Guilherme Gabriel Silva Pereira e Lucas Ângelo Oliveira Martins Rocha
{ggspereira,laomrocha}@sga.pucminas.br

Documento de Visão para a plataforma GitGrade

14 de fevereiro de 2023

Proposta do(s) aluno(s) Guilherme Gabriel Silva Pereira e Lucas Ângelo Oliveira Martins Rocha ao curso de Engenharia de Software como projeto de Trabalho de Conclusão de Curso (TCC) sob orientação de conteúdo do professor Cleiton Silva Tavares e orientação acadêmica do professor José Laerte Pires Xavier Junior.

OBJETIVOS

Este documento visa documentar o escopo previsto para o projeto que será desenvolvido pelos alunos Guilherme Gabriel Silva Pereira e Lucas Ângelo Oliveira Martins Rocha para o curso de Bacharelado em Engenharia de *Software* como projeto de Trabalho de Conclusão de Curso (TCC).

Este projeto planeja suprir a demanda dos professores do curso, mais especificamente que lecionam matérias de Trabalho Interdisciplinar, de uma plataforma para auxiliar o processo de avaliação da disciplina. A plataforma GitGrade que será desenvolvida permitirá aos docentes, de maneira geral, visualizarem as contribuições dos alunos em seus respectivos trabalhos, assim como informações divididas por integrante e *sprint*, analisar entregas de tarefas e avaliar a qualidade do código produzido a partir de integrações com ferramentas de análise estática de código.

ESCOPO

Os professores do curso de Bacharelado em Engenharia de *Software* do Instituto De Ciências Exatas e Informática da Pontifícia Universidade Católica de Minas Gerais (PUC Minas) semestralmente lecionam disciplinas de Trabalho Interdisciplinar. Nessas disciplinas, as turmas são divididas em diversos grupos de alunos, que desenvolvem, ao longo do semestre, um trabalho interdisciplinar de *software*. Por parte do professor, o processo de avaliação consiste

em verificar se todos os documentos, artefatos, arquivos e código-fonte foram entregues no repositório do trabalho na organização ICEI-PUC-Minas-PPLES-TI[1] do GitHub.

Além disso, verifica-se se o sistema desenvolvido possui boa qualidade e cumpre com os requisitos funcionais e não funcionais, bem como as restrições acordadas no início do trabalho. Atualmente, essa etapa é realizada manualmente pelos docentes, tornando-se uma tarefa árdua. Consiste nas atividades feitas manualmente: abrir o repositório, validar a entrega de artefatos (exemplo: CITATION.cff, arquivo de documentação, vídeo de apresentação, etc), verificar as contribuições de *commits*, qualidade das descrições de *commits*, adições e remoções de linhas de código e arquivos, resoluções de *issues*, entre outras informações para cada integrante de cada trabalho, em períodos de tempos delimitados por *sprint*.

A partir do que foi supracitado, nota-se que os docentes do curso de Engenharia de *Software* tem dificuldades no processo de avaliação dos trabalhos das disciplinas de Trabalho Interdisciplinar. Dessa forma, é desejável uma plataforma *web* que auxilie os professores apresentando as informações citadas anteriormente, para auxiliá-los a avaliarem qualitativamente e quantitativamente as contribuições dos alunos integrantes de cada trabalho, os trabalhos de forma geral e as entregas de cada *sprint*. Além disso, inspeções estáticas da qualidade dos códigos produzidos pelos alunos será conveniente para avaliação dos trabalhos.

Diante disso, o sistema proposto consiste em uma plataforma *web* de apoio às avaliações de trabalhos no GitHub que será utilizado pelos professores das disciplinas de Trabalho Interdisciplinar. A plataforma nomeada de GitGrade possibilitará aos docentes cadastrarem métodos avaliativos para cada oferta de disciplina e visualizar as informações resultantes dos repositórios dos trabalhos conforme o método avaliativo selecionado. A necessidade de uma plataforma com essas funções origina-se da carência de aplicações que auxilie docentes a avaliarem repositórios de código e os artefatos de documentação dos trabalhos, por meio de filtros temporais e por integrantes.

Na plataforma GitGrade, os métodos avaliativos funcionam como um agrupador de repositórios, onde professores adicionam repositórios do GitHub para serem avaliados. Cada método inclui regras de consistência, que definem os arquivos e documentos que devem ser entregues pelos alunos até a data final de cada *sprint*. As *sprints* são períodos estabelecidos para a entrega desses artefatos. Caso as entregas não atendam às regras de consistência, são geradas *issues* padronizadas no repositório do GitHub, facilitando a comunicação e a correção de falhas no processo de desenvolvimento do projeto.

Em relação aos indicadores de contribuição, as métricas dos alunos são avaliadas com detalhes significativos para uma compreensão abrangente do engajamento nos projetos. A quantidade de

[1] https://github.com/ICEI-PUC-Minas-PPLES-TI

commits fornece uma visão sobre a frequência e regularidade das contribuições de cada aluno ao repositório. Em relação aos tipos de arquivos, a análise se concentra na variedade das extensões dos arquivos contribuídos, como .md, .js, .png, entre outros, refletindo a amplitude da participação do aluno em diferentes aspectos do projeto. A quantidade de linhas de código adicionadas e removidas oferece uma medida quantitativa do impacto direto do aluno no desenvolvimento do código. A quantidade de arquivos contribuídos é outra métrica importante, indicando o volume de trabalho realizado. A qualidade da descrição dos *commits* é avaliada com base em padrões como o Angular Conventional Commits[2], enfatizando a clareza e a eficácia da comunicação nas mudanças realizadas. Por fim, a participação em *issues*, tanto como autores quanto como atribuídos, mostra o envolvimento do aluno na resolução de problemas e colaboração com a equipe. Todas essas métricas podem ser filtradas pelos contribuidores do repositório e por datas customizadas, incluindo os períodos definidos pelo início e fim de cada *sprint* do método avaliativo associado ao repositório.

O fluxo de funcionamento da plataforma iniciará com o cadastro da oferta de uma disciplina em um semestre específico, com isso, o professor dessa disciplina irá selecionar os repositórios dessa oferta. A partir disso, deverá ser cadastrado um método avaliativo para essa disciplina, informando quais são os arquivos e seus respectivos diretórios onde deverão estar criados e preenchidos. Por exemplo, na plataforma deverá ser cadastrada a oferta da disciplina de “Trabalho Interdisciplinar 5: Aplicações Distribuídas 1/2023” e seu método avaliativo contendo uma regra de consistência que exija a existência do arquivo “Documentacao/documento_de_arquitetura.md” contendo no mínimo 750 caracteres. A partir disso, todos os repositórios dessa oferta de disciplina verificarão essa regra. Quando essa regra não for seguida por algum repositório de um trabalho com esse método avaliativo, haverá uma opção para abrir uma *issue* padronizada no repositório respectivo.

Em relação ao controle temporal, também poderão ser cadastradas *sprints* para ofertas da disciplina, o que possibilitará avaliar contribuições de alunos em trabalhos em diferentes períodos temporais. Seguindo o mesmo exemplo da oferta da disciplina de “Trabalho Interdisciplinar 5: Aplicações Distribuídas 1/2023”, poderão ser cadastradas seis *sprints* para esse primeiro semestre de 2023. Cada *sprint* com períodos pré-definidos de tempo, o que poderá ser utilizado para filtrar contribuições de integrantes em trabalhos para cada uma dessas seis *sprints*. Por exemplo, para a regra de consistência que exija a existência do arquivo “Documentacao/documento_de_arquitetura.md”, poderá ser associada uma *sprint* que determinará que até a data final da *sprint* esse documento deverá estar criado ou abrirá uma *issue* padronizada.

[2] https://www.conventionalcommits.org/en/v1.0.0-beta.4/

A partir da coleta dos dados dos repositórios do GitHub, resultados das validações das regras de consistências dos métodos avaliativos, contribuições de arquivos, linhas de código, *issues* e qualidade de código, a plataforma informará as contribuições no trabalho, por aluno e prazos de *sprints* para cada trabalho. Além disso, apresentará uma visualização do histórico de *commits* com filtro para alunos e *sprint*. Ademais, para cada repositório será apresentada a quantidade de *issues* abertas e fechadas por cada integrante de cada trabalho, também sendo possível filtrar essa informação por *sprint*.

Esta plataforma contará com uma opção para efetuar uma inspeção da qualidade dos códigos SonarQube[3] para cada repositório, essa inspeção calculará a qualidade do código do trabalho. O SonarQube foi escolhido em virtude da necessidade de suportar uma vasta gama de linguagens de programação, bibliotecas e *frameworks*, assegurando assim uma avaliação justa e abrangente dos códigos dos trabalhos submetidos pelos alunos. Outrossim, com objetivo de facilitar o uso por meio dos docentes, o sistema de autenticação e autorização utilizará o OAuth do próprio GitHub. Dessa forma, os docentes serão auxiliados na avaliação qualitativa e quantitativa das entregas de artefatos, documentação e qualidade de código dos trabalhos desenvolvidos pelas equipes de alunos.

O GitGrade se diferencia significativamente do Insights do GitHub, oferecendo funcionalidades avançadas e específicas que são essenciais para o contexto educacional, principalmente na avaliação de trabalhos interdisciplinares em cursos de Engenharia de Software. Enquanto o Insights do GitHub fornece uma visão geral básica das contribuições dos desenvolvedores, como número de *commits* e linhas de código adicionadas ou removidas, o GitGrade vai além, oferecendo métricas detalhadas e funcionalidades adaptadas às necessidades dos professores.

Primeiramente, o GitGrade introduz uma série de indicadores de contribuição não disponíveis no Insights, como os tipos de arquivos contribuídos, a qualidade da descrição dos *commits*, e a participação ativa em *issues*. Essas métricas fornecem uma compreensão mais profunda e detalhada do envolvimento dos alunos nos projetos, indo além da mera contagem de *commits* e linhas de código.

Além disso, o GitGrade facilita a organização e avaliação dos trabalhos por meio de métodos avaliativos, que são comparáveis a turmas, com regras de consistência e *sprints* pré-definidas. Essa estrutura permite aos professores avaliar de forma mais eficiente e organizada os repositórios de uma mesma turma, algo que o Insights do GitHub não oferece.

A plataforma também inclui a detecção de *force pushes*, uma funcionalidade crítica para entender as práticas de versionamento dos alunos, que o Insights não fornece. Além disso, a integração com ferramentas como o SonarQube para análise da qualidade do código adiciona

[3] https://www.sonarsource.com/products/sonarqube/

uma dimensão importante à avaliação, concentrando-se na qualidade do código, e não apenas na quantidade.

Por fim, a capacidade de abrir *issues* padronizadas automaticamente para regras de consistência não atendidas durante *sprints* é uma característica única do GitGrade, facilitando a comunicação e o acompanhamento de requisitos não cumpridos, algo que o Insights não é capaz de realizar.

FORA DO ESCOPO

Como o foco do sistema são as avaliações dos trabalhos feitos pelos alunos para as disciplinas de Trabalho Interdisciplinar, é importante destacar alguns pontos que não serão solucionados pelo *software* a ser desenvolvido. O primeiro ponto que não será tratado neste projeto é o gerenciamento das turmas e dos trabalhos das disciplinas na organização do GitHub, isso inclui o não tratamento de nenhuma demanda relacionada com criar e administrar repositórios por meio de uma organização pelo GitHub Classroom. Outro ponto recomendável destacar é que não será trabalhado neste *software* a integração de notas com a plataforma Canvas PUC Minas.

Em relação a pontos que a plataforma *web* que será desenvolvida neste projeto não cobrirá, será a avaliação de requisitos funcionais, ou seja, não será implementado nenhuma estratégia ou inteligência artificial que consiga verificar se os requisitos funcionais levantados para cada trabalho foram realmente codificados ou estejam coesos. Em vista da avaliação automática de requisitos não funcionais, as limitações serão por parte da ferramenta utilizada, que neste caso será a SonarQube, no qual efetuará inspeções da qualidade dos códigos, não se estendendo às funcionalidades além das oferecidas.

GESTORES, USUÁRIOS E OUTROS INTERESSADOS

Nome	José Laerte Pires Xavier Junior.
Qualificação	Coordenador das Unidades Curriculares de TCC.
Responsabilidades	Usuário da plataforma — cliente responsável pelo levantamento de necessidades e requisitos. É sua responsabilidade identificar e comunicar claramente as necessidades e requisitos do projeto para garantir que a plataforma atenda às suas expectativas e necessidades específicas.

Nome	Guilherme Gabriel Silva Pereira.
Qualificação	Aluno integrante do grupo responsável pelo desenvolvimento do TCC.

Responsabilidades	Responsável pela documentação e levantamento dos requisitos junto ao Coordenador das Unidades Curriculares de TCC. Além disso, com o apoio do outro Aluno integrante do grupo responsável pelo desenvolvimento do TCC, fará a modelagem dos diagramas da plataforma, definição da arquitetura, criação do *design* das interfaces de usuário do sistema e por fim a codificação de código com foco no *frontend* da plataforma.

Nome	Lucas Ângelo Oliveira Martins Rocha.
Qualificação	Aluno integrante do grupo responsável pelo desenvolvimento do TCC.
Responsabilidades	Responsável pela documentação e levantamento dos requisitos junto ao Coordenador das Unidades Curriculares de TCC. Além disso, com o apoio do outro Aluno integrante do grupo responsável pelo desenvolvimento do TCC, fará a modelagem dos diagramas da plataforma, definição da arquitetura, criação do *design* das interfaces de usuário do sistema e por fim a codificação de código com foco no *backend* da plataforma.

Nome	Docentes
Qualificação	Professores das disciplinas de Trabalho Interdisciplinar.
Responsabilidades	Responsáveis por analisar a documentação do projeto e utilizar a plataforma com intuito de validar o que está sendo desenvolvido. Isso, pois, serão os futuros usuários da plataforma.

LEVANTAMENTO DE NECESSIDADES

1. Os professores precisam se autenticar pelo GitHub. Devido à necessidade de verificar quais usuários terão acesso à plataforma, deverá haver integração com o GitHub OAuth. Esta integração detectará os *owners* da organização ICEI-PUC-Minas-PPLES-TI, com isso, liberando acesso à plataforma deste projeto.
2. Avaliação de artefatos customizados para cada oferta de disciplina. Os professores das disciplinas de Trabalho Interdisciplinar necessitam de auxílio para avaliarem com mais praticidade os artefatos de código e documentação dos trabalhos nos repositórios do GitHub. Contudo, o problema é que esses artefatos diferem para cada disciplina de Trabalho Interdisciplinar. Diante disso, a plataforma possibilitará cadastrar métodos avaliativos customizados para cada oferta de disciplina, nos quais cada método terá

regras de consistências para os artefatos que deverão ser entregues, possibilitando efetuar uma avaliação quantitativa das entregas.

3. Avaliação de entrega de artefatos por *sprints* em métodos avaliativos. As entregas dos trabalhos interdisciplinares de uma disciplina são divididas em *sprints*, nas quais os alunos devem entregar os artefatos em um período pré-definido. Com isso, os professores carecem de uma funcionalidade que permita filtrar as entregas dos artefatos nos repositórios em um período. A partir disso, a plataforma possibilita cadastrar em cada regra de consistência quais artefatos deverão ser entregues em cada *sprint*, possibilitando os professores filtrarem os artefatos entregues em cada *sprint* de cada trabalho e avaliarem as tarefas com mais praticidade.
4. Avaliação das entregas de um integrante de um trabalho. Os professores efetuam uma tarefa custosa de verificar manualmente todas as contribuições de um integrante em um repositório, a fim de atribuir notas para as tarefas da disciplina de Trabalho Interdisciplinar. Diante disso, a plataforma possibilitará verificar se todos os integrantes dos trabalhos estão participando ativamente de entregas, filtrando as entregas de um integrante por meio das regras de consistência dos métodos avaliativos, análise quantitativa de contribuições filtrando por *sprints*, além disso, também contará como contribuição o fechamento de *issues* por cada integrante.
5. Automação da avaliação da qualidade de código. No presente momento, os professores possuem uma carência de meios para avaliar de maneira equitativa a qualidade do código submetido pelos alunos em diferentes linguagens de programação, utilizando uma única ferramenta para essa análise. Com intuito de facilitar a avaliação estática da qualidade de código desenvolvido nos trabalhos, a plataforma possibilitará os docentes efetuarem uma análise qualitativa do código de cada trabalho ou integrante sem passar por problemas de configuração local dos trabalhos nem ter de analisar manualmente o código e seu fluxo.
6. Avaliação de trabalhos de forma geral. Os professores necessitam avaliar entregas de artefatos, contribuições de integrantes e qualidade de código, para julgar quais foram os melhores trabalhos de cada semestre. Contudo, esse é um processo muito trabalhoso e repetitivo para os docentes, por precisarem abrir cada repositório e efetuar essas avaliações manualmente. Diante disso, a plataforma apresentará informações quantitativas de *commits*, descrição dos *commits*, tamanho em caracteres da descrição dos *commits*, tipos de arquivos entregues, sendo possível filtrar por cada integrante e *sprint*, o que irá auxiliar os professores no julgamento desses trabalhos. Ademais, os professores também serão auxiliados pelos resultados da ferramenta de análise estática de código (SonarQube).
7. Abertura de *issues* padronizadas para soluções de problemas detectados pelos métodos avaliativos. Atualmente os professores possuem a trabalhosa tarefa de validar se algum artefato ou arquivo não foi entregue nos repositórios de cada trabalho e alertar os alunos.

À vista disso, a plataforma possibilitará cadastrar títulos e descrições para abertura de *issues* padronizadas para regras de consistências não seguidas na entrega de uma *sprint* do método avaliativo.

8. Detectar más práticas dos alunos no uso do Git para trabalhos interdisciplinares. Pelo fato da execução de *squashes* e *rebases* afetar a análise de contribuidores de repositórios Git, será necessário detectar quando essas operações forem efetuadas, além disso, a falta de união de *branches* ao fim de entregas também é considerada má prática pela dificuldade de busca de contribuições. Com isso, a plataforma deverá informar quando alunos efetuarem qualquer uma dessas operações em um repositório do trabalho interdisciplinar.

FUNCIONALIDADES DO PRODUTO

Necessidade: Os professores precisão se autenticar pelo GitHub	
Funcionalidade	**Categoria**
1. O professor poderá se autenticar na plataforma por meio das suas credenciais do GitHub.	Crítico.
2. O sistema verificará se a autenticação do usuário tem permissão de *owner* na organização ICEI-PUC-Minas-PPLES-TI.	Crítico.

Necessidade: Avaliação de artefatos customizados para cada oferta de disciplina. (Consistência de entrega)	
Funcionalidade	**Categoria**
1. O professor deverá associar o repositório de trabalho a um método avaliativo. (Para não ter que cadastrar toda vez um método avaliativo para cada repositório).	Crítico.
2. O professor poderá visualizar a lista de todos os repositórios de trabalhos interdisciplinares.	Crítico.
3. O professor poderá cadastrar métodos avaliativos para cada oferta de disciplina. Exemplo: “Método Avaliativo 001 — Trabalho Interdisciplinar: Aplicações Distribuídas 1/2023”.	Crítico.
4. O professor poderá desativar (exclusão lógica) um método avaliativo.	Útil.
5. O professor poderá vincular ao método avaliativo, regras de	Crítico

consistência pré-definidas.	
6. O professor, para cada método avaliativo, deverá cadastrar as regras de consistências para os artefatos (arquivos) e seus respectivos diretórios e *sprints* de entrega. Exemplos: Validação do CITATION.cff, existência de um vídeo de apresentação, README.md, etc.	Crítico.
7. O professor poderá adicionar validações extra de consistência que validem o conteúdo de um artefato cadastrado.	Útil.
8. O professor poderá atualizar nomes dos artefatos, diretórios dos arquivos e *sprint* de entrega de um método avaliativo.	Importante.
9. O professor poderá visualizar todas as regras de consistências cadastradas para um método avaliativo da oferta de uma disciplina.	Importante.
10. O professor terá a opção de definir que a validação de uma regra de consistência seja aplicada especificamente ao tipo de arquivo CITATION.cff.	Crítico.
11. A plataforma validará se a estrutura das regras do tipo *citation file* estão corretas.	Crítico.

Necessidade: Avaliação de entrega de artefatos por *sprints* em métodos avaliativos.	
Funcionalidade	**Categoria**
1. O professor poderá cadastrar *sprints* para cada método avaliativo.	Crítico.
2. O professor deverá relacionar quais artefatos (regras de consistência) deverão estar criados ou atualizados (no caso de código) ao fim de uma *sprint*.	Crítico.
3. O professor poderá visualizar quais foram os artefatos entregues e não entregues no prazo de uma *sprint* na página de cada trabalho.	Importante.
4. O professor visualizará a qual *sprint* cada artefato (regra de consistência) de um método avaliativo está relacionada.	Útil.

Necessidade: Avaliação das entregas de um integrante de um trabalho. (Análise Quantitativa)	
Funcionalidade	**Categoria**

1. O professor visualizará a lista de todos os integrantes que contribuíram anteriormente na página do repositório de um trabalho.	Crítico.
2. O professor poderá visualizar as contribuições de quantidade e porcentagem de *commits* de cada integrante na página de um trabalho, filtrar por *sprint* e ordenar por crescente ou decrescente.	Crítico.
3. O professor poderá visualizar as contribuições de quantidade de linhas alteradas de cada integrante na página de um trabalho, filtrar por *sprint* e ordenar por crescente ou decrescente.	Útil.
4. O professor poderá visualizar as contribuições de quantidade de *issues* fechadas por integrante na página de um trabalho, filtrar por *sprint* e ordenar por crescente ou decrescente.	Importante.
5. O professor poderá visualizar as contribuições de quantidade e porcentagem de arquivos alterados de cada integrante na página de um trabalho, filtrar por *sprint* e ordenar por crescente ou decrescente.	Útil.
6. O professor poderá visualizar a listagem de *commits* de cada integrante na página de um trabalho, filtrar por *sprint* e ordenar por crescente ou decrescente.	Útil.
7. O professor poderá visualizar a listagem de arquivos alterados de cada integrante na página de um trabalho, filtrar por *sprint* e ordenar por crescente ou decrescente.	Útil.
8. O professor visualizará os tipos de arquivos criados ou alterados por cada integrante na página de um trabalho, filtrar por sprint e ordenar por crescente ou decrescente.	Útil.
9. O professor poderá visualizar a qualidade da descrição dos *commits* de cada integrante na página de um trabalho, filtrar por *sprint* e ordenar por crescente ou decrescente.	Útil.

Necessidade: Automação da avaliação da qualidade de código. (Avaliação Qualitativa).	
Funcionalidade	**Categoria**
1. O professor poderá executar uma análise estática do código (diretório “./Codigo”) pelo SonarQube na página inicial de cada trabalho para cada *sprint*. Válido ressaltar que esta análise utilizará o último *commit* do repositório.	Crítico.

Necessidade: Avaliação de trabalhos de forma geral.	
Funcionalidade	**Categoria**
1. O professor visualizará na página inicial do repositório de um trabalho: à qual método avaliativo ele está relacionada e os seus contribuidores (integrantes do grupo que contribuíram).	Crítico.
2. O professor poderá filtrar todas as informações na página inicial de cada trabalho pelas *sprints* do método avaliativo que o trabalho está relacionado.	Útil.
3. O professor poderá visualizar a listagem dos artefatos entregues e não entregues na página de cada trabalho e filtrar por *sprint*.	Útil.
4. O professor poderá visualizar a quantidade e o histórico de *commits* na página inicial de cada trabalho.	Importante.

Necessidade: Abertura de *issues* padronizadas para soluções de problemas detectados pelos métodos avaliativos.	
Funcionalidade	**Categoria**
1. O professor deverá cadastrar um título padrão para abertura de *issue* para cada artefato cadastrado em um método avaliativo.	Crítico.
2. O professor poderá executar a abertura da *issue* padronizada para cada artefato do método avaliativo na página inicial de cada trabalho.	Crítico.

Necessidade: Auxílio aos alunos sobre boas práticas do uso do GitHub.	
Funcionalidade	**Categoria**
1. O sistema deverá detectar *squashes* e *rebases* no repositório *git* dos alunos e criar *issues* de alerta para informar que esta não é uma boa prática.	Útil.
2. O sistema deve detectar *branches* inativas que não passaram por *merge* até o fim de uma *sprint* e criar *issues* como forma de alerta aos alunos que esta não é uma boa prática.	Útil.

INTERLIGAÇÃO COM OUTROS SISTEMAS

Este projeto irá integrar com a Interface de Programação de Aplicação (API, do inglês *Application Programming Interface*) do GitHub de três modos para conseguir executar todas as funcionalidades. O primeiro modo será por meio do GitHub OAuth[4], o qual possibilitará efetuar as operações de autenticação dos usuários na plataforma e detectar se são *owners* da organização ICEI-PUC-Minas-PPLES-TI. O segundo modo será o principal para coleta de dados, no qual consistirá na execução de requisições utilizando o GitHub Octokit REST[5] com objetivo de buscar os dados necessários para as avaliações dos repositórios trabalhos. Por último, também será utilizado o GitHub Octokit REST para inserção de dados, com objetivo majoritário de abrir *issues* nos repositórios dos trabalhos.

Além disso, também haverá uma integração interna entre contêineres Docker[6] de imagens do SonarQube e SonarScanner CLI[7], o qual fornecerá as interfaces necessárias para executar análises estáticas dos códigos produzidos em repositórios dos trabalhos. Diante disso, será necessário executar um contêiner que dê suporte ao SonarQube inicialmente, para que por meio de linha de comando do SonarScanner CLI, também executando sobre um contêiner, inicie-se a avaliação dos códigos-fontes.

RESTRIÇÕES

Esta plataforma possui as seguintes restrições (requisitos não funcionais):

- A plataforma deverá estar disponível por meio da *web*;
- A plataforma deve ser responsiva para proporcionar o uso de todas as funcionalidades providas pelos requisitos funcionais para dispositivos móveis e *web*;
- A plataforma deve possuir testes unitários para os cálculos que não envolvem integração externa;
- A plataforma deverá rodar sobre um Docker-Compose[8];
- O SonarQube utilizado neste projeto deverá rodar em um Docker-Compose juntamente com o SonarScanner CLI;

[4] https://docs.github.com/en/apps/using-github-apps/connecting-with-third-party-applications
[5] https://octokit.github.io/rest.js/v20
[6] https://www.docker.com/
[7] https://github.com/SonarSource/sonar-scanner-cli
[8] https://docs.docker.com/compose/

- A plataforma deverá sincronizar e salvar os dados coletados da API do GitHub no Sistemas Gerenciadores de Banco de Dados (SGBD) MySQL[9];
- A sincronização das informações dos repositórios de trabalho nesta plataforma deverá ser executada manualmente por meio de botões na tela de interface do usuário, para não exceder o limite de requisições a API do GitHub[10];
- O *frontend* da plataforma deverá ser desenvolvido em React;
- O *backend* da plataforma deverá ser desenvolvido NodeJS[11] com o Mapeamento Objeto-Relacional (ORM, do inglês *Object-Relational Mapping*) SequelizeJS;
- A plataforma deverá seguir a arquitetura MVC (Modelo, Visão, Controlador);
- A comunicação com a API da plataforma deve seguir o padrão RESTful;
- A licença do sistema será *Creative Commons Attribution 4.0 International*;

DOCUMENTAÇÃO

- A plataforma deverá contar com o arquivo README.md no diretório raiz do repositório do GitHub contendo as informações e instruções necessárias para efetuar a configuração, execução local e implantação;
- Faz-se necessário também que o diretório "./Artefatos" contenha os seguintes diagramas que modelam a aplicação:
    - Diagrama de Entidade e Relacionamento (Modelo Lógico) que será utilizado para modelar os dados da plataforma e as relações entre esses dados. Esse diagrama serve para visualizar as entidades (objetos, conceitos, eventos) envolvidos em um sistema, bem como as relações entre eles. O modelo lógico de banco de dados será criado a partir deste diagrama, permitindo criar o banco de dados no MySQL com as tabelas e colunas apropriadas para armazenar as informações da plataforma;
    - Diagrama de Caso de Uso será responsável por descrever as interações entre a plataforma deste projeto e seus usuários ou outros sistemas. Ele mostrará as ações que um usuário ou sistema pode realizar em relação à plataforma em questão. Será feito com o objetivo descrever os requisitos funcionais da plataforma em termos de casos de uso, sendo as principais interações entre a plataforma e seus usuários;
    - Diagrama de Classe o qual será muito útil para descrever a estrutura de classes desta plataforma, pois, será um sistema seguindo o paradigma de orientação a objetos. Com isso, apresentará as classes e seus atributos, métodos e relações entre classes, com o objetivo de fornecer uma visão geral das classes e objetos

[9] https://dev.mysql.com/downloads/mysql/
[10] https://docs.github.com/pt/rest/rate-limit
[11] https://nodejs.org/en/

envolvidos na plataforma , permitindo que o desenvolvimento da plataforma seja mais compreensível sobre como as classes interagem entre si;
    - Diagrama de Componentes será utilizado para descrever a estrutura de componentes da plataforma e como interagem entre si para formar o sistema completo. Com o objetivo de ajudar os desenvolvedores a entender as dependências entre os componentes e como eles se integram para formar a plataforma;
    - Diagrama de Comunicação terá o papel de explicar as interações entre objetos da plataforma. Mostrará como os objetos envolvidos em um determinado processo comunicam entre si para realizar uma tarefa específica. Foi escolhido com o objetivo de fornecer uma visão geral das interações entre os objetos e como eles cooperam para realizar uma tarefa;
    - Diagrama de Estados será usado para descrever o comportamento de um objeto em na plataforma. Mostrará como o objeto se comporta em diferentes estados e como ele transita de um estado para outro. Terá o objetivo de fornecer uma visão geral do comportamento do objeto e como ele responde a diferentes eventos ou estímulos;
    - Diagrama de Implantação usado para descrever a implantação da plataforma em um ambiente de nuvem. Explicará como os componentes de *software* são implantados em diferentes servidores ou dispositivos e como eles interagem entre si. Tendo como objetivo fornecer uma visão geral da arquitetura física da plataforma e como ela é implantada em um ambiente de produção;
    - Diagrama de Pacotes descreverá a organização da plataforma em módulos lógicos ou pacotes, apresentando como os pacotes são organizados em camadas e como eles se relacionam entre si. Será feito para que todos possam entender como é a arquitetura da plataforma e como ela é dividida em diferentes partes funcionais.
    - Diagrama de Sequência terá o papel crucial de descrever a interação entre objetos da plataforma, mostrando a ordem que as mensagens são trocadas entre eles. Ele representará graficamente a sequência de eventos que ocorrem em um processo ou funcionalidade da plataforma, ilustrando como os objetos interagem entre si e como o sistema responde a determinados eventos. O objetivo do diagrama de sequência é fornecer uma visão geral do fluxo de controle e interação da plataforma, permitindo que os desenvolvedores entendam como o sistema funciona e como as classes e objetos interagem entre si.
- No diretório “./Artefatos” também haverá uma pasta contendo os protótipos de tela criados na ferramenta Figma;
- Deverá haver no diretório de “./Documentacao” do repositório do projeto o “DocumentodeViabilidade.md” que conterá qual caso previsto na Resolução de TCC I

que o trabalho de desenvolvimento se enquadra e descrição fundamentada de como as restrições e exigências da resolução estão sendo seguidas;

- Haverá o também o arquivo “DocumentoDoProjeto.pdf” de documentação do projeto no diretório “./Documentacao” contendo toda a documentação escrita e modelada da plataforma.
- Também no diretório “./Documentacao” terá este arquivo “DocumentoDeVisao.pdf” contendo a visão de negócio e projeto desta plataforma.
- Por fim, os arquivos e vídeo para divulgação estarão no diretório “./Divulgacao”.