva proporciona uma compreensão mais profunda da saúde e direção do projeto.

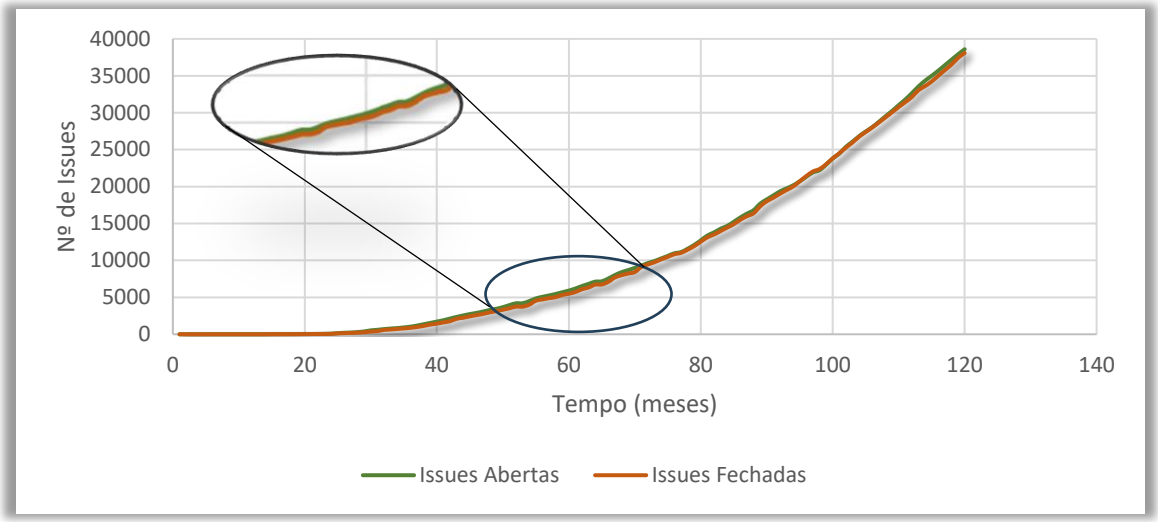


Figura 81 - Tendência Acumulada do Número de *Issues* no Projeto OSS.

Ao longo da evolução do projeto, as linhas representativas de *issues* abertas e fechadas evidenciam um crescimento ascendente contínuo e estável. Esta tendência demonstra uma atividade constante no projeto. A quase sobreposição das linhas indica uma gestão ativa e eficiente das *issues*, sugerindo que a equipa está empenhada em resolver prontamente os problemas à medida que são identificados.

Esta proximidade entre as linhas entre as *issues* abertas e fechadas é um indicador positivo. Significa que, conforme surgem novas *issues*, estas são rapidamente tratadas e resolvidas, evitando a acumulação de um vasto número de problemas por resolver. A distinção mínima entre ambas no gráfico acumulativo reforça que, para cada nova *issue* identificada, existe uma resposta quase imediata, refletindo uma gestão responsiva.

A evolução estável e a gestão ativa das issues são indicativos de um projeto bem gerido. A atenção contínua às *issues* sugere que a equipa está atenta ao *feedback* e preocupações da comunidade, garantindo que o projeto continue a evoluir de acordo com as necessidades dos utilizadores.

O crescimento sustentado do número total de *issues* ao longo dos meses é representativo de um projeto ativo e em constante evolução, com uma comunidade envolvida que procura continuamente melhorias e soluções. A proximidade entre o número de *issues* abertas e o número de *issues* resolvidas salienta que a equipa dá a devida atenção e resolve grande parte das issues em tempo útil. Contudo, é importante notar que, se o número de *issues* abertas continua a crescer, existem questões pendentes que requerem atenção.

A análise da métrica de *issues* revela um compromisso ativo da comunidade e uma gestão atenta por parte da equipa. A evolução contínua e a resolução eficaz da maioria das *issues* são indicativos de um projeto que está em permanente desenvolvimento, apoiado por uma comunidade proactiva e colaborativa. Estas métricas são essenciais para avaliar a saúde, sucesso e sustentabilidade de um projeto, uma vez que representam diretamente o *feedback* da comunidade e a capacidade de resposta da equipa.

- **Tempo Médio *Issue***

O tempo médio entre a abertura de uma *issue* e a sua primeira resposta é uma métrica crucial na avaliação da eficácia e responsividade de uma equipa de projeto. Quando os contribuidores ou utilizadores abrem uma *issue*, esperam uma resposta ou reconhecimento oportuno da parte da equipa de desenvolvimento. Este tempo de resposta pode influenciar a perceção do contribuidor em relação ao compromisso da equipa com a comunidade e à qualidade geral do suporte.

A análise conjunta do número de *issues* e do tempo médio de resolução fornece insights valiosos sobre o desempenho da equipa e a saúde do projeto. Uma equipa que responde prontamente às *issues*, mantendo um tempo médio de resolução estável, é um indicador de um projeto bem gerido. Este equilíbrio entre a quantidade de *issues* e a rapidez na resolução é crucial para a sustentabilidade e sucesso de um projeto.

A evolução mensal do tempo médio entre a abertura de uma *issue* e a sua primeira resposta e número de *issues* abertas observa-se na Figura 82.

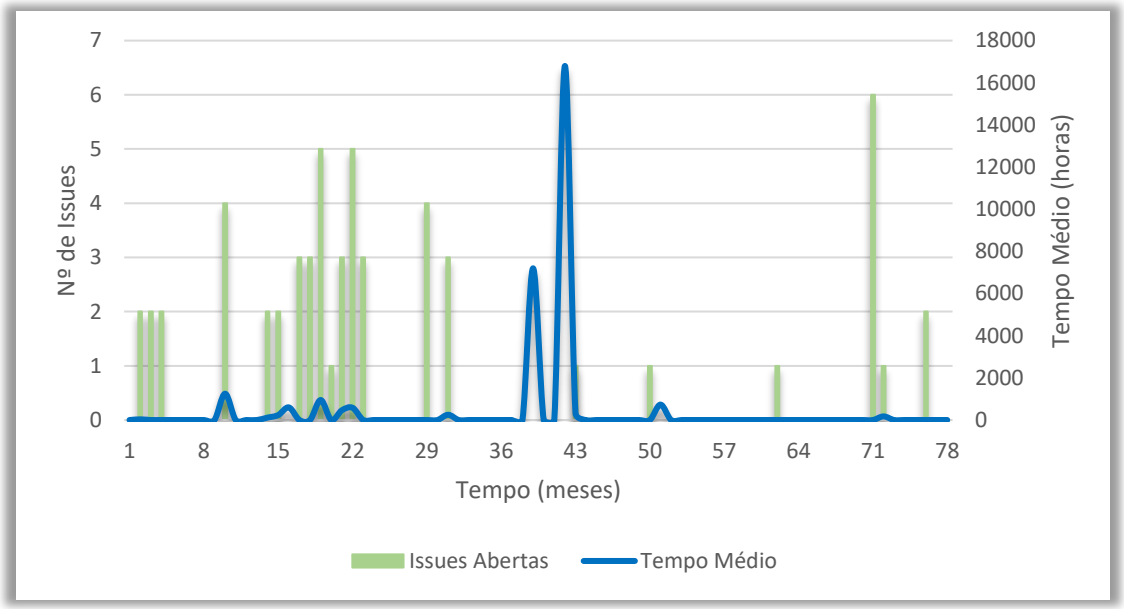


Figura 82 - Representação Gráfica da Variação Mensal do Tempo Médio *Issue* no Projeto OSH.

Em determinados períodos, observa-se que um crescimento no número de *issues* abertas corresponde a um incremento no tempo médio de resolução. Esta relação sugere que, em fases de elevado *feedback* ou deteção de problemas, cada *issue* necessita de um período alargado para a sua resolução. Tal correlação pode refletir a complexa natureza das *issues* identificadas ou um acréscimo que requer uma abordagem mais prolongada.

Por outro lado, existem momentos em que, mesmo com um volume considerável de *issues* abertas, o tempo médio revela-se reduzido. Estes momentos denotam eficácia no projeto, aludindo a uma equipa que prontamente atende às *issues*, as quais poderão ter um grau de complexidade menor ou soluções mais diretas.

No âmbito do desenvolvimento de *hardware* em código aberto, estas oscilações assumem uma interpretação adicional. O carácter palpável do *hardware* implica que qualquer *issue* pode requerer revisões no *design*, prototipagem ou ajustes nos processos produtivos. Assim, um prolongamento no tempo médio pode refletir *issues* intrinsecamente ligadas ao desenho ou funcionalidade base

do *hardware*. Em contraste, *issues* prontamente solucionadas podem estar associadas a aspetos documentais, propostas de aperfeiçoamento ou compatibilidades.

É imperativo, para a vitalidade e êxito do projeto, manter um diálogo transparente com a comunidade, especialmente quando algumas *issues* exigem períodos dilatados para resolução. A clareza acerca dos obstáculos e dos tempos de atendimento fortalece a confiança e participação da comunidade.

O comportamento do tempo médio de *issue* e o volume de *issues* abertas atestam uma gestão diligente das mesmas. A equipa responde assertivamente ao *feedback*, mas a complexidade e tangibilidade do *hardware* podem moldar o tempo requerido para solucionar certas *issues*.

Em conclusão, a inter-relação entre as métricas - *Issues* Abertas e Tempo Médio de *Issue* - revela um projeto em contínua adaptação, que procura responder e evoluir perante os desafios surgidos. A interação entre estas métricas não só espelha a robustez do projeto, mas também as especificidades e desafios inerentes à gestão de um projeto de Open Source Hardware.

Prossegue-se agora para análise da métrica no projeto *de Open Source Software*. A Figura 83 apresenta um gráfico que ilustra a relação entre o número de *issues* abertas e o tempo médio de resolução ao longo de um período dos primeiros 60 meses de desenvolvimento do projeto. Este gráfico oferece uma visão clara sobre como o projeto evoluiu no que se refere à gestão de *issues*, fornecendo uma ferramenta valiosa para avaliação e planeamento estratégico.

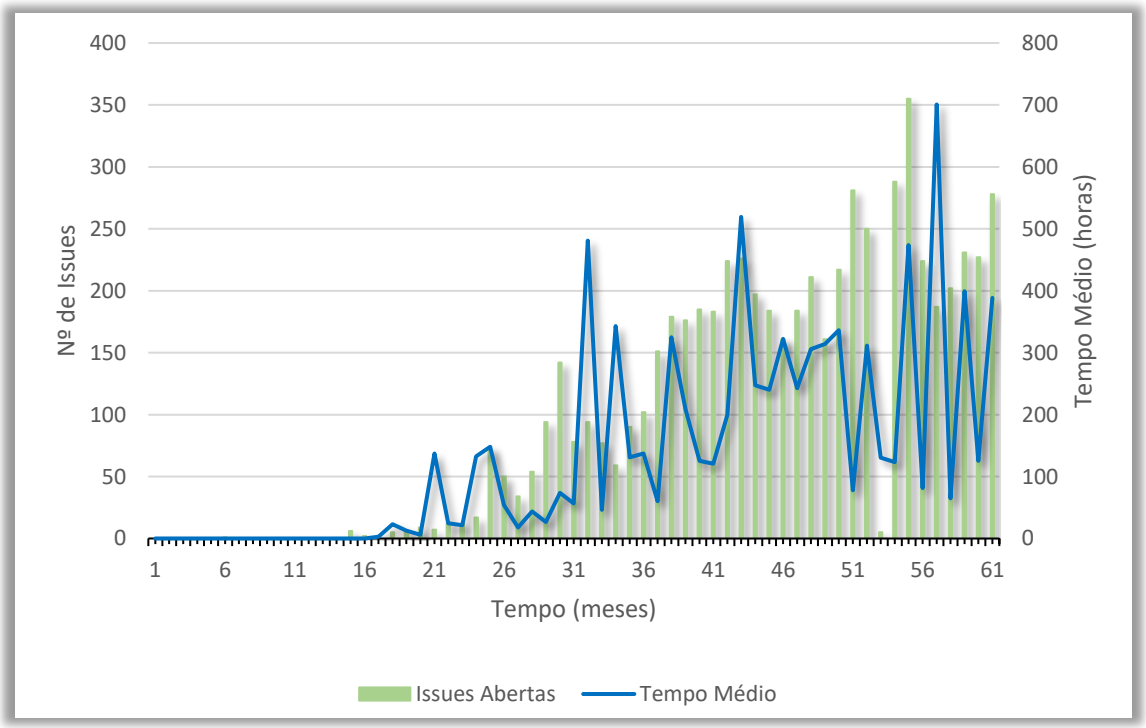


Figura 83 - Representação Gráfica da Variação Mensal do Tempo Médio *Issue* no Projeto OSS.

Durante os primeiros 16 meses do projeto, a atividade em termos de *issues* abertas e fechadas é praticamente inexistente, refletindo em um tempo médio de resposta nulo. A partir do mês 16, há um ligeiro aumento na atividade, mas é somente por volta do mês 21 que se torna visível um crescimento significativo no número de *issues* abertas. Isso indica que o projeto começou a ganhar tração e a atrair maior interação da comunidade.

O comportamento oscilante do tempo médio de resposta é notório e, em muitos momentos, parece contraintuitivo. Por exemplo, entre os meses 21 e 25, apesar do aumento expressivo no número de *issues* abertas, o tempo médio de resposta também cresce. Essa tendência sugere que, embora o projeto estivesse a atrair mais atenção e contribuições, a equipa poderia estar a enfrentar desafios na gestão eficiente do volume crescente de *issues*.

Por volta do mês 32, ocorre um pico no tempo médio de resposta, mesmo com um número de *issues* abertas não tão elevado em comparação com os meses subsequentes. Isso pode indicar que as *issues* apresentadas nesse período eram particularmente desafiadoras ou que a equipa estava sobrecarregada.

Entre os meses 32 e 43, há uma tendência clara de aumento no número de *issues* abertas, mas o tempo médio de resposta oscila. Por volta do mês 43, ocorre outro pico notável no tempo médio, que pode ser explicado pelo acúmulo de *issues* mais complexas ou por limitações de recursos da equipa.

A partir do mês 40 até o mês 60, o número de *issues* abertas tende a estabilizar, com pequenas oscilações. No entanto, o tempo médio de resposta continua a mostrar variações significativas, sugerindo que a natureza e a complexidade das *issues* podem ter variado consideravelmente de mês para mês.

A Figura 84 apresenta um gráfico que ilustra a relação entre o número de *issues* abertas e o tempo médio de resolução ao longo de um período entre os 61 a 112 meses.

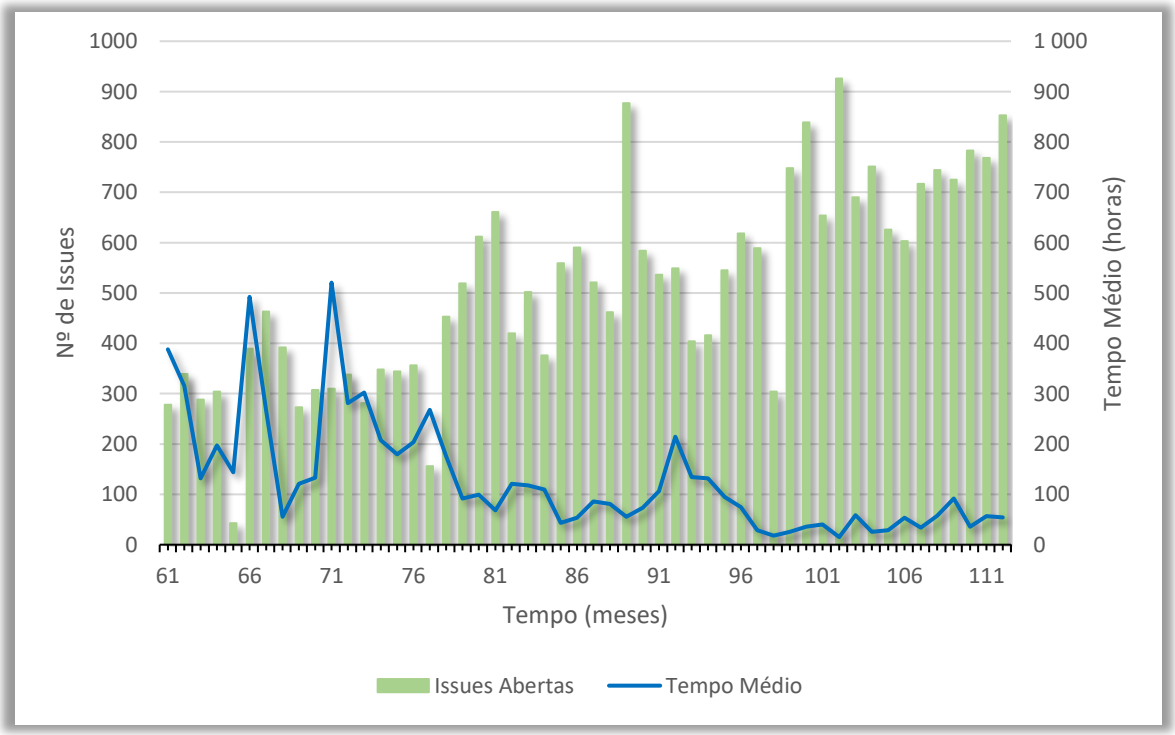


Figura 84 - Representação Gráfica da Variação Mensal do Tempo Médio *Issue* no Projeto OSS.

Durante os meses 61 a 70, registou-se uma tendência ascendente no número de *issues* abertas. Paralelamente, o tempo médio de resposta cresceu, atingindo um ápice no mês 66.

Nos meses 71 a 111, é um período dinâmico para o projeto. O número de *issues* abertas continua a aumentar, com ligeiras flutuações. e por outro lado, o tempo médio de resposta apresenta oscilações consideráveis, contudo já apresenta um declínio gradual, sugerindo uma melhoria na gestão das *issues*.

A Figura 85 apresenta um gráfico que ilustra a relação entre o número de *issues* abertas e o tempo médio de resolução ao longo de um período entre os 111 a 120 meses.

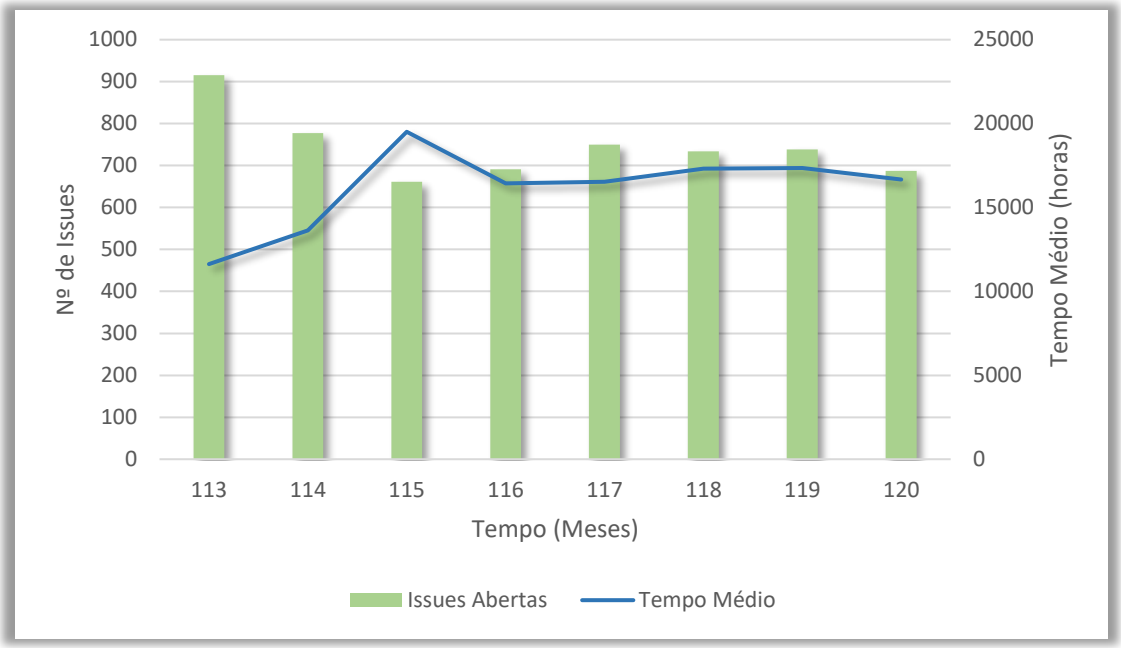


Figura 85 - Representação Gráfica da Variação Mensal do Tempo Médio *Issue* no Projeto OSS.

Entre os meses 111 e 120, continuou a verificar-se um elevado número de *issues*. Porém, o tempo médio de resposta evidenciou um aumento expressivo, o que pode indicar questões particularmente desafiantes ou fatores externos que impactaram a eficiência da equipa.

As flutuações observadas podem dever-se a diversos fatores:

- Complexidade das *Issues*: Questões mais complexas podem exigir uma análise mais meticulosa, aumentando o tempo de resposta.
- Priorização de *Issues*: A equipa pode ter focalizado determinadas *issues* em detrimento de outras, influenciando o tempo médio.
- Recursos da Equipa: A capacidade e disponibilidade da equipa, variáveis ao longo do tempo, influenciam diretamente esta métrica.
- Aprendizagem e Adaptação: Com a acumulação de experiência, a equipa pode otimizar o tempo de resposta.
- Natureza das *Issues*: A diversidade das *issues*, sejam simples sugestões ou falhas críticas, pode influenciar as flutuações observadas.

Dessa forma, um menor número de *issues* abertas geralmente correlaciona-se com um tempo de resposta mais extenso, interpretado como uma dedicação a *issues* complexas. Por outro lado, um número elevado de *issues* abertas está frequentemente associado a tempos de resposta mais curtos, refletindo um esforço concentrado da equipa.

4.1.2. Análise das Matrizes de Correlação e Inter-relações entre Métricas

No contexto da análise, a correlação é uma medida que descreve a extensão da relação linear entre duas variáveis quantitativas. Por outras palavras, esta oferece uma visão sobre como uma variável pode mudar quando outra variável muda. A matriz de correlação, por sua vez, é uma tabela onde cada célula representa a correlação entre duas variáveis específicas.

A matriz de correlação gerada considera todos os meses disponíveis nos dados. Dessa forma, as correlações refletem as relações entre as métricas ao longo de todo o período coberto pelo conjunto de dados.

O coeficiente de correlação, frequentemente representado por r, varia entre -1 e 1, com valores próximos a 1 indicando uma correlação positiva forte, valores próximos a -1 indicando uma correlação negativa forte, e valores próximos a 0 indicando pouca ou nenhuma correlação linear. Especificamente:

- $r = 1$: Correlação positiva perfeita. Significa que as variáveis se movem exatamente na mesma direção.
- $r = -1$: Correlação negativa perfeita. As variáveis movem-se exatamente na direção oposta.
- $r = 0$: Sem correlação linear. As variáveis não mostram uma tendência clara de movimento conjunto.
- $0 < r < 1$: Correlação positiva. Quanto mais próximo de 1, mais forte a correlação.
- $-1 < r < 0$: Correlação negativa. Quanto mais próximo de -1, mais forte a correlação.

É fundamental entender que a correlação não implica causalidade. Uma correlação alta entre duas variáveis não significa que uma causa a outra, apenas que elas tendem a se mover juntas. Além disso, é importante ter cuidado ao interpretar correlações em situações onde múltiplos fatores podem influenciar as variáveis em estudo.

Na análise realizada, é abordada a matriz de correlação, instrumento crucial para identificar e compreender as inter-relações entre as distintas métricas avaliadas. Inicialmente, a análise centrar-se-á na matriz de correlação relativa ao projeto de *Open Source Hardware*. Posteriormente, proceder-se-á à análise da matriz correspondente ao projeto de *Open Source Software*, possibilitando, desta forma, uma comparação estruturada entre as dinâmicas inerentes a ambos os projetos.

A Figura 86 apresenta a matriz de correlação para o projeto de *Open Source Hardware*, para explorar as relações entre as diferentes métricas do projeto, ajudando a identificar padrões e tendências que podem não ser imediatamente óbvios.

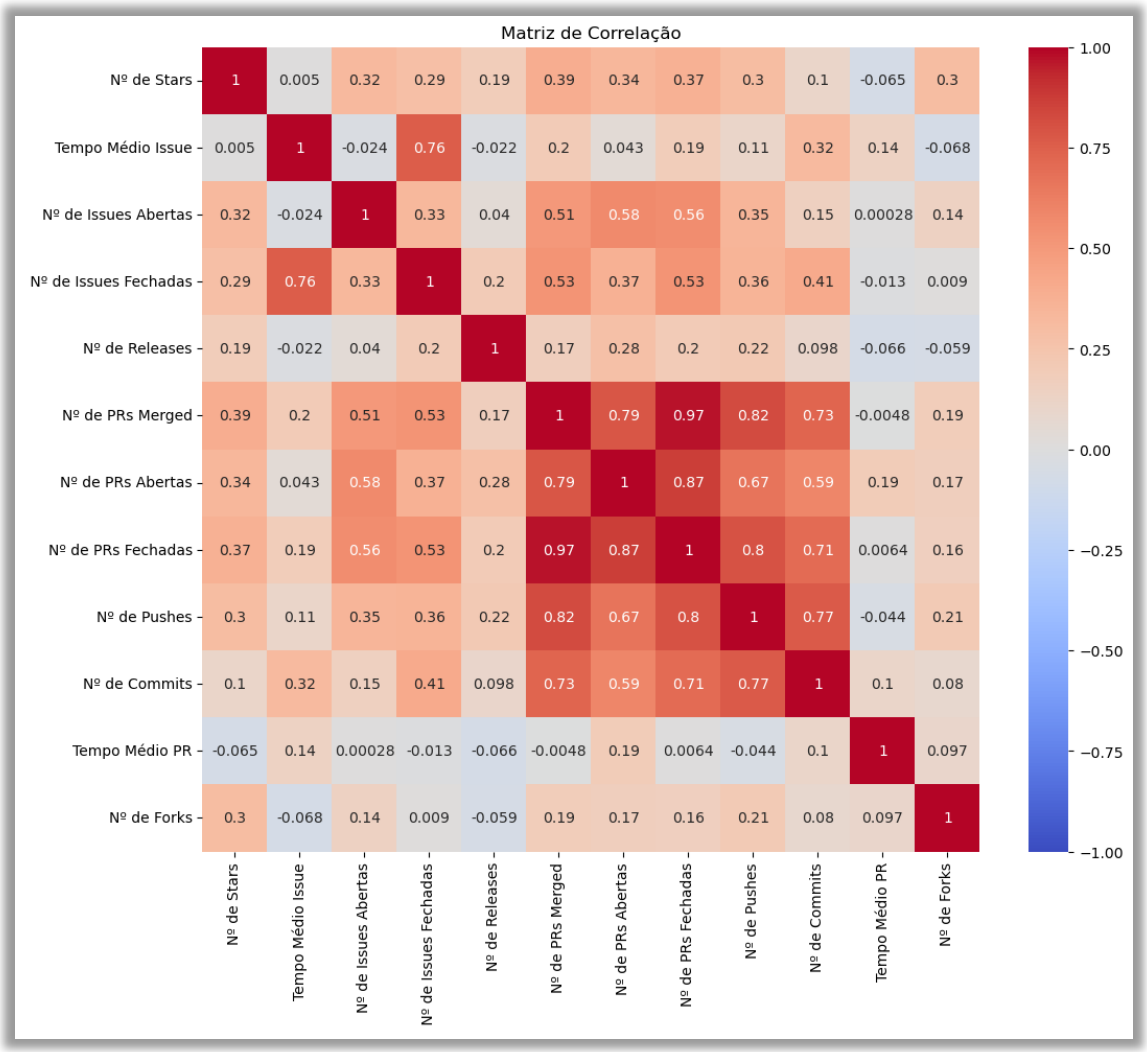


Figura 86 - Matriz de Correlação das Métricas Analisadas no Projeto OSH.

- **Correlação entre Número de *Forks*, *Pull Requests* e *Issues* Abertas**

A análise dos coeficientes de correlação fornecidos revela uma relação positiva, mas fraca, entre o número de *forks* e as várias métricas de *Pull Requests* e *Issues* para o projeto de *Open Source Hardware*. Concretamente, à medida que mais entidades procedem à criação de *forks* do projeto, verifica-se uma ligeira tendência para o aumento no número de PRs abertos, fechados e *merged*, bem como no número de *issues* abertas.

A correlação entre o número de *forks* e os PRs abertos é de 0,17, indicando que, embora haja uma associação positiva, o impacto dos *forks* no número de PRs abertos é limitado. Uma tendência semelhante é observada entre o número de *forks* e os PRs fechados, com um coeficiente de 0,16. Entre estas métricas relativas a PRs, a correlação mais forte é com os PRs *merged*, apresentando um coeficiente de 0,19, o que sugere que os projetos com um número superior de *forks* têm uma ligeira inclinação para ter mais PRs que são efetivamente incorporados no projeto.

Em relação às *issues* abertas, a correlação com o número de *forks* é de 0,14, sendo a mais fraca entre as métricas analisadas. Ainda assim, esta relação positiva sugere que um aumento no número de *forks* pode estar associado a um aumento substancial no número de *issues* abertas.

Apesar de todas as correlações serem positivas, indicando relações na mesma direção, a sua magnitude é modesta. Tal evidencia que existem outros fatores que influenciam a abertura, o fecho e a integração de PRs, assim como a criação de *issues*. Assim, o número de *forks*, por si só, não é um indicador determinante da atividade ou do envolvimento da comunidade no projeto.

- **Correlação entre *Pull Requests* e *Issues***

Analisando a correlação entre as métricas de *pull requests* e *issues* para o projeto de Open Source Hardware, é possível inferir algumas dinâmicas notáveis.

Em primeiro lugar, a correlação entre o número de *issues* abertas e os *pull requests* apresenta coeficientes relativamente fortes. Especificamente, a relação entre *issues* abertas e *pull requests* abertos tem um coeficiente de 0,58, demonstrando uma associação positiva e significativa entre estas duas métricas. De forma semelhante, a correlação entre as *issues* abertas e os *pull requests* fechados é de 0,56, e com os *pull requests merged* é de 0,51. Estes valores indicam que, à medida que mais issues são abertas, existe uma tendência notável para que se observem mais *pull requests* abertos, fechados e *merged*. Isto pode ser interpretado como um sinal de que o projeto está ativo na gestão de *feedback* e sugestões da comunidade.

Relativamente às *issues* fechadas, a correlação com os *pull requests* abertos é de 0,37, sendo esta uma relação positiva, mas de intensidade moderada. Já a relação entre *issues* fechadas e *pull requests* fechados é mais marcada, com um coeficiente de 0,53, valor este que é idêntico quando comparamos *issues* fechadas com *pull requests merged*. Estes coeficientes sugerem que, quando as *issues* são resolvidas, há uma boa probabilidade de que se observe também uma conclusão efetiva das *pull requests*, seja pelo seu fecho ou a sua integração no projeto.

Estas correlações evidenciam uma interação dinâmica entre as *issues* e os *pull requests* no contexto do projeto. A presença de coeficientes de correlação relativamente significativos, em especial entre as *issues* abertas e as métricas de *pull requests*, sublinha a proatividade e o compromisso da equipa em responder às preocupações e sugestões da comunidade. Contudo, é crucial considerar que, apesar das correlações serem indicativas de tendências, a causa direta entre estas métricas pode ser influenciada por diversos outros fatores inerentes ao projeto.

- **Correlação entre *Pushes* e *Commits***

Examinando a correlação entre as métricas de *pushes* e *commits* para o projeto de *Open Source Hardware*, observa-se um coeficiente significativamente elevado de 0,77. Esta forte correlação positiva sugere uma relação intrínseca entre estas duas atividades no âmbito do projeto.

Especificamente, este elevado coeficiente indica que, à medida que o número de *commits* aumenta, há uma tendência muito clara para que se verifique também um aumento no número de *pushes*. Esta associação pode ser interpretada como uma prática consistente no desenvolvimento: após realizar uma série de *commits* que representam progressos incrementais, estes são consolidados e integrados no repositório principal através de *pushes*.

A proximidade deste valor a 1 sublinha a robustez desta relação, reforçando a ideia de que os *commits* e os *pushes* são ações complementares no fluxo de trabalho do projeto. Em contextos de *Open Source Hardware*, onde a precisão e a meticulosidade são cruciais, esta correlação salienta a importância de manter um registo consistente e regular das atualizações e das incorporações no repositório.

- **Correlação entre Número de *Releases*, *Forks* e *Issues* Abertas**

Analisando a correlação entre o número de *releases*, *forks* e *issues* abertas no contexto do projeto, surgem observações interessantes. A correlação entre o número de *releases* e *forks* é de −0,059. Esta correlação negativa, embora muito próxima de zero, sugere que não existe uma relação forte entre estes dois parâmetros. O valor quase nulo indica que a frequência de *releases* do projeto não tem uma influência direta e significativa sobre o número de *forks*. Em termos práticos, isto pode significar que as *releases* não são o principal motivador para a criação de *forks* por parte da comunidade.

Em relação à correlação entre o número de *releases* e *issues* abertas, que é de 0,04, esta é positiva, mas muito próxima de zero. Esta proximidade ao zero sugere que, novamente, não existe uma associação significativa entre estas duas métricas. Assim, a quantidade de *releases* não parece influenciar diretamente a criação de novas *issues.*

Deste modo, pode-se inferir que, no contexto deste projeto de *Open Source Hardware*, as *releases* operam de forma relativamente independente, sem um impacto direto e marcante sobre a atividade de *forks* ou a criação de *issues* abertas. Esta característica pode refletir uma estratégia de gestão de projeto na qual as *releases* são definidas com base em critérios internos, sem serem fortemente influenciadas pela dinâmica externa da comunidade.

- **Correlação entre Número de Estrelas e Número de *Forks***

Ao analisar a correlação entre o número de estrelas e o número de *forks* no contexto do projeto, observa-se que correlação entre o número de estrelas e *forks* é de 0,3. Esta correlação positiva indica que há uma relação moderada entre estas duas métricas. Em termos práticos, à medida que são atribuídas mais estrelas ao projeto (indicativo de popularidade ou apreciação na plataforma), observa-se também um aumento no número de *forks* (indicativo de interesse em trabalhar ou adaptar o projeto). No entanto, é importante notar que a correlação, sendo moderada, sugere que outros fatores podem também influenciar o número de *forks.*

Este valor de correlação pode sugerir que, enquanto a popularidade do projeto (refletida pelo número de estrelas) tem alguma influência sobre a atividade de *forks*, não é o único determinante. Existem outros aspetos ou características do projeto que podem estar a motivar os utilizadores a criarem *forks.*

Enquanto o reconhecimento e apreço pela comunidade, evidenciado pelo número de estrelas, parece influenciar a atividade de *forks*, é evidente que o projeto possui outras qualidades ou características que atraem a atenção e interesse dos contribuidores.

- **Correlação entre Tempo Médio de *Issue* e Número de *Issues***

Analisando a correlação entre o tempo médio de *issue* e o número de *issue*s abertas e fechadas, pode-se destacar que a correlação entre o tempo médio de *issue* e o número de *issues* abertas é de -0,024. Este valor próximo de zero indica uma correlação muito fraca, quase inexistente, entre estas duas métricas no contexto do projeto. Em termos práticos, o tempo que uma *issue* leva para ser tratada não parece ser diretamente influenciado pelo volume de *issues* que são abertas em determinado período.

Por outro lado, a correlação entre o tempo médio de *issue* e o número de *issues* fechadas é de 0,76. Este valor demonstra uma correlação forte e positiva, o que sugere que, à medida que o número

de *issues* fechadas aumenta, o tempo médio que uma *issue* leva para ser resolvida também tende a aumentar. Esta observação pode indicar que, em períodos de maior atividade de resolução de *issue*s, o tempo de resposta e tratamento das mesmas é superior. Esta situação pode ser reflexo da complexidade ou da natureza das *issues* tratadas ou de uma maior carga de trabalho no projeto.

Enquanto o número de *issues* abertas não parece ter um impacto significativo no tempo médio de resolução, o número de *issues* fechadas apresenta uma influência notável. Isto sugere que, no contexto deste projeto, a gestão e resolução de problemas são aspetos críticos que exigem atenção e recursos significativos, especialmente em períodos de maior atividade.

- **Correlação entre Tempo Médio de PR e Número de PRs Abertas**

A análise da correlação entre o tempo médio de *pull request* e o número de *pull requests* abertas revela um coeficiente de 0,19. Esta correlação positiva, embora moderada, sugere que há uma relação entre estas duas métricas no contexto do projeto de *Open Source Hardware*.

Especificamente, um coeficiente de 0,19 indica que, à medida que o número de *pull requests* abertas aumenta, há também uma tendência, ainda que ligeira, para que o tempo médio de uma PR aumente. Isto pode ser interpretado como um indicativo de que, em momentos em que se verifica um aumento no número de contribuições propostas, o projeto pode necessitar de mais tempo para avaliar, discutir e integrar cada uma dessas contribuições.

Ainda assim, este valor sugere que a dinâmica e a gestão das *pull requests* no projeto são influenciadas pela quantidade de contribuições recebidas. Em períodos de maior atividade contributiva, poderá ser necessário despender mais tempo e recursos para garantir uma avaliação adequada de cada PR.

- **Correlação entre Número de Estrelas e Atividade de Desenvolvimento (*Commits* e *Pushes*)**

A análise da correlação entre o número de estrelas e diversas métricas de atividade de desenvolvimento oferece uma compreensão mais profunda da relação entre o reconhecimento do projeto e o seu dinamismo.

Existe uma correlação de 0,1 entre o número de estrelas e *commits*. Esta relação positiva, embora fraca, sugere que à medida que o projeto se torna mais ativo em termos de contribuições ao código (*commits*), há uma tendência para um ligeiro aumento no seu reconhecimento, refletido pelo número de estrelas.

Relativamente aos *pushes*, a correlação com o número de estrelas é de 0,3. Esta relação moderada indica que os períodos de intensa atividade em integrações no repositório principal (*pushes*) estão associados a um reconhecimento mais pronunciado do projeto.

A correlação entre o número de estrelas e *issues* abertas é de 0,32, e com *issues* fechadas é de 0,29. Estes valores refletem uma relação moderada, sugerindo que à medida que o projeto intensifica a sua gestão de *issues*, há um correspondente aumento no seu reconhecimento.

Em relação às *pull requests*, observa-se uma tendência consistente: o número de estrelas possui uma correlação de 0,34 com *pull requests* abertas, 0,37 com *pull requests* fechadas e 0,39 com *pull requests merged*. Estas correlações moderadas indicam que o projeto, ao gerir e integrar contribuições da comunidade, tende a ganhar um reconhecimento crescente.

Estas correlações revelam que tanto a atividade direta de desenvolvimento (como *commits* e *pushes*) como a gestão comunitária (representada por *pull requests* e *issues*) influenciam positivamente o reconhecimento do projeto. O número de estrelas, como indicador de popularidade e reconhecimento, reflete a dinâmica e a recetividade do projeto perante a sua comunidade.

Após a análise das correlações das métricas observadas no projeto de *Open Source Hardware*, procedeu-se ao seguimento da análise para o projeto *Open Source Software*. Para tal, recorreu-se à matriz de correlação. Na Figura 87, apresenta-se a matriz de correlação relativa ao projeto de *Open Source Software*, permitindo uma comparação e contraste com os padrões identificados no projeto anterior.

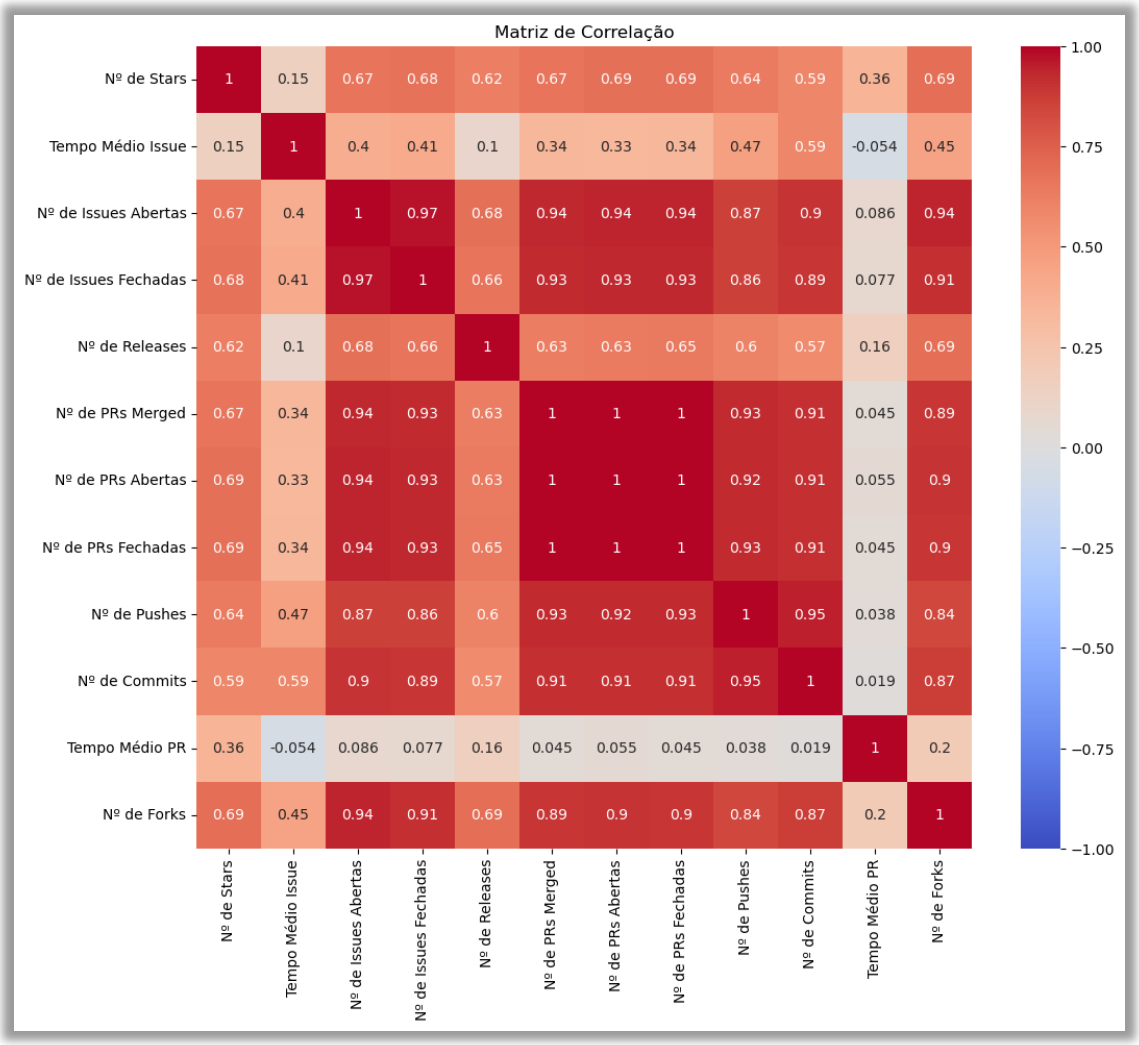


Figura 87 - Matriz de Correlação das Métricas Analisadas no Projeto OSS.

- **Correlação entre Número de *Forks*, *Pull Requests* e *Issues* Abertas**

A análise da relação entre o número de *forks*, *pull requests* e *issues* abertas oferece uma perspetiva significativa sobre a saúde e dinâmica de um projeto de software de código aberto. Uma correlação de 0,9 e de 0,89 entre o número de *forks* e as *pull requests* sublinha uma forte ligação positiva. Isso implica que um aumento no número de *forks* geralmente se reflete num crescimento no número de pull *requests*. Este comportamento indica que contribuidores externos estão ativamente a

colaborar e a propor alterações ao código. Embora um elevado número de *forks* seja frequentemente visto como um indicativo de sucesso na comunidade *open source*, um volume significativo de *pull requests* não resolvidas pode sinalizar potenciais desafios na gestão das mesmas.

Similarmente, uma correlação de 0,94 entre o número de *forks* e *issues* abertas denota uma relação quase direta. Sugere-se que a ampla exploração do projeto leva ao destaque de potenciais melhorias ou problemas. Esta atividade mostra o compromisso da comunidade em otimizar o projeto. Contudo, um acúmulo excessivo de *issues* pode sinalizar dificuldades na sua gestão pela equipe central, possivelmente afetando a sustentabilidade do projeto.

- **Correlação entre Número de *Releases*, *Forks* e *Issues* Abertas**

A correlação de 0,68 entre o Número de *Releases* e o Número de *Forks* sugere que mais versões lançadas resultam em maior frequência de bifurcações. Isso pode indicar que novos lançamentos intensificam o interesse da comunidade de desenvolvimento, conduzindo a um aumento de *forks*, que refletem o interesse em adaptar ou melhorar o projeto base.

Uma correlação de 0,67 entre o Número de *Releases* e *Issues* Abertas demonstra que o lançamento de mais versões coincide com a reportagem de mais *issues*. Contrariamente à perceção inicial, isto pode representar um projeto em evolução e ativamente testado pela comunidade. O aumento das *releases* traduz-se em mais funcionalidades e potenciais desafios identificados.

A combinação destas correlações oferece insights sobre a vitalidade e dinâmica do projeto. O crescimento no número de *releases* indica atividade e evolução, enquanto o aumento de *forks* e *issues* ressalta o valor percebido e o envolvimento da comunidade.

Assim, é possível concluir que a inter-relação entre o número de *releases*, *forks* e *issues* abertas evidencia um ecossistema robusto e ativo em torno do projeto, caracterizado por inovação, *feedback* e melhoria contínua, assegurando a sua relevância e sustentabilidade futura.

- **Correlação entre *Pull Requests* e *Issues***

A correlação entre as métricas de *Pull Requests* (abertas, fechadas e *merged*) e *Issues* (abertas e fechadas) varia entre 0,9 e 0,97, evidenciando uma interdependência marcada entre estas. Este alto coeficiente indica que o aumento de *issues* leva a um crescimento quase proporcional no número de pull *requests* e vice-versa.

Esta tendência reflete uma gestão ativa e sistemática das *issues* pela equipa de desenvolvimento. Quando surge uma *issue*, frequentemente culmina num *pull request*. Inversamente, *pull requests* que adicionam funcionalidades ou corrigem erros podem gerar *issues* adicionais para futuros ajustes.

A sincronização entre *issues* e *pull requests* denota a saúde e eficácia do projeto, demonstrando atividade constante, revisão e adaptação. Tal dinâmica indica um envolvimento profundo com feedbacks e necessidades dos utilizadores, crucial para a sustentabilidade do projeto.

A correlação entre estas métricas também sugere uma comunicação fluida e colaboração entre os intervenientes, traduzindo-se numa abordagem transparente e alinhada no desenvolvimento. Em síntese, a correlação entre *Pull Requests* e *Issues* realça uma gestão eficaz e um compromisso com a evolução contínua, elementos essenciais para a robustez e sucesso de um projeto de software.

- **Correlação entre *Pushes* e *Commits***

A correlação entre o número de *pushes* e *commits* evidencia-se com um coeficiente marcado de 0,95, denotando uma relação linear acentuada entre ambas as métricas. Esta tendência reflete que, em face de um incremento de *commits*, verifica-se um crescimento quase equivalente no número de *pushes*, coerente com a prática habitual de atualizar os repositórios remotos após *commits*.

Tal correlação sublinha que os desenvolvedores, ao introduzirem alterações, procuram regularmente sincronizar o repositório central. Esta cadência homogénea entre *commits* e *pushes* indica uma base de código em constante atualização e um fluxo de trabalho otimizado. A natureza consistente desta relação sugere uma frequente harmonização das alterações locais com o repositório principal, facilitando a colaboração e minimizando potenciais conflitos.

Este ritmo de atividade e alinhamento sinaliza uma saúde robusta do projeto e uma equipa comprometida, que prioriza manter todos os intervenientes informados sobre evoluções recentes. Esta dinâmica é vital para a longevidade de um projeto, assegurando que todos os contribuintes estejam alinhados e a par das atualizações dos pares.

- **Correlação entre Número de Estrelas e Número de *Forks***

Ao analisar a correlação entre o número de estrelas e o número de *forks*, observa-se um coeficiente de correlação de 0,69, demonstrando uma relação positiva significativa entre ambas as métricas. Esta tendência sugere que projetos mais reconhecidos, evidenciado por um elevado número de estrelas, tendem a atrair um maior número de contribuintes, resultando num crescimento no número de *forks*. No entanto, é crucial salientar que a relação, embora positiva, não é absoluta. Existem projetos que, devido à sua especialização, podem acumular muitas estrelas, mas poucos *forks*, ou vice-versa.

O número de estrelas é frequentemente interpretado como um indicador da popularidade ou do valor atribuído a um projeto na comunidade de *open source*. Em contraste, o número de forks denota o interesse da comunidade em colaborar ou personalizar o projeto.

Deste modo, esta correlação sugere que projetos valorizados tendem a gerar mais interesse colaborativo, sendo benéfico para a sua saúde e resiliência. Uma correlação sólida entre estas métricas indica que o projeto é não apenas bem-recebido, mas também suportado por uma comunidade ativa, facilitando a sua evolução.

Concluindo, a inter-relação entre estrelas e *forks* é um indicador da robustez de um projeto, denotando reconhecimento e uma comunidade contributiva ativa – fundamentos essenciais para a prosperidade de qualquer iniciativa.

- **Correlação entre Tempo Médio de *Issue* e Número de *Commits***

Há uma correlação positiva moderada, com coeficiente de 0,59, entre o tempo médio de resposta a uma *issue* e o número de *commits* no projeto. Esta correlação fornece insights significativos:

- Dinâmica Variável das *Issues*: O aumento de *issues* não determina consistentemente o tempo de resposta. Fatores externos, como a complexidade da *issue* e a capacidade da equipa, influenciam essa dinâmica.

- Indicadores de Carga de Trabalho: Mais *issues* indicam um aumento na carga de trabalho da equipa, abrangendo a resolução de bugs, avaliação de recursos, entre outros. Isto pode, consequentemente, diluir a atenção da equipa e prolongar os tempos de resposta.
- Complexidade Variada: *Issues* podem variar desde simples questões até problemas críticos ou pedidos de recursos complexos. Com um volume crescente de *issues*, é provável que haja uma diversidade nas suas naturezas, afetando o tempo de resposta.

A relação entre o tempo médio de resposta e o número de *commits* sugere que, à medida que um projeto cresce em complexidade (refletido pelos *commits*), a resolução de *issues* pode exigir mais tempo. Esta dinâmica pode indicar um projeto em evolução com desafios mais intrincados ou um processo de revisão mais rigoroso que foca na qualidade.

No panorama geral do projeto, um tempo de resposta mais prolongado pode ser interpretado como um compromisso com a qualidade e robustez, fortalecendo a confiabilidade a longo prazo. No entanto, é crucial manter um equilíbrio para que o tempo de resposta não desencoraje a comunidade.

Em suma, a correlação entre o Tempo Médio de *Issue* e o Número de *Commits* oferece uma visão sobre a qualidade e complexidade crescente do projeto. É vital contextualizar esta relação para entender completamente as prioridades e operações do projeto.

- **Correlação entre Número de Estrelas e Atividade de Desenvolvimento (*Commits* e *Pushes*)**

Há uma correlação positiva moderada de 0,59 entre o número de estrelas e o número de commits, sugerindo que a popularidade de um projeto (refletida pelo número de estrelas) está relacionada com a sua atividade de desenvolvimento. À medida que um projeto ganha reconhecimento, parece haver um aumento na atividade de desenvolvimento, seja para atender às expectativas da comunidade, responder ao feedback ou manter o projeto atualizado.

Quando se observa a relação entre o número de estrelas e o número de *pushes,* a correlação é ainda mais forte, com um valor de 0,64. Isto sugere que a popularidade de um projeto também influencia a frequência de atualizações no repositório. A maior frequência de pushes pode indicar uma colaboração crescente e uma comunidade ativa que contribui regularmente para o projeto.

Estas correlações indicam uma dinâmica notável: a popularidade de um projeto está estreitamente ligada à sua atividade de desenvolvimento. Por um lado, um projeto bem recebido pela comunidade parece ser mais ativo. Por outro lado, a atividade constante pode ser uma razão pela qual um projeto é bem avaliado e, assim, ganha mais estrelas. A relação sublinha a relevância de manter um projeto ativo e alinhado com os interesses da comunidade.

4.2. Discussão da Análise

A avaliação de projetos de código aberto requer uma análise meticulosa. As métricas desempenham um papel crucial, proporcionando *insights* que fundamentam decisões informadas. No decorrer deste estudo, foram analisados diversos indicadores de dois projetos específicos do GitHub, um relacionado ao desenvolvimento de *hardware* de código aberto e outro ao desenvolvimento de *software* de código aberto, permitindo a extração de conclusões pertinentes.

Ao comparar ambos os projetos, observou-se que, no projeto de *hardware* de código aberto, todas as métricas revelaram uma atividade esporádica. Esta tendência pode ser justificada pelas características intrínsecas do *hardware* de código aberto, que apresenta barreiras significativas à contribuição. Contudo, um reduzido número de lançamentos ou contribuições não implica que o projeto seja desinteressante ou inútil. Apesar da notoriedade do *hardware* de código aberto, este ainda não conseguiu alcançar o patamar do *software* de código aberto, e não é previsível que o alcance num futuro próximo.

Diferentemente do *software* de código aberto, que é facilmente acessível e modificável online, o *hardware* exige mais do que simplesmente programação; requer compreensão em áreas específicas e frequentemente recursos financeiros para a sua materialização.

Em relação ao número de estrelas e outras métricas, como contribuições e lançamentos, vários fatores podem influenciar esta discrepância, pelo facto de apesar da atividade do projeto não ser em níveis elevados, o número de estrelas demonstrou uma maior notoriedade. Atribuir uma estrela a um projeto é um ato simples, sem compromisso substancial, ao passo que contribuir ativamente exige maior envolvimento. Esta ação pode ser vista como um reconhecimento do valor do projeto, mesmo que não se traduza em contribuição ativa. A visibilidade do projeto em plataformas externas, como *blogs* ou conferências, pode também levar a um aumento de estrelas, sem corresponder a um aumento proporcional de contribuições.

Embora as contribuições ao projeto sejam esporádicas, a sua gestão demonstra ser eficiente. Isto é evidenciado pela capacidade da equipa em responder adequadamente às expectativas, mesmo face à complexidade das tarefas. O projeto exibiu inicialmente um volume significativo de contribuições, mas os últimos meses foram caracterizados por uma atividade reduzida, sugerindo uma possível concretização dos objetivos iniciais.

A análise das correlações no âmbito do projeto de *Open Source Hardware* evidencia relações diversificadas entre diferentes métricas. No que concerne à relação entre o número de *forks* e métricas associadas a *Pull Requests* e *Issues*, verifica-se uma ligação positiva, embora moderada. Isto significa que um aumento nos *forks*, que denota um crescente interesse no projeto, não se traduz proporcionalmente em contribuições ativas. Este cenário aponta para a presença de outros fatores que moldam a dinâmica de contribuições, para além do mero interesse manifestado através dos *forks*.

Quando se observa a interação entre PRs e *Issues*, destaca-se uma correlação forte, sinalizando uma gestão ativa e responsiva ao *feedback* da comunidade. Esta interação revela que o surgimento de sugestões ou problemas leva, frequentemente, a uma resposta na forma de PRs.

No que toca à relação entre *pushes* e *commits*, a forte correlação sugere um processo de desenvolvimento metódico e integrado, refletindo uma maturidade processual no projeto. Por outro lado, as baixas correlações entre o número de *releases*, *forks* e *issues* abertas indicam que as decisões de lançamento de novas versões não são fortemente influenciadas pela dinâmica de *forks* ou pelo surgimento de novas *issues*.

Quanto ao número de estrelas e *forks*, a correlação, embora presente, é moderada. Isto sugere que a popularidade do projeto, embora relevante, não é o único motor da atividade de *forks*. A correlação expressiva entre o tempo médio de uma *issue* e o número de *issues* fechadas sugere uma sintonia entre a gestão de *issues* e o seu tempo de resolução, podendo refletir a natureza das

issues ou as prioridades da equipa. Já o coeficiente de correlação entre o tempo médio de PR e o número de PRs abertas indica que períodos mais ativos podem exigir uma avaliação mais aprofundada das contribuições.

As correlações entre o número de estrelas e métricas de desenvolvimento, como *commits* e *pushes*, sublinham que o reconhecimento do projeto está intrinsecamente ligado à sua dinâmica e atividade. O retrato que emerge das correlações aponta para um projeto de *Open Source Hardware* que, apesar de reconhecido e valorizado pela comunidade, enfrenta desafios multifacetados.

Através da análise, foi possível aferir que a fase inicial do projeto *Open Source Software* é de suma importância, uma vez que se observou ausência de atividade nas métricas analisadas, entre os primeiros 16 a 20 meses, sendo possível concluir que esta fase inicial estabelece as fundações para o desenvolvimento subsequente do projeto. Neste contexto, surgem diversas considerações que os gestores devem ponderar, a fim de assegurar um arranque sólido e promissor para o projeto:

1) Estratégia de Inicialização

Durante a fase inicial, é imperativo que os gestores delineiem e implementem uma estratégia abrangente. Esta deve visar não apenas a promoção do projeto, mas também a criação de canais eficazes que facilitem a interação com a comunidade. Através destes canais, é possível atrair potenciais colaboradores e *stakeholders*, fundamentais para o crescimento e sucesso do projeto.

2) Documentação Completa e Clara

Um dos desafios frequentemente encontrados em projetos de código aberto é a ausência de contribuições, muitas vezes refletida pela ausência de *forks*. Esta situação pode ser consequência de documentação insuficiente ou ambígua. É, portanto, essencial que os gestores assegurem uma documentação completa e clara, que oriente os potenciais contribuidores sobre como podem participar e enriquecer o projeto.

3) Feedback Inicial

A recolha de *feedback* na fase inicial é vital. Os gestores devem ativamente procurar opiniões e comentários daqueles que interagem ou mostram interesse no projeto. Estes *feedbacks* são valiosos, pois podem revelar áreas de melhoria, lacunas ou potenciais desafios que necessitam de ser abordados.

Com base na análise das métricas, conclui-se que o projeto em questão, de *Open Source Software*, revela sinais cruciais de sucesso e sustentabilidade. O projeto mostrou uma tendência ascendente em diversas métricas, evidenciando uma saúde robusta e uma forte ressonância na comunidade. Esta saúde é destacada pelo crescimento constante em métricas como o número de *forks*, estrelas e *releases*, apontando para uma maturidade e adaptabilidade que se mantêm ao longo do tempo. A gestão eficiente dos PRs, particularmente o tempo médio de revisão, demonstra a capacidade do projeto em manter um equilíbrio entre a qualidade do trabalho e a eficiência do desenvolvimento colaborativo. Adicionalmente, o comprometimento da equipa e da comunidade torna-se evidente através do aumento sustentado no número de *issues* e pelo tempo médio de resposta que se manteve em níveis considerados razoáveis.

Ao explorar o projeto de *software* em profundidade, as correlações ajudam a compreender as relações interdependentes entre as diferentes métricas. A medida da atividade de desenvolvimento, que se baseia no número de *commits*, *pushes* e *pull requests*, revelou um projeto

vibrante com uma comunidade altamente envolvida, sendo este um indicativo positivo da saúde e vitalidade do projeto. A quantidade significativa de *pull requests* abertas, juntamente com a proximidade entre as *pull requests* que foram abertas e posteriormente fechadas, não só mostrou uma comunidade envolvida, mas também uma equipa de projeto que age de forma ágil e responsiva.

O projeto analisado demonstrou possuir características que são típicas de um projeto saudável e bem gerido. Métricas de atividade, como o número de *commits*, *pushes* e *pull requests*, são indicativos de um projeto ativo. Estes, juntamente com o elevado número de *issues* abertas e fechadas e o tempo médio de resposta a estas, destacam uma equipa que dá grande valor ao *feedback* e que está pronta para se adaptar às necessidades do projeto e da comunidade. Esta adaptabilidade e resposta rápida são características de uma gestão ágil, onde o *feedback* é recebido e implementado rapidamente.

Dentro do desenvolvimento ágil, a colaboração entre desenvolvedores e *stakeholders* é essencial, e tudo indica que esta colaboração está a ser eficazmente realizada no projeto em análise. Este tipo de colaboração e adaptabilidade pode ser ainda mais evidente quando se observa a gestão das *issues* e *pull requests*, sugerindo a aplicação de metodologias ágeis como *Scrum* ou *Kanban*, que se centram em ciclos de desenvolvimento curtos e entregas incrementais. A correlação entre o número de *issues* e *pull requests* sugere um ciclo contínuo de identificação de novos desafios e implementação de soluções.

As correlações entre a métrica do número de estrelas e outras, como *commits* e *pushes*, indicam que a popularidade e a atividade de desenvolvimento estão intimamente ligadas. Isto é reforçado pela observação de que projetos populares tendem a ser mais ativos em termos de desenvolvimento, sugerindo que a atividade contínua pode ser um dos fatores que contribuem para um projeto ser bem recebido e, por consequência, bem-sucedido a longo prazo.

Na Figura 88, é apresentado um quadro geral que ilustra os diversos gráficos e métricas considerados essenciais para a avaliação de projetos *open source*. Esta representação visual procura consolidar, de forma clara e concisa, os indicadores-chave que foram identificados como cruciais para avaliar a saúde, o sucesso e a sustentabilidade de um projeto. A figura serve como um guia visual para os leitores, permitindo uma compreensão rápida e integrada dos parâmetros avaliativos abordados nesta dissertação.

Ambos os domínios, apesar das diferenças, demonstram compromisso, adaptabilidade e relevância no contexto do desenvolvimento *Open Source*. Reconhecer essas diferenças e interpretar as métricas no contexto apropriado é essencial para compreender a saúde e direção de tais projetos no panorama *Open Source*.

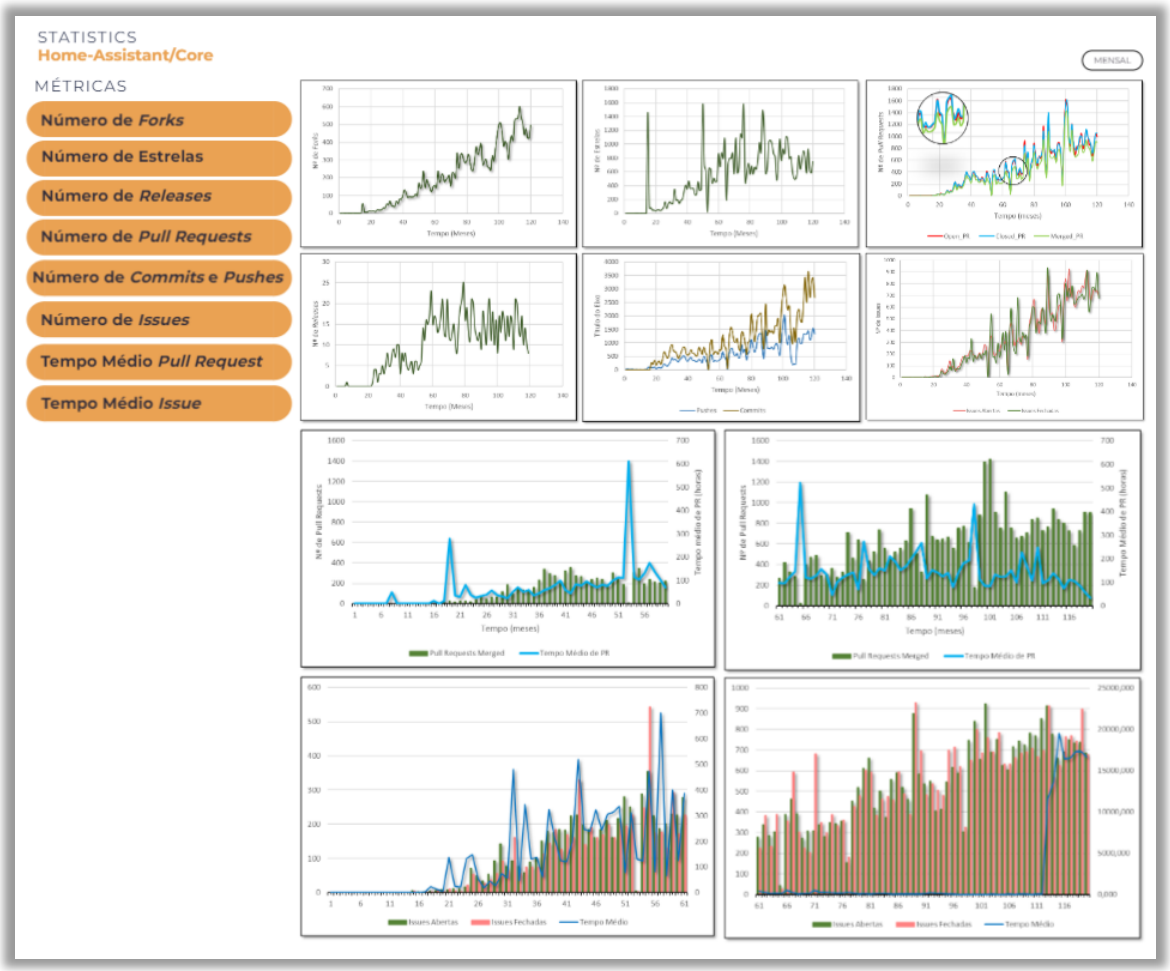


Figura 88 - Panorama Geral das Métricas e Gráficos para Avaliação de Projetos Open Source.

As métricas e correlações analisadas oferecem uma estrutura robusta para avaliar a saúde, sucesso e sustentabilidade de qualquer projeto de código aberto. Estas métricas são essenciais para qualquer equipa ou organização que aspire criar, gerir e manter um projeto bem-sucedido. Para aqueles que pretendem ter sucesso na comunidade de código aberto ou avaliar projetos *open source*, estas métricas são fundamentais. Estas métricas, analisadas em conjunto, oferecem uma visão completa da saúde e potencial de sucesso de um projeto de código aberto, permitindo decisões informadas e contribuições significativas para a comunidade *open source*. A abordagem proposta oferece uma visão completa do estado e direção futura dos projetos, sendo uma ferramenta robusta e holística que pode guiar os *desenvolvedores* na avaliação e direção de um projeto de código aberto, destacando não só a saúde atual, mas também proporcionando insights sobre onde investir esforços para assegurar o seu crescimento e sucesso contínuo.

5. CONCLUSÃO

Neste capítulo, são apresentadas as conclusões finais do trabalho desenvolvido. Aqui, realizar-se-á uma recapitulação das informações discutidas nos capítulos anteriores, enfatizando-se a relevância da revisão bibliográfica, da metodologia adotada e da análise das métricas para a avaliação do sucesso e sustentabilidade de projetos *Open Source*. Também serão abordadas as limitações encontradas durante a realização deste trabalho e serão identificadas áreas que poderão ser exploradas em futuras pesquisas.

5.1. Conclusões finais

A convergência entre a gestão ágil e o *Open Design*, em comunidades de código aberto, representa uma intersecção fascinante de teoria e prática. Através desta dissertação, explorou-se essa interseção, revelando a dinâmica, desafios e oportunidades inerentes a projetos *Open Source* e à aplicabilidade das práticas ágeis nesse contexto.

A investigação revelou uma importância fundamental em aplicar princípios e práticas ágeis em comunidades de projetos abertos. Esta abordagem ágil não apenas proporciona flexibilidade e adaptabilidade, mas também promove uma colaboração eficaz, vital para responder prontamente a mudanças e desafios. Diferentemente das equipas que adotam abordagens tradicionais, as comunidades de projetos abertos apresentam desafios específicos. A natureza aberta e distribuída destas comunidades exige transparência total, comunicação aberta e estruturas de tomada de decisão colaborativas.

O estudo do caso OHWR foi um ponto crucial na dissertação. As interações e discussões com representantes da comunidade OHWR proporcionaram *insights* valiosos sobre o funcionamento interno de uma plataforma de projeto aberto. Foi possível compreender o processo de admissão de novos projetos e a postura do OHWR face à abertura e colaboração, bem como identificar áreas potenciais de desenvolvimento e aprimoramento.

Centrando-se na análise de métricas extraídas de projetos Open Source, particularmente de um projeto específico do GitHub, obteve-se um panorama detalhado sobre o que define a saúde, sucesso e sustentabilidade de um projeto *Open Source*. Através destas métricas, discerniram-se tendências e padrões, formando a base para o modelo de métricas proposto. Esse modelo visa fornecer uma estrutura clara e abrangente, integrando métricas quantitativas para avaliar projetos.

Quando analisamos o Open Source Software e o Open Source Hardware, é possível observar uma série de semelhanças e diferenças. Ambos são movidos pela filosofia de compartilhamento e colaboração. O OSS é caracterizado pela sua acessibilidade: o código-fonte é disponibilizado publicamente, permitindo que qualquer pessoa, em qualquer parte do mundo, visualize, modifique e distribua o software. Esta abertura democratiza o acesso à tecnologia, eliminando barreiras geográficas e financeiras.

Para empresas, especialmente pequenas e médias, que muitas vezes enfrentam restrições orçamentais, o OSS apresenta uma solução de valor inestimável. Ao invés de investir em soluções de software dispendiosas, estas empresas têm à sua disposição ferramentas e sistemas de

qualidade, prontos a serem adaptados às suas necessidades, sem os custos associados ao licenciamento tradicional.

Mais do que uma mera alternativa económica, o OSS fomenta a inovação. A colaboração global, inerente ao universo de código aberto, permite uma troca constante de ideias e soluções. Este intercâmbio resulta em software mais robusto, seguro e adaptável às necessidades em constante evolução do mundo empresarial.

Contudo, a natureza tangível do hardware, aliada à sua complexidade e aos desafios logísticos associados à sua produção e distribuição, torna o OSHW um campo de atuação distinto e com desafios únicos.

Atualmente, as abordagens no âmbito do *hardware* de código aberto concentram-se, em grande parte, nas fases preliminares de *design*, nomeadamente na prototipagem. A evolução destes projetos para uma fase de maturidade que os torne prontos para o mercado é uma etapa complexa, e poucos conseguem atingir este patamar no contexto do *Open Design*.

O conceito de *Open Design*, em especial quando se aborda em contexto com o *Hardware* de Código Aberto, insere-se num movimento contemporâneo que procura redefinir as narrativas tradicionais. Este movimento desafia a forma convencional como as indústrias e os consumidores interagem, e destaca valores essenciais como a sustentabilidade económica, a valorização da autonomia local e a centralidade do ser humano.

A proposta de um modelo de gestão ágil para comunidades *open source* representa uma contribuição notável no campo do *design aberto* e gestão ágil. Este modelo não apenas serve como uma ferramenta prática para desenvolvedores, mas também estabelece diretrizes para futuras investigações. Além disso, a dissertação destaca a importância da adaptabilidade e da rápida resposta às mudanças, qualidades essenciais no dinâmico mundo do código aberto.

Ao concluir, esta dissertação avança significativamente na compreensão da gestão de projetos em comunidades *open source* e *Open Design*. Através da fusão de *design* aberto com práticas ágeis, a pesquisa propõe um caminho para uma realização mais eficaz e colaborativa de projetos em comunidades digitais e abertas, com implicações profundas para o futuro da inovação aberta.

5.2. Limitações e trabalhos futuros

A presente investigação, embora abrangente, teve algumas limitações inerentes ao processo de recolha e análise de dados. Primeiramente, a análise baseou-se na análise de dois projetos, o que potencialmente restringe a diversidade dos *insights* obtidos. Além disso, ao centrar a atenção predominantemente no GitHub, outras plataformas relevantes, como GitLab ou Bitbucket, poderiam ter sido negligenciadas, resultando em possíveis métricas diferenciadas ou relevantes que não foram contempladas.

Outra limitação relevante refere-se às APIs utilizadas para extrair métricas. A API do GitHub, apesar de útil, pode apresentar limitações quanto à capacidade, frequência de chamadas e profundidade dos dados acessíveis. Tais restrições afetaram a aquisição de um conjunto completo de métricas desejadas. Adicionalmente, latências na resposta da API impactaram a eficiência da recolha, especialmente quando se trata de um volume elevado de dados.

No que diz respeito a trabalhos futuros, várias direções promissoras podem ser exploradas. Seria pertinente considerar uma amostra mais ampla e diversificada de projetos, abrangendo diferentes domínios e escalas. A análise poderia ser expandida para englobar outras plataformas de código aberto, aprofundando a compreensão das métricas e padrões. Uma investigação mais aprofundada sobre métricas específicas poderia oferecer *insights* valiosos sobre a qualidade intrínseca do código.

Casos de estudo detalhados em projetos específicos poderiam elucidar a relação entre métricas e o sucesso tangível dos projetos. Uma interação mais próxima com as comunidades de código aberto poderia enriquecer a pesquisa com *feedback* qualitativo, complementando a análise quantitativa. Adicionalmente, a comparação entre projetos de sucesso variável poderia identificar fatores determinantes para o sucesso de um projeto.

Por fim, explorar outras APIs ou, até mesmo, desenvolver ferramentas proprietárias para extração de dados pode ser uma estratégia valiosa para superar as limitações das ferramentas atuais. Uma colaboração mais estreita com plataformas de hospedagem de código aberto também poderia desbloquear um acesso mais detalhado aos dados necessários.

Em suma, apesar das limitações identificadas, esta investigação lança as bases para futuras explorações no domínio da gestão de projetos de código aberto, e as recomendações para trabalhos futuros apontam para um horizonte repleto de oportunidades promissoras.

5.3. Contribuições

A principal contribuição, até ao momento, da presente dissertação em trabalho de pesquisa foi um artigo de conferência submetido e aceite para publicação pela *Procedia Computer Science*, intitulado como "*Open Design Communities: A bibliometric analysis of community-based management"*, segundo a *International Conference CENTERIS on ENTERprise Information Systems Organising Committee*.

REFERÊNCIAS BIBLIOGRÁFICAS

Acosta, R., Von Hippel, E., Supervisor, T., & Hale, P. (2009). *Open Source Hardware*.

Afuah, A., & Tucci, C. L. (2012). Crowdsourcing as a solution to distant search. In *Academy of Management Review* (Vol. 37, Issue 3, pp. 355–375). https://doi.org/10.5465/amr.2010.0146

Aitamurto, T., Holland, D., & Hussain, S. (2015). The open paradigm in design research. *Design Issues*, *31*(4), 17–29. https://doi.org/10.1162/DESI_a_00348

Alamer, G., & Alyahya, S. (2017). Open Source Software Hosting Platforms: A Collaborative Perspective's Review. *Journal of Software*, *12*(4), 274–291. https://doi.org/10.17706/jsw.12.4.274-291

Alexander M. (2018). *Agile project management: A comprehensive guide*.

Al-Saqqa, S., Sawalha, S., & Abdelnabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, *14*(11), 246–270. https://doi.org/10.3991/ijim.v14i11.13269

Antoniou, R., Bonvoisin, J., Hsing, P. Y., Dekoninck, E., & Defazio, D. (2022). Defining success in open source hardware development projects: A survey of practitioners. *Design Science*, *8*. https://doi.org/10.1017/dsj.2021.30

Augustine, S. (2005). *Managing agile projects*. Prentice Hall PTR: Upper Saddle River.

Avital, M. (2011). *The generative bedrock of open design Inclusivity ww View project PhD in Information Technology Management View project*. https://www.researchgate.net/publication/279409150

Azanha, A., Argoud, A. R. T. T., de Camargo Junior, J. B., & Antoniolli, P. D. (2017). Agile project management with Scrum: A case study of a Brazilian pharmaceutical company IT project. *International Journal of Managing Projects in Business*, *10*(1), 121–142. https://doi.org/10.1108/IJMPB-06-2016-0054

Azenha, F. C., Copola Azenha, F., Reis, D. A., & Fleury, A. L. (2021). The Role and Characteristics of Hybrid Approaches to Project Management in the Development of Technology-Based Products and Services. In *Project Management Journal* (Vol. 52, Issue 1).

Balka, K., Raasch, C., & Herstatt, C. (2010). How open is open source? - software and beyond. *Creativity and Innovation Management*, *19*(3), 248–256. https://doi.org/10.1111/j.1467-8691.2010.00569.x

Balka, K., Raasch, C., & Herstatt, C. (2014). The effect of selective openness on value creation in user innovation communities. *Journal of Product Innovation Management*, *31*(2), 392–407. https://doi.org/10.1111/jpim.12102

Barcellini, F., Détienne, F., & Burkhardt, J. M. (2014). A situated approach of roles and participation in open source software communities. In *Human-Computer Interaction* (Vol. 29, Issue 3, pp. 205–255). Taylor and Francis Inc. https://doi.org/10.1080/07370024.2013.812409

Basmer, S., Buxbaum-Conradi, S., Krenz, P., Redlich, T., Wulfsberg, J. P., & Bruhns, F. L. (2015). Open production: Chances for social sustainability in manufacturing. *Procedia CIRP*, *26*, 46–51. https://doi.org/10.1016/j.procir.2014.07.102

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., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifesto for Agile Software Development Twelve Principles of Agile Software*. http://www.agilemanifesto.org

Bleiel, N. (2016). *Collaborating in GitHub*.

Blincoe, K., Sheoran, J., Goggins, S., Petakovic, E., & Damian, D. (2016). Understanding the popular users: Following, affiliation influence and leadership on GitHub. *Information and Software Technology*, *70*, 30–39. https://doi.org/10.1016/j.infsof.2015.10.002

Bonvoisin, J., & Mies, R. (2018). Measuring Openness in Open Source Hardware with the Open-o-Meter. *Procedia CIRP*, *78*, 388–393. https://doi.org/10.1016/j.procir.2018.08.306

Bonvoisin, J., Mies, R., & Boujut, J. F. (2021). Seven observations and research questions about Open Design and Open Source Hardware. *Design Science*, *7*. https://doi.org/10.1017/dsj.2021.14

Bonvoisin, J., Mies, R., Stark, R. G., & Jochem, R. (2016). *Business models supporting open source hardware View project OPEN! Methods and tools for community-based product development View project*. https://www.researchgate.net/publication/311722221

Braunschweig, K., Eberius, J., Thiele, M., & Lehner, W. (2012). The State of Open Data: Limits of Current Open Data Platforms. *Faculty of Computer Science, Database Technology Group*.

Carroll, J. (2022). *The ever-growing developer community*. https://octoverse.github.com/2022/developer-community

Castro, H., & Ávila, P. (2020). *Open Design, Open Process and Open Production*.

Castro, H., Putnik, G., Castro, A., & Fontana, R. D. B. (2019). Could open design learn from wikipedia? *Procedia CIRP*, *84*, 1112–1115. https://doi.org/10.1016/j.procir.2019.07.001

Cervone, H. F. (2011). Understanding agile project management methods using Scrum. In *OCLC Systems and Services* (Vol. 27, Issue 1, pp. 18–22). https://doi.org/10.1108/10650751111106528

Chauvette, A., Schick-Makaroff, K., & Molzahn, A. E. (2019). Open Data in Qualitative Research. *International Journal of Qualitative Methods*, *18*. https://doi.org/10.1177/1609406918823863

Chen, Q., & Liu, Z. (2019). How does openness to innovation drive organizational ambidexterity? the mediating role of organizational learning goal orientation. *IEEE Transactions on Engineering Management*, *66*(2), 156–169. https://doi.org/10.1109/TEM.2018.2834505

Chesbrough, H. (2006). *Open business models: How to thrive in the new innovation landscape*. Harvard Business Press.

Chesbrough, H. (2012). Open innovation: Where we've been and where we're going. In *Research Technology Management* (Vol. 55, Issue 4, pp. 20–27). https://doi.org/10.5437/08956308X5504085

Chesbrough, H., Vanhaverbeke, W., & West, J. (2006). Open innovation: Researching a new paradigm. In *Oxford University Press on Demand*.

Chesbrough, H. W. (2003). *Chesbrough, H. W. (2003). Open innovation: The new imperative for creating and profiting from technology*. Harvard Business Press.

Christian, J., & Vu, A. N. (2020). *Task-based structures in open source software: revisiting the onion model*.

Coelho, J., Valente, M. T., Milen, L., & Silva, L. L. (2020). Is this GitHub project maintained? Measuring the level of maintenance activity of open-source projects. *Information and Software Technology*, *122*. https://doi.org/10.1016/j.infsof.2020.106274

Conforto, E. C., Amaral, D. C., da Silva, S. L., Di Felippo, A., & Kamikawachi, D. S. L. (2016). The agility construct on project management theory. *International Journal of Project Management*, *34*(4), 660–674. https://doi.org/10.1016/j.ijproman.2016.01.007

Conforto, E. C., Salum, F., Amaral, D. C., Da Silva, S. L., & De Almeida, L. F. M. (2014). Can agile project management be adopted by industries other than software development? *Project Management Journal*, *45*(3), 21–34. https://doi.org/10.1002/pmj.21410

Cooper, R. G., & Sommer, A. F. (2016). The Agile–Stage-Gate Hybrid Model: A Promising New Approach and a New Research Opportunity. *Journal of Product Innovation Management*, *33*(5), 513–526. https://doi.org/10.1111/jpim.12314

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2017). A Systematic Mapping Study of Software Development with GitHub. *IEEE Access*, *5*, 7173–7192. https://doi.org/10.1109/ACCESS.2017.2682323

de Bruijn, E. (2010). *On the viability of the Open Source Development model for the design of physical object: Lessons learned from the RepRap project*.

Dittrich, K., & Duysters, G. (2007). *Networking as a Means to Strategy Change: The Case of Open Innovation in Mobile Telephony*.

Elbanna, A., & Sarker, S. (2016). The Risks of Agile Software Development: Learning from Adopters. *IEEE Software*, *33*(5), 72–79. https://doi.org/10.1109/MS.2015.150

Forbes, H., & Schaefer, D. (2017). Social Product Development: The Democratization of Design, Manufacture and Innovation. *Procedia CIRP*, *60*, 404–409. https://doi.org/10.1016/j.procir.2017.02.029

Gassmann, O., & Enkel, E. (2004). *Towards a Theory of Open Innovation: Three Core Process Archetypes*.

Gehring, W. (2022). *OCTOVERSE 2022 - The state of open source software*. https://octoverse.github.com/

Gloor, P. A. (2006). *Swarm Creativity: Competitive Advantage through Collaborative Innovation Networks*.

Goldstein, V., & Euchner, J. (2017). Transformation for Growth at GE: An Interview with Viv GoldsteinViv Goldstein talks with Jim Euchner about the transformation at GE to make the company faster, more agile, and more intensely focused on the customer. In *Research Technology Management* (Vol. 60, Issue 6, pp. 14–19). Taylor and Francis Inc. https://doi.org/10.1080/08956308.2017.1373045

Greco, M., Grimaldi, M., & Cricelli, L. (2016). An analysis of the open innovation effect on firm performance. *European Management Journal*, *34*(5), 501–516. https://doi.org/10.1016/j.emj.2016.02.008

Greco, M., Locatelli, G., & Lisi, S. (2017). Open innovation in the power & energy sector: Bringing together government policies, companies' interests, and academic essence. *Energy Policy*, *104*, 316–324. https://doi.org/10.1016/j.enpol.2017.01.049

Gunasekaran, A. (1998). Agile manufacturing: Enablers and an implementation framework. *International Journal of Production Research*, *36*(5), 1223–1247. https://doi.org/10.1080/002075498193291

Highsmith, J. (2009). *Agile project management: creating innovative products*. Pearson education.

Hoda, R., Noble, J., & Marshall, S. (2008). *Agile Project Management*.

Huizingh, E. K. R. E. (2011). Open innovation: State of the art and future perspectives. *Technovation*, *31*(1), 2–9. https://doi.org/10.1016/j.technovation.2010.10.002

Hung, K. P., & Chou, C. (2013). The impact of open innovation on firm performance: The moderating effects of internal R&D and environmental turbulence. *Technovation*, *33*(10–11), 368–380. https://doi.org/10.1016/j.technovation.2013.06.006

Iriberri, A., & Leroy, G. (2009). A life-cycle perspective on online community success. *ACM Computing Surveys*, *41*(2). https://doi.org/10.1145/1459352.1459356

Ito, J. (2017). The Antidisciplinary Approach: IRI Medal AddressThe antidisciplinary approach of the MIT Media Lab demonstrates how organizations might adapt to and take advantage of the evolving world of permissionless innovation. *Research Technology Management*, *60*(6), 22–28. https://doi.org/10.1080/08956308.2017.1373047

Jarczyk, O., Gruszka, B., Jaroszewicz, S., Bukowski, L., & Wierzbicki, A. (2014). *GitHub Projects. Quality Analysis of Open-Source Software*.

Jensen, C., & Scacchi, W. (2007). *Role Migration and Advancement Processes in OSSD Projects: A Comparative Case Study*.

Jergensen, C., Sarma, A., & Wagstrom, P. (2011). *The Onion Patch: Migration in Open Source Ecosystems*.

Jiménez, V., Afonso, P., & Fernandes, G. (2020). Using agile project management in the design and implementation of activity-based costing systems. *Sustainability (Switzerland)*, *12*(24), 1–23. https://doi.org/10.3390/su122410352

Kapoor, K., Weerakkody, V., & Sivarajah, U. (2015). Open data platforms and their usability: Proposing a framework for evaluating citizen intentions. *Lecture Notes in Computer Science*

(Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), *9373*, 261–271. https://doi.org/10.1007/978-3-319-25013-7_21

Kaur, R., Kaur Chahal, K., & Saini, M. (2022). Understanding community participation and engagement in open source software Projects: A systematic mapping study. In *Journal of King Saud University - Computer and Information Sciences* (Vol. 34, Issue 7, pp. 4607–4625). King Saud bin Abdulaziz University. https://doi.org/10.1016/j.jksuci.2020.10.020

Kilamo, T., Hammouda, I., Mikkonen, T., & Aaltonen, T. (2012). From proprietary to open source - Growing an open source ecosystem. *Journal of Systems and Software*, *85*(7), 1467–1478. https://doi.org/10.1016/j.jss.2011.06.071

Kim, T., & Shin, D. H. (2016). Social platform innovation of open source hardware in South Korea. In *Telematics and Informatics* (Vol. 33, Issue 1, pp. 217–226). Elsevier Ltd. https://doi.org/10.1016/j.tele.2015.07.004

Li, R., Fu, L., & Liu, Z. (2020). Does openness to innovation matter? The moderating role of open innovation between organizational ambidexterity and innovation performance. *Asian Journal of Technology Innovation*, *28*(2), 251–271. https://doi.org/10.1080/19761597.2020.1734037

Lichtenthaler, U. (2008). Open innovation in practice: An analysis of strategic approaches to technology transactions. *IEEE Transactions on Engineering Management*, *55*(1), 148–157. https://doi.org/10.1109/TEM.2007.912932

Lindvall, M., Basili, V., Boehm, B., Costa, P., Dangle, K., Shull, F., & Zelkowitz, M. (2002). *Empirical findings in agile methods. InConference on Extreme Programming and Agile Methods*. 197–207.

Link, G., & Vargas, S. (2022). *8 ideas for measuring your open source software usage*. https://opensource.com/article/22/12/open-source-usage-metrics

Lock, J. (2013). *Open Source Hardware Can embedded electronics companies thrive through the use and/or development of open source hardware?*

Loiro, C., Castro, H., Ávila, P., Cruz-Cunha, M. M., Putnik, G. D., & Ferreira, L. (2019). Agile Project Management: A Communicational Workflow Proposal. *Procedia Computer Science*, *164*, 485–490. https://doi.org/10.1016/j.procs.2019.12.210

Mahdavinejad, M., & Abedi, M. (2011). Community-oriented landscape design for sustainability in architecture and planning. *Procedia Engineering*, *21*, 337–344. https://doi.org/10.1016/j.proeng.2011.11.2024

Martinsuo, M. (2019). Strategic Value at the Front End of a Radical Innovation Program. *Project Management Journal*, *50*(4), 431–446. https://doi.org/10.1177/8756972819853438

McDonald, N., & Goggins, S. (2013). Perfomance and Participation in Open Source Software on GitHub. *CHI 2013 Extended Abstracts*.

Melo, J. C. F. de, Salerno, M. S., Freitas, J. S., Bagno, R. B., & Brasil, V. C. (2020). From open innovation projects to open innovation project management capabilities: A process-based approach. *International Journal of Project Management*, *38*(5), 278–290. https://doi.org/10.1016/j.ijproman.2020.06.006

Michlmayr, M. (2009). *Community Management in Open Source Projects*.

Middleton, J., Murphy-Hill, E., Green, D., Meade, A., Mayer, R., White, D., & McDonald, S. (2018a). Which contributions predict whether developers are accepted into github teams. *Proceedings - International Conference on Software Engineering*, 403–413. https://doi.org/10.1145/3196398.3196429

Middleton, J., Murphy-Hill, E., Green, D., Meade, A., Mayer, R., White, D., & McDonald, S. (2018b). Which contributions predict whether developers are accepted into github teams. *Proceedings - International Conference on Software Engineering*, 403–413. https://doi.org/10.1145/3196398.3196429

Midler, C., Maniak, R., & de Campigneulles, T. (2019). Ambidextrous Program Management: The Case of Autonomous Mobility. *Project Management Journal*, *50*(5), 571–586. https://doi.org/10.1177/8756972819869091

Mkoba, E., & Marnewick, C. (2020). Conceptual Framework for Auditing Agile Projects. *IEEE Access*, *8*, 126460–126476. https://doi.org/10.1109/ACCESS.2020.3007874

Molloy, J. C. (2011). The open knowledge foundation: open data means better science. *PLoS Biology*.

Mombach, T., & Valente, M. T. (2018). *GitHub REST API vs GHTorrent vs GitHub Archive: A Comparative Study*. https://developer.github.com/v3/

Moritz, M., Redlich, T., & Wulfsberg, J. (2018). Best practices and pitfalls in open source hardware. *Advances in Intelligent Systems and Computing*, *721*, 200–210. https://doi.org/10.1007/978-3-319-73450-7_20

Murray-Rust, P. (2008). *Open Data in Science*.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., & Ye, Y. (2002). *Evolution Patterns of Open-Source Software Systems and Communities*. http://www.gnu.org/

Open Hardware Repository. (2023). *Companies*. Companies. https://ohwr.org/companies

Open Source Hardware Association. (2022, January 12). *The Open Source Hardware Definition*. https://www.oshwa.org/Definition/

Open Source Initiative. (2022, March 22). *The Open Source Definition 1.0*. https://opensource.org/osd

Padhye, R., Mani, S., & Sinha, V. S. (2014). A study of external community contribution to Open-source projects on GitHub. *11th Working Conference on Mining Software Repositories, MSR 2014 - Proceedings*, 332–335. https://doi.org/10.1145/2597073.2597113

Papadopoulos, G. (2015). Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects. *Procedia - Social and Behavioral Sciences*, *175*, 455–463. https://doi.org/10.1016/j.sbspro.2015.01.1223

Pellizzoni, E., Trabucchi, D., & Buganza, T. (2019). When agility meets open innovation: two approaches to manage inbound projects. *Creativity and Innovation Management*, *28*(4), 464–476. https://doi.org/10.1111/caim.12337

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. (2016a). 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

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. (2016b). 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

Perkmann, M., & Walsh, K. (2007). University-industry relationships and open innovation: Towards a research agenda. *International Journal of Management Reviews*, *9*(4), 259–280. https://doi.org/10.1111/j.1468-2370.2007.00225.x

Peterson, K. (2013). *The GitHub Open Source Development Process*. http://kevinp. me/github-process-research/github-processresearch

Pipinellis, A. (2015). *GitHub essentials : unleash the power of collaborative workflow development using GitHub, one step at a time*.

Plonka, F. E. (1997). Developing a Lean and Agile Work Force. In *Human Factors and Ergonomics in Manufacturing* (Vol. 7, Issue 1).

P'shewchuk, J. (1998). *Agile Manufacturing: One Size Does Not Fit All*.

R. M. Wideman. (2006). *Agile Project Management*. Anacom.

Raasch, C., Herstatt, C., & Balka, K. (2009). *On the open design of tangible goods*.

Ramírez-Mora, S. L., Oktaba, H., Gómez-Adorno, H., & Sierra, G. (2021). Exploring the communication functions of comments during bug fixing in Open Source Software projects. *Information and Software Technology*, *136*. https://doi.org/10.1016/j.infsof.2021.106584

Rasnacis, A., & Berzisa, S. (2016). Method for Adaptation and Implementation of Agile Project Management Methodology. *Procedia Computer Science*, *104*, 43–50. https://doi.org/10.1016/j.procs.2017.01.055

Rebensdorf, A., Gergert, A., Oosthuizen, G. A., & Böhm, S. (2015). Open community manufacturing - Development Challenge as a concept for value creation for sustainable manufacturing in South Africa. *Procedia CIRP*, *26*, 167–172. https://doi.org/10.1016/j.procir.2015.01.012

Redlich, T. (2010). *Open Production Gestaltungsmodell für die Wertschöpfung in der Bottom-up-Ökonomie*.

Rothaermel, F. T., & Alexandre, M. T. (2009). Ambidexterity in technology sourcing: The moderating role of absorptive capacity. *Organization Science*, *20*(4), 759–780. https://doi.org/10.1287/orsc.1080.0404

Saebi, T., & Foss, N. J. (2015). Business models for open innovation: Matching heterogeneous open innovation strategies with business model dimensions. *European Management Journal*, *33*(3), 201–213. https://doi.org/10.1016/j.emj.2014.11.002

Sanchez, L. M., & Nagi, R. (2001). A review of agile manufacturing systems. *International Journal of Production Research*, *39*(16), 3561–3600. https://doi.org/10.1080/00207540110068790

Schmidt, C. (2016). *Agile Software Development* (pp. 7–35). https://doi.org/10.1007/978-3-319-26057-0_2

Schueller, W., Wachs, J., Servedio, V. D. P., Thurner, S., & Loreto, V. (2022). Evolving collaboration, dependencies, and use in the Rust Open Source Software ecosystem. *Scientific Data*, *9*(1). https://doi.org/10.1038/s41597-022-01819-z

Schwaber, K. (1997). *SCRUM Development Process*. http://vai1.al.adzona.edu/rugby/rad/rookie

Schwaber, K. (2004). *Agile Project Management with SCRUM* (M. Press, Ed.).

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum* (Vol. 1). Upper Saddle River: Prentice Hall.

Schwaber, K., & Sutherland, J. (2016). *The Scrum guide - the definitive guide to Scrum: The rules of the game*.

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

Senabre Hidalgo, E. (2019). *Adapting the scrum framework for agile project management in science: case study of a distributed research initiative*. https://doi.org/10.1016/j.heliyon.2019

Serrador, P., & Pinto, J. K. (2015). Does Agile work? - A quantitative analysis of agile project success. *International Journal of Project Management*, *33*(5), 1040–1051. https://doi.org/10.1016/j.ijproman.2015.01.006

Shaikh, S., & Abro, S. (2019). COMPARISON OF TRADITIONAL AND AGILE SOFTWARE DEVELOPMENT METHODOLOGY: A SHORT SURVEY. *International Journal of Software Engineering and Computer Systems*, *5*(2), 1–14. https://doi.org/10.15282/ijsecs.5.2.2019.1.0057

Shari, H., Colquhoun, G., Barclay, I., & Dann, Z. (2000). *Agile manufacturing: a management and operational framework*.

Sheffield, J., & Lemétayer, J. (2013). Factors associated with the software development agility of successful projects. *International Journal of Project Management*, *31*(3), 459–472. https://doi.org/10.1016/j.ijproman.2012.09.011

Sherehiy, B., Karwowski, W., & Layer, J. K. (2007). A review of enterprise agility: Concepts, frameworks, and attributes. *International Journal of Industrial Ergonomics*, *37*(5), 445–460. https://doi.org/10.1016/j.ergon.2007.01.007

Stankovic, D., Nikolic, V., Djordjevic, M., & Cao, D. B. (2013). A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. *Journal of Systems and Software*, *86*(6), 1663–1678. https://doi.org/10.1016/j.jss.2013.02.027

State of Agile. (2022). *State of Agile Report*. https://stateofagile.com/

Suetin, S., Vikhodtseva, E., Nikitin, S., Lyalin, A., & Brikoshina, I. (2016). *Results of agile project management implementation in software engineering companies*. https://doi.org/10.1051/C

Tiemann, M. (2009). *How Open Source Software Can Save the ICT Industry One Trillion Dollars per Year*.

Tooze, J., Baurley, S., Phillips, R., Smith, P., Foote, E., & Silve, S. (2014). Open design: Contributions, solutions, processes and projects. *Design Journal*, *17*(4), 538–559. https://doi.org/10.2752/175630614X14056185480069

Torrecilla-Salinas, C. J., Sedeño, J., Escalona, M. J., & Mejías, M. (2015). Estimating, planning and managing Agile Web development projects under a value-based perspective. *Information and Software Technology*, *61*, 124–144. https://doi.org/10.1016/j.infsof.2015.01.006

Vallance, R., Kiani, S., & Nayfeh, S. (2001). *OPEN DESIGN OF MANUFACTURING EQUIPMENT*. http://www.opendesign.org

van der Bij, E. (2018). *Open source hardware and KiCAD*.

Wagstrom, P., Jergensen, C., & Sarma, A. (2012). *Roles in a Networked Software Development Ecosystem: A Case Study in GitHub*. http://digitalcommons.unl.edu/csetechreportshttp://digitalcommons.unl.edu/csetechreports/149

Wassmer, U., Li, S., & Madhok, A. (2017). Resource ambidexterity through alliance portfolios and firm performance. *Strategic Management Journal*, *38*(2), 384–394. https://doi.org/10.1002/smj.2488

Wessel, M., Mens, T., Decan, A., & Mazrae, P. R. (2023). *The GitHub Development Workflow Automation Ecosystems*. http://arxiv.org/abs/2305.04772

West, J., & Gallagher, S. (2006). *Challenges of open innovation: the paradox of firm investment in open-source software*.

Xia, X., Weng, Z., Wang, W., & Zhao, S. (2022). Exploring Activity and Contributors on GitHub: Who, What, When, and Where. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, *2022-December*, 11–20. https://doi.org/10.1109/APSEC57359.2022.00013

Young, C. (2013). Community management that works: How to build and sustain a thriving online health community. In *Journal of Medical Internet Research* (Vol. 15, Issue 6). JMIR Publications Inc. https://doi.org/10.2196/jmir.2501

Zöller, N., Morgan, J. H., & Schröder, T. (2020). A topology of groups: What GitHub can tell us about online collaboration. *Technological Forecasting and Social Change*, *161*. https://doi.org/10.1016/j.techfore.2020.120291

Žužek, T., Gosar, Ž., Kušar, J., & Berlec, T. (2020). Adopting agile project management practices in non-software SMEs: A case study of a slovenian medium-sized manufacturing company. *Sustainability (Switzerland)*, *12*(21), 1–17. https://doi.org/10.3390/su12219245

APÊNDICE A – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA NÚMERO DE *RELEASES*

```python
import requests
import pandas as pd

TOKEN = "token de acesso pessoal"

HEADERS = {
    'Authorization': f'token {TOKEN}'
}

def get_releases(repo):
    releases = []
    page = 1
    while True:
        url = f"https://api.github.com/repos/{repo}/releases?per_page=100&page={page}"
        response = requests.get(url, headers=HEADERS)
        data = response.json()
        if not data:
            break
        releases.extend(data)
        page += 1
    return releases

repo = "fossasia/pslab-hardware"

releases = get_releases(repo)

df = pd.DataFrame({
    'Release Date': [release['published_at'] for release in releases]
})

df['Release Date'] = pd.to_datetime(df['Release Date']).dt.tz_localize(None)

df = df[df['Release Date'] <= '2023-08-31']

monthly_releases = df.resample('M', on='Release Date').count()

path = "C:/Users/Utilizador/Documentos/ProjetosDissertacao/Número de Releases.xlsx"

with pd.ExcelWriter(path) as writer:
    monthly_releases.to_excel(writer, sheet_name='Monthly Releases')
```

APÊNDICE B – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA NÚMERO DE *FORKS*

```python
import requests
import pandas as pd

TOKEN = "token de acesso pessoal"

HEADERS = {
    'Authorization': f'token {TOKEN}'
}

def get_forks(repo):
    forks = []
    page = 1

    while True:
        url = f"https://api.github.com/repos/{repo}/forks?per_page=100&page={page}"
        response = requests.get(url, headers=HEADERS)
        data = response.json()
        if not data:
            break
        forks.extend(data)
        page += 1
    return forks

repo = "fossasia/pslab-hardware"

forks = get_forks(repo)

data = []
for fork in forks:
    owner = fork['owner']['login']
    data.append([fork['created_at'], owner])

df = pd.DataFrame(data, columns=["Date", "Owner"])
df['Date'] = pd.to_datetime(df['Date']).dt.tz_localize(None)

df = df[df['Date'] <= '2023-08-31']

monthly_counts = df.resample('M', on='Date').size()

path = "C:/Users/Utilizador/Documentos/ProjetosDissertacao/Número de Forks.xlsx"
with pd.ExcelWriter(path) as writer:
    monthly_counts.to_excel(writer, sheet_name="Forks Mensal")
    df.to_excel(writer, sheet_name="Detalhes dos Forks", index=False)

print(f"Os dados foram salvos em {path}")
```

APÊNDICE C – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA NÚMERO DE *COMMITS* E *PUSHES*

```sql
WITH YearData AS (
    SELECT
        FORMAT_TIMESTAMP('%Y-%m-01', TIMESTAMP(created_at)) as event_month,
        COUNT(*) AS pushes,
        SUM(ARRAY_LENGTH(JSON_EXTRACT_ARRAY(payload, '$.commits'))) AS commits
    FROM `githubarchive.year.*`
    WHERE repo.id = 86472815 and type = 'PushEvent'
    GROUP BY event_month
),
DayData AS (
    SELECT
        FORMAT_TIMESTAMP('%Y-%m-01', TIMESTAMP(created_at)) as event_month,
        COUNT(*) AS pushes,
        SUM(ARRAY_LENGTH(JSON_EXTRACT_ARRAY(payload, '$.commits'))) AS commits
    FROM `githubarchive.day.2023*`
    WHERE repo.id = 86472815 and type = 'PushEvent'
    AND created_at <= '2023-08-31'
    GROUP BY event_month
)
SELECT event_month, SUM(pushes) AS pushes, SUM(commits) AS commits
FROM (
    SELECT * FROM YearData
    UNION ALL
    SELECT * FROM DayData
)
GROUP BY event_month
ORDER BY event_month;
```

APÊNDICE D – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA NÚMERO DE ESTRELAS

```sql
WITH stars AS (
    SELECT
        repo.id AS repo_id,
        created_at,
        ROW_NUMBER() OVER (PARTITION BY repo.id, actor.login) AS row_num
    FROM `githubarchive.year.*`
    WHERE
        type = 'WatchEvent'
        AND repo.id IN (86472815)
    UNION ALL
    SELECT
        repo.id AS repo_id,
        created_at,
        ROW_NUMBER() OVER (PARTITION BY repo.id, actor.login) AS row_num
    FROM `githubarchive.day.2023*`
    WHERE
        type = 'WatchEvent'
        AND repo.id IN (86472815)
        AND created_at <= '2023-08-31'
),
stars_per_month AS (
    SELECT
        repo_id,
        FORMAT_TIMESTAMP('%Y-%m-01', TIMESTAMP(created_at)) AS t_month,
        COUNT(*) AS total
    FROM stars
    WHERE
        row_num = 1
    GROUP BY
        repo_id, t_month
),
acc AS (
    SELECT
        repo_id,
        t_month,
        SUM(total) OVER (PARTITION BY repo_id ORDER BY t_month) AS total,
        ROW_NUMBER() OVER(PARTITION BY repo_id, t_month) AS row_num
    FROM
        stars_per_month
)
SELECT
    repo_id,
    t_month AS event_month,
    total
FROM acc
WHERE row_num = 1
ORDER BY t_month;
```

APÊNDICE E – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA NÚMERO DE *ISSUES* (ABERTAS E FECHADAS)

```
SELECT
  EXTRACT(YEAR FROM created_at) as year,
  EXTRACT(MONTH FROM created_at) as month,
  COUNTIF(action='opened') AS open_issues,
  COUNTIF(action='closed') AS closed_issues,
FROM (
  SELECT
    type AS event_type,
    JSON_EXTRACT_SCALAR(payload, '$.action') AS action,
    actor.id AS actor_id,
    JSON_EXTRACT_SCALAR(payload, '$.issue.user.id') AS user_id,
    JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS issue_number,
    created_at
  FROM `githubarchive.year.*`
  WHERE repo.id = 86472815
  AND type IN ('IssuesEvent')
  UNION ALL
  SELECT
    type AS event_type,
    JSON_EXTRACT_SCALAR(payload, '$.action') AS action,
    actor.id AS actor_id,
    JSON_EXTRACT_SCALAR(payload, '$.issue.user.id') AS user_id,
    JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS issue_number,
    created_at
  FROM `githubarchive.day.2023*`
  WHERE repo.id = 86472815
  AND type IN ('IssuesEvent')
  AND created_at <= '2023-08-31'
)
GROUP BY year, month
ORDER BY year, month;
```

APÊNDICE F – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA NÚMERO DE *PULL REQUESTS* (ABERTOS, FECHADOS E *MERGED*)

```
WITH all_data AS (
  SELECT
    type AS event_type,
    JSON_EXTRACT_SCALAR(payload, '$.action') AS action,
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged') AS merged,
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.user.id') AS user_id,
    actor.id AS actor_id,
    EXTRACT(YEAR FROM created_at) as year,
    EXTRACT(MONTH FROM created_at) as month
  FROM `githubarchive.year.*`
  WHERE repo.id = 86472815
  AND type IN ('PullRequestEvent')
  UNION ALL
  SELECT
    type AS event_type,
    JSON_EXTRACT_SCALAR(payload, '$.action') AS action,
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged') AS merged,
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.user.id') AS user_id,
    actor.id AS actor_id,
    EXTRACT(YEAR FROM created_at) as year,
    EXTRACT(MONTH FROM created_at) as month
  FROM `githubarchive.day.2023*`
  WHERE repo.id = 86472815
  AND type IN ('PullRequestEvent')
  AND created_at <= '2023-08-31'
)

SELECT
  year,
  month,
  COUNT(*) AS total_prs,
  COUNTIF(action='opened') AS open_prs,
  COUNTIF(action='closed') AS closed_prs,
  COUNTIF(merged='true') AS merged_prs,
FROM all_data
GROUP BY year, month
ORDER BY year, month;
```

APÊNDICE G – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA TEMPO MÉDIO DE RESPOSTA/FECHO *ISSUE*

```sql
WITH issue_with_first_responded_at AS (
    SELECT
        JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS number,
        MIN(FORMAT_TIMESTAMP('%Y-%m', TIMESTAMP(created_at))) AS t_month,
        MIN(created_at) AS first_responded_at
    FROM `githubarchive.year.*`
    WHERE repo.id = 86472815
    AND (type = 'IssueCommentEvent' OR (type = 'IssuesEvent' AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'closed'))
    AND created_at >= '2013-08-19'
    GROUP BY JSON_EXTRACT_SCALAR(payload, '$.issue.number')
    UNION ALL
    SELECT
        JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS number,
        MIN(FORMAT_TIMESTAMP('%Y-%m', TIMESTAMP(created_at))) AS t_month,
        MIN(created_at) AS first_responded_at
    FROM `githubarchive.day.2023*`
    WHERE repo.id = 86472815
    AND (type = 'IssueCommentEvent' OR (type = 'IssuesEvent' AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'closed'))
    AND created_at <= '2023-08-31'
    GROUP BY JSON_EXTRACT_SCALAR(payload, '$.issue.number')
),
issue_with_opened_at AS (
    SELECT
        JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS number,
        created_at AS opened_at
    FROM `githubarchive.year.*`
    WHERE repo.id = 86472815
    AND type = 'IssuesEvent' AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'opened'
    AND created_at >= '2013-08-19'
    UNION ALL
    SELECT
        JSON_EXTRACT_SCALAR(payload, '$.issue.number') AS number,
        created_at AS opened_at
    FROM `githubarchive.day.2023*`
    WHERE repo.id = 86472815
    AND type = 'IssuesEvent' AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'opened'
    AND created_at <= '2023-08-31'
),
tdiff AS (
    SELECT
        t_month,
        AVG(UNIX_SECONDS(TIMESTAMP(iwfr.first_responded_at)) - UNIX_SECONDS(TIMESTAMP(iwo.opened_at))) / 3600 AS avg_diff_hours
    FROM
        issue_with_opened_at iwo
        JOIN issue_with_first_responded_at iwfr ON iwo.number = iwfr.number AND iwfr.first_responded_at > iwo.opened_at
    GROUP BY t_month
)
SELECT
    t_month AS event_month,
    avg_diff_hours
FROM tdiff
ORDER BY event_month;
```

APÊNDICE H – *SCRIPT* PARA A EXTRAÇÃO DA MÉTRICA TEMPO MÉDIO *PULL REQUEST*

```
WITH pr_times AS (
  SELECT
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.created_at') AS pr_created_at,
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged_at') AS pr_merged_at,
    TIMESTAMP_DIFF(
      TIMESTAMP(JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged_at')),
      TIMESTAMP(JSON_EXTRACT_SCALAR(payload, '$.pull_request.created_at')),
      HOUR) AS pr_time_hours
  FROM `githubarchive.year.*`
  WHERE repo.id = 86472815
  AND type = 'PullRequestEvent'
  AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'closed'
  AND JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged') = 'true'
  UNION ALL
  SELECT
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.created_at') AS pr_created_at,
    JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged_at') AS pr_merged_at,
    TIMESTAMP_DIFF(
      TIMESTAMP(JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged_at')),
      TIMESTAMP(JSON_EXTRACT_SCALAR(payload, '$.pull_request.created_at')),
      HOUR) AS pr_time_hours
  FROM `githubarchive.day.2023*`
  WHERE repo.id = 86472815
  AND type = 'PullRequestEvent'
  AND JSON_EXTRACT_SCALAR(payload, '$.action') = 'closed'
  AND JSON_EXTRACT_SCALAR(payload, '$.pull_request.merged') = 'true'
  AND created_at <= '2023-08-31'
)

SELECT
  EXTRACT(YEAR FROM TIMESTAMP(pr_created_at)) as year,
  EXTRACT(MONTH FROM TIMESTAMP(pr_created_at)) as month,
  AVG(pr_time_hours) AS avg_pr_time_hours
FROM pr_times
GROUP BY year, month
ORDER BY year, month;
```