Classwork: Uma ferramenta de acompanhamento em tempo real da contribuição individual de alunos de cursos de computação no desenvolvimento de projetos de software acadêmicos hospedados no GitHub

Lucas Marcel Silva de Brito[1], João Helis Bernardo[1]

[1]Instituto Federal do Rio Grande do Norte (IFRN)
Macau – RN – Brasil

lucas.marcel@escolar.ifrn.edu.br, joao.helis@ifrn.edu.br

Abstract. *Tools that provide information to software project managers about their developers' contributions have gained prominence in recent years. However, little research has focused on the use of tools that present managerial information to computer science teachers, who are responsible for monitoring software projects developed by their students, usually manually. In this context, this work aims to propose and evaluate the Classwork tool, which provides to computer teachers, in an automated way, information about the individual contribution of their students in the development of academic software projects hosted on GitHub.*

Resumo. *Ferramentas que fornecem informações a gerentes de projetos de software sobre a contribuição dos seus desenvolvedores têm ganhado destaque nos últimos anos. Todavia, poucas pesquisas têm focado na utilização de ferramentas que apresentem informações gerenciais a docentes da área de computação, que têm a responsabilidade de realizar o acompanhamento dos projetos de software desenvolvidos por seus alunos, usualmente de forma manual. Neste contexto, este trabalho tem o objetivo de propor e avaliar a ferramenta Classwork, que apresenta aos docentes de computação, de forma automatizada, informações sobre a contribuição individual dos seus alunos no desenvolvimento de projetos de software acadêmicos hospedados no GitHub.*

1. Introdução

A engenharia de *software* é a área da computação responsável pelo estudo das ferramentas, métodos e técnicas utilizadas no processo de desenvolvimento de software, com o objetivo de garantir que este seja desenvolvido dentro do prazo, custo e qualidade pré-definidos [Pressman 2005]. Os cursos de computação, de nível técnico e superior, em todo o mundo, têm a disciplina de engenharia de software em suas bases curriculares e usualmente utilizam a aplicação dos conceitos teóricos referentes a esta disciplina sob uma perspectiva prática, de forma que alunos, normalmente em grupos, contribuem colaborativamente para simular situações experienciadas durante o exercício da profissão, a exemplo do desenvolvimento de um projeto de software, e por esta colaboração são avaliados sob o ponto de vista acadêmico.

DOI: 10.5753/sbie.2021.217491

Uma das sub-áreas com considerável destaque dentro da Engenharia de Software é a gerência de projetos de software, tendo em vista que o gerenciamento ineficaz de um projeto pode ser considerado um fator determinante para o seu fracasso [Sommerville 2003]. Neste sentido, a figura do gerente de projetos tem grande responsabilidade na garantia da execução dos projetos que gerenciam. Analogamente, em um contexto educacional de um curso de computação, têm-se a figura do docente, que deve gerenciar e acompanhar efetivamente a evolução dos projetos de software desenvolvidos por seus alunos. Este acompanhamento, contudo, pode ser um fator determinante para a garantia da boa execução do projeto proposto, bem como, servir como mecanismo de avaliação da aprendizagem do aluno, que deve utilizar-se efetivamente da união da teoria com a prática como instrumento impulsionador do seu processo educativo [Dearden 2011]. Neste sentido, o diagnóstico prévio de possíveis índices de inatividade de determinado aluno durante o desenvolvimento de um projeto, o que sugere o seu distanciamento da prática, pode permitir que o docente possa agir proativamente, visando a garantia da efetividade do processo de ensino-aprendizagem.

Tanto no contexto industrial quanto no acadêmico, é de fundamental importância que gerentes de projetos e docentes utilizem estratégias capazes de prover informações em tempo real sobre a contribuição individual de cada um dos membros das equipes que gerenciam, de forma a embasar melhor suas decisões, de cunho estratégico e operacional. Todavia, a coleta manual dessas informações, no âmbito industrial, pode demandar um alto consumo de tempo e aumentar os custos do projeto [Costa 2013], enquanto no âmbito educacional, pode tornar-se inviável dado a quantidade de alunos e projetos que cada docente deve monitorar durante a ministração de uma disciplina de desenvolvimento de software.

Como alternativa para automatização do acompanhamento da contribuição individual de alunos de cursos de computação em projetos de software, pode-se levar em consideração que desenvolvedores, estejam estes na indústria ou academia, utilizam Sistemas Descentralizados de Controle de Versão (SDCV) para hospedar os seus projetos, a exemplo do GitHub. Além desses sistemas, utilizam também aplicativos de comunicação (e.g. Slack), e-mails, logs de sistema, bancos de dados, dentre outros repositórios que guardam informações sobre suas ações, bem como dados relevantes sobre a evolução do projeto [Hassan 2006]. A extração, processamento e transformação desses dados em informações úteis tem sido o foco da área de Mineração de Repositório de Software (MRS). Diversos trabalhos têm utilizado métricas de software extraídas desses repositórios [Silva 2017; Wang *et al*. 2013; Zhang & Lee 2013], para embasar a construção de ferramentas que coletam e apresentam informações gerenciais sobre projetos de software. No entanto, pouco foi pesquisado para melhor entender os desafios enfrentados por docentes de computação no que concerne o gerenciamento das contribuições individuais dos seus alunos em projetos de software. Adicionalmente, ainda existe uma lacuna na literatura sobre pesquisas que utilizem técnicas de MSR para criação de ferramentas que proveem informações automatizadas sobre dados de repositórios de software no contexto educacional de ensino.

O presente trabalho tem o objetivo de propor e avaliar a ferramenta Classwork, desenvolvida com o intuito de fornecer aos docentes de computação, de forma automatizada, informações sobre a contribuição individual dos seus alunos no desenvolvimento de projetos de software hospedados no GitHub. A ferramenta Classwork destina-se, portanto, a qualquer docente da área de computação que pretenda

monitorar suas turmas e alunos, sob o ponto de vista da contribuição de código fonte em projetos de software.

2. A Mineração de Repositórios de Software na Extração de Informações de Contribuições em Projetos do GitHub

A Mineração de Repositórios de Software (MSR) é uma sub-área da computação que utiliza um conjunto de ferramentas e técnicas para extrair dados pertinentes ao desenvolvimento e evolução de projetos de software, por meio da mineração dos seus mais diversos repositórios, a exemplo de servidores de e-mail, chats, sistemas de logs ou sistema de controle de versão [Meireles 2013]. Estes repositórios têm sido amplamente utilizados por pesquisadores da área com base em dois propósitos específicos, que consistem em (i) utilizar dados reais de projetos para validar novas técnicas ou abordagens previamente propostas; e (ii) elaborar técnicas para extrair e apresentar informações relevantes sobre um determinado projeto de software [Hassan 2006].

Um conjunto numeroso de trabalhos podem ser citados como exemplos de pesquisas da área de MSR que utilizam dados de projetos para validar abordagens teóricas previamente propostas [Bernardo *et al*. 2018; Vasilescu *et al*. 2015; Zhao *et al*. 2017]. O trabalho de Bernardo *et al*. (2018), por exemplo, investiga dados de 87 projetos do GitHub para avaliar o impacto da adoção da Integração Contínua no tempo de entrega de *Pull-Requests*. Adicionalmente, os trabalhos de Silva (2017) e Treude & Storey (2010), podem ser considerados esforços em direção a apresentação de informações visuais de projetos de software para finalidades gerenciais. Neste contexto, dá-se destaque também ao crescimento das plataformas online de versionamento de código de projetos, a exemplo do GitHub, que têm possibilitado inúmeras pesquisas a partir da disponibilização dos dados dos projetos de código aberto nelas hospedados.

O GitHub é o provedor de hospedagem de código com maior popularidade entre os desenvolvedores de software de todo o mundo. É amplamente utilizado no contexto acadêmico e industrial. A plataforma conta com mais de 50 milhões de usuários e 100 milhões de repositórios de software. Possui também uma API (*Application Programming Interface*) de acesso aos dados dos projetos que estão versionados na plataforma, desta forma, possibilitando a extração destes dados para embasar a análise de inúmeras pesquisas na área de MSR. Adicionalmente, a API do GitHub permite ainda a extração de dados em tempo real dos seus projetos, o que pode levar a análise e monitoramento em tempo real da contribuição de desenvolvedores, por exemplo.

Contudo, é necessário ainda que métricas de software sejam aplicadas aos dados extraídos do GitHub, a fim de que seja possível quantificar a evolução dos projetos. De acordo com Costa (2013), é possível medir a contribuição individual de desenvolvedores de software, sob o ponto de vista do tamanho da contribuição, a partir da utilização da métrica *chrun*, que corresponde a soma da quantidade de linhas adicionadas e removidas em uma determinada contribuição. Ainda de acordo com Meireles (2013), a indicação do tamanho do *churn* nas contribuições realizadas em um determinado intervalo de tempo pode ser considerado um indicador à saúde do projeto. Além do *churn*, existem também várias outras métricas relacionadas a um projeto de software que podem ser consideradas

em sua análise, bem como fatores qualitativos que não podem ser extraídos apenas por meio de análise estática de código.

3. Trabalhos Relacionados

Ao longo dos últimos anos, especialmente a partir da popularização dos Sistemas de Controle de Versão (SCV), a exemplo do SVN[1] e Git[2], pesquisas vêm sendo realizadas com o propósito de mensurar a contribuição de desenvolvedores de software em seus respectivos projetos. O trabalho de Costa (2013) investiga projetos versionados no SVN para identificar a contribuição dos seus desenvolvedores sob três perspectivas: (i) *commits* defeituosos; (ii) tamanho dos *commits*; e (iii) resolução de *bugs*. De forma complementar, Lima (2014) propõe uma abordagem que estende o trabalho de Costa (2013), adicionando uma nova métrica na avaliação da contribuição dos desenvolvedores de software, a Complexidade Ciclomática.

O trabalho de Silva (2019) apresenta a SPM, uma ferramenta para gerenciamento de projetos do GitHub baseada em técnicas de MSR. A ferramenta fornece uma interface de comunicação entre o cliente do sistema e o gerente de projetos. O cliente pode acompanhar as atividades em desenvolvimento nos projetos monitorados via interface web ou por mensagens no Telegram. Já os gerentes de projetos podem ter acesso a métricas de software relacionadas às contribuições dos desenvolvedores das equipes que gerenciam.

A Classwork, ferramenta proposta neste trabalho, tem o seu foco no monitoramento de contribuições de alunos de cursos de computação em projetos de software hospedados no GitHub. Diferentemente dos trabalhos de Costa (2013) e Lima (2014), os quais propõem ferramentas que suportam projetos advindos do SVN, e necessitam baixar o código do projeto localmente para extrair métricas de software relativas às contribuições dos desenvolvedores, a Classwork extrai informações de forma automatizada, a partir de uma interface web, que monitora dados em tempo real da contribuição de desenvolvedores/alunos advindas de projetos do GitHub. Essa extração automatizada também é realizada no trabalho de Silva (2019), contudo, a Classwork apresenta funcionalidades que permitem realizar análise da contribuição de alunos em seus respectivos repositórios de software em bloco, o que facilita o trabalho de docentes ao avaliarem os projetos de software desenvolvidos por alunos das suas respectivas turmas. A partir do Classwork, é permitido a criação de Turmas, e dentro destas turmas a inclusão de Equipes, as quais são associadas a um determinado repositório do GitHub. A partir dessa associação, de forma automatizada é possível a extração de informações sobre a contribuição de código realizada em toda turma, bem como de forma específica em cada equipe cadastrada.

4. Abordagem Metodológica

Este trabalho caracteriza-se, sob o ponto de vista metodológico, como uma pesquisa aplicada, por seu interesse prático, ao prover a docentes da área de computação uma ferramenta para acompanhamento em tempo real da contribuição dos seus alunos em projetos de softwares acadêmicos. Segundo Marconi e Lakatos (2017), na pesquisa

[1] https://subversion.apache.org/
[2] https://git-scm.com/

aplicada pretende-se que seus "resultados sejam aplicados ou utilizados, imediatamente, na solução de problemas que ocorrem na realidade". Aplicou-se, portanto, o método *Design Science Research* (DSR) [Wieringa 2014]. Este método é adequado quando a pesquisa se objetiva no projeto e desenvolvimento de artefatos. A partir da compreensão do problema de pesquisa, o método DSR busca elaborar e avaliar artefatos que alterem situações e melhorem a realização de atividades.

A metodologia de execução da pesquisa se deu a partir da execução das seguintes ações: (i) Levantamento bibliográfico sobre métricas de software utilizadas para o monitoramento de projetos de software; (ii) Estudo sobre o arcabouço tecnológico a ser utilizado para o desenvolvimento da ferramenta; (iii) Elaboração do projeto e arquitetura da ferramenta; (iv) Desenvolvimento da ferramenta; e (v) Avaliação da ferramenta com um conjunto de potenciais usuários.

De forma a coordenar a execução das ações, utilizou-se o *framework* de gerenciamento de projetos *Scrum*. O *Scrum* é um *framework* para desenvolvimento, entrega e manutenção de produtos complexos. Este *framework* foi desenvolvido no início dos anos 90 e tem sido extensivamente utilizado para o gerenciamento de projetos dos mais diversos tipos, a exemplo do desenvolvimento e entrega de produtos de *software* [Sutherland & Schwaber 2013]. O *Scrum* é composto por um conjunto de papéis, eventos, artefatos e regras para gerência de projetos, que podem ser adaptadas a partir das necessidades específicas de cada projeto. Dentre os eventos do *Scrum*, destaca-se a *Sprint*, que é um intervalo fixo de tempo que vai de 2 a 4 semanas, utilizado para a execução de ações predeterminadas antes do seu início, em uma reunião chamada de reunião de planejamento (realizada entre todos os membros do projeto). No contexto do projeto em questão, foram utilizadas *Sprints* de 2 semanas.

4.1. Avaliação de usabilidade e o formulário SUS

Para avaliação da ferramenta, foi aplicado o questionário *System Usability Scale* (SUS). Este questionário pré-estabelece um conjunto de 10 questões relativas à usabilidade de um sistema, com o intuito de mensurar o grau de aceitação dos usuários em relação às suas múltiplas funcionalidades [Martins *et al.* 2015]. As questões do formulário SUS seguem o padrão de respostas baseadas na escala likert de 5 pontos [JOSHI 2015], contendo as seguintes opções: discordo completamente, discordo, neutro, concordo, concordo completamente. A avaliação se dá a partir das respostas advindas da aplicação do questionário, de forma que, a partir da média total da pontuação atribuída por cada participante da avaliação, chega-se a uma pontuação final para a ferramenta. As categorias são: (i) pior imaginável: menor que 20,5 pontos, (ii) pobre: 21 a 38,5 pontos, (iii) mediano: 39 a 52,5 pontos, (iv) bom: 53 a 73,5 pontos, (v) excelente: 74 a 85,5 pontos e (vi) melhor imaginável: 86 a 100 pontos [Martins *et al.* 2015]. O questionário foi submetido a docentes da área de Ciências da Computação, público-alvo da pesquisa. A escolha dos docentes foi realizada a partir de contato prévio com os autores da pesquisa. A aplicação do questionário foi realizada entre os dias 24 de junho e 3 de julho de 2021 e obteve um total de 8 respostas.

5. Arquitetura da Ferramenta e Tecnologias Utilizadas

Por se tratar de um Software que precisa realizar uma grande quantidade de processos no lado do servidor, a Classwork foi desenvolvida utilizando a arquitetura baseada no

conceito de REST (*Representational State Transfer*), onde as funções da aplicação são interligadas por uma API, e seguindo o modelo conhecido como SPA (*Single Page Application*), que consiste em criar uma aplicação de página única, otimizando alguns processos da ferramenta ao baixar todos os recursos da página web, como o CSS (*Cascading Style Sheets*), HTML (*HyperText Markup Language*) e o Javascript, com antecedência.

Ao seguir essa arquitetura, podemos dividir a Classwork em dois módulos distintos: a aplicação web, que está relacionada ao lado do cliente, e a API, representada pelo lado do servidor. A divisão entre o front-end e o back-end permite que o servidor se dedique apenas a responder as requisições HTTP (*HyperText Transfer Protocol*) feitas pela aplicação web com os dados solicitados devidamente processados e dentro do formato JSON (*JavaScript Object Notation*). No modelo tradicional, que foi opção descartada, o servidor tem tanto a obrigação de fornecer esses dados, como também o conteúdo que será renderizado no navegador, diminuindo o custo de processamento para a aplicação no lado do cliente e aumentando o do servidor. A Figura 1 apresenta a arquitetura de comunicação da Classwork, destacando principalmente o papel da API dentro do modelo escolhido.

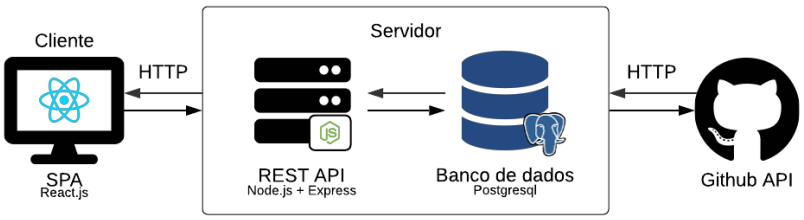

Figura 1. Arquitetura de comunicação da Classwork

A documentação completa da ferramenta está acessível a partir da Seção 9.

6. A ferramenta Classwork

A Classwork pode ser compreendida como uma ferramenta de monitoramento de repositórios públicos hospedados no Github que, com o auxílio de sua interface, possibilita acompanhar os resultados gerais dos repositórios de software monitorados, como também mensurar o envolvimento dos seus membros ao longo do tempo. A partir da ferramenta, possibilita-se o gerenciamento de **Turmas** e de blocos menores dentro dela, as **Equipes** associadas aos **Repositórios**. A Figura 2 apresenta o diagrama de

Entidade Relacionamento (ER) das principais entidades da ferramenta, elucidando como se dá o seu relacionamento.

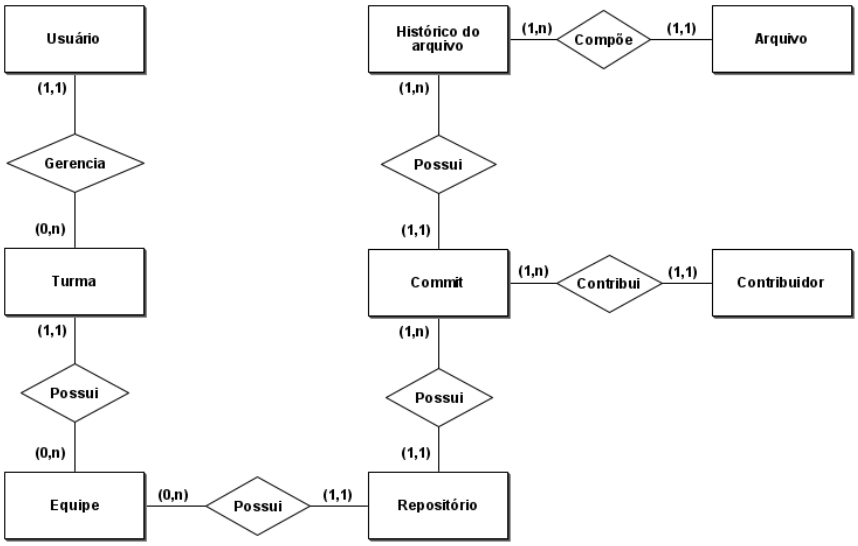

Figura 2. Diagrama de ER das entidades da Classwork

É usual que disciplinas de programação dentro de cursos de computação promovam o desenvolvimento colaborativo de software entre os seus alunos, com propósito de avaliação da aprendizagem. Quando estes softwares são mantidos por meio de repositórios públicos no GitHub, é possível a integração com a Classowrk. A ferramenta em questão torna as informações que são obtidas através da extração de métricas do código fonte ainda mais fáceis de gerenciar, a partir da consolidação de um ambiente de trabalho simplificado, caracterizado pela divisão dos repositórios em dois blocos: as *turmas* e as *equipes*. A Figura 3 apresenta um fragmento da página de uma turma após seu cadastro na ferramenta.

Figura 3. Fragmento da página de Turma

Com a turma é possível reunir, a partir de dois gráficos, todas as informações coletadas a partir das métricas extraídas de cada equipe, que sempre está associada a um repositório. O primeiro gráfico tem como objetivo informar a pontuação atribuída pela Classwork ao código fonte analisado, valor que pode ser calculado através de uma média

ponderada entre o número de alterações nos arquivos, isto é, a soma das linhas adicionadas e removidas deles, e a complexidade ciclomática, que é uma métrica que mensura a complexidade do código com base no número de possíveis resultados a serem obtidos dentro de um método. A Figura 4 apresenta um exemplo do gráfico da pontuação das equipes de uma turma.

Figura 4. Exemplo do gráfico da pontuação das equipes

No entanto, o objetivo da pontuação não é qualificar o desempenho da equipe, visto que um código fonte muito complexo para um software que realiza uma tarefa simples pode ser algo insatisfatório, mas identificar o quanto cada contribuidor se envolveu na sua produção. Por essa razão, o segundo gráfico apresenta um valor em porcentagem que diz respeito a relação que a pontuação de cada contribuidor tem com a pontuação total do repositório: o *equilíbrio*. Uma equipe que tem o seu equilíbrio em 100% significa que todos os seus colaboradores/alunos contribuíram igualmente no desenvolvimento do software. A Figura 5 apresenta um exemplo do gráfico do equilíbrio das equipes de uma turma.

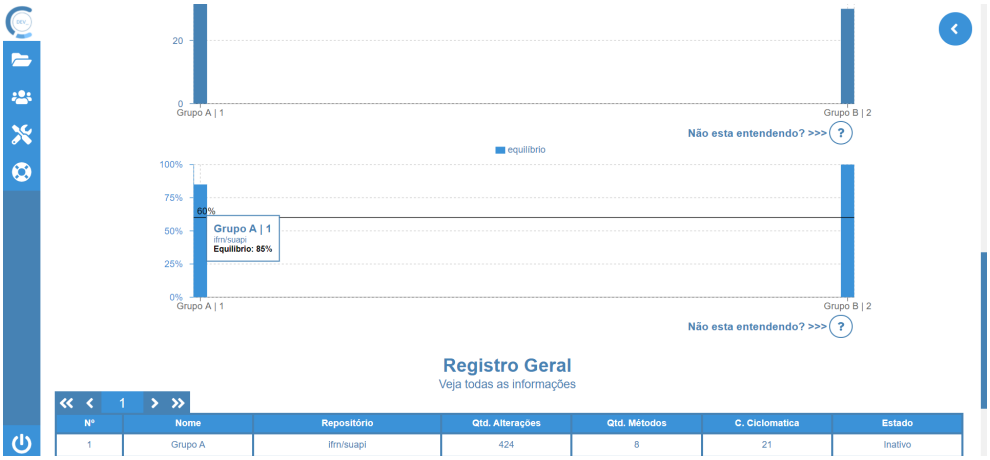

Figura 5. Exemplo do gráfico do equilíbrio das equipes

Por fim, tendo em vista que todos os dados de repositórios de software fornecidos pela API do Github se organizam cronologicamente de acordo com as alterações

efetuadas pelos seus contribuidores, a Classwork foi projetada para recriar o conteúdo do código fonte seguindo a ordem de suas alterações e salvando seu estado atual em um banco de dados local, mantendo uma espécie de histórico próprio das versões do software. Essas características da interação do GitHub com a ferramenta dão os recursos necessários para que ela possa reduzir o tempo na leitura de repositórios que já foram carregados anteriormente e se encontram salvos no banco de dados local, e entregar aos usuários uma lista de equipes inativas, isto é, que não registraram alterações dentro do intervalo de tempo solicitado. Identificar, de forma automatizada, equipes inativas em disciplinas de programação, possibilita que docentes possam, de forma proativa, tomar medidas que retomem o desenvolvimento do projeto por parte dos alunos, de forma a reestabelecer os objetivos de aprendizagem predefinidos para a disciplina. A Figura 6 apresenta um fragmento da página da ferramenta que lista equipes inativas de uma turma dentro de um intervalo de tempo determinado.

Figura 6. Exemplo da lista de equipes inativas

7. Avaliação de Usabilidade

Como resultado da avaliação de usabilidade, que contou com 8 participantes, a ferramenta Classwork obteve um total de 80,94 pontos, o que a coloca dentro da categoria excelente: 74 a 85,5 pontos, de acordo com os critérios pré-estabelecidos pelo questionário SUS [Martins *et al.* 2015]. Tal pontuação indica que as funcionalidades da ferramenta em seu estado atual foram avaliadas positivamente pelos seus potenciais usuários, docentes de computação. A Tabela 1 apresenta a pontuação que cada participante deu à ferramenta.

Tabela 1. Resultado do questionário SUS da ferramenta Classwork

Participante	Pontuação	Participante	Pontuação
Participante 01	85	Participante 05	67,5
Participante 02	87,5	Participante 06	85
Participante 03	95	Participante 07	65
Participante 04	77,5	Participante 08	85
Pontuação Geral			
80,94			

8. Considerações Finais

A aplicação web proposta neste trabalho obteve sucesso na execução de suas principais funcionalidades, assim como se demonstrou promissora na manipulação e extração de grandes bases dados. Além disso, a interface gráfica desenvolvida permitiu que os usuários pudessem acessar as funcionalidades de forma satisfatória, conforme resultado da avaliação de usabilidade da ferramenta. Neste sentido, a Classwork apresenta-se como uma alternativa viável de ser utilizada por docentes de cursos de computação, especialmente os que lecionam disciplinas relacionadas ao desenvolvimento de software, como instrumento de monitoramento da contribuição de alunos em projetos de software hospedados no GitHub. Esta ferramenta busca prover melhoria no processo de ensino-

aprendizagem de disciplinas de programação, uma vez que possibilita, de forma automatizada, identificar índices de inatividades de equipes de desenvolvimento (compostas por alunos) ao longo do tempo, desta forma, permitindo ações proativas por parte dos docentes, a fim de garantir os objetivos de aprendizagem da disciplina. Além disso, são fornecidas uma série de métricas de código para as turmas, equipes e alunos envolvidos nos projetos monitorados, com o intuito de prover mecanismos que subsidiem a avaliação e acompanhamento dos alunos em tempo real, a partir da utilização de técnicas de análise estática de código.

Dentre as limitações da ferramenta, destaca-se a falta de suporte integral a outras linguagens de programação, com exceção do Java. Adicionalmente, pela ferramenta utilizar a API do GitHub para extrair os dados dos repositórios monitorados, deve-se atentar para o limite de 5 mil requisições por hora definidos por esta API, o que pode levar projetos de software com grande quantidade de *commits* e arquivos a demorarem um tempo considerável para terem suas métricas de software extraídas pela ferramenta.

Como trabalhos futuros, sugere-se ainda: (i) implementar a identificação de contribuidores inativos dentro de um determinado intervalo de tempo, haja vista que atualmente a Classwork permite identificar apenas equipes inativas; (ii) expandir a quantidade de linguagens suportadas; (iii) integrar a Classwork com *Issue Tracker* do GitHub, de forma que se possa mensurar também a quantidade de tarefas executadas por cada contribuidor de uma equipe de software; e (iv) realizar um estudo de caso da aplicação da ferramenta em um ambiente real, de forma que docentes possam avaliar a sua efetividade dentro de um contexto em que estejam de fato ofertando uma disciplina de desenvolvimento de software, bem com os potenciais pedagógicos da ferramenta.

9. URLs

A ferramenta acompanha um guia para os usuários que contém as instruções necessárias para sua utilização, além disso, todos os dados são publicizados com intuito de facilitar a colaboração de desenvolvedores externos que possuam interesse em contribuir com a evolução da ferramenta, uma vez que esta é de código aberto, sob a licença MIT.

- Documentação: https://dev-classwork.gitbook.io/classwork
- Guia para usuários: https://dev-classwork.gitbook.io/classwork/para-usuarios/iniciando-com-o-classwork
- URL para teste da Classwork: https://classwork-master.herokuapp.com
- Repositório da Classwork: https://github.com/dev-classwork/classwork
- Repositório da API: https://github.com/dev-classwork/classwork-server
- Vídeo explicativo: https://youtu.be/Do1IaSwGy5E

Referencias Bibliográficas

Bernardo, J. H., Costa, D. A., Kulesza, U. (2018) Studying the impact of adopting continuous integration on the delivery time of *pull requests*. In: 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR). IEEE, p. 131-141.

Costa, D. A. (2013) Avaliação da contribuição de desenvolvedores para projetos de software usando mineração de repositórios de software e mineração de processos. Dissertação de Mestrado. Universidade Federal do Rio Grande do Norte.

Dearden, R. F. (Ed.). (2011) Theory and Practice in Education (RLE Edu K). Routledge.

Hassan, A. E. (2006) Mining software repositories to assist developers and support managers. In: 2006 22nd IEEE International Conference on Software Maintenance. IEEE, p. 339-342.

Joshi, A. et al. (2015) Likert scale: Explored and explained. British Journal of Applied Science & Technology, v. 7, n. 4, p. 396.

Lima, J. R. F. (2014) Uma abordagem de apoio à gerência de projetos de software para análise da contribuição de desenvolvedores. 2014. 127f. Dissertação (Mestrado Em Sistemas E Computação) - Centro De Ciências Exatas E Da Terra, Universidade Federal do Rio Grande do Norte, Natal.

Marconi, M. A., Lakatos, E. M. (2017) Técnicas de Pesquisa. 8. Ed. São Paulo. Atlas.

Martins, A. I. *et al*. (2015) European portuguese validation of the system usability scale (SUS). Procedia Computer Science, v. 67, p. 293-300.

Meirelles, P. R. M. (2013) Monitoramento de métricas de código-fonte em projetos de software livre. Tese de Doutorado. Universidade de São Paulo.

Pressman, R. S. (2005) Software engineering: a practitioner's approach. Palgrave macmillan.

Silva, H. A. P. (2019) SPM: Uma Ferramenta para Gerenciamento de Projetos do GitHub Baseada em Técnicas de Mineração de Repositório de Software. (Trabalho de Conclusão de Curso) Graduação em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte.

Silva, L. M. (2017) PerfMiner Visualizer: uma ferramenta para análise da evolução do atributo de qualidade de desempenho em sistemas de software. Dissertação de Mestrado. Brasil.

Sommerville, I. (2003) Engenharia de Software. Person Addicon Wesley. São Paulo.

Sutherland, J., Schwaber, K. (2013) The scrum guide. The definitive guide to scrum: The rules of the game. Scrum. org, v. 268.

Treude, C., Storey, M. (2010) Awareness 2.0: staying aware of projects, developers and tasks using dashboards and feeds. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1. p. 365-374.

Vasilescu, B. *et al*. (2015) Quality and productivity outcomes relating to continuous integration in GitHub. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. p. 805-816.

Wang, S., Lo, D; Jiang, L. (2013) An empirical study on developer interactions in stackoverflow. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing. p. 1019-1024.

Wieringa, R. J. (2014) Design science methodology for information systems and software engineering. Dordrecht: Springer.

Zhang, T., Lee, B. (2013) A hybrid bug triage algorithm for developer recommendation. In: Proceedings of the 28th annual ACM symposium on applied computing. p. 1088-1094.

Zhao, Y. *et al*. (2017) The impact of continuous integration on other software development practices: a large-scale empirical study. In: 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE. p. 60-71.