UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DA UFC EM QUIXADÁ

CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

DIRLÂNDIA DE OLIVEIRA DE SOUSA

RIST: UMA SOLUÇÃO PARA A ALOCAÇÃO AUTOMÁTICA DE TAREFAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE BASEADOS NO GITHUB E TRELLO

QUIXADÁ

2023

DIRLÂNDIA DE OLIVEIRA DE SOUSA

RIST: UMA SOLUÇÃO PARA A ALOCAÇÃO AUTOMÁTICA DE TAREFAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE BASEADOS NO GITHUB E TRELLO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em SISTEMAS DE INFORMAÇÃO do Campus da UFC em Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em SISTEMAS DE INFORMAÇÃO.

Orientador: Prof. Dr. Enyo José Tavares Gonçalves.

QUIXADÁ

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S696r Sousa, Dirlândia de Oliveira de.
RIST: Uma solução para a alocação automática de tarefas em projetos de desenvolvimento de software baseados no Github e Trello. / Dirlândia de Oliveira de Sousa. – 2023.
62 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2023.
Orientação: Prof. Dr. Enyo José Tavares Gonçalves.

1. Alocação de Tarefas. Algoritmo Genético. Truck Factor. Trello. Github.. I. Título.

CDD 004

DIRLÂNDIA DE OLIVEIRA DE SOUSA

RIST: UMA SOLUÇÃO PARA A ALOCAÇÃO AUTOMÁTICA DE TAREFAS EM PROJETOS DE DESENVOLVIMENTO DE SOFTWARE BASEADOS NO GITHUB E TRELLO

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em SISTEMAS DE INFORMAÇÃO do Campus da UFC em Quixadá da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em SISTEMAS DE INFORMAÇÃO.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Enyo José Tavares Gonçalves (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Marcos Antônio de Oliveira
Universidade Federal do Ceará (UFC)

Prof. Caetano Vieira Neto Segundo
Faculdade Paraíso (FAP)

À minha mãe Maria Auxiliadora, mulher batalhadora que sempre incentivou que a educação é a melhor e maior maneira de transformar o mundo.

AGRADECIMENTOS

Primeiramente, expresso minha infinita gratidão a Deus, cujas bênçãos têm sido a força motriz da minha vida. Sua presença é a luz que guia cada passo que dou, fornecendo direção e conforto nos momentos mais desafiadores.

À minha mãe e aos meus irmãos, vocês são a verdadeira essência do amor incondicional e da força. Juntos, vivenciamos as alegrias e superamos os desafios da vida. Sou imensamente grata por cada esforço e sacrifício feito por vocês, que moldaram a pessoa que sou hoje. Seu apoio inabalável é meu maior tesouro.

Aos meus queridos sobrinhos, vocês iluminam cada dia com sua alegria e inocência. Observar o crescimento e desenvolvimento de cada um de vocês é um privilégio e uma fonte constante de felicidade. Sou grata por todos os sorrisos, abraços e momentos alegres que compartilhamos.

Um agradecimento especial ao meu orientador, cuja orientação, sabedoria e apoio inestimáveis foram pilares fundamentais em minha jornada acadêmica. Sua paciência, conhecimento e inspiração foram essenciais na busca pela excelência em meu trabalho. Sua dedicação e ensinamentos deixaram marcas profundas e eternas.

À banca examinadora, agradeço pela avaliação criteriosa e pelos feedbacks construtivos. Suas perspectivas e sugestões foram vitais para enriquecer meu projeto, contribuindo significativamente para meu crescimento pessoal e profissional.

Por fim, mas não menos importante, aos meus amigos e parceiros de trajetória acadêmica: vocês foram o suporte emocional e intelectual necessário para tornar esta jornada mais leve e significativa. Cada risada compartilhada, cada palavra de encorajamento, e cada momento de companheirismo foram fundamentais para meu sucesso e bem-estar.

Com profunda gratidão e amor.

“A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original.”
(Albert Einstein)

RESUMO

Diversas etapas e atividades estão envolvidas no processo de desenvolvimento de software, uma destas atividades é a alocação de tarefas. Esta atividade está relacionada ao gerenciamento de projetos e é determinante para o sucesso ou fracasso do projeto, visto que envolve riscos relacionados a tempo e consequentemente custos. No entanto, é importante ressaltar a rotatividade de membros dentro de uma equipe de desenvolvimento, o que pode ocasionar em impactos negativos nos prazos e custo, podendo resultar até na descontinuidade do projeto. Este impacto negativo é maximizado quando ocorre uma concentração de conhecimento em parte da equipe. Logo, faz-se necessário que todo o conhecimento relacionado ao projeto seja distribuído igualmente sobre todos os membros da equipe de modo a mitigar os impactos negativos de possíveis saídas de membros do time. Esse trabalho propõe uma solução web para alocação de tarefas considerando o nível de conhecimento do repositório dos membros do time. Esta solução é baseada em algoritmo genético e cálculo do truck factor (uma métrica que representa o nível de conhecimento do time no repositório), além de ser integrada a ferramentas utilizadas no desenvolvimento como github e trello.

Palavras-chave: alocação de tarefas; algoritmo genético; truck factor; Trello; Github.

ABSTRACT

Several steps and activities are involved in the software development process, one of these activities is task allocation. This activity is related to project management and is decisive for the success or failure of the project, since it involves risks related to time and consequently costs. However, it is important to emphasize the rotation of members within a development team, which can have negative impacts on deadlines and cost, and may even result in the discontinuation of the project. This negative impact is maximized when there is a concentration of knowledge in part of the team. Therefore, it is necessary that all knowledge related to the project is distributed equally over all team members in order to mitigate the negative impacts of possible departures of team members. This work proposes a web solution for task allocation considering the knowledge level of the team members' repository. This solution is based on a genetic algorithm and truck factor calculation (a metric that represents the team's knowledge level in the repository), in addition to being integrated with tools used in development such as github and trello.

Keywords: task allocation; genetic algorithm; truck factor; Trello; Github.

LISTA DE QUADROS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

1 INTRODUÇÃO

O Standish Group (2020) aponta que 19% de projetos de software falharam, 50% dos projetos tiveram algum problema e somente 31% destes projetos tiveram sucesso. Este relatório também aponta um conjunto de fatores críticos de sucesso para projetos de TI como: i) equipe capacitada, consistindo na habilidade de adquirir, gerenciar e controlar os recursos certos no momento certo, lidando com a rotatividade, bem como desenvolver e manter competências e ii) otimização, visando obter o máximo de valor para o negócio com o mínimo de riscos.

A rotatividade de membros do time é apontada como um dos principais fatores que pode impactar no insucesso do projeto por conta do impacto que pode causar na produtividade do time, atrasos nas entregas, risco de descontinuidade do projeto e impacto no custo do projeto. Uma das causas deste risco se deve à concentração do conhecimento do projeto em poucos membros. Assim, uma vez que a saída destes membros ocorre, o impacto no projeto pode ser percebido.

A alta demanda por software nos últimos anos, sobretudo após o início da pandemia de COVID-19, somado à oferta de trabalho remoto tem acelerado a rotatividade de software na área de TI. Uma busca no Linkedin [1] por vagas na área de TI apontou uma disponibilidade de 3648 vagas no Ceará, 51.621 vagas no Brasil e 3.226.277 vagas no mundo. Logo, são necessárias estratégias adequadas para auxiliar as empresas de TI a lidar com a rotatividade e reduzir seu impacto negativo nos projetos.

O desenvolvimento de um projeto de software envolve várias etapas, as quais são compostas por atividades. Uma destas atividades é a alocação de tarefas. Nesta atividade, o líder técnico ou gerente de projeto irá determinar quais tarefas precisam ser executadas e quem ficará responsável por tal atividade. Uma má distribuição destas tarefas pode causar a concentração do nível do conhecimento do repositório em poucos membros do time e consequentemente aumentar o impacto da rotatividade de pessoal e elevar os riscos do projeto.

O Truck factor é uma métrica que pode ser utilizada neste contexto, pois representa o nível de distribuição do conhecimento do repositório projeto. Para que o projeto continue funcionando é importante que todo o conhecimento em relação ao projeto seja distribuído de forma igualitária e que caso ocorra saídas de membros do time o projeto continue funcionando sem nenhuma interrupção (Avelino *et al.*, 2016).

Diversas soluções vêm sendo propostas para tratar a alocação de tarefas de software,

[1] Linkedin. Disponível em: https://www.linkedin.com/feed/. Acesso em: 12 jun. 2022

parte delas baseadas em otimização. Algoritmos de otimização utilizam de ferramentas matemáticas que fazem uma execução de forma iterativa, comparando várias possíveis soluções, até encontrar uma solução considerada ótima para resolução do problema.Os algoritmos genéticos, que são métodos de otimização, baseiam-se na teoria evolucionista de Charles Darwin. Esta teoria propõe que as espécies sofrem mutações e os indivíduos mais adaptados (considerados os melhores) têm maiores chances de sobreviver. Portanto, o processo de adaptação é um aspecto crucial nos algoritmos genéticos, refletindo a dinâmica da seleção natural proposta em 'A Origem das Espécies' (Darwin, 1859).

Neste contexto, o trabalho de SEGUNDO (2022) apresenta uma proposta de alocação de tarefas em um time de desenvolvimento considerando o truck factor e algoritmos genéticos. A solução é proposta como uma simulação multiagente que representa o time, o conhecimento do projeto e nível de maturidade como elementos da simulação. Esta solução apresentou bons resultados na simulação. No entanto, é necessária uma integração desta solução com ferramentas utilizadas em projetos de desenvolvimento reais como GitHub [2] e Trello[3], de modo a termos uma solução que pode ser utilizada por gerentes de projeto e líderes técnicos em projetos reais.

Abordagens, processos, modelos, métodos e frameworks para a alocação de tarefas em projetos de desenvolvimento de software distribuído são explorados na literatura, destacando a importância da tomada de decisão multicritérios qualitativa e análise de decisão verbal. Este tema é vital para profissionais e pesquisadores na área de sistemas de informação, sublinhando a contribuição potencial da sua solução para avanços neste campo (Oliveira *et al.*, 2023).

Neste trabalho, desenvolvemos uma solução web que integra o algoritmo de "truck factor"e algoritmos genéticos apresentados no trabalho de SEGUNDO (2022) com as ferramentas do GitHub e Trello. Essa integração propicia uma distribuição de conhecimento mais eficaz entre os membros do time de desenvolvimento. Com o objetivo de mitigar os riscos associados à alocação de tarefas e melhorar a eficiência operacional, a solução é projetada para ser aplicada em projetos de software reais. A sua implementação promete não só otimizar o processo de alocação de tarefas, mas também facilitar a gestão de projetos e a colaboração entre equipes.

[2] GitHub. Disponível em: https://github.com. Acesso em: 16 dezembro 2023
[3] Trello. Disponível em: https://trello.com. Acesso em: 16 dezembro 2023

1.1 Objetivo

Desenvolver uma aplicação web integrada com as ferramentas de desenvolvimento GitHub e Trello, capaz de recomendar a alocação de tarefas em equipes de desenvolvimento de software. Esta recomendação será realizada de forma otimizada considerando a distribuição do conhecimento do projeto e facilitando a colaboração e o gerenciamento de projetos através destas plataformas.

1.1.1 Objetivos específicos

- Implementar a utilização de Banco de dados, GitHub e Trello na aplicação.
- Desenvolver uma aplicação web integrada às plataformas GitHUB e Trello, permitindo uma gestão de tarefas.
- Disponibilizar a informação do truck factor dos projetos tratados pela ferramenta.
- Realizar a sugestão da alocação de tarefas de forma otimizada e automática, com base em algoritmo genético e levando em consideração o 'truck factor'.
- Ilustrar o uso da solução proposta para alocar tarefas em um projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os principais conceitos utilizados durante o desenvolvimento deste trabalho.

2.1 Metodologias Ágeis de Desenvolvimento

Durante o processo de desenvolvimento os times de software necessitam trocar informações, atribuir tarefas, e assumir diferentes papéis, com isso a utilização de uma metodologia de desenvolvimento a ser seguida é de suma importância para atingir os objetivos determinados durante o processo. Na engenharia de software existem diferentes técnicas para o desenvolvimento de um sistema, elas são usadas de acordo com o contexto de cada equipe.

De acordo com Fadel e Silveira (2010) ele faz referência a um evento que ocorreu em 2001 e contou com a presença de 17 líderes que trabalhavam no contra-fluxo dos padrões da indústria de software, durante o encontro eles debateram sobre formas de trabalho, a fim de chegar a uma nova metodologia de desenvolvimento, com isso eles obtiveram como resultados 12 princípios, gerando então a publicação do Manifesto Ágil (Beck *et al.*, 2001), que é composto por quatro premissas:

- Os Indivíduos e as interações possuem mais importância do que as ferramentas e os processos utilizados.
- O software ter funcionamento possuem mais importância do que uma documentação completa.
- Ter colaboração com o cliente, tem uma maior importância de negociações de contratos.
- Ser adaptável as mudanças possuem uma maior importância do que ter um plano inicial.

Segundo Sommerville (2011) as metodologias ágeis tem como objetivo reduzir a burocracia do processo, evitando o trabalho de valores incertos de longo prazo, e que apesar de serem baseadas em entregas e desenvolvimento, é proposto diferentes processos para alcançar tal objetivo. Entre os métodos ágeis está presente o Scrum que apesar de ser uma abordagem simples é capaz de auxiliar no processo de gerenciamento de tarefas complexas. O método tem como base os princípios utilizando equipes pequenas, requisitos desconhecidos e poucas interações, O desenvolvimento é dividido em intervalos, que recebem o nome de Sprints (Koschevic, 2012).

No Scrum existem alguns papéis a serem assumidos, as tarefas a serem executadas e artefatos desenvolvidos.

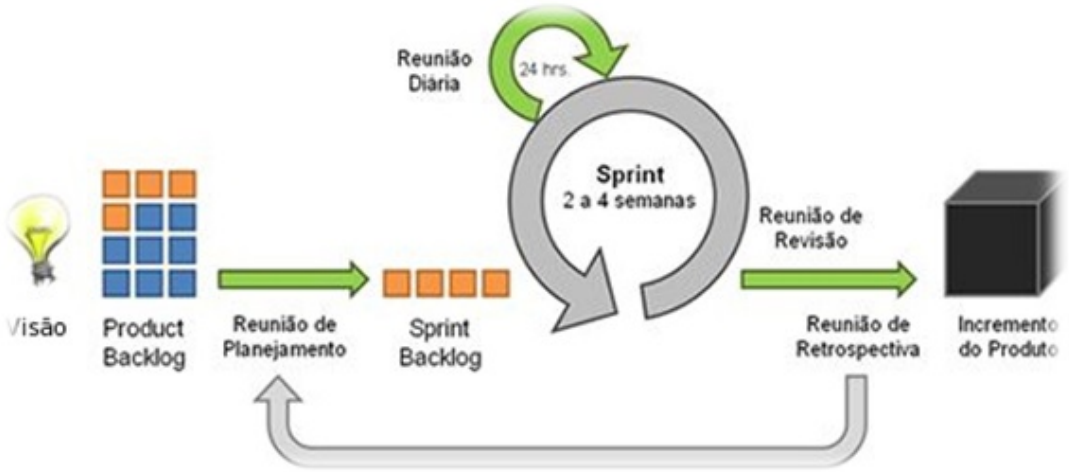
Entre os papéis fundamentais pode ser destacado o Scrum Master que é o responsável pela gerência do time, ele é quem o maior conhecimento sobre o framework, será ele quem determinará quais práticas e princípios devem ser seguidos. Além de ser um facilitador para o time, buscando potencializar as habilidades dos integrantes como citado por Coutinho (2020), outro papel fundamental é o de Product Owner é o elo de ligação entre cliente e equipe de desenvolvimento, ele deve colaborar em ambos os times para que haja compreensão de ambas as partes, tanto o time necessita ter uma visão ampla sobre quais tarefas executar em cada sprint, quanto os stakeholders precisam ter um meio de feedback e entregas e todos necessitam seguir a mesma visão definida para o produto . Developer Team, estão incluídos profissionais multidisciplinares, visto que são, programadores, arquitetos de software, testadores.

O método Scrum tem como principal tarefa a execução da 'sprint', que é um ciclo de atividades com duração máxima de um mês. O objetivo é entregar parte das tarefas determinadas. Assim que uma sprint termina, outra se inicia. Durante a sprint, outras atividades são realizadas, incluindo: planejamento da sprint, revisão da sprint e retrospectiva da sprint.

Existem três artefatos principais no Scrum: Product Backlog: uma lista de todas as alterações a serem feitas no produto. Sprint Backlog: um subconjunto do Product Backlog, consistindo em uma lista com informações técnicas, incluindo os itens do backlog que a equipe Scrum entregará durante a sprint atual, e um plano para a entrega desses itens. Product Increment: a saída da sprint. Este é uma versão utilizável e potencialmente liberável do produto, representando a junção dos itens do Product Backlog fornecidos em cada sprint.

A utilização do Scrum, com seus ciclos de curta duração, é de grande importância para obter as métricas de desempenho de cada membro da equipe do projeto e adaptar o trabalho de acordo com as necessidades.

Figura 1 – Visão geral do Scrum



Fonte: **??**)

2.2 Gerenciamento de projetos

Existem diversas definições sobre gerenciamento de projetos. O Project Management Institute (2017) define o gerenciamento de projetos como a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto para atender aos seus requisitos. O gerenciamento de projetos é realizado através da aplicação e integração de processos de iniciação, planejamento, execução, monitoramento e controle, e encerramento. O Foco é alcançar os objetivos do projeto enquanto se honra as premissas de escopo, tempo, custo, qualidade, recursos e risco.

O processo de Gerenciamento de projetos, representado por Patel (2010), aborda diversas fases. Essas fases incluem a iniciação, planejamento, execução, monitoramento e controle, e encerramento do projeto. Em cada uma delas o de projetos e a equipe de projeto aplicam conhecimentos, habilidades e técnicas apropriadas para alcançar os objetivos do projeto e entregar os resultados esperados. Este processo assegura que o projeto seja concluído de forma eficiente e eficaz, atendendo aos requisitos de qualidade, prazo e custo. Assim abordando o objetivo do gerenciamento de projetos que é garantir que a finalização do projeto seja concluído dentro do escopo, tempo, custo e qualidade acordados, ao mesmo tempo em que se otimizam a alocação de recursos e a integração dos inputs necessários para atender aos objetivos e expectativas das partes interessadas.

A gestão de projetos apresentada pelo Project Management Institute (2017), apresenta três elementos fundamentais, que são respectivamente: papéis do projeto, ciclo de vida do projeto e áreas de gerenciamento. Um projeto possui diversas partes interessadas que irão interferir

na execução do projeto, porém para que ele aconteça, é necessário que exista a participação da equipe e de três papéis principais: Sponsor é quem patrocina o projeto e escolhe o gerente, prestando o suporte e ele nas ações, representa a figura de maior autoridade no projeto. O gerente de projetos é o responsável pelas entregas e resultados do projeto, ele coordena o planejamento, a execução e a entrega, sendo responsável por liderar a equipe e fazer a mediação entre sponsor. O cliente não está envolvido necessariamente, o foco dele é apenas a entrega final, pois o produto é destinado a ele. Em alguns casos esses três papéis podem ser realizados por uma única pessoa, porém com funções diferentes que estarão atribuídas a um mesmo indivíduo naquele contexto.

Em Project Management Institute (2017), também é abordado que para uma boa gestão de projetos é necessário seguir cinco etapas fundamentais. Iniciação, primeira etapa representando o início do projeto, onde será alinhado as possibilidades e definir quesitos básicos, buscando responder às seguintes perguntas: O quê? Como? Por quê? Quando?. Essa fase fornece o direcionamento do projeto. Planejamento, onde será detalhado tudo o que precisa ser feito, definindo as atividades, prazos e responsáveis. Nessa etapa, será elaborado um plano distinto para cada área de conhecimento em gerenciamento., detalhando: escopo, cronograma, custos, equipe, qualidade, recursos, comunicação, riscos, aquisições e partes interessadas.Na fase de Execução, o projeto ganha vida, com a implementação das atividades planejadas para atingir os objetivos estabelecidos. Durante o monitoramento, verifica-se o progresso para assegurar que o projeto se mantenha alinhado ao planejado, ajustando-se conforme necessário. O Encerramento envolve a revisão final e a análise de desempenho, aprendendo com as experiências para melhorar processos futuros.

O gerenciamento de projetos, conforme delineado pelo Project Management Institute (2017), é uma prática multifacetada que envolve a orquestração cuidadosa de várias etapas, desde a iniciação até o encerramento do projeto. Essencialmente, ele foca na realização de objetivos específicos dentro de parâmetros de escopo, tempo, custo e qualidade, exigindo a integração eficiente de recursos e a coordenação entre diferentes partes interessadas. Com papéis claramente definidos e uma estrutura de ciclo de vida bem estabelecida, o gerenciamento de projetos se torna uma ferramenta vital para alcançar sucesso, eficiência e resultados de alta qualidade em empreendimentos temporários e únicos.

2.3 Alocação de tarefas

De acordo com Patterson (2022) a alocação de tarefas é um processo fundamental no gerenciamento de projetos que envolve a distribuição de atividades específicas a membros da equipe ou recursos disponíveis, a fim de atingir os objetivos do projeto de forma eficiente e eficaz. Isso inclui identificar as tarefas necessárias, avaliar a habilidade e a disponibilidade dos membros da equipe, e atribuir as tarefas de maneira que maximize a produtividade e minimize os riscos. A alocação eficaz de tarefas garante que cada membro da equipe tenha uma carga de trabalho adequada, contribuindo para o cumprimento dos prazos e para a qualidade do trabalho realizado.

Para fazer uma alocação inteligente de tarefas no gerenciamento de projetos, é essencial seguir um processo estruturado. Primeiro, alinhe-se com o escopo do seu projeto, compreendendo sua complexidade e necessidades de recursos. Em seguida, divida o projeto em tarefas e subtarefas, atribuindo-as com base nas habilidades e disponibilidade dos membros da equipe. Use ferramentas profissionais, como softwares de gestão de projetos, para facilitar esse processo. Esses softwares podem oferecer visualizações como gráficos de Gantt para um melhor acompanhamento do progresso. Lembre-se de gerenciar as cargas de trabalho para evitar sobrecarga ou subutilização dos recursos, e esteja pronto para realocar recursos conforme necessário para manter o projeto dentro do orçamento e cumprir os prazos estabelecidos (GanttPRO, 2023)

Em relação ao conhecimento das habilidades dos membros da equipe durante o processo de alocação, Project Management Institute (2017), aborda essa relação principalmente nas áreas de conhecimento de Gerenciamento de Recursos Humanos e Planejamento de Recursos Humanos. A abordagem ocorre em três etapas, durante o processo de gerenciamento de recursos humanos, onde ressalta que a identificação das habilidades necessárias e a correspondência com os membros da equipe apropriados são elementos-chave nesse processo.

Durante o planejamento de recursos humanos, onde inclui a definição das necessidades de recursos humanos para o projeto, que envolve a identificação das habilidades necessárias para concluir o trabalho do projeto com sucesso e no processo de Desenvolvimento da Equipe que envolve a melhoria das habilidades e competências dos membros da equipe ao longo do projeto. Isso pode incluir o treinamento para desenvolver habilidades específicas necessárias para as tarefas atribuídas. Além disso, o Project Management Institute (2017) enfatiza a importância de considerar o conhecimento das habilidades dos membros da equipe ao alocar tarefas em

projetos. Isso é fundamental para garantir que as tarefas sejam executadas de maneira eficaz e que os objetivos do projeto sejam alcançados com sucesso. Portanto, o Project Management Institute (2017) apoia a prática de alocação de tarefas com base nas competências individuais da equipe

2.4 Rotatividade em equipes de software

Chiavenato (1997), define turnover ou rotatividade de recursos humanos como o movimento de pessoas entre uma organização e seu ambiente de trabalho, ou seja, o vínculo de pessoas e organizações, contendo como ambiente o número de pessoas que entram e saem da organização. De acordo com o autor, se essa movimentação ocorrer em pequeno número, será saudável para a organização, pois indica uma melhoria de renovação em relação ao conhecimento dentro do ambiente de trabalho.

No entanto, a rotatividade é retratada como um problema para muitas empresas, especialmente, quando envolve a reposição de pessoal, pois por trás dessas reposições existe uma grande quantidade de custo em relação a treinamentos para os funcionários, que acaba se tornando perdido para a empresa quando os profissionais saem dessa organização, isto é, o empregador investe em treinamento para os colaboradores, e quando ocorre a rotatividade o profissional acaba levando junto consigo todo o investimento em treinamento (Pontes *et al.*, 2017).

Além disso como citado no artigo de Chaves *et al.* (2022) existem alguns fatores que influenciam nos impactos causados pelo turnover, entre os citados está presente a autonomia, sendo representada como um fator essencial e também motivador para a engenharia de software perante esse contexto, uma insatisfação entre autonomia necessita de procedência de engenheiros de software e o grau de autonomia dos engenheiros de software tem impacto na motivação e no desempenho da aplicação.

A rotatividade no setor de TI é destacada como a mais alta entre todas as indústrias, chegando a 13,2% em 2017. As principais razões para a troca de emprego por profissionais de tecnologia incluem a busca por melhor remuneração, falta de oportunidades de avanço, insatisfação com a liderança ou cultura da empresa, desejo de enfrentar maiores desafios e a necessidade de tarefas mais criativas. Além disso, o artigo sugere métodos para mitigar a rotatividade de pessoal técnico, como a criação de um equilíbrio positivo entre trabalho e vida pessoal, flexibilidade, oportunidades de desenvolvimento, tarefas significativas e reconhecimento,

e caminhos de carreira claros com atualizações salariais regulares (Idego Group, 2021).

O turnover é um problema recorrente em projetos de desenvolvimento de software. Para mitigar seus impactos negativos na produtividade e eficiência, é importante adotar estratégias para gerenciar esse fenômeno. Um software de alocação de tarefas baseado em algoritmos genéticos pode ser uma solução interessante para esse fim.

Os algoritmos genéticos são uma técnica de otimização que podem ser utilizados para encontrar a melhor combinação de tarefas para cada membro da equipe, levando em consideração o nível de conhecimento de cada um. Essa abordagem pode ajudar a garantir que as tarefas sejam atribuídas de forma mais eficiente, levando em consideração as habilidades e conhecimentos de cada membro da equipe. Além disso, um software de alocação de tarefas baseado em algoritmos genéticos pode ajudar a identificar possíveis gargalos na distribuição de tarefas, permitindo que os gerentes de projeto possam ajustar a alocação de tarefas para garantir que a equipe esteja trabalhando de forma eficiente e produtiva.

No entanto, é importante lembrar que um software de alocação de tarefas baseado em algoritmos genéticos não é uma solução completa para gerenciar o turnover. É necessário adotar uma abordagem mais ampla, que inclua medidas para reter talentos, como programas de mentoria e integração eficaz de novos membros da equipe, além de investir em treinamentos para os profissionais, de modo a aumentar sua capacitação e reduzir a necessidade de substituição (Carvalho, 2012).

Portanto, um software de alocação de tarefas baseado em algoritmos genéticos pode ser uma ferramenta útil para ajudar a gerenciar o turnover em projetos de desenvolvimento de software, mas é importante adotar uma abordagem mais ampla para minimizar seus impactos negativos na produtividade e eficiência dos projetos.

2.5 Truck Factor

Truck Factor é uma analogia que aborda o contexto sobre a quantidade de pessoas que devem ser atropeladas por um caminhão para que um projeto de software seja descontinuado, ou seja o quanto o time de desenvolvimento está preparado para a rotatividade de membros dentro da equipe que está desenvolvendo o projeto. O TF tem como foco fazer a detecção de como o conhecimento do código-fonte está compartilhado entre os participantes do time de software (Ferreira *et al.*, 2016).

Diversos algoritmos foram propostos para calcular o Truck Factor, incluindo aqueles

apresentados em estudos por Avelino *et al.* (2016), Cosentino *et al.* (2015), e Rigby *et al.* (2016). Embora todos esses algoritmos visem ao mesmo objetivo, eles apresentam diferenças em suas abordagens e metodologias. Uma análise comparativa desses algoritmos, destacando suas particularidades, foi realizada no estudo de (Ferreira *et al.*, 2017).

O autor aponta que o algoritmo proposto por Avelino *et al.* (2016) tem como base principal a medida DOA(Degree-of-Authorship), que trata-se do grau de autoria dos desenvolvedores dentro do projeto, assim definindo quais desenvolvedores são considerados autores de cada arquivo. Para determinar os valores de DOA são realizado cálculos de acordo com o histórico de commits onde: f é a representação da criação de um arquivo por um desenvolvedor d, assim inicializando o valor de DOA(d, f); commits adicionais em f por d aumentam o valor de DOA(d, f).

Os valores dos pesos usados são definidos de acordo com experimentos empíricos realizados Fritz *et al.* (2014), com isso os valores DOA são normalizados por arquivo, o desenvolvedor com maior DOA em um arquivo f tem seu DOA normalizado igual a 1, se o desenvolvedor obter DOA normalizado maior que 0,75, ele é considerado autor de um arquivo. Esse limite foi definido após o experimento empírico validado manualmente Avelino *et al.* (2016), utilizando uma amostra de 120 arquivos de código-fonte de seis sistemas de código aberto.

O algoritmo denominado AVL recebe como entrada uma lista dos principais autores, essa lista é ordenada pelo número de arquivos em que os desenvolvedores possuem autoria, essa lista é calculada usando a medida DOA. O algoritmo AVL simula continuamente a remoção do autor principal do sistema. Após cada remoção do autor principal, a estimativa de TF atual é incrementada em 1 e o autor removido é adicionado ao conjunto de desenvolvedores de TF. O algoritmo termina quando mais de 50% dos arquivos são abandonados. Um arquivo é considerado abandonado quando todos os seus autores foram removidos do sistema.

O algoritmo proposto por Rigby *et al.* (2016) possue uma avaliação que quantifica a suscetibilidade de projetos de software à rotatividade de desenvolvedores. O algoritmo utiliza de uma abordagem baseada em confirmação, semelhante ao algoritmo de Avelino *et al.* (2016). Conforme implementado em sistemas de controle de versão do git, o recurso de culpa indica o desenvolvedor que alterou pela última vez cada linha de um arquivo.

O algoritmo proposto define que uma linha de código é abandonada quando o comando git-blame, que é utilizado para examinar o conteúdo de um arquivo, linha por linha, e ver quando cada linha foi modificada pela última vez e quem foi o autor das modificações,

determina como autor um desenvolvedor que saiu do projeto. Além disso o algoritmo também define como um arquivo abandonado quando ao menos 90% das suas linhas são abandonadas. Os autores afirmam que usam esse limite alto para excluir desenvolvedores com contribuições triviais para um arquivo.

Assim o algoritmo proposto por Rigby *et al.* (2016) funciona simulando vários cenários do Truck Factor, primeiro ocorre uma variação do tamanho do grupo g de desenvolvedores que saem, variam de 1 a 200 desenvolvedores. Então, o algoritmo aleatoriamente seleciona grupos de desenvolvedores para sair, cada um com tamanho g, essa simulação ocorre durante aproximadamente 1000 vezes, para cada tamanho de grupo, o algoritmo também calcula a probabilidade de cada cenário de desastre, ou seja, a saída do grupo selecionado de desenvolvedores, usando uma técnica estatística originalmente proposta para gerenciar possiveis riscos financeiros, para realização da comparação entre o algoritmo de Avelino *et al.* (2016) e o algoritmo Rigby *et al.* (2016), os autores Ferreira *et al.* (2017) consideraram o valor de TF retornado pelo algoritmo de Rigby como sendo o menor g que implica em mais de 50% dos arquivos sendo abandonados no sistema, usando a definição baseada em culpa de arquivos abandonados. De fato, os autores do algoritmo discutem o uso desta solução para comparar seus resultados como outros algoritmos de Truck factor, como originalmente definido por (Zazworka *et al.*, 2010).

Duas características importantes do algoritmo é que ele não é deterministico, ou seja, devido ao uso de uma amostra aleatória de desenvolvedores, os resultados produzidos pelo algoritmo variam de uma execução para outra. A segunda caracteristica é a de que o algoritmo pode determinar sem computar um resultado de Truck Fator válido, isso acontece quando todos os grupos de desenvolvedores não atendem às condições de um cenário Truck Factor.

Já o algoritmo proposto em Cosentino *et al.* (2015), calcula o fator de caminhão utilizando dois conjuntos de desenvolvedores. Desenvolvedores considerados primários P, que são os que possuem um conhecimento minimo Kp sobre determinado artefato de software. Desenvolvedores secundários S são aqueles que possuem pelo menos um conhecimento Ks, onde Ks < Kp. os autores sugerem que Kp deve ser definido como $\frac{1}{d}$, onde d é o número de desenvolvedores que já alteraram o artefato e que Ks deve ser definido como $\frac{Kp}{2}$.

O fator de caminhão de um artefato é definido como $|P \cup S|$, ou seja, o número de desenvolvedores classificados como desenvolvedores primários ou secundários. Para calcular o conhecimento de um desenvolvedor em um arquivo, os autores propõem um conjunto de métricas, incluindo a última alteração leva tudo, todo o conhecimento de um arquivo é atribuído ao último

desenvolvedor que modificou e várias alterações igualmente consideradas, o conhecimento de um desenvolvedor d em um arquivo é definido como $\frac{Cd}{C}$, onde Cd é o número de commits no artefato executado pelo desenvolvedor d e C é o número total de commits no artefato.

Além disso é proposto uma estratégia para agregar dados de conhecimento ao nível de diretório, filial ou projeto. Essa agregação é calculada somando o conhecimento sobre arquivos individuais e dimensionando os resultados considerando o número total de arquivos em um diretório, filial ou projeto.

2.6 Algoritmos Genéticos

Um algoritmo genético é a representação de um modelo computacional da evolução biologica, ele utiliza metódos de busca para resolução de problemas e para modelar sistemas evolutivos. O algoritmo genético possue como base a técnica de busca e otimização, inspirado no principio de evolução das espécies desenvolvido por Darwin (1859), com fundamentação na seleção natural onde, individuos mais aptos a permanecerem em determinado ambiente sejam selecionados para reprodução, assim perpetuando a herança genética.

Na construção de algoritmos computacionais o principio não é diferente, uma vez que buscam uma melhor solução para determinado problema, através da evolução de populações de soluções codificadas em dados, que representam uma das possiveis soluções do espaço de busca do problema.

Algoritmos Genéticos(AGs), são mecanismos de otimização e busca orientado nos mecanismos de evolução de populações de seres vivos. Esse algoritmo teve como introdutor John Holland e disseminados por seu aluno, David Goldberg (Lacerda; Carvalho, 1999). Os AGs buscam encontrar uma solução ótima para determinado problema, o ambiente tem diferentes indivíduos, e cada um possui uma diferente solução para determinados problemas, e cada indivíduo pode ser avaliado de formas distintas.

Os AGs são aplicados em problemas complexos de otimização: problemas esses que abordam diversos parâmetros ou características que necessitam de uma melhor solução e que não podem ser representados matematicamente. (Pacheco *et al.*, 1999).

A Figura 2 representa os passos de um algoritmo genético, onde a população inicial é gerada automaticamente de forma aleatória onde cada indivíduo representa uma solução possível, em seguida é feito o cálculo da função fitness de cada indivíduo da população, ela mede o quão adaptado um indivíduo está de acordo com o problema. A seleção escolhe os indivíduos mais

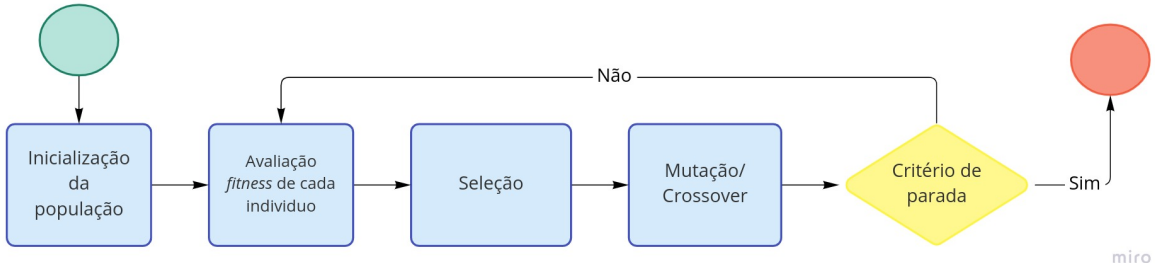
aptos para continuarem para as próximas gerações e assim eles passam por operadores genéticos de mutação e crossover com o objetivo de expandir seus genes.

Enquanto o critério de parada não é atingido é realizada uma nova geração com os indivíduos gerados pelos operadores de mutação e crossover e, caso o critério de parada for atingido, o algoritmo se encerra. O operador de cruzamento faz uma combinação de partes de dois indivíduos selecionados para gerar novos indivíduos. Então, o operador de mutação modifica aleatoriamente um indivíduo descendente, a população descendente criada a partir dos operadores de busca substitui a população anterior.

Em resumo, os algoritmos genéticos emergem como uma abordagem poderosa e flexível na alocação de tarefas, um desafio complexo encontrado em muitos domínios da ciência e da indústria. A capacidade dos AGs de simular processos evolutivos, adaptando-se a ambientes e requisitos variáveis, os torna particularmente adequados para resolver problemas de alocação onde as variáveis e restrições são numerosas e dinâmicas. Esta eficácia é fundamentada em trabalhos como o de Goldberg (1989), que fornece insights valiosos sobre a aplicação dos AGs em problemas de otimização.

O potencial dos AGs na alocação de tarefas é vasto, abrangendo desde a distribuição de recursos em redes de computadores até a gestão de mão-de-obra em sistemas de manufatura. A natureza iterativa e a adaptabilidade dos AGs permitem que eles encontrem soluções que não apenas atendam aos critérios de eficiência e eficácia, mas também se adaptem a mudanças nas demandas e condições operacionais. Com o avanço contínuo na computação evolutiva, como explorado por Bäck *et al.* (1997), espera-se que os AGs desempenhem um papel ainda mais crucial na otimização de processos complexos de alocação de tarefas, impulsionando inovações e eficiências em diversos setores.

Figura 2 – Passos de um algoritmo genético



Fonte: elaborado pelo autora.

3 TRABALHOS RELACIONADOS

Nesta seção, discutiremos três estudos relevantes no campo do desenvolvimento de software: um se concentra especificamente no Truck Factor, aplicando simulação multiagente e algoritmos genéticos para otimizar a alocação de tarefas em equipes de desenvolvimento, enquanto o outro foca na utilização de aprendizado de máquina para recomendar revisores de código, melhorando a eficiência do processo

3.1 Uma simulação multiagente para alocação de tarefas em projetos de software baseada no truck factor

O trabalho de SEGUNDO (2022) propõe o uso de uma simulação multiagente, empregando técnicas de Search-Based Software Engineering (SBSE), para otimizar a alocação de tarefas em times de desenvolvimento de software, que é considerado um problema NP-difícil. A pesquisa foca especificamente na minimização dos impactos causados pelo Truck Factor, utilizando para isso algoritmos de otimização como o algoritmo genético, para melhor distribuir o conhecimento e as responsabilidades entre os membros da equipe.

Através da simulação multiagente é possível modelar diferentes cenários e avaliar estratégias de alocação de tarefas, levando em consideração o impacto do Truck Factor. A pesquisa busca utilizar essa abordagem para minimizar os impactos causados pela concentração de conhecimento e pela rotatividade de profissionais de TI na alocação de tarefas em equipes de desenvolvimento de software, contribuindo para a otimização do Truck Factor e para a melhoria do desempenho dos projetos de software.

Assim foi desenvolvido um algoritmo genético usando a linguagem python e a biblioteca DEAP, que é uma biblioteca que fornece suporte para a construção de algoritmos genéticos. O algoritmo foi utilizado com o intuito de otimizar a alocação de tarefas em equipes de desenvolvimento de software, considerando a métrica do truck factor, que foi utilizado com o objetivo de maximizar, indicando um equilíbrio na distribuição de conhecimento sobre o projeto. A métrica busca definir a quantidade de pessoas que detêm o conhecimento sobre o projeto, utilizando como entrada o repositório de software para encontrar o valor do TF.

A abordagem para calcular o TF, conforme definido por Avelino *et al.* (2016), foi utilizada na dissertação de SEGUNDO (2022). Esta abordagem é descrita como "gulosa"para o cálculo do TF, e foi adotada após um estudo sobre ferramentas de simulação para ambientes

multi-agentes. A abordagem de Avelino *et al.* (2016) foi considerada a mais adequada, que visava maximizar o TF como parte do processo de otimização da alocação de tarefas em equipes de desenvolvimento de software

O algoritmo genético foi implementado com uma função de aptidão que busca gerar uma população visando maximizar o Truck Factor. Esta abordagem permitiu a exploração de diferentes cenários e abordagens em um ambiente simulado, variando desde equipes pequenas até projetos maiores com mais pessoas e tarefas. A função de aptidão do algoritmo genético foi baseada na contribuição do agente ao repositório de software e nos resultados fornecidos pela métrica do Truck Factor. O objetivo era encontrar a solução ótima para a distribuição das tarefas, maximizando assim a distribuição do conhecimento do projeto entre os membros da equipe.

Além disso, foi utilizado a plataforma NetLogo como ambiente de simulação de um ambiente real de desenvolvimento, pois através dela era possível criar ambientes multiagentes de simulação. A plataforma também oferece uma vasta quantidade de recursos que foram importantes para fazer a validação da pesquisa, tornando mais ágil, uma vez que na simulação é possível utilizar somente variáveis necessárias para o ambiente de simulação. Através da plataforma também foi possível adicionar e executar a utilização dos agentes que seriam utilizados no ambiente, assim a plataforma oferece recursos e funcionalidades que são essenciais para a modelagem, interação e validação da proposta de trabalho relacionada aos sistemas multi-agentes.

Em resumo, SEGUNDO (2022) apresenta uma abordagem inovadora para lidar com o problema da alocação de tarefas em equipes de desenvolvimento de software, que considera o Truck Factor e utiliza técnicas de SBSE. Portanto, o uso do algoritmo genético e o foco no Truck Factor foram centrais para desenvolver uma abordagem eficaz para a alocação de tarefas, visando a otimização da distribuição de conhecimento e a eficiência dos processos em equipes de desenvolvimento de software. No entanto, há a necessidade desta soução ser integrada aos ambientes comumente utilizados no desenvolvimento de projetos de software.

3.2 Usando uma abordagem baseada no contexto para recomendar revisores de código: resultados de um estudo de caso industrial

O estudo de caso apresentado por Strand *et al.* (2020) apresenta uma plataforma para solucionar o problema de alocação de desenvolvedores no processo de revisão de código. Na maioria das empresas, inclusive na empresa utilizada no estudo, essa alocação ocorre manualmente, e isto pode levar a alguns problemas como atraso na alocação, limitação de candidatos e

risco de alocação excessiva de alguns revisores. Buscando evitar tais problemas foi desenvolvido uma ferramenta denominada Carrot, que é baseada no aprendizado de máquina para recomendar revisores de códigos utiliza uma abordagem que leva em consideração fatores como a experiência do revisor, complexidade do código e a carga de trabalho dos revisores para recomendar os revisores mais adequados para uma determinada tarefa de revisão de código.

A ferramenta Carrot baseia-se na filtragem colaborativa e leva em consideração o contexto das mudanças apresentadas no código e as características dos revisores potenciais, buscando equilibrar a carga de trabalho, uma vez que a ferramenta visa distribuir a carga de trabalho de maneira uniforme entre os revisores, evitando sobrecarregar os indivíduos. Além disto, a plataforma também fornece um feedback quando um desenvolvedor faz a solicitação de algum revisor, melhorando a eficiência e eficácia, isso ajuda a assegurar a distribuição mais justa e eficiente das tarefas de revisão.

A arquitetura da ferramenta Carrot é composta por diversos micro serviços, cada um desempenhando funções específicas na recomendação de revisores de código. O Gerrit é utilizado no micro serviço de extração para fornecer dados cruciais como solicitações de revisão de código, atividades dos revisores e histórico de revisões.

Estes dados desempenham um papel crucial no processo de ETL (Extração, Transformação e Carregamento). Inicialmente extraídos, são posteriormente convertidos para um formato apropriado e carregados para uso. Após essa etapa de processamento, os dados são armazenados em um banco de dados. Posteriormente, essas informações são aplicadas no treinamento de um modelo de machine learning.Em seguida o modelo é utilizado para fazer recomendações (ML MODEL), essas recomendações geradas serão armazenadas em um núcleo(CORE) e encaminhadas para a interface de usuário, onde os desenvolvedores podem selecionar revisores sugeridos para as suas solicitações, na própria interface o usuário tem a possibilidade de fornecer feedback sobre a adequação das recomendações de revisores, e assim pode ser utilizado para refinar e melhorar o modelo de machine learning.

O estudo de caso apresentado demonstra que a ferramenta Carrot, desenvolvida e testada na Ericsson, oferece uma solução inovadora para o problema da alocação manual de revisores de código. Utilizando algoritmos de aprendizado de máquina e filtragem colaborativa, a Carrot consegue equilibrar a carga de trabalho entre os revisores, ao mesmo tempo que considera a experiência, a complexidade do código e outras variáveis relevantes para fazer recomendações precisas. A arquitetura de microserviços, incluindo o uso do Gerrit para extração de dados e

o feedback contínuo dos usuários, assegura uma melhoria contínua da ferramenta, tornando o processo de revisão de código mais eficiente e justo.

3.3 Uma Abordagem Multicritério para Apoiar a Alocação de Tarefas em Projetos de Desenvolvimento de Software Distribuído

O artigo de Filho *et al.* (2019) detalha a metodologia DiVA, que se destina a aperfeiçoar a alocação de tarefas em ambientes de Desenvolvimento Distribuído de Software (DSD). Ele explora o uso da Análise de Decisão Verbal (VDA) para enfrentar a complexidade inerente à distribuição de tarefas em equipes geograficamente dispersas. A Análise de Decisão Verbal (VDA) é uma técnica usada para facilitar a tomada de decisões complexas onde fatores qualitativos são significativos. Ela permite que os decisores traduzam suas avaliações subjetivas em um formato estruturado e verbalizado que pode ser analisado sistematicamente. No contexto da metodologia DiVA, a VDA é aplicada para avaliar e classificar os fatores que influenciam a alocação de tarefas e as unidades executoras dentro de projetos de software distribuídos. Isso ajuda a criar um processo de decisão mais objetivo e transparente.

A abordagem DiVA emprega uma metodologia específica para categorizar tarefas e equipes, considerando diversos níveis de adequação a critérios estabelecidos. Inicialmente, as tarefas e as equipes são classificadas de acordo com esses critérios. Em seguida, um processo de ordenação é aplicado para alinhar as categorias classificadas com as prioridades do projeto de Desenvolvimento Distribuído de Software (DSD). As prioridades dentro do contexto da metodologia DiVA podem variar, mas geralmente incluem fatores como habilidades técnicas das equipes, carga de trabalho, comunicação entre as equipes, prazos do projeto, e outros critérios relevantes para a eficiência do desenvolvimento do software e a distribuição eficaz de tarefas.

Esta abordagem é notavelmente flexível, adaptando-se a diferentes conjuntos de prioridades em cada projeto. Esse processo sequencial assegura uma alocação de tarefas mais objetiva e considera as características únicas e as prioridades de cada projeto de DSD, facilitando a tomada de decisões de forma justificável e transparente.

A implementação da metodologia DiVA nas cinco empresas revelou sua capacidade de adaptar-se a diferentes cenários e requisitos organizacionais, demonstrando sua aplicabilidade prática além da teoria. Em cada empresa, a DiVA proporcionou uma estrutura clara e metódica para a alocação de tarefas, o que se traduziu em uma gestão de projetos mais eficaz e decisões mais estratégicas. Isso ressalta a utilidade da DiVA como uma ferramenta valiosa no campo do

Desenvolvimento Distribuído de Software, ajudando a superar desafios comuns e melhorando a colaboração entre equipes distribuídas.

Os resultados indicam um avanço notável na distribuição de recursos, contribuindo para aprimorar tanto a colaboração interequipes quanto a eficácia global dos projetos. A metodologia foi destacada por sua versatilidade e simplicidade na implementação. Ademais, além de aperfeiçoar a gestão de recursos e a sinergia entre as equipes, observou-se também um incremento na comunicação e no entendimento das habilidades das equipes. Isso levou a uma designação de tarefas mais acertada e produtiva. A DiVA ganhou reconhecimento por sua capacidade de identificar as combinações mais eficientes de tarefas e equipes, resultando em melhorias na produtividade e qualidade dos projetos executados.

Finalmente, o artigo conclui que a aplicação da DiVA pode oferecer um valor significativo para a gestão de projetos de software, especialmente naqueles que enfrentam os desafios do DSD. A metodologia não apenas melhora a alocação de tarefas, mas também pode servir como um guia para outras organizações que buscam soluções semelhantes para desafios de gestão de tarefas. O artigo ressalta que a DiVA oferece uma abordagem inovadora para a gestão de projetos de software, especialmente valiosa em cenários de DSD, onde a colaboração eficaz entre equipes distribuídas é crucial. A metodologia não só aprimora a alocação de tarefas, mas também fornece insights valiosos para aprimorar processos e estratégias de gestão em diversos contextos organizacionais.

3.4 Comparação dos Trabalhos relacionados

No Quadro 1, foi realizada uma análise comparativa, levando em consideração vários critérios importantes. Esses critérios incluem desenvolvimento de modelos de simulação, abordagem para alocação de tarefas dentro da equipe de desenvolvimento, uso do algoritmo "Truck Factor" e integração com repositórios e ferramentas de gerenciamento de tarefas. Esta análise fornece uma visão detalhada desse estudo, permitindo identificar a contribuição específica de cada estudo.

Quadro 1 – Quadro comparativo de trabalhos

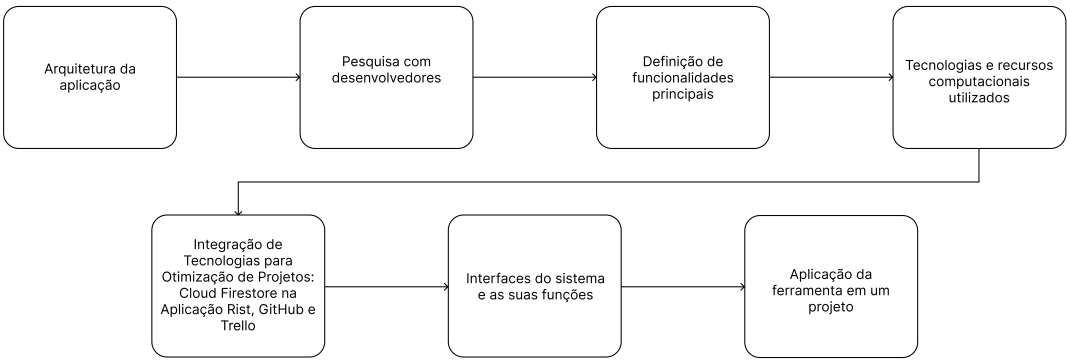
Critérios	**(SEGUNDO, 2022)**	**(Strand *et al.*, 2020)**	**Diva** (Filho *et al.*, 2019)	**Presente trabalho**
Desenvolvimento de um modelo de simulação	Sim	Não	Não	Não
Aborda alocação de tarefas em times	Sim	Não	Sim	Sim
Utiliza o algoritmo truck factor	Sim	Não	Não	Sim
Integração com ferramentas de repositório e de gerenciamento de tarefas	Não	Não	Não	Sim
Foco em Simulação Baseada em Agentes	Sim	Não	Não	Não
Uso de Search-Based Software Engineering (SBSE)	Sim	Não	Não	Sim
Solução Web	Não	Sim	Não	Sim
Uso de Algoritmo Genético	Sim	Não	Não	Sim
Integração com GitHub e Trello	Não	Não	Não	Sim
Recomendação de revisores de código	Não	Sim	Não	Não
Uso de Machine Learning	Não	Sim	Não	Não
Avaliação em ambiente simulado	Sim	Não	Não	Não
Avaliação em ambiente real	Não	Sim	Sim	Sim

Fonte: elaborado pela autora

4 METODOLOGIA

Nesta seção, são detalhadas as etapas envolvidas no desenvolvimento deste projeto.A Figura 3, que representa o fluxo das atividades realizadas, oferece uma visão de todos os passos envolvidos neste trabalho.

Figura 3 – Fluxo das atividades realizadas



Fonte: elaborado pelo autor.

A condução desse trabalho envolve desafios adicionais, incluindo a integração com ferramentas de gerenciamento de projetos, como Trello, e ferramentas de controle de versões, como GitHub. Assim, inicialmente foi definida a arquitetura da aplicação, com a finalidade de identificar como as plataformas(GitHub e Trello) se relacionaram com a aplicação e como os resultados obtidos se integrariam com o algoritmo genético.

Após determinada arquitetura foi importante aplicar um questionário com os desenvolvedores. Assim, foi conduzida uma pesquisa com desenvolvedores envolvidos em projetos de desenvolvimento de software. O objetivo principal deste questionário foi identificar lacunas na alocação de tarefas durante o planejamento da Sprint e na ferramentas utilizadas.

Após a pesquisa com os desenvolvedores foi analisado quais funcionalidades deveriam ser priorizadas como essenciais para a aplicação. Assim, foi possível estabelecer as tecnologias e recursos adotados para a construção de uma interface web destinada aos usuários, com integração tanto ao Trello quanto ao GitHub.

Na sequência, foi feita a integração entre as ferramentas/tecnologias envolvidas e as interfaces com o usuário foram implementadas. Por fim, a solução foi utilizada com o repositório do github e quadro trelo de um projeto para ilustrar seu uso.

5 RESULTADOS

Este capitulo apresenta os resultados envolvidos na criação da solução para alocação de tarefas. Inicialmente são apresentados os resultados de um questionário que tem como objetivo entender como a alocação de tarefas em equipes de software acontece e quais problemas os usuários enfrentavam em cada plataforma. Com base no resultado deste questionário, foi determinado quais funcionalidades deveriam existir na aplicação. Na sequência, é detalhado como o sistema de alocação de tarefas foi desenvolvido. Assim, foi detalhada a definição das tecnologias utilizadas, integração entre das plataformas, envolvidas e apresentação das funcionalidades disponíveis. Por fim é apresentada uma ilustração do uso da ferramenta em um repositório do Github e é apresentado como a ferramenta pode ser acessada por usuários interessados no seu uso.

5.1 Arquitetura da aplicação

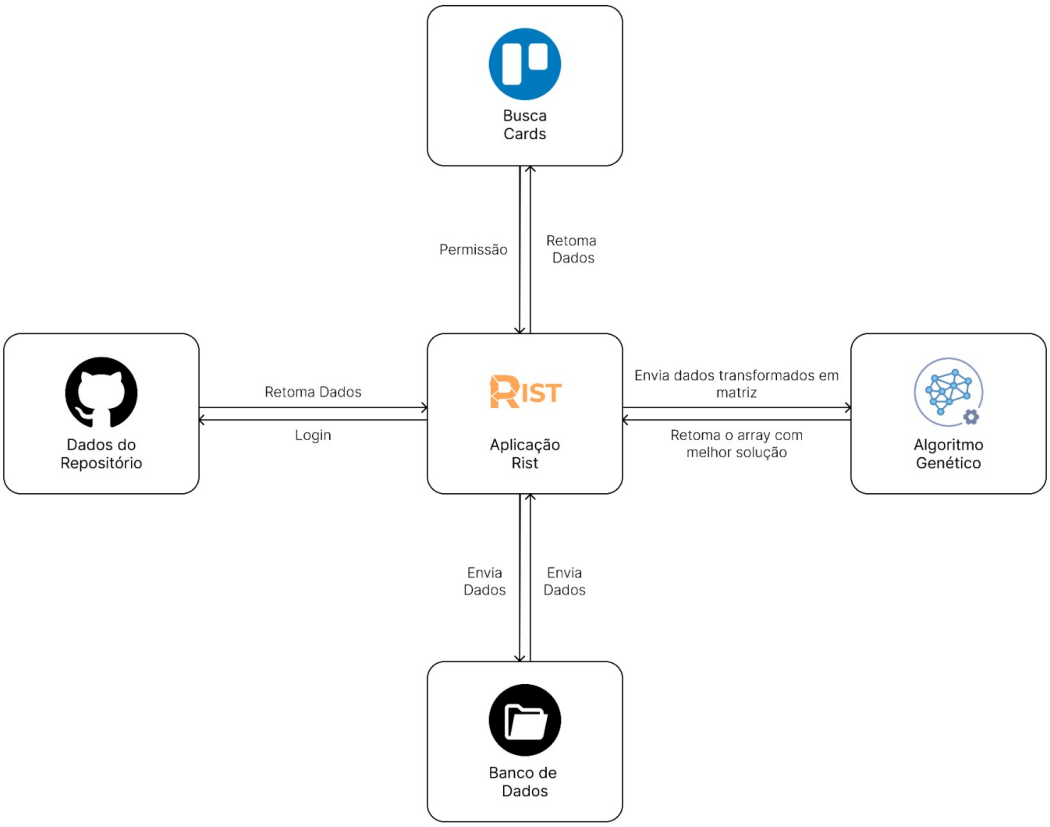
A Figura 4 descreve o fluxo da arquitetura da aplicação RIST, que inicia quando o usuário faz login através do GitHub, obtendo informações de um repositório, essa etapa é essencial pois é através dela que o usuário irá conseguir acessar a aplicação. O login no Trello permite a busca de cartões de tarefas, essa integração com o Trello é importante para obtenção de dados durante a alocação. O banco de dados atua como núcleo da aplicação, pois é nele que ocorre o armazenamento de dados do GitHub e Trello, e após o armazenamento, os dados são reorganizados em matrizes pela aplicação e enviados ao algoritmo genético, também desenvolvido por SEGUNDO (2022), que retorna o valor do Truck Factor do repositório e a melhor solução de alocação de tarefas.

5.1.1 Pesquisa com desenvolvedores

Nesta seção, são apresentados os resultados de uma pesquisa conduzida junto aos profissionais do desenvolvimento de software. O objetivo deste estudo é obter informações sobre a alocação de tarefas, bem como identificar as dificuldades enfrentadas durante esse processo de alocação de tarefas. O estudo foi baseado em questionário com questões abertas, aplicado online via google forms. As análises e conclusões desta seção proporcionaram uma visão aprofundada das percepções e experiências dos desenvolvedores em relação à gestão de tarefas.

O formulário de coleta é composto pelas 7 perguntas abertas apresentadas a seguir:

Figura 4 – Fluxo das atividades realizadas



Fonte: elaborado pelo autor.

Questão 1. Há quanto tempo você está no seu projeto atual?

Questão 2. Como são alocadas as tarefas em sua equipe?

Questão 3. Qual(is) plataforma(s) é (são) utilizadas para alocação de tarefas atualmente pela sua equipe?

Questão 4. Há dificuldade para utilizar essa(s) plataforma(s)? Se sim, quais?

Questão 5. Quais fatores são considerados na hora de alocar uma tarefa?

Questão 6. Você identifica desafios no processo de alocação de tarefas? Quais?

Questão 7. Você acha que a alocação de tarefas poderia ser melhorada? Se sim, como??

Essas perguntas foram aplicadas a um formulário em um grupo de 11 desenvolvedores que estão inseridos no mercado de trabalho. A análise das respostas oferece insights valiosos sobre a alocação de tarefas em equipes de desenvolvimento e acerca da diversidade de experiências, métodos de gestão, desafios e ferramentas utilizadas em equipes de desenvolvimento de software. Os resultados mostraram uma ampla variação no tempo de envolvimento dos membros

nos projetos, com experiências variando de "entre 1 mês e 6 meses"a "mais de 2 anos", refletindo uma diversidade significativa de experiências dentro das equipes.

Quanto ao método de alocação de tarefas, observou-se uma tendência à liderança centralizada, com a maioria das tarefas sendo alocadas por gerentes de projeto ou líderes técnicos. Isso sugere um modelo de gestão hierárquico. Em relação aos critérios para alocação, as habilidades dos membros da equipe, a prioridade e os prazos das tarefas emergiram como fatores decisivos, indicando um foco em competências e urgência.

Foram identificados desafios significativos no processo de alocação, especialmente no que diz respeito à comunicação e flexibilidade. Problemas como falta de clareza nas responsabilidades e falhas de comunicação apontam para áreas potenciais de melhoria. Opiniões divergentes foram expressas sobre possíveis melhorias, com algumas pessoas sugerindo a necessidade de melhor descrição das tarefas, maior clareza no escopo e um balanceamento mais equitativo da carga de trabalho. É importante ressaltar que a alocação de tarefas baseada no conceito de truck factor pode levar a um equilíbrio mais justo na carga de trabalho. Isso ocorre porque tal abordagem visa a uma distribuição homogênea do conhecimento sobre o repositório entre os membros da equipe.

A diversidade também foi observada nas ferramentas de gestão de tarefas utilizadas pelas equipes, com plataformas como Jira, Notion e Trello sendo as mais comuns. A maioria dos participantes reportou uma boa usabilidade dessas ferramentas, embora algumas dificuldades tenham sido apontadas, relacionadas principalmente à personalização e ao gerenciamento eficiente do tempo.

Em resumo, os resultados da pesquisa oferecem uma visão abrangente do estado atual da alocação de tarefas em equipes de desenvolvimento de software, destacando áreas de sucesso e oportunidades para otimização. A diversidade de experiências e a centralização na gestão de tarefas coexistem com desafios de comunicação e necessidades de melhorias nas ferramentas utilizadas, sugerindo caminhos para aprimorar a eficiência e a satisfação das equipes de desenvolvimento.

As equipes de desenvolvimento operam em um ambiente onde a experiência dos membros varia, e as tarefas são alocadas centralmente com foco em habilidades e prioridades. Enquanto muitos estão satisfeitos com o processo atual e as ferramentas usadas, há áreas claras para melhoria, especialmente em termos de comunicação, flexibilidade e especificidades das ferramentas de gerenciamento de tarefas. Estas conclusões podem ser usadas para informar

decisões de gestão de equipe, escolha de ferramentas e estratégias de melhoria de processos.

5.1.2 Definição de funcionalidades principais

Após a aplicação do questionário e a obtenção de informações, foi necessário analisar quais funcionalidades deveriam ser incorporadas à aplicação desenvolvida, priorizando-as com base em sua importância. A primeira versão do aplicativo inclui as seguintes funcionalidades:

1. Login com Github: - Permite que o gerente de projetos acesse repositórios nos quais está envolvido, facilitando o acesso a informações do projeto, como participantes, contribuições em cada arquivo e informações dos contribuidores.
2. Autenticação com Trello: - Essencial para o processo de alocação de tarefas, permite que o gerente de projetos busque e extraia informações de cards de tarefas na coluna "A Fazer"de seus quadros no Trello.
3. Adicionar Habilidades Necessárias para Tarefa: - Após recuperar as tarefas do Trello, permite determinar o conjunto de habilidades necessárias para cada tarefa, essencial para a aplicação do algoritmo genético.
4. Adicionar Nova Tarefa: - Oferece a opção de adicionar tarefas manualmente, caso o gerente de projetos opte por não extrair as tarefas do Trello.
5. Cálculo do Truck Factor: - Após o login no Github e seleção do repositório, requer que o usuário preencha dados de habilidades de tarefas e habilidades dos desenvolvedores para gerar a análise e retornar o valor do Truck Factor.
6. Alocação de Tarefas: - Funcionalidade principal da aplicação, onde os desenvolvedores são atribuídos às tarefas por meio do algoritmo genético, podendo realizar novas análises se necessário.

Após essa análise inicial, foi conduzida uma análise mais técnica para definir as tecnologias a serem utilizadas no desenvolvimento do aplicativo. Isso incluiu a utilização de algoritmos existentes, como o algoritmo de Truck FactorAvelino *et al.* (2016) e o algoritmo genético implementado no trabalho de SEGUNDO (2022). Além disso, a aplicação requer a integração com as APIs do Github e Trello para extrair dados dessas plataformas.

5.1.3 Tecnologias e recursos computacionais utilizados

Na fase de desenvolvimento desta aplicação, foi empregado um conjunto diversificado de tecnologias e ferramentas, cada uma desempenhando um papel crucial para atender aos

requisitos funcionais e estéticos do projeto. Aqui está uma expansão detalhada das tecnologias utilizadas:

- JavaScript: - Como espinha dorsal do desenvolvimento web moderno, o JavaScript se destacou por sua flexibilidade e versatilidade. Utilizado tanto no cliente quanto no servidor, esta linguagem de script desempenha um papel fundamental na criação de páginas web dinâmicas e interativas. Graças ao seu ecossistema extenso, incluindo uma vasta gama de bibliotecas e frameworks, o JavaScript se tornou indispensável para o desenvolvimento web rápido e eficiente.
- Node.js: - O Node.js, adotado para o backend, garantiu uma execução eficiente e escalável do lado do servidor. Sua capacidade de lidar com múltiplas solicitações simultaneamente, graças ao seu modelo de I/O não bloqueador, tornou-o ideal para aplicações web modernas que exigem alto desempenho e disponibilidade.
- Flask: - O uso do Flask para conectar a aplicação ao algoritmo genético em Python demonstrou a flexibilidade e a eficiência deste framework. O Flask, conhecido por sua simplicidade e capacidade de se integrar facilmente com outras tecnologias, foi uma escolha estratégica para criar uma ponte entre o frontend em JavaScript e o processamento de backend em Python.
- React: - A escolha do React como biblioteca principal para a construção da interface do usuário reflete a busca por eficiência e reatividade. Com seu modelo baseado em componentes, o React permite uma abordagem declarativa e modular para a construção de interfaces, facilitando a manutenção e a escalabilidade do código. Além disso, a utilização do estado e do ciclo de vida dos componentes do React aprimora a experiência do usuário, permitindo atualizações em tempo real e uma interatividade fluida.
- CSS e Styled-Components: - A combinação do CSS tradicional com Styled-Components criou uma solução poderosa para a estilização. Enquanto o CSS oferece a base para o design visual, os Styled-Components proporcionam uma maneira de encapsular estilos em componentes específicos, melhorando a manutenção e a reutilização do código, além de reduzir a complexidade e os conflitos de estilos.
- API do Trello: - A integração com a API do Trello trouxe uma camada adicional de funcionalidade, permitindo uma interação direta com dados externos. Isso enriqueceu a aplicação com a capacidade de se conectar a um ecossistema mais amplo, abrindo portas para a automação e sincronização de tarefas em projetos.

- API do GitHub: - A API do GitHub ampliou o alcance da aplicação ao mundo dos repositórios de código. Esta conexão direta com o GitHub permitiu uma visão mais profunda dos projetos de desenvolvimento, trazendo informações valiosas sobre commits, branches, pull requests e muito mais, diretamente para a interface da aplicação.
- Firebase e Cloud Firestore: - A escolha do Firebase e, mais especificamente, do Cloud Firestore, refletiu a necessidade de um armazenamento de dados ágil e em tempo real. O Firestore, com sua abordagem NoSQL e sincronização em tempo real, ofereceu uma solução eficaz para o armazenamento e recuperação de dados dos contribuidores e das tarefas extraídas do trello e adicionadas manualmente, suportando a natureza dinâmica da aplicação.

Este conjunto de tecnologias não somente simplificou a implementação de funcionalidades avançadas, mas também assegurou a escalabilidade, a manutenção e a usabilidade da aplicação. A junção entre essas ferramentas resultou em um processo de desenvolvimento dinâmico e eficaz, ao mesmo tempo fornecendo aos usuários uma aplicação confiável, responsiva e de fácil navegação.

5.1.4 Integração de Tecnologias para Otimização de Projetos: Cloud Firestore na Aplicação Rist, GitHub e Trello

5.1.4.1 Banco de Dados NoSQL-Cloud Firestore

O Cloud Firestore, um banco de dados NoSQL fornecido pelo Firebase, é ideal para desenvolvimento em plataformas móveis, web e servidores. Este serviço, oferecido pelo Firebase e Google Cloud, compartilha funcionalidades com o Firebase Realtime Database, permitindo a sincronização de dados em tempo real em aplicativos clientes através de listeners. Essa característica é essencial para armazenar informações sobre contribuidores e tarefas.

Para integrar o banco de dados à aplicação, foi necessário inicialmente criar um projeto no console do Firebase. Durante a criação desse projeto, é essencial estabelecer o Cloud Firestore como o banco de dados. Nas configurações do projeto, um script é gerado e este foi utilizado na aplicação para facilitar a interação entre a aplicação e o banco de dados. Com essa integração, tornou-se possível armazenar e atualizar em tempo real todos os dados referentes a desenvolvedores e tarefas, incluindo funcionalidades para criar, editar e excluir tarefas.

O Diagrama de Classes mostrado na Figura 5 representa as seguintes classes:

A classe Repositório representa um repositŕorio do GITHUB com atributos s como ID, proprietário, nome, arquivos e colaboradores .

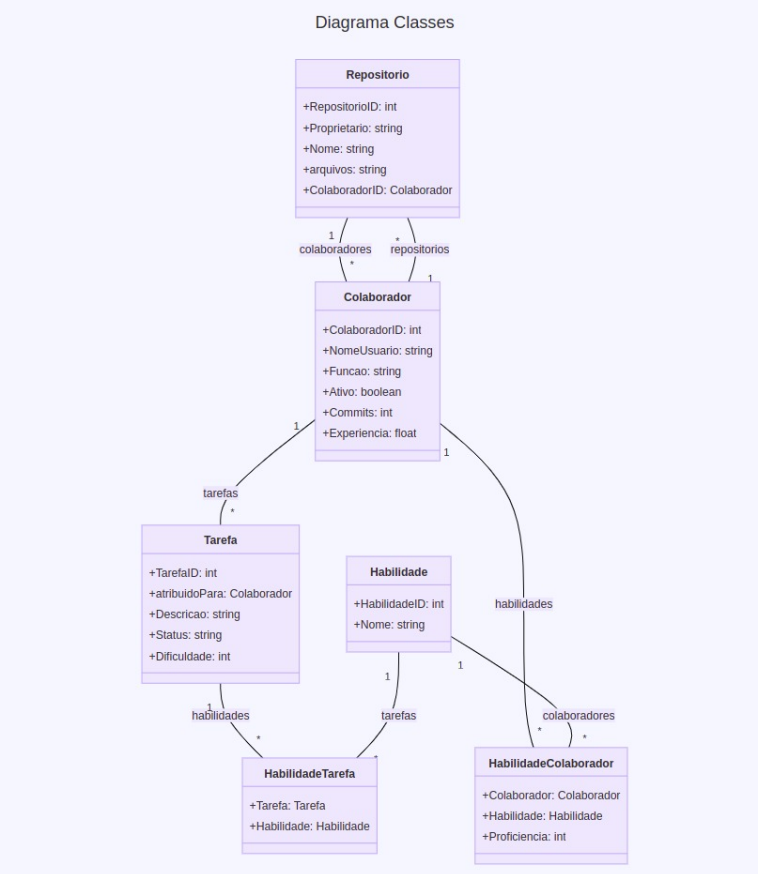
A classe Colaborador representa os usuários do GITHUB, que contribuíram dentro daquele repositório e que serão alocado as tarefas. Os atributos incluem ID, nome de usuário, função, se está ativo, número de commits e experiência.

A classe Tarefa representa uma tarefa atribuída a um colaborador, com atributos como ID, descrição, status e dificuldade.

A Classe Habilidade lista as habilidades que podem ser necessárias para realizar uma tarefa ou que um colaborador pode possuir

As Classes HabilidadeTarefa e HabilidadeColaborador, são classes associativas que ligam as habilidades às tarefas e colaboradores, respectivamente. Essas classes representam a associação de múltiplas habilidades a uma tarefa e habilidades a colaboradores com um certo nível de proficiência.

Figura 5 – Diagrama



Fonte: elaborado pelo autor.

5.1.4.2 Github

O GitHub é uma plataforma de hospedagem de código-fonte e controle de versão usando Git. Ele permite a colaboração de programadores em projetos, facilitando o compartilhamento de códigos, controle de versões e integração contínua. O GitHub oferece funcionalidades como gerenciamento de tarefas e solicitações de pull para revisão de código, além de hospedar uma variedade de projetos de código aberto. Através da API do Github é possível obter dados relevantes para a aplicação Rist, visto que a API oferece diversas funcionalidades como: Acessar repositórios, obter informações do usuário e realizar autenticação e diversas outras funcionalidades.

Além disso a API do GitHub foi utilizada para obter dados dos repositórios, visto que era necessário uma matriz que contivesse, arquivos do repositório e a contribuição de cada desenvolvedor dentro do arquivo específico, essa matriz formada também é utilizada durante a Análise do Algoritmo Genético.

A plataforma utilizou a API do Github para que o usuário fizesse Login com a aplicação. Em seguida a plataforma obtem dados do usuário e de repositórios que ele tem acesso. Assim, após o usuário fazer Login, ele é redirecionado para uma página onde irá determinar o dono do repositório e qual repositório ele quer obter informações. Após isso todos os desenvolvedores que são contribuidores daquele repositório serão listados e a Matriz de contribuições do repositório é gerada.

Um exemplo de Matriz de Contribuições é apresentado pela Matriz (5.1). Ela representa a quantidade de contribuições em cada arquivo do repositório, sendo que cada coluna representa um arquivo e cada linha representa um contribuidor. O total de arquivos analisados é 8 e cada contribuidor, observando a matriz é possível perceber que o arquivo 3 possui contribuição de todos os participantes, porém o contribuidor 3 fez uma quantidade de contribuições superior em relação aos outros contribuidores.

$$A = \begin{bmatrix} 1 & 1 & 6 & 1 & 2 & 1 & 1 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.1}$$

É possivel visualizar a função implementada em Javascript no apendice C, com essa matriz (5.1) feitas os dados são enviados para a análise com o Algoritmo genético implementado por (SEGUNDO, 2022).

5.1.4.3 Trello

De acordo com o Trello[1], a ferramenta é descrita como "a ferramenta visual que possibilita ao time o gerenciamento de qualquer tipo de projeto, fluxo de trabalho ou monitoramento de tarefas.".

Além disso, através do Trello existe a opção de construir aplicações em cima do Trello criando um Power-Up. O Power-Up do trello é uma forma de aproveitar a API para estender as funcionalidades do Trello e integrá-lo com outras ferramentas e serviços. A API oferece endpoints para manipular quadros, cartões, listas, membros da equipe e muito mais. Para utilizar a API do Trello é necessário criar um token de API e chave de API no site do Trello. Esses dados são adquiridos após criar um Power-Up no trello.

Foi necessário desenvolver um Power-Up no Trello para acessar dados específicos de cartões, incluindo cartões de tarefas, utilizando as chaves da API na aplicação. Ao utilizar essa API a aplicação Rist consegue solicitar dados do usuário que está conectado com o Trello, no caso da aplicação é mostrado todos os Board de Tarefas daquele usuário, através do Board Selecionado é possível fazer a listagem de todas as tarefas que estão na coluna “A Fazer”.

Após isso é necessário que o usuário selecione as habilidades que os desenvolvedores precisam ter para que a tarefa seja executada. Todas as tarefas são armazenadas em um banco de dados de nuvem NoSQL. Os dados utilizados dessas tarefas são representados apenas por índices da tarefa. A identificação da tarefa e habilidades da tarefa juntos formaram uma matriz. A Matriz de Habilidades das Tarefas é utilizada para representar as tarefas que devem ser alocadas. Um

[1] https://trello.com/

exemplo desta matriz é apresentado pela Matriz (5.2), a qual é composta por um conjunto de 5 tarefas (linhas) e 10 habilidades (colunas). Caso esteja marcado com 1 significa que aquela habilidade é importante na tarefa.

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5.2}$$

É possivel visualizar a implementação da função de criação da matriz no apendice B. Esta matriz e as outras são repassadas por meio da utilização Flash em python para o Algoritmo Genético implementado por (SEGUNDO, 2022).

5.1.4.4 Algoritmo Genético

O trabalho de SEGUNDO (2022) aborda a utilização de um algoritmo genético, implementado em Python utilizando a biblioteca DEAP (Distributed Evolutionary Algorithms in Python) utilizada para implementar e gerenciar o algoritmo genético que otimiza a distribuição de tarefas em projetos de software. Ela proporciona as ferramentas necessárias para a criação, avaliação, evolução e seleção de soluções, desempenhando um papel vital na melhoria da eficiência e eficácia do gerenciamento do projeto. Para utilizar esse algoritmo em Python dentro da aplicação em Javascript, foi necessário utilizar o framework web em Python, Flask para criar uma API web, através dele foi possível utilizar a implementação do algoritmo genético.

O algoritmo Genético utiliza 3 matrizes que representam agentes, tarefas e repositório, essas matrizes representam respectivamente: A matriz de agentes (5.1) é composta pelos contribuidores do projeto, cada um caracterizado por um conjunto único de habilidades. A função dessa matriz é identificar quais agentes estão mais aptos a executar determinadas tarefas, com base em suas habilidades específicas. A matriz de tarefas (5.2) contém um detalhamento das tarefas necessárias para a conclusão do projeto, incluindo as habilidades requeridas para cada uma delas. O algoritmo utiliza essas informações para casar tarefas com agentes, garantindo que

cada tarefa seja atribuída ao agente mais qualificado. A matriz de repositório (5.2): representa a contribuição de cada agente nos diversos arquivos do repositório do projeto. Essa matriz é crucial para entender o nível de experiência e envolvimento de cada agente com diferentes seções do código, influenciando assim a alocação de tarefas relacionadas a esses arquivos.

Na aplicação, os dados relacionados às habilidades de desenvolvedores são inseridos manualmente pelo usuário. A Matriz de valor de Habilidade apresentada na Matriz C (5.3) representa as competências de seis contribuidores diferentes (linhas) em dez áreas distintas (colunas). As competências são avaliadas numa escala de 1 a 10.

$$C = \begin{bmatrix} 5 & 7 & 6 & 7 & 6 & 5 & 3 & 2 & 8 & 8 \\ 8 & 6 & 5 & 7 & 6 & 6 & 5 & 6 & 7 & 7 \\ 10 & 5 & 6 & 7 & 6 & 6 & 6 & 7 & 6 & 6 \\ 10 & 9 & 8 & 8 & 6 & 7 & 5 & 5 & 8 & 5 \\ 9 & 6 & 8 & 8 & 7 & 7 & 7 & 5 & 8 & 6 \\ 5 & 8 & 8 & 7 & 5 & 8 & 5 & 7 & 5 & 6 \end{bmatrix} \tag{5.3}$$

Utilizando estas matrizes que são enviadas para o algoritmo utilizando a função demonstada no apendice D, o algoritmo genético gera várias soluções possíveis para a alocação de tarefas, onde cada solução representa uma combinação específica de tarefas e agentes. Cada solução é avaliada com base em critérios como a compatibilidade das habilidades dos agentes com as tarefas e a distribuição equilibrada do trabalho. Através de seleção, cruzamento e mutação, o algoritmo busca melhorar continuamente estas soluções, visando encontrar a configuração mais eficiente e eficaz para a alocação de tarefas.

5.2 Interfaces do sistema e as suas funções

5.2.0.1 Tela Inicial

A página inicial do sistema mostrada na Figura 6 é onde o usuário terá o primeiro contato com a aplicação e nela que o usuário será redirecionado para fazer Login com Github, ao clicar no botão "Login com Github"o usuário é redirecionado para a página de login do GitHub, após o login ser realizado ele é redirecionado novamente para a aplicação Figura 7.

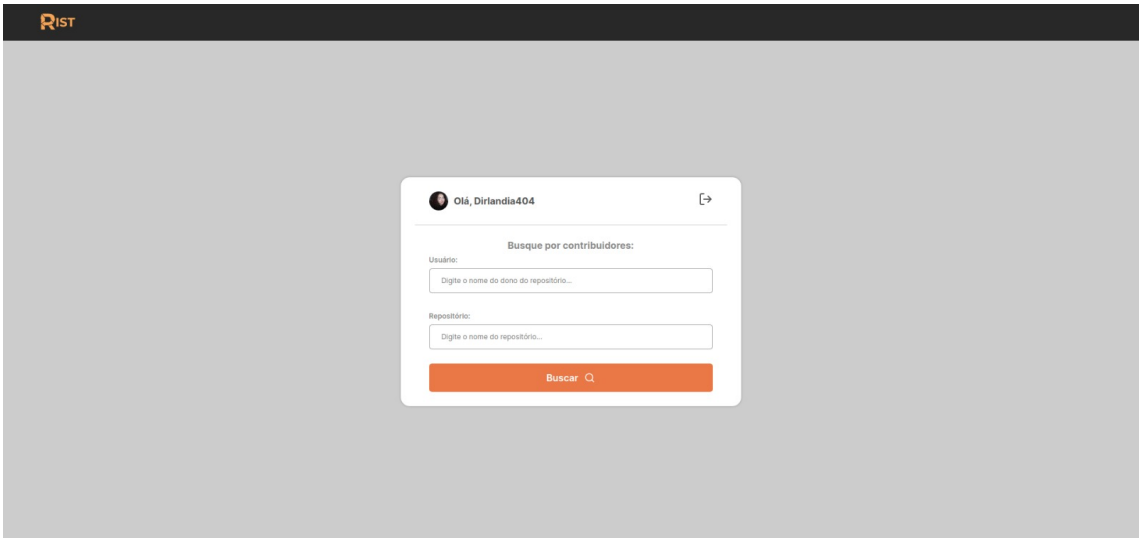
Figura 6 – Tela Inicial

Fonte: elaborado pelo autor.

5.2.0.2 Tela de busca de repositório

Na Tela de busca de repositório exibida na Figura 7, o usuário deve inserir dados sobre o repositório que ele deseja buscar, caso seja um repositório público o usuário deve inserir dados do dono do repositório e nome do repositório que deseja buscar, caso o repositório seja privado o usuário deve ter permissões dentro do repositório para fazer as buscas. Após inserir os dados corretamente e realizar a busca o usuário é direcionado a página de board da aplicação, exibida na Figura 7.

Figura 7 – Tela de Busca de repositório

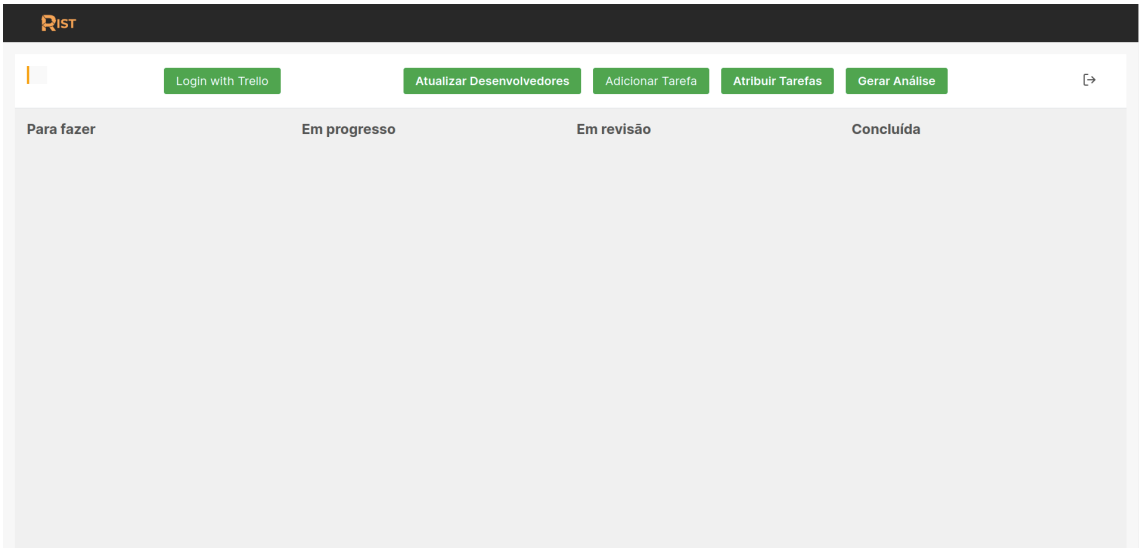


Fonte: elaborado pelo autor.

5.2.0.3 Autenticação com Trello e busca de Tarefas

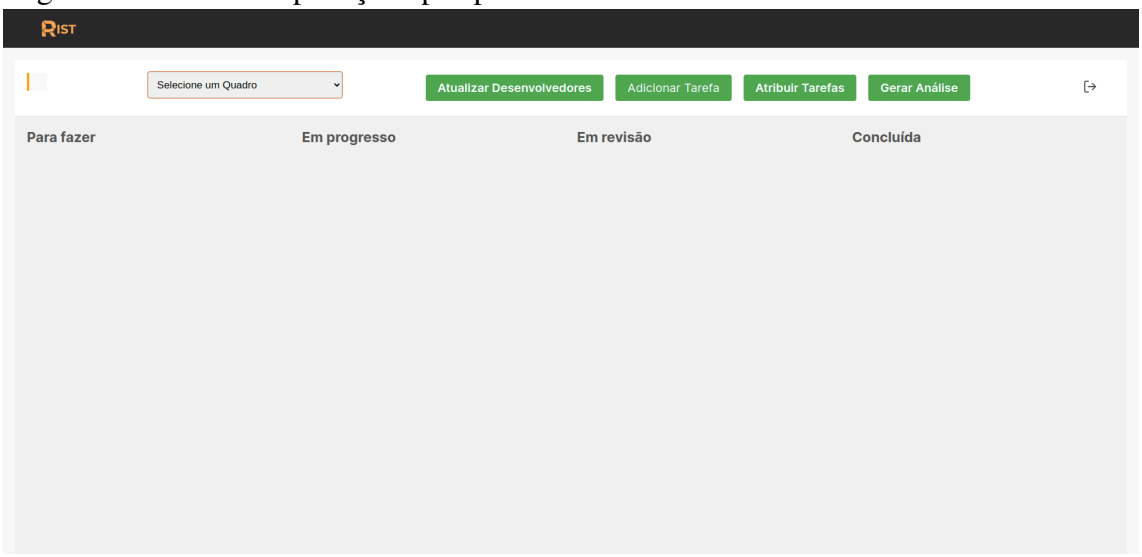
Na tela de Board Figura 8 o usuário possui a possibilidade de buscar tarefas diretamente do Trello, clicando no botão Login Trello, no entanto para realizar essa busca é necessário que o usuário solicite a permissão, após realizar a permissão o usuário é redirecionado para a página de Board onde o usuário pode selecionar o quadro que deseja buscar as tarefas, após essa etapa já é possível visualizar as tarefas na coluna denominada “Para fazer”.

Figura 8 – Board da aplicação



Fonte: elaborado pelo autor.

Figura 9 – Board da aplicação após permissões do Trello

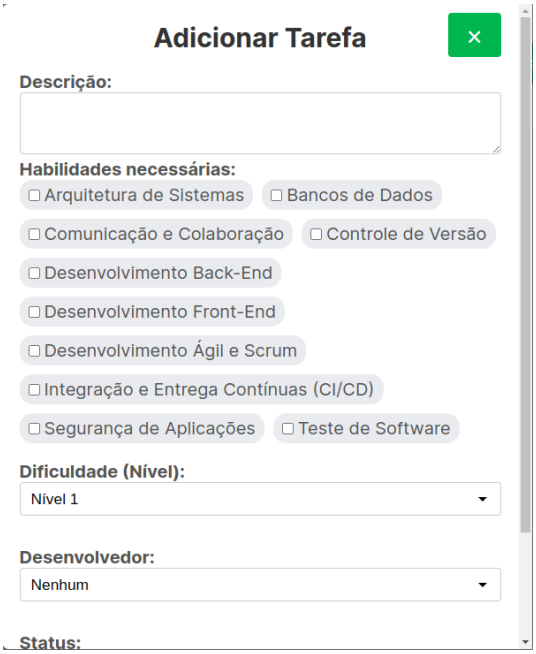


Fonte: elaborado pelo autor.

5.2.0.4 *Adicionando nova Tarefa*

Se o usuário optar por não sincronizar tarefas do Trello ou precisar complementar a lista com tarefas que não estão presentes na plataforma, ele pode prosseguir com a inserção manual. Este processo envolve o preenchimento detalhado das informações Figura10, incluindo a descrição da tarefa e a seleção das habilidades necessárias específicas para sua execução.

Figura 10 – Adicionando nova tarefa



Fonte: elaborado pelo autor.

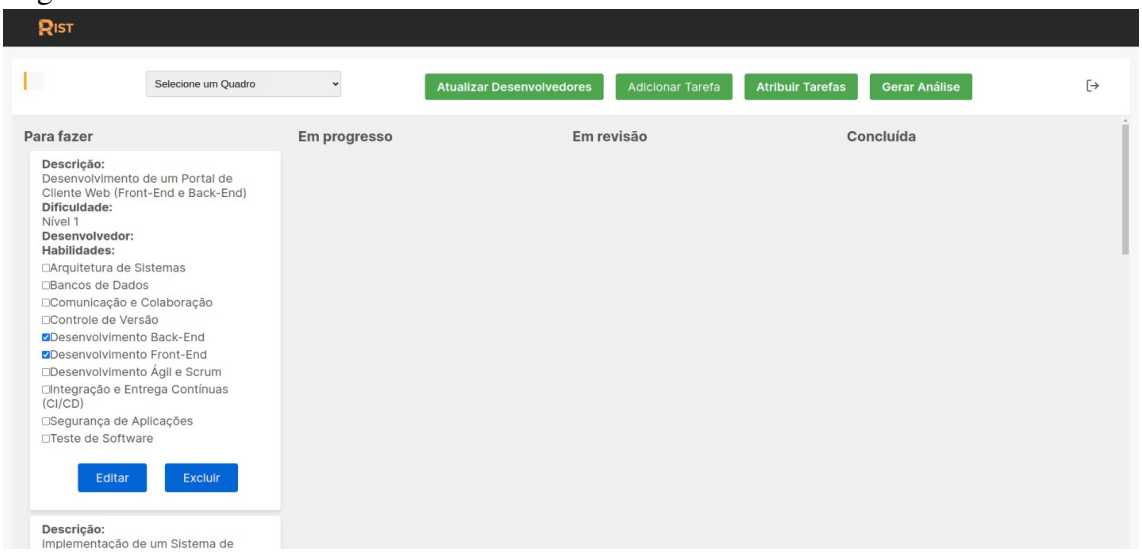
5.2.0.5 *Adicionando habilidades necessárias em cada Tarefa*

Uma vez que as tarefas são determinadas, é crucial que o usuário assinale as competências específicas necessárias para cumprir cada uma delas. As habilidades são atribuídas marcando-se as caixas de seleção correspondentes, Figura11.

5.2.0.6 *Adicionando habilidades necessárias aos desenvolvedores*

As competências dos colaboradores associados ao repositório em questão requerem atribuição manual, dado que não há um meio de obter automaticamente essas informações do GitHub. A interface para essa atribuição, conforme ilustrada na Figura12, apresenta uma relação de competências onde valores específicos devem ser designados a cada habilidade aplicável.

Figura 11 – Adicionando habilidades as tarefas



Fonte: elaborado pelo autor.

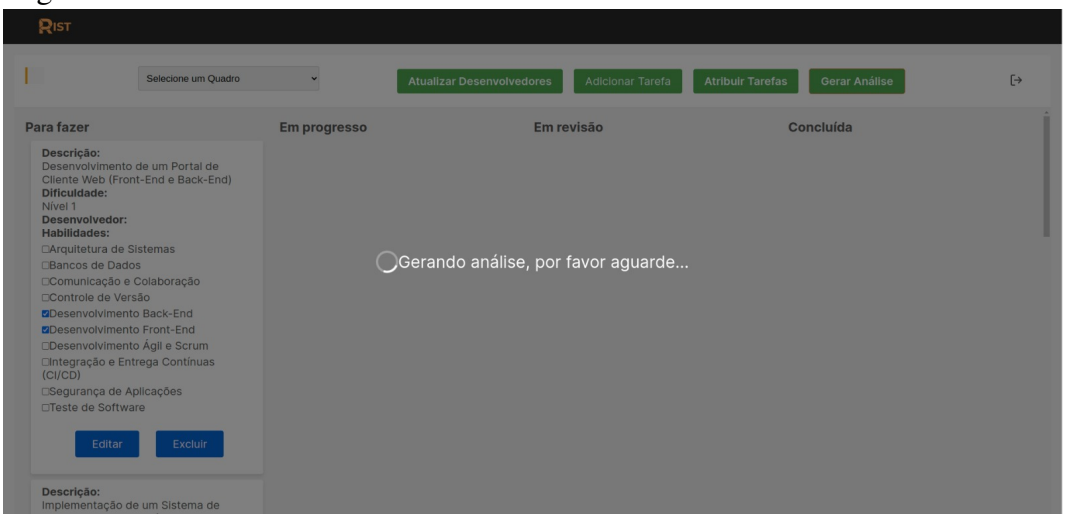
Figura 12 – Designação valores de habilidades

Fonte: elaborado pelo autor.

5.2.0.7 *Gerar Análise e atribuindo tarefas aos desenvolvedores*

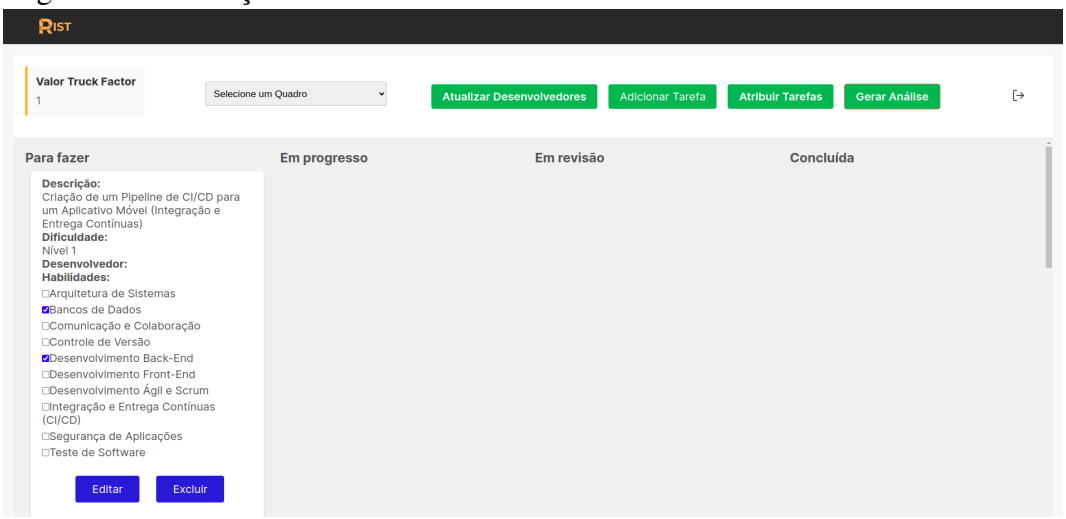
Uma vez definidas as habilidades necessárias para as tarefas e as competências dos desenvolvedores, a análise do Truck Factor pode ser executada selecionando a opção "Gerar análise". Concluída a análise, o valor do Truck Factor é apresentado na interface, Figura14, permitindo ao usuário proceder com a alocação das tarefas aos colaboradores.

Figura 13 – Gerando análise



Fonte: elaborado pelo autor.

Figura 14 – Exibição Valor Truck Factor



Fonte: elaborado pelo autor.

5.3 Exemplo de uso da ferramenta em um projeto

Nesta seção, o uso da aplicação é demonstrado por meio do projeto “fupisfun”. Este é um projeto voltado ao desenvolvimento de questões de fup para disciplinas de programação. O projeto possui 6 colaboradores e é um repositório público. O repositório do projeto está disponível em https://github.com/senapk/fupisfun/.

O Link demonstração da ilustração exibe a utilização da aplicação no ambiente citado anteriormente. A seguir serão exibidos os resultados de alocação gerados durante a utilização da aplicação.

As matrizes geradas a partir do projeto foram apresentadas na Seção 5.1.4 para ilustrar cada uma das matrizes utilizadas pela solução desenvolvida. A Matriz de Contribuições

do projeto foi apresentada por meio da Matriz A (5.1).

A Matriz de Tarefas do projeto foi apresentada por meio da Matriz B (5.2). Ela é utilizada para representar as tarefas que devem ser alocadas, compostas por um conjunto de 5 tarefas buscadas do trello, que são respectivamente mostradas na Tabela abaixo:

Tarefa
Desenvolvimento de um Portal de Cliente Web
Implementação de um Sistema de Autenticação Segura
Otimização do Desempenho do Banco de Dados
Desenvolvimento de um Framework de Testes Automatizados
Criação de um Pipeline de CI/CD para um Aplicativo Móvel

Cada coluna representa o conjunto de habilidades, composta por um grupo de 10 habilidades, que são respectivamente:

Habilidades
Arquitetura de Sistemas
Bancos de Dados
Comunicação e Colaboração
Controle de Versão
Desenvolvimento Back-End
Desenvolvimento Front-End
Desenvolvimento Ágil e Scrum
Integração e Entrega Contínuas (CI/CD)
Segurança de Aplicações
Teste de Software

Segue uma relação das tarefas com as habilidades correspondentes que são requeridas para cada uma:

Tarefa	**Habilidades**
Desenvolvimento de um Portal de Cliente Web	Desenvolvimento Back-End, Desenvolvimento Front-End
Implementação de um Sistema de Autenticação Segura	Segurança de Aplicações, Teste de Software
Otimização do Desempenho do Banco de Dados	Bancos de Dados, Arquitetura de Sistemas
Desenvolvimento de um Framework de Testes Automatizados	Teste de Software, Arquitetura de Sistemas
Criação de um Pipeline de CI/CD para um Aplicativo Móvel	Integração e Entrega Contínuas (CI/CD), Bancos de Dados

Enquanto a Matriz de valor de habilidade apresentada na Seção 5.1.4 por meio da Matriz C (5.3). Cada linha representa um contribuidor e cada coluna representa uma área, essas áreas são respectivamente: "Arquitetura de Sistemas", "Bancos de Dados", "Comunicação e Colaboração", "Controle de Versão","Desenvolvimento Back-End", "Desenvolvimento Front-End", "Desenvolvimento Ágil e Scrum","Integração e Entrega Contínuas (CI/CD)", "Segurança de Aplicações", "Teste de Software".

Na Figura 15 é possivel visualizar os seis desenvolvedores e os valores atribuidas as 10 habilidades existentes, esse valores para cada habilidade foi determinado para representar o quanto cada desenvolvedor tinha de doniminio referente a cada habilidade.

Na Figura 16 é possivel visualizar as 5 tarefas existentes, essas tarefas foram extraidas do trello para a aplicação e quais foram as habilidades necessárias atribuidas a cada uma. As habilidades definidas para cada foram determinadas de acordo com o que a tarefa necessitava para ser realizada.

Ao aplicar o envio dessas matrizes para o algoritmo genético durante 10 vezes, ele gerou as seguintes soluções: geração 1, geração 2, geração 3, geração 4, geração 5, geração 6, geração 7, geração 8, geração 9, geração 10. As diferentes soluções indicam a exploração eficiente do algoritmo genético no espaço de possíveis soluções.

Figura 15 – Valores de cada habilidade dos desenvolvedores

Fonte: print retirado do Firestore

Figura 16 – Tarefas e habilidades necessárias

Fonte: print retirado do Firestore

$$1 = \boxed{5}\boxed{1}\boxed{0}\boxed{4}\boxed{3} \tag{5.4}$$

$$2 = \boxed{5}\boxed{4}\boxed{0}\boxed{1}\boxed{3} \tag{5.5}$$

$$3 = \boxed{4}\boxed{3}\boxed{5}\boxed{1}\boxed{0} \tag{5.6}$$

$$4 = \boxed{4}\boxed{5}\boxed{1}\boxed{3}\boxed{0} \tag{5.7}$$

$$5 = \boxed{0}\boxed{5}\boxed{4}\boxed{1}\boxed{3} \tag{5.8}$$

$$6 = \boxed{0}\boxed{1}\boxed{5}\boxed{4}\boxed{3} \tag{5.9}$$

$$7 = \boxed{0}\boxed{4}\boxed{5}\boxed{1}\boxed{3} \tag{5.10}$$

$$8 = \boxed{1}\boxed{5}\boxed{3}\boxed{4}\boxed{0} \tag{5.11}$$

$$9 = \boxed{3}\boxed{5}\boxed{0}\boxed{1}\boxed{4} \tag{5.12}$$

$$10 = \boxed{5}\boxed{4}\boxed{1}\boxed{3}\boxed{0} \tag{5.13}$$

Essas soluções podem ser utilizadas para alocar o desenvolvedor a cada tarefa especifica. A Figura 17 apresenta a alocação da geração 10, onde o desenvolvedor 5, foi alocado para a tarefa de indice 0, o desenvolvedor 4 foi alocado para a tarefa de indice 1, o desenvolvedor 1 foi alocado para a tarefa de indice 2, o desenvolvedor 3 foi alocado para a tarefa de indice 3 e por ultimo o desenvolvedor 0 foi alocado para a tarefa de indice 5. Vale ressaltar que o indice de cada desenvolvedor é determinado pela lista de Time mostrada na Figura 18.

Figura 17 – Alocação realizada

Resultados da consulta

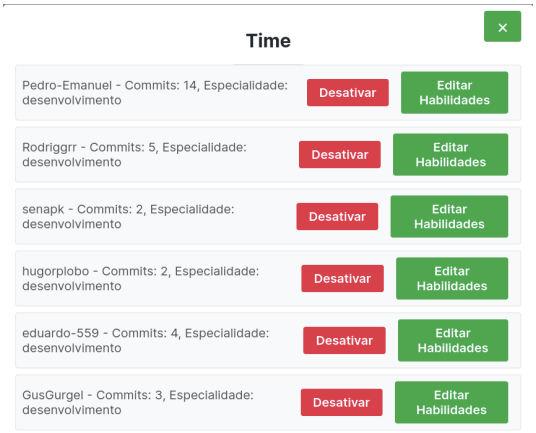
Document ID	description	desenvolvedor	difficulty	id	skills	status
Bp49VG0Zli6Yl5bHsLdc	"Desenvolvimento de um Portal de Cliente Web (Front-End e Back-End)"	"GusGurgel"	"1"	"Bp49VG0Zli6Yl5bHsLdc"	["Desenvolvimento Back-End", "Desenvolvimento Front-End"]	"in_progress"
KktftkNpj4uvzWeFSeI7	"Implementação de um Sistema de Autenticação Segura (Segurança de Aplicações)"	"eduardo-559"	"1"	"KktftkNpj4uvzWeFSeI7"	["Segurança de Aplicações", "Teste de Software"]	"in_progress"
RFEWkK6un4l0SjsxlfZX	"Otimização do Desempenho do Banco de Dados (Bancos de Dados e Arquitetura de Sistemas)"	"Rodriggrr"	"1"	"RFEWkK6un4l0SjsxlfZX"	["Bancos de Dados", "Arquitetura de Sistemas"]	"in_progress"
dTy4njAadjBxKa3P3mbW	"Desenvolvimento de um Framework de Testes Automatizados (Teste de Software) Habilidades necessárias:"	"hugorplobo"	"1"	"dTy4njAadjBxKa3P3mbW"	["Teste de Software", "Arquitetura de Sistemas"]	"in_progress"
oiVEPkVnTUfSnKPSY95g	"Criação de um Pipeline de CI/CD para um Aplicativo Móvel (Integração e Entrega Contínuas)"	"Pedro-Emanuel"	"1"	"oiVEPkVnTUfSnKPSY95g"	["Integração e Entrega Contínuas (CI/CD)", "Bancos de Dados"]	"in_progress"

Items per page: 50 1 - 5 de 5

Local do banco de dados: nam5

Fonte: print retirado do Firestore

Figura 18 – Lista de Desenvolvedores



Fonte: print retirado da aplicação

5.4 Como utilizar a aplicação

A implementação da aplicação está disponível em https://github.com/Dirlandia404/Rist-Reducing-Impact-turnover. Para utilizar a aplicação, o usuario deve executar a aplicação localmente, gerar os tokens de acesso no GitHub e no trello.

A criação de um token no GitHub é detalhada na documentação oficial, e o processo envolve várias etapas importantes que começam com o acesso à sua conta no GitHub. Primeiramente, é necessário navegar até as configurações do perfil, prosseguir para as configurações de desenvolvedor e lá, o usuário poderá adicionar um novo aplicativo OAuth. Neste processo, o usuário receberá um ID de cliente e um segredo de cliente, que são essenciais para a autenticação OAuth. Com essas credenciais e o token de acesso obtido, você poderá realizar a integração com a aplicação, após substituir na aplicação.

Para utilização da API do Trello é necessário a criação de um Power-Up, Link para documentação https://developer.atlassian.com/cloud/trello/guides/power-ups/your-first-power-up/, uma vez que, será utilizado dentro de quadros do Trello, para isso é necessario a página https://trello.com/power-ups/admin, criar um novo power-up, selecionar as permissões de acesso e ativar, após criado é necessário fazer a substituição da chave de API dentro da aplicação.

Também é necessário substituir as informações do banco de dados dentro da aplicação, para isso o usuário deve acessar o console do Firebase, adicionar um novo projeto, selecionar o tipo no caso, web, selecionar o Cloud Firestore e criar um banco de dados, após essas configurações o usuário deve ir até as configurações do projeto é lá que irá ter o script que deve ser substituído para inicialização do banco de dados dentro da aplicação. Após essas configurações o usuário já consegue utilizar a aplicação localmente.

6 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresenta uma solução web para alocação de tarefas automática com base no truck factor e de forma integrada com Github e Trello. A solução foi desenvolvida com base em necessidades confirmadas por profissionais de desenvolvimento de software participantes de um estudo baseado em questionário.

O desenvolvimento envolveu um conjunto de tecnologias. As funcionalidades disponibilizadas permitem a recuperação de dados de repositórios do github e de projetos do Trello, permite ajustar os dados de habilidades de cada participante, bem como visualizar a informação de truck factor atual do projeto e realizar alocação de tarefas de forma automática, por meio de algoritmo genético, considerando o valor do truck factor e as habilidades do time de desenvolvimento.

6.1 Impedimentos

Limites da API do GitHub: Limitações quanto ao número de solicitações que podem ser feitas durante um determinado período de tempo, essa limitação afeta diretamente na busca de repositórios de projetos grandes.

Limitações de Armazenamento e Leitura do Firestore: O Firestore tem limites na quantidade de dados que posso armazenar e no número de operações de leitura e escrita. Ocorre limitações quando o banco de dados é consultado frequentemente.

6.2 Trabalhos futuros

Durante o desenvolvimento da solução surgiram alguns impedimentos, visto que os unicos dados possiveis extraídos do github eram quantidades de commits que cada desenvolvedor realizou, porém essa metrica não é muito eficaz para determinar o truck factor, visto que existe outros fatores bem como: o quão importante aquele commit foi relevante para a aplicação, quais dados ele adicionou ao documento. Infelizmente não foi possivel extrair esses dados de forma exata, por esse motivo o truck factor foi determinado somente pela quantidade de commits. Esta é uma possibilidade de aprimoramento da ferramenta desenvolvida

Apresentam-se a seguir outras sugestões para realização de trabalhos futuros

– Realização de testes de sistema com uma equipe de desenvolvimento durante um periodo para obter conhecimento do quão eficas a ferramenta rist foi para o processo de alocação

de tarefas.

- Adicionar funcionalidades bem como: a opção de selecionar apenas algumas tarefas para fazer a locação, selecionar mais de um desenvolvedor, adicionar mais fatores que contribua para a geração do algoritmo genetico como outras habilidades, por exemplo: conhecimento de testes, desenvolvimento back-end, entre outras habilidades.
- Melhorar a interface de usuario respeitando os padrões de usabilidade, utilizando as Heurísticas de Nielsen.

REFERÊNCIAS

AVELINO, G.; PASSOS, L.; HORA, A.; VALENTE, M. T. A novel approach for estimating truck factors. In: **2016 IEEE 24th International Conference on Program Comprehension (ICPC)**. [*S. l.*: *s. n.*], 2016. p. 1–10.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Handbook of Evolutionary Computation**. [*S. l.*]: Oxford University Press, 1997.

BECK, K.; BEEDLE, M.; BENNEKUM, A. V.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R. *et al.* **Manifesto for agile software development**. 2001.

CARVALHO, W. C. d. S. **Análise dos efeitos do turnover na produtividade de processos de software tradicionais e híbridos**. Tese (Doutorado) – Universidade Federal de Uberlândia, 2012.

CHAVES, A.; AMORIM, L.; MARINHO, M.; JúNIOR, I. d. F.; MOURA, H. Autonomy and turnover in distributed software development projects: A systematic literature review. **Cibse**, p. 15, 2022.

CHIAVENATO, I. **Recursos Humanos**: Capitulo 3. 4. ed. São Paulo: Atlas, 1997.

COSENTINO, V.; IZQUIERDO, J. L. C.; CABOT, J. Assessing the bus factor of git repositories. In: IEEE. **2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER)**. [*S. l.*], 2015. p. 499–503.

COUTINHO, T. **Entenda as funções de um Scrum Master e sua importância em um Scrum Team**. 2020. Disponível em: https://www.voitto.com.br/blog/artigo/scrum-master. Acesso em: 12 jun. 2022.

DARWIN, C. **On the Origin of Species**. Londres: John Murray, 1859.

FADEL, A. C.; SILVEIRA, H. d. M. Metodologias ágeis no contexto de desenvolvimento de software: Xp, scrum e lean. **Monografia do Curso de Mestrado FT-027-Gestão de Projetos e Qualidade da Faculdade de Tecnologia–UNICAMP**, v. 98, p. 101, 2010.

FERREIRA, M.; VALENTE, M. T.; FERREIRA, K. A comparison of three algorithms for computing truck factors. In: **2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)**. [*S. l.*: *s. n.*], 2017. p. 207–217.

FERREIRA, M. M.; FERREIRA, K. A. M.; VALENTE, M. T. Distribuiçao de conhecimento de código em times de desenvolvimento-uma análise arquitetural. In: **IV Workshop on Software Visualization, Evolution and Maintenance (VEM 2016)**. [*S. l.*: *s. n.*], 2016. p. 1–8.

FILHO, M. S.; PINHEIRO, P. R.; ALBUQUERQUE, A. B.; SIMÃO, R. P. S.; AZEVEDO, R. S. N.; NUNES, L. C. **A Multicriteria Approach to Support Task Allocation in Projects of Distributed Software Development**. [*S. l.*]: hindawi, 2019. Intervalo de páginas p.

FRITZ, T.; MURPHY, G.; MURPHY-HILL, E.; OU, J.; HILL, E. Grau de conhecimento: modelando o conhecimento de código de um desenvolvedor. **ACM Transactions on Software Engineering and Methodology (TOSEM)**, v. 23, 2014.

GanttPRO. **GanttPRO Blog**. 2023. https://ganttpro.com/pt/?redirectByBrowserDetectedLocale. Acesso em: 18 dez. 2023.

GOLDBERG, D. E. **Genetic Algorithms in Search, Optimization, and Machine Learning**. [*S. l.*]: Addison-Wesley, 1989.

Idego Group. **Software Engineering Team's Turnover**. 2021. Disponível em: https://idego-group.com/blog/2021/11/19/software-engineering-teams-turnover/. Acesso em: 18 dez. 2023.

INSTITUTE, P. M. **A Guide to the Project Management Body of Knowledge (PMBOK® Guide)**. 6th. ed. [*S. l.*]: Project Management Institute, 2017.

KOSCHEVIC, M. T. **Gerenciamento de processos com metodologias ágeis**. 2012. Disponível em: https://repositorio.utfpr.edu.br/jspui/bitstream/1/23527/2/MD_ENGESS_I_2012_15.pdf. Acesso em: 16 dez. 2023.

LACERDA, E. G. M.; CARVALHO, A. C. P. d. L. F. **Introdução aos algoritmos genéticos**. Rio de Janeiro: EntreLugar, 1999.

OLIVEIRA, L. F. *et al.* **Task Allocation in Distributed Software Development**: A systematic literature review. 2023. Disponível em: https://www.hindawi.com/journals/jcse/2023/1234567/. Acesso em: 26 jun. 2022.

PACHECO, M. A. C. *et al.* **Algoritmos genéticos**: princípios e aplicações. [*S. l.*]: ICA: Laboratório de Inteligência Computacional Aplicada. Departamento de Engenharia Elétrica. Pontifícia Universidade Católica do Rio de Janeiro., 1999.

PATEL, B. **Project Management**. 2. ed. [*S. l.*]: Vikas, 2010.

PATTERSON, D. **Strategic Project Management for Human Resources**. London, Ontario: Fanshawe College Pressbooks, 2022. Licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License, except where otherwise noted.

PONTES, L. G.; NOVAES, M. A.; CARVALHO, G. R. d. C.; SERPA, M. J. M. Fatores que influenciam a rotatividade na organização. **XVIII Encontro Latino Americano de Iniciação Científica, XIV Encontro Latino Americano de Pós-Graduação e IV Encontro de Iniciação à Docência – Universidade do Vale do Paraíba**, p. 1–7, 2017.

RIGBY, P. C.; ZHU, Y. C.; DONADELLI, S. M.; MOCKUS, A. Quantifying and mitigating turnover-induced knowledge loss: case studies of chrome and a project at avaya. In: IEEE. **2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)**. [*S. l.*], 2016. p. 1006–1016.

SEGUNDO, C. V. N. **Uma Simulação Multiagente para Alocação de Tarefas em Projetos de Software Baseada no Truck Factor**. Dissertação (Dissertação (Mestrado em Computação)) – Universidade Federal do Ceará, Quixadá, 2022. Programa de Pós-Graduação em Computação. Disponível em: https://repositorio.ufc.br/handle/riufc/70817. Acesso em: 18 dez. 2023.

SOMMERVILLE, I. **Engenharia de Software**. 9.. ed. São Paulo: Pearson Prentice Hall, 2011. ISBN 978-85-7936-108-1.

STANDISH GROUP. **Chaos Report**. 2020. Disponível em: https://www.standishgroup.com. Acesso em: 26 jun. 2022.

STRAND, A.; GUNNARSON, M.; BRITTO, R.; USMAN, M. Using a context-aware approach to recommend code reviewers: Findings from an industrial case study. In: IEEE. **Proceedings of the 42nd ACM/IEEE International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)**. [*S. l.*], 2020.

ZAZWORKA, N.; STAPEL, K.; KNAUSS, E.; SHULL, F.; BASILI, V. R.; SCHNEIDER, K. Are developers complying with the process: an xp study. In: **Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement**. [*S. l.*: *s. n.*], 2010. p. 1–10.

APÊNDICE A – CÓDIGOS-FONTES UTILIZADOS PARA MAPEAR DADOS DO REPOSITORIO

```
async function fetchContributionData() {
  const response = await axios.get(
    `https://api.github.com/repos/${owner}/${repo}/contents`
  );
  const files = response.data.filter((item) => item.type === "file");
  const contributionsMap = new Map();
  for (const file of files) {
    const commitsResponse = await axios.get(
      `https://api.github.com/repos/${owner}/${repo}/commits`,
      { params: { path: file.path } }
    );
    commitsResponse.data.forEach((commit) => {
      const authorLogin = commit.author?.login;
      if (authorLogin) {
        if (!contributionsMap.has(authorLogin)) {
          contributionsMap.set(authorLogin, {});
        }
        const userContributions = contributionsMap.get(authorLogin);
        if (!userContributions[file.name]) {
          userContributions[file.name] = 0;
        }
        userContributions[file.name]++;
        contributionsMap.set(authorLogin, userContributions);
      }
    });
  }
  const contributorsList = Array.from(contributionsMap.keys());
  const filesList = files.map((file) => file.name);
  const contributionMatrix = contributorsList.map((login) => {
    return filesList.map((fileName) => {
      const userContributions = contributionsMap.get(login) || {};
      return userContributions[fileName] || 0;
    });
  });
  return contributionMatrix;
}
```

Código-fonte 1 – Função javascript para obter Dados dos contribuidores do repositorio

APÊNDICE B – CÓDIGOS-FONTES UTILIZADOS PARA MAPEAR DADOS DAS TAREFAS

```
async function fetchTasksData() {
  const tasksCollection = collection(
    db,
    "repositories",
    `${owner}_${repo}`,
    "tasks"
  );
  const tasksSnapshot = await getDocs(tasksCollection);

  const tasksMatrix = tasksSnapshot.docs.map((doc) => {
    const taskData = doc.data();
    const skillsArray = availableSkills.map((skill) =>
      Array.isArray(taskData.skills) && taskData.skills.includes(skill)
        ? 1
        : 0
    );
    return skillsArray;
  });
  return tasksMatrix;
}
```

Código-fonte 2 – Função javascript para obter Dados das Tarefas

APÊNDICE C – CÓDIGOS-FONTES UTILIZADOS PARA MAPEAR DADOS DOS CONTRIBUIDORES

```
async function fetchContributorsFromFirestore() {
  let contributorsMatrix = [];
  try {
    const repoDocRef = doc(db, "repositories", `${owner}_${repo}`);
    const contributorsCollection = collection(repoDocRef, "contributors");
    const contributorSnapshot = await getDocs(contributorsCollection);

    contributorSnapshot.docs.forEach((doc) => {
      let contributorData = doc.data();

      if (!contributorData.skills) {
        contributorData.skills = {};
        availableSkills.forEach((skill) => {
          contributorData.skills[skill] = 0;
        });
      }
      let skillsArray = availableSkills.map(
        (skill) => contributorData.skills[skill] || 0
      );

      contributorsMatrix.push(skillsArray);
    });

    console.log("Matriz de contribuidores carregada:", contributorsMatrix);
  } catch (error) {
    console.error("Erro ao buscar contribuidores do Firestore:", error);
  }
  return contributorsMatrix;
}
```

Código-fonte 3 – Função javascript para obter Dados Matriz de Habilidades dos contribuidores

APÊNDICE D – FUNÇÃO QUE ENVIA AS MATRIZES AO ALGORITMO

```
async function sendOptimizationRequest() {
  setIsAnalysisLoading(true);
  try {
    const taskTable = await fetchTasksData(tasksToDo);
    const repositoryData = await fetchContributionData();
    const agentsTable = await fetchContributorsFromFirestore();

    const data = {
      repository: repositoryData,
      agents_table: agentsTable,
      task_table: taskTable,
    };

    const response = await fetch("http://localhost:5000/optimize", {
      method: "POST",
      headers: {
        "Content-Type": "application/json",
      },
      body: JSON.stringify(data),
    });

    const responseData = await response.json();
    setResponse(responseData);

    setIsAnalysisLoading(false);
  } catch (error) {
    console.error("Error:", error);
    setIsAnalysisLoading(false);
  }
}
```

Código-fonte 4 – Função javascript para enviar as matrizes ao algoritmo