Commit Explorer V2.0: Uma Solução para Extração e Avaliação de Commits e Códigos

Mateus F. Back[1], **Frank W. C. de Oliveira**[1], **Marcelo F. Terenciani**[1]

[1] Instituto Federal do Paraná – Campus Paranavaí
Paranavaí – PR – Brazil

mateusfback14@gmail.com.br, frank.willian@ifpr.edu.br

marcelo.terenciani@ifpr.edu.br

Abstract. *This paper presents the development experience and evaluation of Commit Explorer, an open-source tool for extracting, analyzing, and visualizing data from GitHub repositories, with a focus on educational contexts. The work is grounded in literature on software quality, static code analysis, and teaching practices supported by distributed version control systems. Data collection involved semi-structured interviews with Software Engineering instructors and the implementation of a functional prototype. The tool integrates a customized PMD configuration to assess code quality and provides interactive dashboards for metric visualization. Expected outcomes include supporting educators in monitoring students' coding practices, streamlining assessments, and fostering improved programming skills through structured feedback.*

Resumo. *Este trabalho apresenta a experiência de desenvolvimento e a avaliação do Commit Explorer, uma ferramenta voltada à extração, análise e visualização de informações de repositórios GitHub, com foco no contexto educacional. A pesquisa é fundamentada na literatura sobre qualidade de software, análise estática de código e práticas de ensino apoiadas por sistemas de controle de versão. A coleta de dados para o desenvolvimento da aplicação envolveu entrevistas semiestruturadas com docentes de Engenharia de Software e a implementação de um protótipo funcional. A ferramenta integra uma configuração do PMD para avaliar a qualidade do código e disponibiliza painéis interativos para visualização de métricas. Os resultados esperados incluem apoiar docentes no acompanhamento das práticas de programação dos estudantes, agilizar o processo avaliativo e promover a melhoria das habilidades de codificação por meio de feedback estruturado.*

1. Introdução

O uso de repositórios Git em esferas educacionais é uma prática comum para o ensino de programação e desenvolvimento de software. O GitHub, em particular, é adotado por docentes devido à sua capacidade de registrar, rastrear e organizar contribuições de estudantes em projetos desenvolvidos. A integração de desenvolvimento com Git favorece a aproximação com práticas reais de mercado, além de apresentar uma visão da evolução dos projetos ao longo do tempo [Orvalho et al. 2024]. No entanto, o processo de avaliação destes repositórios ainda é um desafio, principalmente em projetos maiores de turmas

grandes, a análise realizada de forma manual e subjetiva acarreta um alto custo de tempo para os professores.

Esse problema se intensifica à medida que o número de estudantes e repositórios cresce. Professores precisam lidar com um grande número de projetos simultaneamente, cada um com um histórico único de *commits*, estilos de escrita de código, estruturas de pastas e padrões de versionamento distintos. O tempo necessário para compreender as contribuições de cada aluno, verificar se seguem boas práticas, e identificar erros recorrentes se torna impraticável. Além disso, a ausência de critérios objetivos ou de padronização no processo avaliativo faz com que o julgamento da qualidade do código fique condicionado à percepção individual do avaliador. No contexto docente, esse tipo de avaliação manual e subjetiva torna-se um gargalo para a efetividade das práticas pedagógicas em disciplinas que utilizam o GitHub como instrumento avaliativo ou um meio de obter dados para aplicar suas métricas de avaliação.

Nesse cenário, ferramentas de análise estática como o PMD, SonarQube se apresentam como promissoras por oferecerem uma análise automática e diagnósticos objetivos sobre a qualidade do código. O PMD, em específico, identifica falhas comuns - chamadas de *code smells* - como variáveis não utilizadas, duplicações, complexidade excessiva. Essas falhas, a longo prazo, tendem a comprometer a manutenibilidade e legibilidade do software. Trabalhos recentes apontam que sua aplicação no ambiente educacional, quando combinada com feedback estruturado, tem potencial para melhorar o aprendizado dos estudantes e reduzir a carga de trabalho dos docentes [AlOmar et al. 2023]. Ainda assim, outros estudos destacam limitações de analisadores estáticos, como falhas induzidas por anotações ou falsos positivos, o que reforça a necessidade de uma configuração personalizada para contextos educacionais [Zhang et al. 2024].

Neste contexto, este trabalho descreve o desenvolvimento do Commit Explorer, uma ferramenta para extração, análise e visualização de *commits* e códigos armazenados em repositórios GitHub. A solução visa automatizar a aplicação de métricas de qualidade com o PMD, gerar relatórios visuais e facilitar o processo de avaliação em disciplinas práticas de programação. Por meio de uma API integrada a um dashboard interativo, a ferramenta permite aos docentes monitorar aspectos como frequência de *commits*, padrão de mensagens, presença de *code smells* e complexidade das alterações.

Diante da complexidade desse contexto e assumindo o risco de não atender às necessidades, o objetivo desse trabalho é desenvolver uma solução computacional que auxilie professores no processo de avaliação de atividades práticas de programação realizadas por meio do GitHub. Para isso, foi proposta uma ferramenta configurável com suporte à análise estática de código via PMD, que permite extrair, analisar e visualizar informações dos commits realizados pelos estudantes, oferecendo métricas e relatórios que facilitem o processo de correção e acompanhamento da qualidade do código entregue. Para alcançar esse objetivo, este trabalho busca contemplar os seguintes aspectos:

- Levantar necessidades e critérios utilizados por docentes na avaliação de atividades práticas com GitHub, por meio de entrevistas semiestruturadas;
- Integrar a API com repositórios GitHub para extração de *commits* e código;
- Propor uma configuração customizada do PMD adequada ao ambiente acadêmico;
- Avaliar boas práticas de codificação utilizando métricas como *code smells*, padrão

das mensagens de *commit* e frequência das contribuições;

- Desenvolver uma interface gráfica interativa para visualização das análises realizadas;
- Validar a ferramenta junto a docentes da instituição, por meio da aplicação prática e coleta de feedback qualitativo;
- Documentar a API e a interface para uso por professores, alunos e instituições;
- Registrar todo o processo de desenvolvimento e implementação em relatório técnico e artigo científico;
- Submeter os resultados obtidos em evento acadêmico da área.

Dessa forma, o presente trabalho está estruturado da seguinte forma: A Seção 2 apresenta a fundamentação teórica, abordando conceitos sobre controle de versão, qualidade de *software*, análise estática e métricas de código. A Seção 3 discute os principais trabalhos relacionados na área. A Seção 4 descreve a metodologia adotada para o desenvolvimento e validação da ferramenta. A Seção 5 apresenta o Commit Explorer, com foco em sua arquitetura e principais funcionalidades. A Seção 6 traz os resultados obtidos a partir dos testes realizados. Por fim, a Seção 7 apresenta as conclusões e sugestões para trabalhos futuros.

2. Fundamentação Teórica

O ensino de programação e engenharia de *software* passa por transformações à medida que novas ferramentas e práticas profissionais são incorporadas na esfera educacional. Nesse sentido, o uso de sistemas de controle de versão como o Git e plataformas colaborativas como o GitHub se tornaram centrais no ensino de disciplinas práticas. O Git é um sistema de versão distribuído criado por Linus Torvalds em 2005, que permite rastrear mudanças no código-fonte, colaborar em equipe e reverter alterações de forma direta e eficiente. Já o GitHub, por sua vez, oferece uma interface que integra os repositórios Git com funcionalidades como *issues*, *pull requests*, histórico de *commits* e ferramentas de integração continua, fazendo com que ele seja uma das principais plataforma de contribuição de projetos de software, inclusive em ambientes educacionais. [Chacon and Straub 2021]

A utilização do GitHub no contexto acadêmico vai além da entrega de trabalhos, uma vez que ele permite a observação do comportamento de desenvolvimento dos estuda dantes, suas práticas de versionamento, frequência de *commits*, clareza nas mensagens e respeitos às boas práticas de engenharia de *software* e programação. Conforme Martin (2008), a granularidade dos *commits* e legibilidade das mensagens são aspectos essenciais da cultura do código limpo *Clean Code*), pois promovem rastreabilidade, colaboração e facilitam a revisão. *Commits* muito extensos ou com mensagens genéricas dificultam a manutenção e compreensão da evolução do projeto, enquanto contribuições pequenas, frequentes e bem documentadas são preferidas por equipes profissionais [Martin 2008]

Do ponto de vista técnico, a qualidade do código entregue pode ser avaliada por meio de métricas extraídas por ferramentas de análise estática. Uma das ferramentas amplamente utilizadas em ambientes educacionais é o PMD, um analisador de código *open source* que detecta automaticamente más práticas de programação conhecidas como *code*

smells [Kaur and Nayyar 2020]. Essas más práticas incluem desde a presença de variáveis não utilizadas até estruturas com alta complexidade ciclomática — métrica proposta por McCabe para medir a complexidade de um método com base no número de caminhos independentes de execução [McCabe 1976]. Quanto maior a complexidade ciclomática, maior a probabilidade de ocorrência de erros e mais difícil se torna o teste e a manutenção do código.

Estudos recentes têm evidenciado o uso de ferramentas como o PMD para fins educacionais. AlOmar et al. (2023) mostraram que o uso da análise estática com *feedback* automatizado promove melhorias no código dos estudantes e fortalece o processo de aprendizagem, desde que acompanhado de orientações instrucionais claras. No entanto, o mesmo estudo destaca que a aplicação direta de regras industriais pode gerar ruído no contexto acadêmico, sendo necessário calibrar as regras para o nível de maturidade dos alunos [AlOmar et al. 2023]. Apesar disso, estudos recentes apontam limitações nos analisadores estáticos, como a incidência de falsos negativos ou falhas provocadas por anotações incorretas no código [Zhang et al. 2024], evidenciando a importância de configurar essas ferramentas de forma personalizada para atender aos objetivos pedagógicos específicos de cada curso.

Nesse mesmo contexto, o conceito de *refactoring* foi introduzido como técnica para melhorar gradualmente o design interno do *software* sem alterar seu comportamento externo, combatendo *code smells* e mantendo o código limpo ao longo do tempo [Fowler 1999]. Ensinar boas práticas de refatoração e integrá-las ao processo educacional exige ferramentas que apoiem o diagnóstico automatizado dessas más práticas.

Além dos aspectos técnicos, é preciso compreender o uso do Git e GitHub como ferramentas pedagógicas alinhadas à avaliação formativa. Segundo Nicol e Macfarlene-Dick (2006), a avaliação formativa eficaz depende de feedbacks frequentes, imediatos e orientados a progresso do aluno, características que podem ser potencializas quando se analisa a evolução dos repositórios de código ao longo do tempo. [Nicol and Macfarlane-Dick 2006] O controle de versão, neste caso, serve como registro do processo de aprendizagem, permitindo que docentes acompanhem não apenas o resultado, mas o caminho, as decisões tomadas e o desenvolvimento das habilidades ao longo do projeto.

Assim, ao se propor o uso de uma ferramenta como o Commit Explorer, parte-se da premissa de que a combinação entre controle de versão, análise estática customizada e visualização de métricas contribui não apenas para a automatização do processo avaliativo, mas também para a melhoria do aprendizado dos alunos, desde que respeitados os princípios pedagógicos do ensino ativo, da transparência e da adequação às necessidades do público-alvo.

3. Trabalhos Relacionados

A seleção dos trabalhos relacionados foi feita a partir de buscas exploratórias no Google Scholar, utilizando termos como ”*automatic code feedback*”, ”*Git educational tools*”e ”*commit analysis for education*”. Foram priorizadas ferramentas com foco educacional ou que se destacam na análise de código em repositórios Git, como o GitSEED e o GitHub Classroom. Além disso, incluiu-se o SonarQube por sua relevância técnica na avaliação de qualidade de código, mesmo não sendo uma ferramenta voltada diretamente ao ensino.

A avaliação automatizada de código tem sido tema recorrente em propostas voltadas ao ensino de programação e engenharia de software. Ferramentas como GitSEED e GitHub Classroom buscam fornecer feedback automatizado e contínuo aos alunos e reduzir a carga de correção por parte dos professores.

Dessa forma, a Tabela 1 apresenta um comparativo entre os principais trabalhos relacionados ao uso de ferramentas Git para avaliação de trabalhos, evidenciando seus objetivos, mecanismos de *feedback*, foco no uso educacional e capacidade de explorar o histórico de *commits*.

Tabela 1. Comparativo entre ferramentas de apoio à avaliação com Git

Trabalho	Objetivo Principal	Análise de Commits	Sugestões ao Aluno	Foco nos Professores	Tecnologias
GitSEED	Feedback automatizado via GitLab com integração de análise de código	Parcial	Sim	Sim	GitLab, SafeExec
Github Classroom	Gerenciar tarefas e corrigir exercícios via Github	Não	Parcial	Sim	Github Actions
SonarQube	Analisar qualidade de código em projetos com foco em manutenção contínua	Sim	Não	Não	Java, Sonar Analyzers, PostgreSQL, Web UI

Diante dos dados apresentados, nota-se que embora as principais ferramentas existente ofereçam recursos válidos na avaliação e ensino da programação, nenhuma delas se propõe a resolver as necessidades dos docentes do Instituto Federal do Paraná - Campus Paranavaí (IFPR), realizar uma exploração no histórico de *commits*, validando*code smells* e periodicidade, por exemplo. À luz desse trabalho, surge o Commit Explorer, visando sanar os objetivos apresentados.

4. Metodologia

Quanto à metodologia utilizada para a abordar a pesquisa e o desenvolvimento, classifica-se de natureza incremental e interativa [Bezerra 2007], com o foco no desenvolvimento de uma ferramenta educacional voltada à análise de *commits* em repositório GitHub. Diante dessa proposta, foram realizadas três entrevistas semiestruturadas [Fraser and Gondim 2004] com professores do colegiado de Engenharia de Software no IFPR - Campus Paranavaí. Essas entrevistas foram realizadas durante o mês de março de 2025, para identificar os critérios utilizados na correção de atividades práticas via GitHub e suas necessidades específicas.

A partir do levantamento de necessidades e requisitos, foi desenvolvida uma API REST utilizando Java com Spring Boot, com a capacidade de acessar repositórios do GitHub, extraindo os históricos de *commits* com base em uma data especificada e repositório. Após a extração a API aplica métricas de qualidades por *commit*, utilizando como base a configuração personalizada do PMD, adaptada as necessidades dos professores e disponibiliza os resultados no dashboard web, implementado com React.js.

Quanto à avaliação da efetividade e aceitação da ferramenta, foi realizada uma análise qualitativa com os docentes entrevistados e com os alunos que utilizaram o sistema durante o desenvolvimento por meio de formulários que buscarão verificar a percepção dos usuários quando a usabilidade a interface, clareza das métricas e efetividade na avaliação docente.

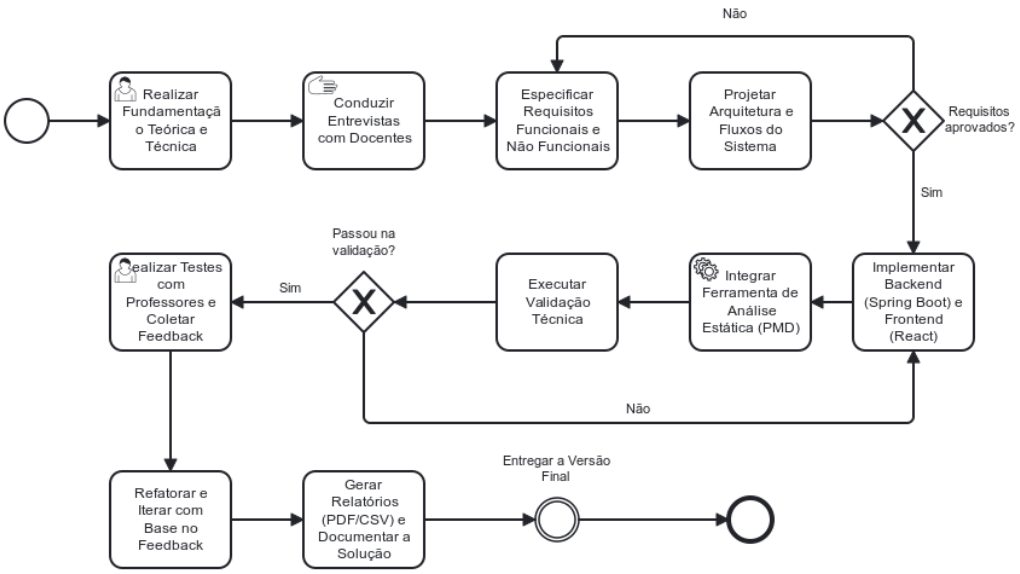

Figura 1. Visualização da Metodologia

4.1. Entrevistas Semiestruturadas e Dados Levantados

As entrevistas realizadas com docentes da área de Engenharia de Software do Instituto Federal do Paraná - Campus Paranavaí evidenciaram padrões relevantes sobre o uso do GitHub na avaliação de projetos acadêmicos. Os professores relataram o uso do repositório principalmente em disciplinas práticas, como Programação Web, Desenvolvimento para Dispositivos Móveis e Construção de Software. Ao discutir os critérios de avaliação, os entrevistados destacaram como aspectos importantes: a frequência e coesão dos *commits*, a clareza das mensagens, a organização do histórico e a utilização adequada de *branches*. Um dos docentes mencionou que *commits* muito extensos ou com baixa frequência pode indicar plágio, ou desorganização, enquanto registros pequenos e frequentes tornam o processo de correção mais transparente.

As principais dificuldades relatadas dizem respeito à ausência de padronização nos repositórios, à dificuldade de rastrear a autoria individual em projetos colaborativos e à sobrecarga de trabalho causada pelo grande volume de *commits* em turmas com maior quantidade de alunos. Também foi mencionada a limitação em acessar informações que permitam compreender como o raciocínio do aluno evoluiu ao longo do tempo.

Sobre métricas que poderiam apoiar a avaliação, os docentes mencionaram a necessidade de ferramentas que identifiquem *code smells*, avaliem a frequência de *commits* e a clareza nas mensagens, e apontem a autoria e a progressão do trabalho. Além disso, foi sugerida a possibilidade de cruzar essas informações com as etapas de um fluxo de trabalho ou issue tracker, como forma de verificar a efetividade das contribuições realizadas.

Todos os participantes demonstraram interesse em utilizar uma ferramenta automatizada, desde que sua usabilidade fosse simples e oferecesse visualizações claras, como gráficos, dashboards e quadros comparativos entre projetos. Um dos professores destacou a importância de uma representação visual que facilite identificar padrões e lacunas recorrentes, tornando a avaliação mais objetiva e transparente.

5. Proposta de Solução

O desenvolvimento do Commit Explorer foi conduzido a partir das necessidades mapeadas nas entrevistas com docentes do curso de Engenharia de Software do IFPR – Campus Paranavaí. Nessas conversas, ficou evidente a importância de uma ferramenta que fosse capaz de importar tanto repositórios públicos quanto privados do GitHub, analisar a frequência e granularidade dos *commits*, identificar más práticas recorrentes no código e, ao mesmo tempo, manter a rastreabilidade da autoria em projetos colaborativos. Também se destacou a demanda por visualizações claras, em forma de linha do tempo ou gráficos comparativos entre projetos.

Outro ponto recorrente foi a necessidade de correlacionar as contribuições dos alunos com tarefas registradas em *issues*, permitindo verificar a coerência entre o que foi planejado e o que efetivamente foi implementado. Para apoiar o professor na detecção de situações atípicas, como longos períodos de inatividade ou possíveis casos de plágio, a ferramenta deve emitir alertas automáticos sempre que padrões incomuns forem identificados. Esses requisitos refletem a expectativa de um sistema que não apenas reduza a carga manual do docente, mas que também forneça insumos objetivos e auditáveis para a avaliação prática do aprendizado.

No que diz respeito a requisitos não funcionais, os professores enfatizaram a importância de uma interface simples e acessível, que facilite a interpretação dos dados sem exigir grande esforço técnico. A arquitetura, concebida em um estilo de desenvolvimento orientado a domínio, privilegia a separação entre lógica de negócio e tecnologia, garantindo modularidade, rastreabilidade e flexibilidade para futuras integrações com outras plataformas educacionais.

5.1. Modelo de Dados Relacional

Com base nas funcionalidades previstas e nas análises realizadas, foi elaborado um modelo entidade-relacionamento (DER) que estrutura os dados essenciais do projeto. O modelo contempla entidades como repositórios, *commits*, autores e análises, organizadas para garantir integridade, rastreabilidade e desempenho nas consultas.

Contemplando também entidades específicas para o processo avaliativo, como *feedback_analise*, *indicadores_feedback* e *pontos_analise*, que permitem consolidar métricas, registrar *feedbacks* automáticos e associar evidências objetivas ao desempenho dos usuários avaliados. Fator que favorece a rastreabilidade das avaliações ao longo do tempo e possibilita futuras extensões, como o suporte a novas métricas.

O modelo apresentado na Figura 2 representa apenas as entidades e seus relacionamentos principais, destacando as chaves primárias e estrangeiras de cada tabela, sem detalhar todos os atributos internos. Essa abordagem prioriza a compreensão estrutural do sistema e facilita a visualização das dependências entre os componentes do banco de dados.

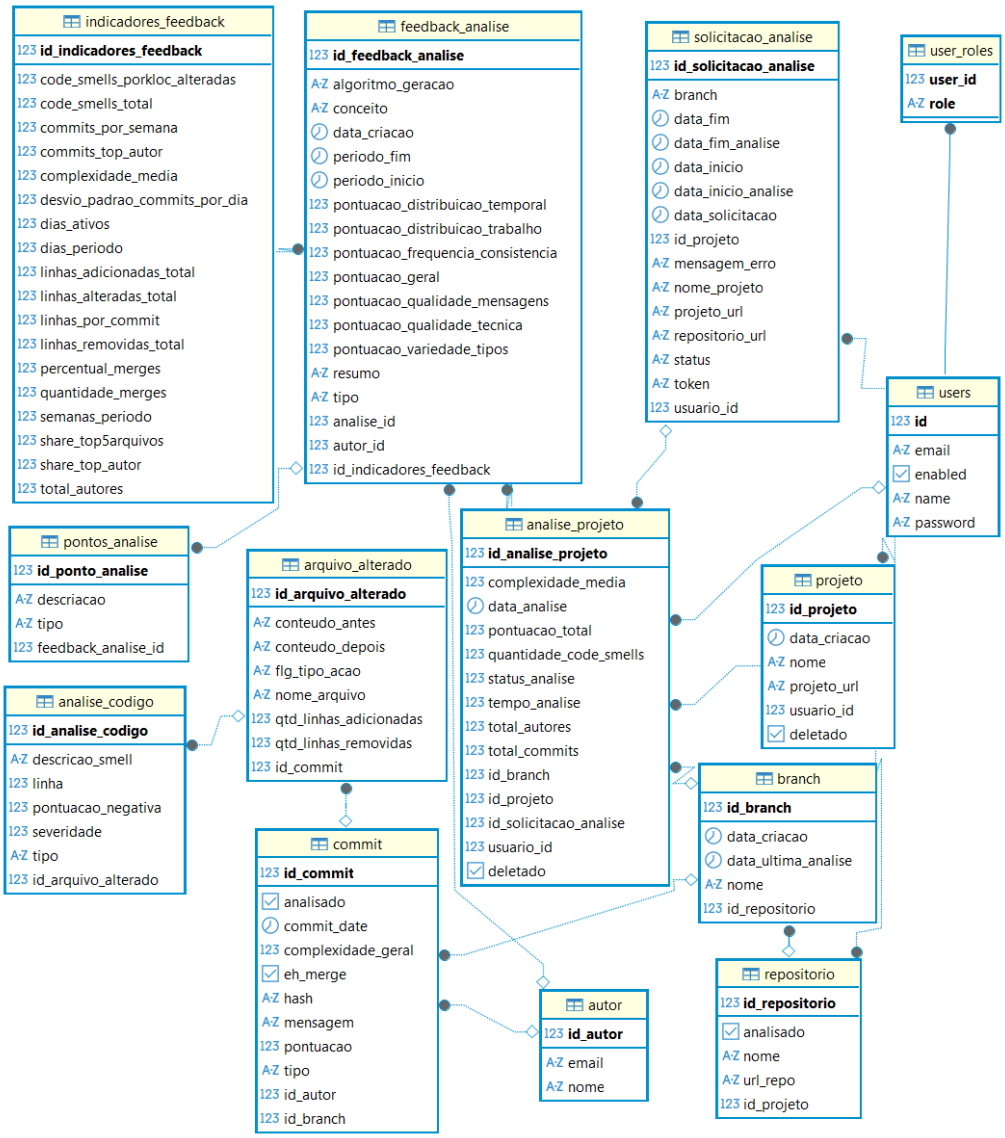

Figura 2. Modelo Entidade-Relacionamento (DER) do Commit Explorer

6. Avaliação/Discussão dos Resultados

Com base na fundamentação teórica e nos objetivos, esta seção visa apresentar os principais resultados obtidos até o momento. O desenvolvimento da ferramenta foi guiado por necessidades práticas identificadas por meio das entrevistas semiestruturadas com docentes do Instituto Federal do Paraná – Campus Paranavaí. Esses dados permitiram validar a relevância do problema em questão, levantar requisitos técnicos e orientar ao desenvolvimento de uma prova de conceito funcional. Além disso, o processo de desenvolvimento inicial possibilitou refletir sobre os desafios técnicos da integração com o GitHub, da análise automatizada de *commits* e da extração de métricas significativas para o ambiente educacional. Sendo assim, serão detalhadas as percepções obtidas nas entrevistas, os requisitos mapeados, a arquitetura implementada e os primeiros resultados da aplicação.

6.1. Forma de Avaliação Acadêmica

A avaliação dos repositórios segue um modelo adaptado para o contexto educacional, incorporando práticas relevantes ao processo de ensino-aprendizagem. Os critérios, definidos a partir de entrevistas com docentes e da literatura analisada, abrangem seis dimensões principais: frequência e consistência (25%), que mede a cadência dos commits e a regularidade ao longo do tempo; qualidade das mensagens (20%), que avalia clareza, especificidade e aderência a padrões como o *Conventional Commits*; variedade e organização de tipos (15%), que verifica a diversidade entre funcionalidades, correções, testes e refatorações; distribuição de trabalho (15%), que observa a colaboração equilibrada entre membros em projetos em grupo; distribuição temporal (10%), que analisa a disciplina de trabalho em diferentes horários; e qualidade técnica (15%), que considera métricas de *code smells*, complexidade e manutenibilidade.

Cada uma dessas dimensões corresponde a indicadores específicos implementados na ferramenta. Por exemplo, a frequência e consistência são calculadas a partir da média de *commits* por semana, dias ativos e intervalos máximos sem contribuições; já a qualidade das mensagens é avaliada por meio de expressões regulares que identificam descrições curtas, genéricas ou aderentes a padrões consolidados. A variedade e organização de tipos é verificada pela distribuição das categorias de *commits*, enquanto a distribuição de trabalho considera tanto a concentração de alterações em cada autor quanto o equilíbrio entre os colaboradores. A distribuição temporal observa a concentração de *commits* em determinados horários e, por fim, a qualidade técnica é apurada pela integração com o PMD, que identifica problemas de padronização, *code smells* e manutenibilidade. Essa estrutura garante que a avaliação vá além da simples contagem de contribuições, contemplando também aspectos qualitativos e pedagógicos do desenvolvimento.

A cada critério é atribuída uma pontuação que compõe a nota final, classificada em quatro níveis: A (85–100), B (70–84), C (55–69) e D (0–54). Além da nota, o sistema gera *feedback* automático com aspectos positivos e sugestões de melhoria, favorecendo tanto a autoavaliação pelos estudantes quanto a objetividade do processo avaliativo pelos docentes.

6.2. Funcionalidades Implementadas

A versão atual do Commit Explorer é plenamente funcional e já está sendo aplicada em contextos acadêmicos. O projeto é capaz de importar repositórios diretamente do GitHub e realizar análises por commit, identificando autoria, frequência de alterações, estrutura das mensagens e qualidade dos arquivos modificados.

A integração com o PMD permite detectar problemas recorrentes no código-fonte em Java, como más práticas de programação e ausência de padronização estrutural. Dessa forma, torna-se viável avaliar a granularidade das contribuições feitas, destacando não apenas a quantidade, mas também a qualidade e clareza das alterações realizadas por cada desenvolvedor.

Os dados extraídos incluem total de *commits*, autores únicos, problemas detectados e outros indicadores relevantes para o processo avaliativo. As Figuras 3 e 4 apresentam a interface da aplicação, que consolida essas informações em um *dashboard* web interativo.

Além do suporte técnico, a ferramenta contribui pedagogicamente ao oferecer relatórios claros e objetivos que auxiliam os docentes na avaliação de projetos em grupo e no acompanhamento individual dos estudantes. Para os discentes, o acesso às métricas e ao *feedback* estruturado possibilita a autoavaliação contínua, estimulando boas práticas de versionamento e o desenvolvimento de competências profissionais relacionadas ao trabalho colaborativo em equipes de software.

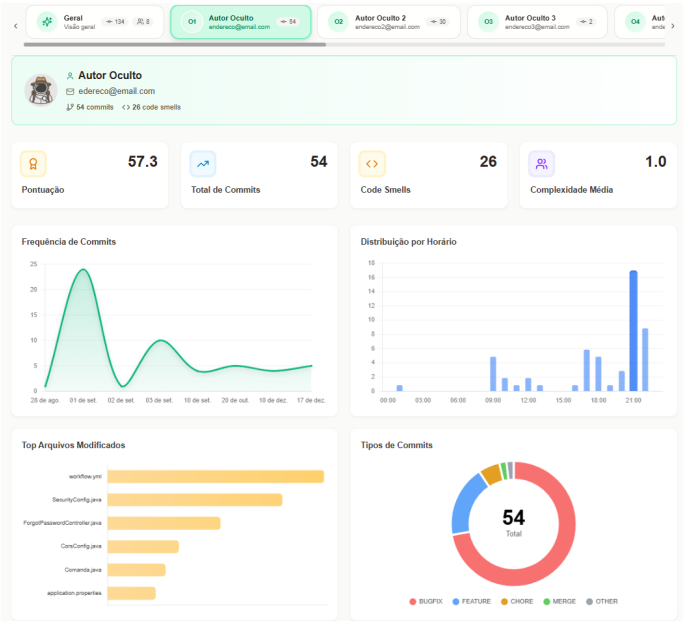

Figura 3. Página de análise de repositório do Commit Explorer V2.0 (Informações de Commits)

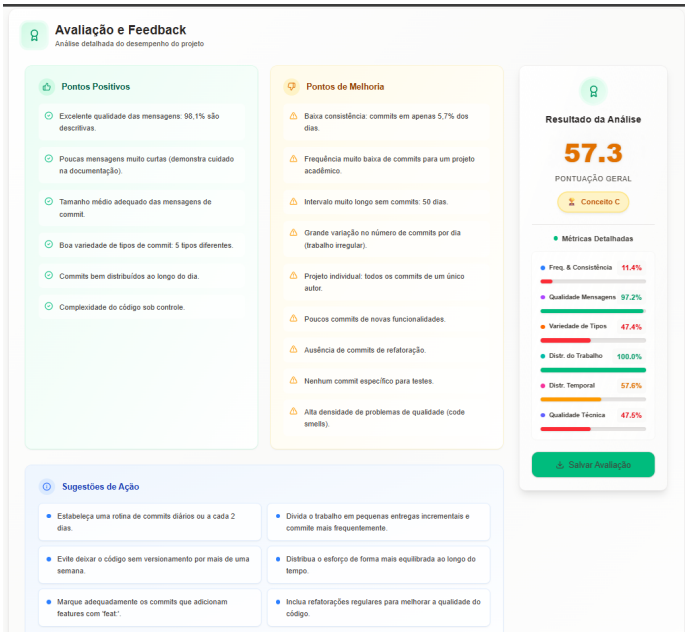

Figura 4. Página de análise de repositório do Commit Explorer V2.0 (Feedback Interativo)

6.3. Validação com Docentes

Além das entrevistas iniciais, a ferramenta foi validada em atividades do projeto integrador do curso de Engenharia de Software, permitindo constatar sua efetividade na análise de repositórios acadêmicos, bem como a clareza das métricas e painéis apresentados. Essa etapa prática evidenciou que a solução pode ser integrada ao cotidiano das disciplinas, ampliando a capacidade de acompanhamento por parte dos professores.

Os docentes também apontaram melhorias necessárias, como a unificação de múltiplos repositórios, correção de inconsistências em análises, criação de perfis específicos para professores, inclusão de opção para exclusão de projetos e ampliação dos recursos de autenticação. Essas observações reforçam a relevância da ferramenta no apoio às práticas pedagógicas e indicam caminhos concretos para seu aperfeiçoamento, de modo a torná-la mais robusta e aderente às demandas institucionais.

7. Conclusão e Trabalhos Futuros

Este trabalho apresentou a experiência de desenvolvimento da ferramenta Commit Explorer, que tem como objetivo apoiar a análise de repositórios GitHub em ambientes educacionais. A partir das entrevistas realizadas com docentes e do levantamento bibliográfico, foi possível identificar pontos essenciais para a avaliação de projetos, como a frequência e granularidade dos *commits*, clareza das mensagens e qualidade do código entregue.

A implementação da ferramenta foi concluída, integrando extração de dados via JGit, análise de métricas com o PMD e visualização das informações por meio de um *dashboard web* interativo. A solução já está em uso em contextos acadêmicos, demonstrando viabilidade técnica e utilidade prática, especialmente em turmas com grande volume de entregas e projetos em grupo. Ao disponibilizar métricas objetivas e *feedback* estruturado, o Commit Explorer contribui tanto para o trabalho docente quanto para a autoavaliação dos estudantes, estimulando melhores práticas de versionamento e desenvolvimento colaborativo.

Existem também possibilidades a serem exploradas adiante, dentre elas, destaca-se a criação de um histórico de avaliações por docente, facilitando o acompanhamento de análises anteriores e o resgate de informações ao longo dos semestres. A ampliação da gamificação, com níveis, insígnias e indicadores visuais vinculados ao desempenho, também surge como alternativa para estimular a participação dos alunos. Outra frente de interesse envolve o suporte a linguagens além do Java, como JavaScript, HTML e CSS, ampliando o uso da ferramenta em disciplinas com foco em desenvolvimento web.

Ainda pensando na flexibilidade da aplicação, a personalização das métricas por projeto ou disciplina aparece como ponto relevante, permitindo ajustar os critérios avaliativos conforme os objetivos de cada curso. Também está prevista a realização de novas avaliações com outros docentes e alunos, ampliando a validação prática da ferramenta. Por fim, uma integração com plataformas de ensino, como o Moodle, poderia facilitar a consolidação dos dados e sua utilização em processos formais de avaliação. Essas direções ampliam o escopo da proposta e apontam caminhos possíveis para consolidar o Commit Explorer como uma solução robusta, flexível e aderente às necessidades reais do contexto acadêmico.

Disponibilidade da Ferramenta e Código-Fonte

A ferramenta Commit Explorer está disponível publicamente em: `https://app.commitexplorer.com`

O código-fonte do *backend* encontra-se em: `https://github.com/mateusback/commitExplorer`

O código-fonte do *frontend* encontra-se em: `https://github.com/mateusback/commit-explorer-f`

Referências

AlOmar, E. A., AlOmar, S. A., and Mkaouer, M. W. (2023). On the use of static analysis to engage students with software quality improvement: An experience with pmd. *arXiv preprint arXiv:2302.05554*. Available at `https://doi.org/10.48550/arXiv.2302.05554`.

Bezerra, E. (2007). *Princípios de Análise e Projeto de Sistemas com UML*. Elsevier Editora Ltda., Rio de Janeiro, 7ª reimpressão edition. Inclui bibliografia. 1. Métodos orientados a objetos (Computação). 2. UML. 3. Análise de sistemas. 4. Projeto de sistemas.

Chacon, S. and Straub, B. (2021). *Pro Git*. Apress.

Fowler, M. (1999). *Refactoring: Improving the design of existing code*. Addison-Wesley.

Fraser, M. T. D. and Gondim, S. M. G. (2004). Da fala do outro ao texto negociado: discussões sobre a entrevista na pesquisa qualitativa. *Paidéia*, 14(28):139–152.

Kaur, A. and Nayyar, R. (2020). A comparative study of static code analysis tools for vulnerability detection in c/c++ and java source code. *Procedia Computer Science*, 167:3214–3223. Available at `https://doi.org/10.1016/j.procs.2020.04.217`.

Martin, R. C. (2008). *Clean code: A handbook of agile software craftsmanship*. Pearson Education.

McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320.

Nicol, D. J. and Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in higher education*, 31(2):199–218.

Orvalho, P., Janota, M., and Manquinho, V. (2024). Gitseed: A git-backed automated assessment tool for software engineering and programming education. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference (SIGCSE Virtual 2024)*, pages 165–171. Association for Computing Machinery. Available at `https://doi.org/10.1145/3649165.3690106`.

Zhang, H., Pei, Y., Liang, S., and Tan, S. H. (2024). Understanding and detecting annotation-induced faults of static analyzers. *Proceedings of the ACM on Software Engineering*, 1(FSE):722–744. Article No.: 33. Available at `https://doi.org/10.1145/3643759`.