UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

GILMAR GOMES DO NASCIMENTO

O USO DE LOGS NO ENSINO DE PROGRAMAÇÃO

CURITIBA

2025

GILMAR GOMES DO NASCIMENTO

O USO DE LOGS NO ENSINO DE PROGRAMAÇÃO

USING LOGS TO SUPPORT PROGRAMMING EDUCATION

Proposta de Dissertação apresentado como requisito para obtenção do título de Mestre em Ciência da Computação do Mestre em Computação Aplicada da Universidade Tecnológica Federal do Paraná.
Orientador(a): Prof. Dr. Laudelino Cordeiro Bastos
Coorientador(a): Profa. Dra. Maria Claudia Figueireido Pereira Emer

CURITIBA

2025

RESUMO

No desenvolvimento de software, é comum utilizar métricas para avaliar a qualidade do código e o tempo despendido em sua produção. No ambiente profissional, essa análise ocorre principalmente por meio de *pull requests*, enquanto no contexto acadêmico, a avaliação costuma ser feita por meio de envios pontuais (como e-mails ou sistemas de gestão de aprendizagem - LMS). Embora os LMS registrem *logs* de atividades e eventos, há uma lacuna na coleta estruturada de dados sobre o processo de aprendizagem em programação, especialmente em ambientes como laboratórios de informática. Este trabalho propõe a criação de um *dataset* com registros (logs) do processo de desenvolvimento de código, capturando o rastro da aprendizagem em tempo real. Diferentemente de abordagens convencionais, que dependem de Sistemas de Gestão de Aprendizagem, a coleta será realizada por meio de um *plugin* integrado a um editor de código amplamente utilizado, voltado tanto para programação quanto para documentação. Espera-se fornecer uma base de dados sobre o ensino-aprendizagem de programação que possa auxiliar na identificação de dificuldades dos estudantes, na personalização do ensino e no aprimoramento das práticas pedagógicas.

Palavras-chave: ensino de programação; log; telemetria; logging.

ABSTRACT

In software development, it is common to use metrics to assess the quality of code and the time spent producing it. In Education is not default, however, this project will help to obtain new informations to recognize standard and metrics. This work proposes the creation of a *dataset* with records (logs) of the code development process, capturing the learning trail in real time. Unlike conventional approaches, which rely on Learning Management Systems, the collection will be carried out using a *plugin* integrated with a widely used code editor, aimed at both programming and documentation. The goal of this paper is to provide a detailed database for future analyses, contributing to the study of programming teaching and learning.

Keywords: programming education; log; telemetry; logging.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Enquanto o ensino tradicional foca na leitura e escrita de textos, os cursos de programação introduzem uma nova de forma de pensamento: os algoritmos. Estes, que são sequências lógicas interpretadas por humanos e máquinas. Essa disciplina, presente em cursos de áreas tecnológicas, é “considerada desafiadora para alunos e professores"(Raabe; Silva, 2005), pois exige o desenvolvimento de uma lógica formal.

Ainda segundo (Raabe; Silva, 2005), as dificuldades no aprendizado de programação podem ser categorizadas em três grupos distintos, que vão além da simples aptidão individual: Problemas de natureza didática (relacionados aos métodos de ensino), cognitiva(relacionados aos desafios de raciocínio lógico e abstração) e afetiva(relacionados à motivação, ansiedade e confiança do aluno). Essa categorização sistêmica ajuda a entender que o desafio requer iniciativas diferentes.

No contexto educacional, a avaliação é entendida como uma etapa ao processo de ensino e aprendizagem, o que a torna um tema permanente de reflexão e estudo entre especialistas da área, que buscam constantemente refinar suas práticas para alcançar melhores resultados (Lima *et al.*, 2022)

A avaliação no curso técnico é um processo bifronte, contemplando tanto a avaliação da aprendizagem do estudante quanto a avaliação do sistema educacional como um todo. No que tange à aprendizagem, o projeto pedagógico estabelece que seu objetivo central é a progressão do discente para o alcance do perfil profissional almejado. Nesse sentido, a Resolução que rege o curso determina que esse processo “é contínua e cumulativa, com prevalência dos aspectos qualitativos sobre os quantitativos, bem como dos resultados ao longo do processo sobre os de eventuais provas finais"(Educação, 2020). Paralelamente, a avaliação do sistema, referente ao rendimento acadêmico, deve ser realizada para cada disciplina individualmente, considerando de forma simultânea e obrigatória a frequência do aluno e seu aproveitamento na aquisição de conhecimentos (Educação, 2020).

O ensino de programação é uma atividade multidisciplinar que demanda uma práxis pedagógica apurada – ou seja, uma constante reflexão que integre teoria e prática em sala de aula. Uma estratégia comum é utilizar exemplos do cotidiano para estimular a resolução de problemas por meio de algoritmos. Contudo, avaliar a qualidade desses algoritmos exige ir além da verificação funcional.

Esses algoritmos, definidos como procedimentos computacionais que transformam entradas em saídas, geram registros conhecidos como logs. Mas como esses logs podem ser utilizados para transformar o ensino e a aprendizagem? Este texto explora a interseção entre tecnologia e educação, mostrando como a análise de logs pode oferecer perspectivas valiosas para personalizar o ensino e identificar dificuldades dos alunos.

Essas interações, além de transformar entradas em saídas, geram logs. De acordo com Sah (2002, apud (Clemente, 2008)), “log é definido como um conjunto de registros com marcação temporal, que suporta apenas inserção, e que representa eventos que aconteceram em

um computador ou equipamento de rede". Além disso, segundo (Gu *et al.*, 2022), “um log geralmente contém um grande número de entradas, também conhecidas como mensagens de registro". Em geral, o nível do log, seu conteúdo e o local adequado para declará- lo são decisões tomadas pelos desenvolvedores durante a criação do software.

O processo de ensino pode ser compreendido analogamente a uma tecnologia que pode ser atualizada, adaptada ou aplicada em diferentes contextos. No entanto, para estimar o tempo de produção de um código, métodos tradicionais de marcação de tempo são insuficientes, sendo necessárias métricas adicionais. Nesse sentido, o armazenamento de logs permite coletar informações detalhadas sobre eventos em dispositivos. Como destacado por (Nguyen; Ha; Chen, 2023), “esse recurso de previsão possibilita que instrutores identifiquem alunos com dificuldades desde o início e ofereçam intervenções direcionadas para apoiar sua jornada de aprendizado”. Ademais, a análise de logs pode revelar padrões de comportamento e envio, fornecendo, assim, dados valiosos não apenas para a intervenção individual, mas para a melhoria contínua e baseada em evidências do processo educacional como um todo.

É aqui que entram métricas de análise de código. para além das tradicionais métricas de complexidade computacional (como tempo de execução), o professor pode lançar mão de métricas de similaridade e adequação estrutural, que avaliam, por exemplo, o uso esperado de variáveis, a aplicação correta de laços de repetição e o emprego adequado das palavras reservadas da linguagem.

O potencial inovador dessa abordagem reside na possibilidade de sistematizar a análise: os dados gerados por essas métricas – coletados em larga escala dos códigos dos estudantes – podem ser agregados em um dataset. Este, por sua vez, permite comparações objetivas não apenas entre indivíduos, mas entre turmas, períodos letivos e diferentes metodologias de ensino para a avaliação em larga escala.

1.1 Contexto, motivação e justificativa

Diferentemente da avaliação quantitativa, que prioriza resultados finais, a avaliação qualitativa é fundamental para aferir o processo de aprendizado. Em cursos de algoritmos, esse acompanhamento processual pode ser significativamente ampliado com o uso de telemetria. Conforme proposto por (Kennedy, 2015), essa abordagem é sinérgica em relação aos mecanismos tradicionais de *feedback*, como notas em laboratórios, tarefas e exames. A telemetria não os substitui, mas oferece uma camada adicional de dados contínuos e objetivos sobre o comportamento do estudante, enriquecendo a análise do docente. Saber o que acontece dentro de um sistema enquanto ele está em execução é benéfico para fins de depuração, manutenção e para fins de análise.(Gatev, 2021) (tradução nossa). Nosso trabalho buscará entender como os discentes produzem os códigos focando em algumas observações geradas por logs.

A telemetria proveniente de seus aplicativos geralmente pode ser dividida em três categorias: logs, métricas e rastros. Idealmente, você deve trabalhar para ter um único painel de

vidro que possa mostrar todas as três categorias para que você possa navegar entre elas. (Gatev, 2021)

Para (Foundation,)“Telemetria é o processo de coleta e transmissão automática de dados de sistemas remotos ou distribuídos para monitorar, medir e rastrear o desempenho ou o status desses sistemas. Os dados de telemetria fornecem percepções em tempo real sobre o desempenho de diferentes partes de um aplicativo".

Apesar do uso consolidado em ferramentas de desenvolvimento de software, a aplicação de telemetria no ensino de programação ainda não é um prática amplamente utilizada. Seu potencial, no entanto, é significativo: as instituições podem empregar os dados telemetrados para identificar dificuldades específicas dos alunos na compreensão do material, avaliar os nível de desafio apresentado por seus cursos e, consequentemente, intervir de forma mais precisa e personalizada.

Diante deste potencial, esta pesquisa tem o intuito de entender o processo de construção de algoritmos, as ferramentas e técnicas de engenharia de software utilizados no ensino, por meio de logs criados e armazenados no processo de ensino-aprendizagem em um IF da região Amazônica. Com esta coleta teremos informações qualiquantitativas sobre a produção de códigos, as quais serão fundamentais para a formulação de modelos com possibilidades de aplicação no ensino de programação e desenvolvimento de soluções para problemas locais. O estabelecimento de tal estrutura, permitiria o compartilhamento de conhecimentos de maneira cumulativa, podendo ser utilizado e aprimorado continuamente por sucessivas turmas de alunos, de diferentes cursos.

Nesse contexto, conforme destacado por (Qian; Lehman, 2017), “esforços para aprimorar o ensino em ciência da computação estão em andamento, e os docentes enfrentam desafios significativos em disciplinas introdutórias de programação, visando facilitar a aprendizagem dos estudantes".

Nos cursos técnicos em informática e ou afins ofertados pela Rede de EPCT, disciplinas como Introdução a Algoritmos, Programação Básica, Estrutura de Dados e Programação Orientada a Objetos são frequentemente associadas a altas taxas de dificuldade e evasão. Além dos obstáculos cognitivos, os estudantes frequentemente lidam com questões emocionais, sentindo-se incapazes de dominar os conceitos e, consequentemente, desenvolvendo a crença de que são inaptos para a área de informática como um todo. A aplicação da métrica proposta neste estudo permitiria uma análise qualiquantitativa do desempenho algorítmico e do uso do computador, oferecendo subsídios valiosos para reduzir esses problemas.

Por fim, esta pesquisa reforça o papel central dos IFs, que é promover o desenvolvimento socioeconômico local e regional, por meio de pesquisas aplicadas e da criação de soluções técnicas e tecnológicas alinhadas às demandas da comunidade, fortalecendo os arranjos produtivos locais (Rodrigues; Gava, 2016), pois demonstra como o corpo docente e técnico buscam alternativas para o aprimoramento do ensino e resultados para a instituição.

Sendo assim, pode-se interpretar esta característica dos IFs como um potencializador do uso de plataformas de códigos, estimulando os alunos a se dedicarem ao desenvolvimento

de soluções para problemas locais, ao mesmo tempo permitindo a interação com outras experiências e aportes colaborativos diversos.

A aplicação de princípios de Engenharia de Software aliada à análise de métricas de uso de ambientes de desenvolvimento integrado (IDE) possibilitará avaliações mais robustas. A opção por uma IDE amplamente adotada no setor profissional busca uniformizar as ferramentas utilizadas no contexto acadêmico, eliminando a dissonância entre disciplinas. A solução proposta consiste em uma extensão de operação transparente, executando-se em segundo plano sem comprometer a experiência de usuários finais (discentes e docentes) durante suas atividades de desenvolvimento.

1.2 Questões de pesquisa

1. QP1: Quais são os principais objetivos e aplicações do uso de logs no ensino de programação?
2. QP2: Quais ferramentas, métodos e métricas são mais utilizados para capturar, armazenar e analisar logs em ambientes de ensino de programação?

1.3 Objetivos

O objetivo deste trabalho é desenvolver e implementar uma extensão para coleta e análise de logs durante o processo de ensino-aprendizagem de programação, com o intuito de fornecer métricas que auxiliem na identificação de dificuldades dos alunos, na personalização do ensino e no aprimoramento das práticas pedagógicas

1.3.1 Objetivos Gerais

1. O que pretende: Investigar o processo de aprendizagem de programação por meio da análise de logs acadêmicos, identificando padrões e dificuldades comuns entre os estudantes.
2. Como pretende: Utilizando ferramentas de captura de eventos durante as aulas de programação, processando os dados coletados e organizando-os em um repositório acessível.
3. Por que pretende: Para melhorar o ensino de programação, oferecendo outras características, objetos de estudos baseados em dados quantitativos e criando um recurso (dataset) que possa ser utilizado por outros pesquisadores e educadores.

1.3.2 Objetivos Específicos

1. Desenvolver uma extensão para registro de eventos durante as aulas de programação.
2. Implementar o plugin na IDE utilizada na disciplina, com a anuência da instituição e dos estudantes.
3. Coletar, sincronizar e armazenar dados de eventos gerados durante o experimento.
4. Desenvolver um dataset com os dados coletados, organizados por turma, instituição e disciplina.

1.4 Contribuições esperadas

Este trabalho almeja contribuir com os pontos mencionados na seção 1.1 atendendo:

1. Um Mapeamento Sistemático da Literatura sobre uso de logs no ensino de programação focado principalmente nas instituições de ensino brasileiras.
2. Desenvolvimento de uma extensão para uma IDE que possa auxiliar na coleta de logs durante o uso do programa e desenvolvimento de códigos.
3. Desenvolvimento e disponibilidade de um dataset dos logs gerados de forma aberta, respeitando a privacidade e dados sigilosos de todos os participantes.

1.5 Estrutura do trabalho

Neste capítulo foram descritos o contexto, motivação e justificativa do trabalhos, suas questões de pesquisa, objetivos (geral e específicos), contribuições esperadas e estrutura. No Capítulo 2 é exposta sua fundamentação teórica, na qual são definidos os termos e conceitos fundamentais ao entendimento da proposta, bem como discorrido acerca dos trabalhos correlatos. Por sua vez, no Capítulo 3 é apresentada a metodologia de pesquisa a ser seguida(com suas fases e cronograma) e por fim, são apresentadas os seus apêndices.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta a fundamentação teórica necessária para o entendimento desta dissertação. Inicialmente, contextualiza os desafios do ensino de programação e introduz o conceito de *logging* como uma prática crucial da Engenharia de Software. Em seguida, explora o uso de *logs* no contexto educacional, detalha a taxonomia e os tipos de informações registradas, e apresenta a metodologia proposta para a análise desses dados. Por fim, é descrita a extensão *Logado*, que constitui a contribuição prática deste trabalho.

2.1 Desafios no Ensino de Programação e a Lacuna Instrucional sobre Logging

O ensino de programação transcende a mera transmissão de conhecimentos técnicos, abrangendo aspectos como boas práticas de desenvolvimento, integridade acadêmica e a compreensão do ciclo de vida do software. No entanto, uma análise da literatura pedagógica fundamental revela uma lacuna significativa.

Para mapear o estado da arte, consultou-se os conteúdos de livros-texto amplamente adotados globalmente (como os de Sommerville e Pressman) e a principal referência nacional, “Engenharia de Software Moderna” de Marco Tulio Valente. Constatou-se que, embora essas obras abordem tópicos como testes e qualidade de software, não dedicam seções ou discussões substantivas ao *logging* como uma prática de engenharia a ser ensinada, com suas próprias estratégias e melhores práticas (Gu *et al.*, 2022).

Esta omissão é notável dada a criticalidade dos *logs* na indústria e contrasta com o potencial educacional da análise dos rastros digitais que os alunos naturalmente geram durante a codificação. No Brasil, o tema é ainda mais incipiente; uma busca no portal da SBC com os termos `log OR logging AND "ensino de programação"` retorna poucos ou nenhum resultado, indicando um nicho de pesquisa promissor.

2.2 Logging: Conceitos e Aplicações na Engenharia de Software

Para estabelecer uma base terminológica unificada, adota-se a taxonomia de (Gu *et al.*, 2022), que define os conceitos fundamentais:

- Log statement
- Log message
- Log
- Log level
- Log content
- Log location
- Log placement
- Logging
- Logging practice

Segundo (Rice; Borgman, 1983), o monitoramento de sistemas pode ser entendido como um registro automático de transações. A importância dos *logs* é vasta, sendo utilizados em tarefas como detecção de anomalias, depuração, diagnóstico de desempenho e modelagem de comportamento de sistemas (Fu *et al.*, 2014). Estudos como o de (Yang *et al.*, 2021) confir-

mam que os desenvolvedores os utilizam primariamente para análise de problemas, buscando informações como a propagação de erros, *timestamps* e o *dataflow* entre componentes.

2.3 A Análise de Logs como Ferramenta Educacional

Aplicar a análise de *logs* no ensino de programação significa transpor essa prática da manutenção de sistemas para a análise do processo de aprendizagem. Os algoritmos de ensino não produzem apenas resultados; geram *logs* e rastros digitais detalhados que, quando analisados, podem fornecer insights sobre a curva de aprendizagem.

Como sugerido por (Silva *et al.*, 2022), variáveis preditoras podem ser extraídas do código, tais como: número de linhas lógicas, uso de estruturas de repetição, quantidade de identificadores, funções utilizadas, entre outras. Esta abordagem está alinhada com a *Learning Analytics*, cuja meta é melhorar a qualidade do ensino e da aprendizagem através da análise de dados sobre o comportamento dos estudantes (Huang *et al.*, 2020). Os *logs* de programação funcionam, assim, como um registro de aprendizagem (*learning record*) digital, revelando a experiência pessoal e as reflexões do estudante (Du; Wagner, 2005).

2.4 Metodologia para Análise de Logs Educacionais

A análise dos dados coletados será realizada em três etapas principais:

2.4.1 Coleta e Pré-processamento

Os dados brutos são gerados a partir da interação dos estudantes com o ambiente de programação, capturando eventos como edições de código, compilações e erros. Esta fase envolve:

- **Registro de Logs**: Captura de *timestamp*, trechos de código editados e eventos de desenvolvimento.
- **Filtragem e Anonimização**: Substituição de dados sensíveis por identificadores únicos para garantir a privacidade.

2.4.2 Geração de Indicadores Educacionais

Os *logs* brutos são transformados em métricas educacionais significativas (e.g., tempo para resolver um erro, padrões de tentativa-e-erro), criando um *dataset* para pesquisa. Este *dataset* permitirá estudos sobre a eficácia de metodologias de ensino e a correlação entre padrões de codificação e desempenho.

2.4.3 Análise de Padrões e Autoria

Com base no perfil de usuário (*user profile*) construído—que inclui frequência de *commits* e uso de estruturas específicas—é possível identificar discrepâncias na autoria de códigos, auxiliando na detecção de plágio e fornecendo uma base para avaliação personalizada.

2.5 A Extensão Logado: Proposta de Implementação

A motivação para o desenvolvimento da extensão *Logado* reside na evolução dos ambientes de programação. Argumenta-se que é mais eficaz integrar a funcionalidade de *logging* em uma IDE já consagrada e amplamente utilizada, evitando a curva de aprendizado associada a novas ferramentas. Dessa forma, a extensão será executada diretamente na IDE do estudante, capturando de forma integrada os eventos de desenvolvimento detalhados na seção 2.4.

3 METODOLOGIA E CRONOGRAMA

Este capítulo descreve a abordagem metodológica adotada, que combina uma Revisão Sistemática da Literatura (detalhada no Apêndice A) e um survey exploratório (resultados completos no Apêndice B) com a metodologia de Design Science Research para o desenvolvimento do artefato proposto.

3.1 Abordagem Metodológica

Este trabalho adota uma abordagem metodológica mista, combinando métodos de pesquisa bibliográfica e empírica para garantir uma fundamentação sólida tanto no estado da arte quanto no contexto prático do problema investigado.

3.1.1 Revisão Sistemática da Literatura

Para mapear o estado da arte sobre o uso de logs no ensino de programação, foi conduzida uma Revisão Sistemática da Literatura (RSL) seguindo as diretrizes de (Kitchenham, 2004) e o protocolo PRISMA (Page *et al.*, 2021). A revisão, detalhada na Seção A.3, foi realizada em cinco bases de dados acadêmicas e resultou na seleção de 17 estudos para análise qualitativa. O processo de seleção, ilustrado no fluxograma PRISMA (Figura 4), seguiu critérios rigorosos de inclusão e exclusão. O protocolo completo encontra-se no **Apêndice A**.

3.1.2 Survey Exploratório

Complementarmente à RSL, foi realizado um survey com 14 participantes, docentes, mestrandos em computação e pesquisadores de computação em foco no ensino de programação com o objetivo de investigar o uso de ferramentas de capturas de logs e construção de avaliações quantitativas e desenvolvimento de dataset acadêmicos. O instrumento de coleta continha 10 questões abordando logs, uso de ferramentas e IDEs. As respostas completas do survey e a análise detalhada encontram-se no **Apêndice B**.

3.1.3 Design Science Research

Este trabalho adota a abordagem de Design Science Research (DSR), conforme proposto por (Peffers *et al.*, 2007) e (Hevner *et al.*, 2004). O DSR é adequado para pesquisas que visam desenvolver e avaliar artefatos que resolvam problemas identificados em contextos organizacionais e educacionais.

O processo de DSR será executado em quatro atividades principais:

3.1.3.1 Atividade 1: Identificação do Problema e Motivação

O problema central identificado é a carência de ferramentas específicas para captura e análise de logs acadêmicos no ensino de programação, especialmente no contexto da região Amazônica. Esta lacuna limita a capacidade de docentes e instituições em identificar dificuldades de aprendizagem de forma objetiva e baseada em dados.

3.1.3.2 Atividade 2: Definição dos Objetivos da Solução

Com base no problema identificado, definiram-se os seguintes objetivos:

Objetivo Geral: Desenvolver e validar uma extensão para coleta e análise de logs durante o processo de ensino-aprendizagem de programação.

Objetivos Específicos:

1. Desenvolver uma extensão para registro de eventos em IDE
2. Implementar o plugin na IDE utilizada na disciplina
3. Coletar, sincronizar e armazenar dados de eventos
4. Desenvolver um dataset com os dados coletados
5. Validar a abordagem por meio de experimentos controlados

3.1.3.3 Atividade 3: Design e Desenvolvimento

3.1.3.4 Especificação de Requisitos do Artefato

Para garantir que a extensão *Logado* atenda aos objetivos da pesquisa e às necessidades dos usuários finais, foi realizada uma etapa de especificação de requisitos. A análise inicial identificou dois atores principais, cujas necessidades são sumarizadas na Tabela 1.

Tabela 1 – Análise de atores e necessidades do sistema

Ação	**Discente**	**Pesquisador**
Funções necessárias	Enviar dados de execução	Desenvolver/testar plugin
Necessidade principal	Instalar e usar o plugin	Validar processamento de logs
Operações CRUD	Enviar (criar) logs	Gerenciar usuários e dataset

Com base nesta análise, foram elaborados diagramas de casos de uso para especificar as funcionalidades do sistema. A Figura 1 ilustra as interações do ator Discente, enquanto a Figura 2 detalha as funcionalidades para o Pesquisador.

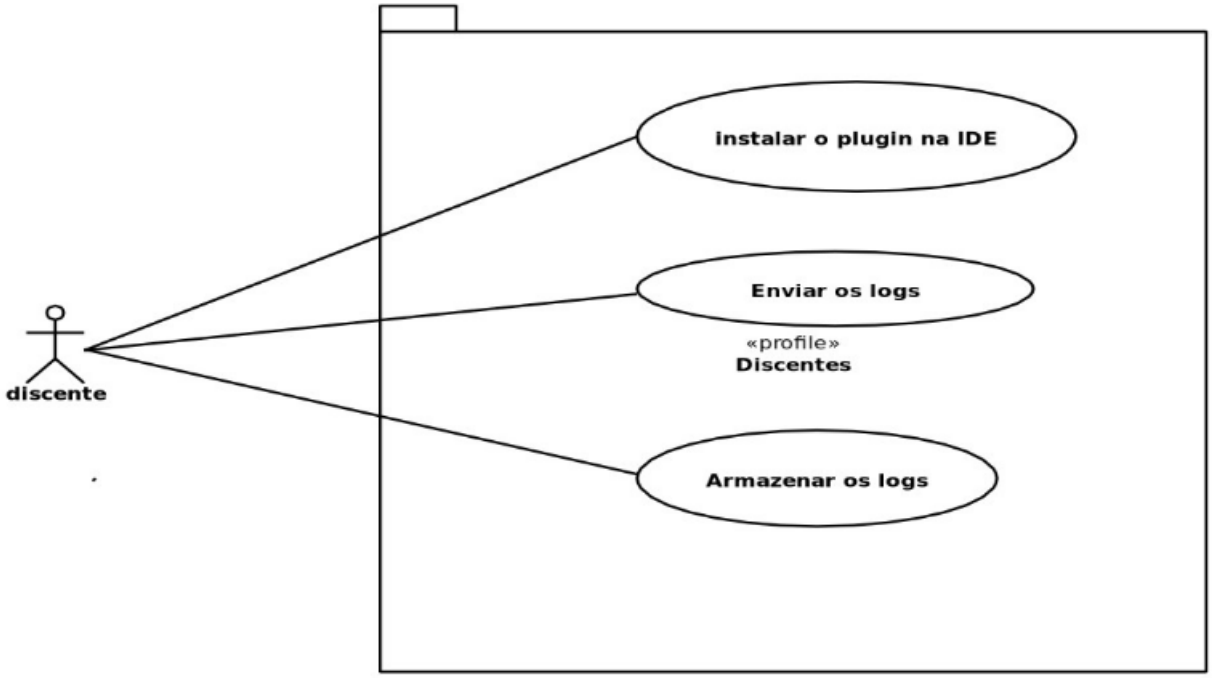

Figura 1 – Casos de uso da extensão Logado - Visão do Discente

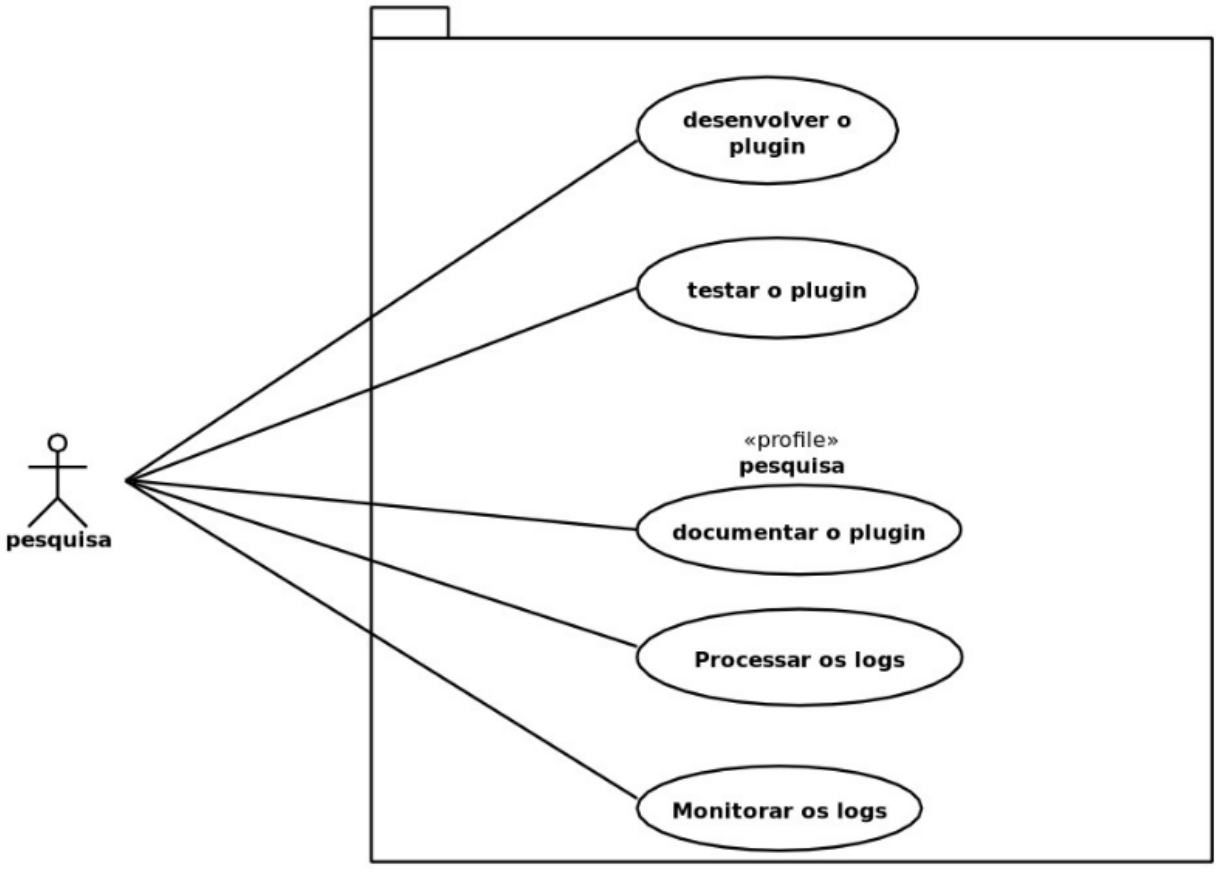

Figura 2 – Casos de uso da extensão Logado - Visão do Pesquisador

A partir dos casos de uso, foram derivados os requisitos formais do sistema:

3.1.3.4.1 Requisitos Funcionais

Os requisitos funcionais definem as funcionalidades que o sistema deve executar. Para o ator **Discente**, a extensão deve:

- **RF01:** Registrar automaticamente eventos de edição de código (e.g., inserção, deleção de caracteres).
- **RF02:** Capturar eventos de compilação e execução de código, incluindo sucesso ou falha.
- **RF03:** Armazenar localmente os logs com *timestamp* preciso.
- **RF04:** Transmitir os logs anonimizados para um servidor remoto de forma assíncrona.
- **RF05:** Exibir notificações de confirmação de operações para o usuário.

Para o ator **Pesquisador**, a extensão deve:

- **RF06:** Fornecer uma interface de configuração para parametrizar os eventos a serem capturados.
- **RF07:** Gerar identificadores únicos anônimos para cada sessão de uso.

3.1.3.4.2 Requisitos Não-Funcionais

Estes requisitos definem os critérios de qualidade para o sistema:

- **RNF01 (Desempenho):** A extensão não deve impactar significativamente o desempenho da IDE. O tempo de resposta deve ser inferior a 100ms para operações de registro.
- **RNF02 (Usabilidade):** A instalação e o uso devem ser simples, não requerendo configuração complexa por parte do discente.
- **RNF03 (Confiabilidade):** A extensão deve garantir a persistência dos logs mesmo em caso de fechamento inesperado da IDE.
- **RNF04 (Segurança e Privacidade):** Todos os dados pessoais devem ser anonimizados antes do envio ao servidor, conforme a LGPD.

3.1.3.4.3 Desenvolvimento do Artefato

Desenvolvimento da extensão LOGADO para Visual Studio Code, com funcionalidades de captura de eventos de programação.

3.1.3.5 Atividade 4: Demonstração e Avaliação

3.1.3.5.1 Contexto e Participantes

Pesquisa será realizada no IFAM Campus Boca do Acre com aproximadamente 80 estudantes do curso técnico em Informática.

3.1.3.5.2 Procedimentos de Coleta

- Duração: 3 semanas de aulas práticas
- Local: Laboratório de Informática do campus
- Ferramenta: Visual Studio Code com extensão LOGADO
- Aspectos éticos: Aprovação do Comitê de Ética, TCLE

3.2 Plano de Análise de Dados

Os dados coletados serão analisados em três etapas:

3.2.1 Pré-processamento e Limpeza

Filtragem e anonimização dos dados, garantindo a privacidade dos participantes.

3.2.2 Geração de Métricas Educacionais

Transformação dos logs brutos em indicadores de aprendizagem.

3.2.3 Análise Estatística e Identificação de Padrões

Uso de técnicas estatísticas para correlacionar padrões de codificação com desempenho acadêmico.

3.3 Cronograma

Nesta seção é exposto o cronograma previsto para a realização deste trabalho e obtenção de grau de mestre (considerando os requisitos apresentados no Regulamento Interno do Programa de Pós-Graduação em Computação Aplicada (PPGCA) da Universidade Tecnológica Federal do Paraná (UTFPR), campus Curitiba. Na figura 3 são listadas as atividades a serem

realizadas e os períodos em que estas pretendem ser executadas. As atividades estão organizadas de acordo com as quatro fases detalhadas abaixo, não sendo apresentadas aquelas já finalizadas(e.g.: revisão da literatura - Fase 1; delineamento inicial da abordagem - parte da Fase 2; obtenção dos créditos exigidos pleo PPGCA em disciplinas ou atividades; e aprovação em exame de suficiência em língua inglesa).

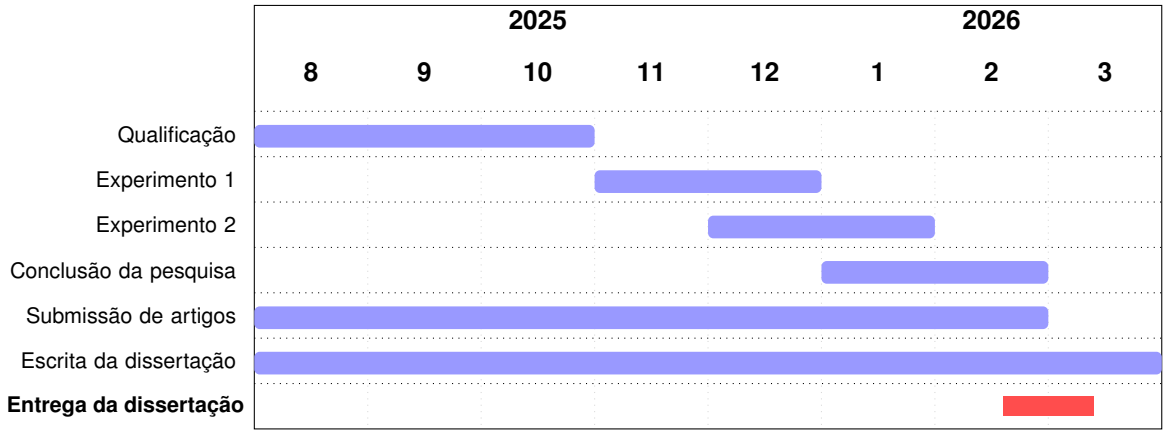

Figura 3 – Cronograma do trabalho/mestrado

Conforme pode ser visualizado na 3, as atividades foram planejadas de modo mensal. Inicialmente, para agosto de 2025 estavam previstas: (1) a qualificação da proposta de dissertação; e (2) a execução da parte da Fase 3, por meio da validação, análise e interpretação dos dados coletados na primeira execução do experimento de avaliação da abordagem (ocorrerá em novembro de 2025). A segunda atividade será finalizada em setembro, sendo seguida pela realização de eventuais ajustes na abordagem (Fase 2), cujo término é previsto para outubro.

Como forma de validação externa da pesquisa, um artigo com o mesmo tema desta dissertação foi submetido à LACLO (Conferência Latinoamericana de Tecnologias de Aprendizagem) e aceito após processo de revisão por pares. Esta aceitação, ocorrida em agosto de 2025, atesta o valor científico do trabalho, que será apresentado no evento em novembro do mesmo ano.

Por sua vez, nos meses de outubro e novembro planeja-se realizar a segunda operação do experimento, bem como análise e interpretação dos dados coletados (seguindo os passos a serem descritos nas subseções 4.2.3 e 4.2.4). Já em dezembro pretende-se concluir a pesquisa (execução das atividades da Fase 4).

Acerca da escrita da dissertação, está ocorrerá entre agosto de 2025 e março de 2026. É importante mencionar que a escrita de todos os capítulos pretende ser finalizada ainda em dezembro de 2025, sendo janeiro e fevereiro do ano seguinte destinado a correções e/ou melhorias recomendadas pelos orientadores (antes do envio do documento à banca). Por sua vez, em março serão realizados eventuais ajustes no texto(sugeridos pela banca) após a defesa da dissertação, prevista para o mesmo mês.

Por fim, após a conclusão das atividades mencionadas acima, ainda em março planeja-se: (1) entregar a dissertação para registro no RIUT; (2) prover feedback dos resultados da pesquisa aos participantes interessados (a indicação de interesse será coletada em campo do

Termo de Consentimento Livre e Esclarecido (TCLE), sendo os resultados enviados por e-mail); e (3) elaborar e entregar o relatório final de pesquisa ao CEP da UTFPR, atividade que encerra as atividades previstas para este trabalho.

REFERÊNCIAS

ANNAMAA, A. Thonny, a python ide for learning programming. *In*: PROCEEDINGS OF THE 2015 ACM CONFERENCE ON INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION. 2015. **Anais [...]** [*S.l.*: *s.n.*], 2015. p. 343–343.

ARABYARMOHAMADY, S.; MORADI, H.; ASADPOUR, M. A coding style-based plagiarism detection. *In*: IEEE. PROCEEDINGS OF 2012 INTERNATIONAL CONFERENCE ON INTERACTIVE MOBILE AND COMPUTER AIDED LEARNING (IMCL). 2012. **Anais [...]** [*S.l.*], 2012. p. 180–186.

CLEMENTE, R. G. Uma arquitetura para processamento de eventos de log em tempo real. **Agosto de**, „, 2008.

DU, H. S.; WAGNER, C. Learning with weblogs: An empirical investigation. *In*: IEEE. PROCEEDINGS OF THE 38TH ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES. 2005. **Anais [...]** [*S.l.*], 2005. p. 7b–7b.

EDUCAÇÃO, C. e. T. d. A. Instituto Federal de. **Projeto Pedagógico do Curso Técnico de Nível Médio em Informática, na Forma Subsequente**. Boca do Acre: , 2020. Disponível em: https:cbda.ifam.edu.br/cursos/subsequentes/info/ppc-info-subsequente.pdf.

FOUNDATION, L. **The Linux Foundation**. [*S.l.*]: , . https://trainingportal.linuxfoundation.org/learn/course/getting-started-with-opentelemetry-lfs148/why-do-we-need-opentelemetry/how-we-got-here?page=5. Accessed: 2024-12-3.

FU, Q. *et al.* Where do developers log? an empirical study on logging practices in industry. *In*: COMPANION PROCEEDINGS OF THE 36TH INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING. 2014, Hyderabad India. **Anais [...]** Hyderabad India: ACM, 2014. p. 24–33. ISBN 978-1-4503-2768-8. Disponível em: https://dl.acm.org/doi/10.1145/2591062.2591175.

GATEV, R. Observability: Logs, metrics, and traces. *In*: **Introducing Distributed Application Runtime (Dapr): Simplifying Microservices Applications Development Through Proven and Reusable Patterns and Practices**. Berkeley, CA: Apress, 2021. p. 233–252. ISBN 978-1-4842-6998-5. Disponível em: https://doi.org/10.1007/978-1-4842-6998-5_12.

GU, S. *et al.* Logging practices in software engineering: A systematic mapping study. **IEEE Transactions on Software Engineering**, IEEE v. 49, n. 2, p. 902–923, 2022.

GUPTA, A.; JINDAL, M.; GOYAL, A. Identification of student programming patterns through clickstream data. *In*: IEEE. 2024 IEEE INTERNATIONAL CONFERENCE ON COMPUTING, POWER AND COMMUNICATION TECHNOLOGIES (IC2PCT). 5., 2024. **Anais [...]** [*S.l.*], 2024. p. 1153–1158.

HEVNER, A. R. *et al.* Design science in information systems research. **MIS quarterly**, JSTOR„ p. 75–105, 2004.

HUANG, A. Y. *et al.* Predicting students' academic performance by using educational big data and learning analytics: evaluation of classification methods and learning logs. **Interactive Learning Environments**, Taylor & Francis v. 28, n. 2, p. 206–230, 2020.

KENNEDY, A. R. **Towards a data-driven analysis of programming tutorials' telemetry to improve the educational experience in introductory programming courses**. 2015. Tese (Doutorado) 2015.

KITCHENHAM, B. Procedures for performing systematic reviews. **Keele, UK, Keele University**, v. 33, n. 2004, p. 1–26, 2004.

LIMA, N. A. *et al.* Avaliação da aprendizagem nos institutos federais em goiás durante a pandemia: uma investigação documental, . Insitituto Federal de Educação, Ciência e Tecnologia de Goiás„, 2022.

LIU, C. *et al.* User behavior discovery from low-level software execution log. **IEEJ Transactions on Electrical and Electronic Engineering**, Wiley Online Library v. 13, n. 11, p. 1624–1632, 2018.

MANGAROSKA, K. *et al.* Exploring students' cognitive and affective states during problem solving through multimodal data: Lessons learned from a programming activity. **Computer Assisted Learning**, v. 38, n. 1, p. 40–59, fev. 2022. ISSN 0266-4909, 1365-2729. Disponível em: https://onlinelibrary.wiley.com/doi/10.1111/jcal.12590.

NGUYEN, B.-A.; HA, T.-V. T.; CHEN, H.-M. Analyzing git log in an code-quality aware automated programming assessment system: A case study. **tvujs**, „, set. 2023. ISSN 2815-6099, 2815-6072. Disponível em: https://journal.tvu.edu.vn/index.php/journal/article/view/2430.

PAGE, M. J. *et al.* The prisma 2020 statement: an updated guideline for reporting systematic reviews. **Systematic reviews**, Springer v. 10, n. 1, p. 1–11, 2021.

PEFFERS, K. *et al.* A design science research methodology for information systems research. **Journal of management information systems**, Taylor & Francis v. 24, n. 3, p. 45–77, 2007.

PEREIRA, F. D. *et al.* Using learning analytics in the amazonas: understanding students' behaviour in introductory programming. **British journal of educational technology**, Wiley Online Library v. 51, n. 4, p. 955–972, 2020.

QIAN, Y.; LEHMAN, J. Students' misconceptions and other difficulties in introductory programming: A literature review. **ACM Transactions on Computing Education (TOCE)**, ACM New York, NY, USA v. 18, n. 1, p. 1–24, 2017.

RAABE, A. L. A.; SILVA, J. d. Um ambiente para atendimento as dificuldades de aprendizagem de algoritmos. *In*: SN. XIII WORKSHOP DE EDUCAÇÃO EM COMPUTAÇÃO (WEI'2005). SÃO LEOPOLDO, RS, BRASIL. 3 n. 5., 2005. **Anais [...]** [*S.l.*], 2005.

RICE, R. E.; BORGMAN, C. L. The use of computer-monitored data in information science and communication research. **Journal of the American Society for Information Science**, Wiley Online Library v. 34, n. 4, p. 247–256, 1983.

RODRIGUES, F. C. R.; GAVA, R. Universidades federais no estado de minas gerais: Um estudo comparativo. **REAd | Porto Alegre – Edição 83**, SciELO Brasil„, 2016.

SILVA, E. S. *et al.* Previsão de indicadores de dificuldade de questões de programação a partir de métricas do código de solução. *In*: SBC. ANAIS DO XXXIII SIMPóSIO BRASILEIRO DE INFORMáTICA NA EDUCAçãO. 2022. **Anais [...]** [*S.l.*], 2022. p. 859–870.

UMEZAWA, K. *et al.* Analysis of logic errors utilizing a large amount of file history during programming learning. *In*: IEEE. 2020 IEEE INTERNATIONAL CONFERENCE ON TEACHING, ASSESSMENT, AND LEARNING FOR ENGINEERING (TALE). 2020. **Anais [...]** [*S.l.*], 2020. p. 630–634.

YANG, N. *et al.* An interview study of how developers use execution logs in embedded software engineering. *In*: IEEE. 2021 IEEE/ACM 43RD INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING: SOFTWARE ENGINEERING IN PRACTICE (ICSE-SEIP). 2021. **Anais [...]** [*S.l.*], 2021. p. 61–70.

APÊNDICE A – Mapeamento Sistemático da Literatura

A.1 Objetivo e questões de pesquisa

As pesquisas conduzidas nos repositórios de trabalhos, com base nas *strings* de busca apresentadas na Tabela 4, resultaram em 163 estudos após um processo de refinamento e testes iterativos das entradas. Esses trabalhos foram então categorizados e analisados em fases sequenciais, visando garantir uma revisão sistemática e abrangente.

A.2 Resultados da Busca e Caracterização dos Estudos

O processo de seleção, ilustrado na Figura 4, resultou na inclusão de 17 estudos para análise qualitativa.

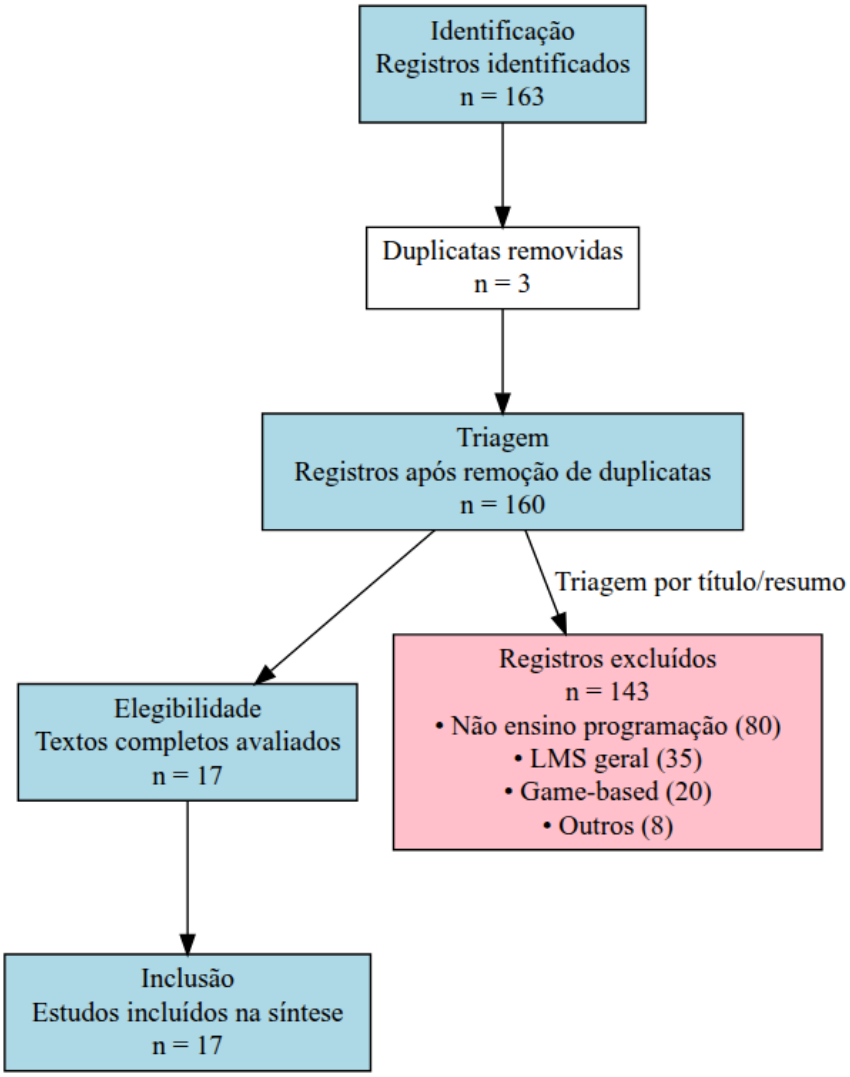

Figura 4 – Fluxograma PRISMA do processo de seleção de estudos

A Tabela 2 sumariza as principais características destes estudos.

Estudo	Ano	Foco Principal	Tipo de Log
Arabyarmohamady et al.	2012	Detecção de plágio	Estilo de codificação
Annamaa	2015	IDE educacional	Logs de sessão Python
Raabe et al.	2005	Dificuldades de aprendizagem	Ambiente educacional
Liu et al.	2018	Comportamento do usuário	Logs de execução
Umezawa et al.	2020	Análise de erros lógicos	Histórico de arquivos
Pereira et al.	2020	Comportamento estudantil	Logs de IDE + Online Judge
Gu et al.	2022	Práticas de logging	Revisão sistemática
Silva et al.	2022	Predição de dificuldade	Métricas de código
Mangaroska et al.	2022	Estados cognitivos	Logs Eclipse + multimodal
Da Silva	2022	Análise de comunidades	Comportamento online
Nguyen et al.	2023	Avaliação automática	Git logs + qualidade
Educomp	2023	Complexidade vs dificuldade	Métricas de exercícios
Gupta et al.	2024	Padrões de programação	Clickstream data
Arguson et al.	2022	Erros de compilação	Logs de compilação
Liz-Domínguez et al.	2022	Desempenho acadêmico	Logs de LMS
Yang et al.	2021	Uso de logs na indústria	Práticas profissionais
Cândido et al.	2021	Monitoramento baseado em logs	Revisão sistemática

Tabela 2 – Características dos 17 estudos incluídos na síntese qualitativa

A.2.1 Perfil dos Estudos Incluídos

A análise dos 17 estudos revela um interesse crescente na área, com 65% das publicações concentradas após 2020. A maioria dos estudos (53%) utiliza logs de IDEs ou ambientes específicos, enquanto apenas 12% abordam aspectos de privacidade e ética.

A análise temporal confirma o interesse crescente na área, com predominância de estudos recentes. Quanto ao foco metodológico, mais da metade utiliza logs de ambientes específicos, enquanto aspectos de privacidade permanecem pouco explorados.

A.2.2 Respostas para QP1: Quais são os principais objetivos e aplicações do uso de logs no ensino de programação?

Os objetivos identificados foram categorizados em quatro áreas principais:

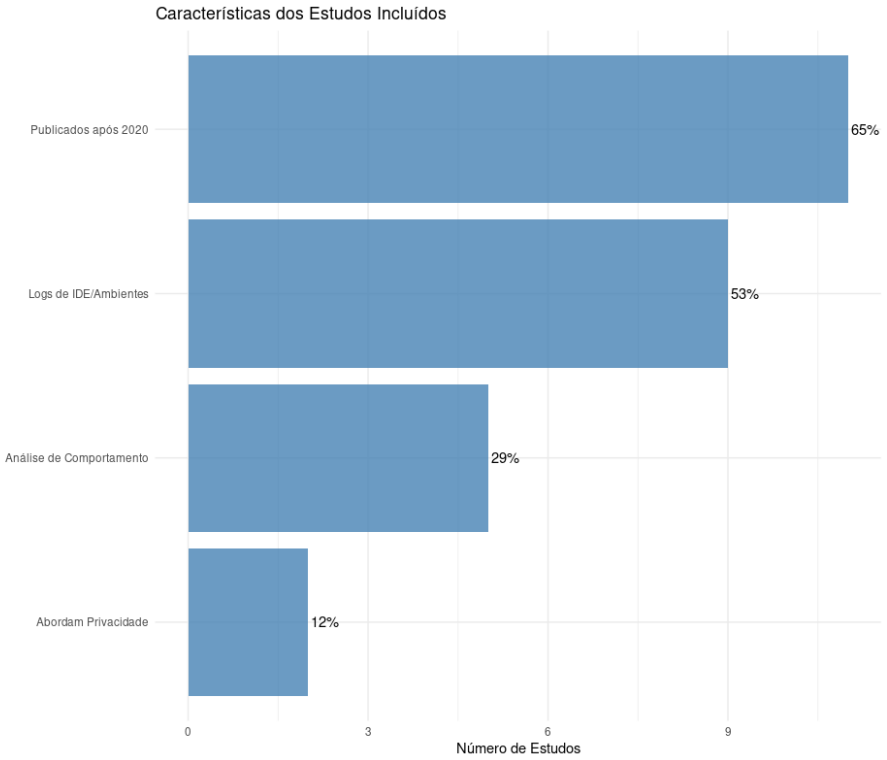

Figura 5 – Distribuição das características principais dos estudos incluídos (n=17)

- **Detecção de Dificuldades** (6 estudos): Umezawa (2020), Pereira (2020), Silva (2022), Educomp (2023), Arguson (2022), Raabe (2005)
- **Análise de Comportamento** (5 estudos): Gupta (2024), Liu (2018), Mangaroska (2022), Da Silva (2022), Liz-Domínguez (2022)
- **Qualidade de Código** (3 estudos): Arabyarmohamady (2012), Nguyen (2023), Annamaa (2015)
- **Práticas e Ferramentas** (3 estudos): Gu (2022), Yang (2021), Cândido (2021)

De acordo com (Gu *et al.*, 2022), “A obtenção de logs é um processo que deve ser guiado por quatro questões fundamentais, conhecidas como 3W1H:

- Why (Por quê): Qual é o objetivo da coleta de logs?
- Where (Onde): Em quais partes do sistema ou código os logs devem ser coletados?
- What (O que): Quais informações ou eventos devem ser registrados nos logs?
- How (Como): De que forma os logs serão coletados, armazenados e analisados?

Essas questões são essenciais para garantir que a geração de logs seja eficiente, relevante e alinhada aos objetivos do projeto, seja no desenvolvimento de software ou no contexto educacional. No ensino de programação, por exemplo, a aplicação do 3W1H pode ajudar a definir métricas e técnicas para monitorar o progresso dos estudantes e identificar dificuldades de aprendizado.

A abordagem feita por (Silva *et al.*, 2022) busca “prever indicadores de dificuldade de exercícios de programação a partir de métricas de complexidade do código dos instrutores. Identificando registros de logs de JOs e relacionando esforço e dificuldade dos estudantes durante o desenvolvimento do código de solução”.

De acordo com (Gupta; Jindal; Goyal, 2024), “os dados de cliques (clickstream) e logs de atividades dos estudantes em plataformas de programação podem ser processados para extrair informações relevantes, como padrões de programação e relações entre características do código e o desempenho dos alunos. Essa análise permite que os instrutores identifiquem previamente estudantes com dificuldades ou baixo desempenho, ajustando suas metodologias de ensino e oferecendo suporte direcionado. Além disso, ao acompanhar a evolução do estilo de codificação dos alunos, os educadores podem avaliar a eficácia de suas estratégias pedagógicas e garantir que nenhum estudante seja deixado para trás em um mundo tecnologicamente avançado”.

O ensino de programação vai além da transmissão de conhecimentos técnicos, incluindo orientações sobre licenciamento de códigos e boas práticas de desenvolvimento. Nesse contexto, o combate ao plágio é uma preocupação relevante, e os logs surgem como uma ferramenta valiosa para apoiar a avaliação dos estudantes. Como argumenta (Arabyarmohamady; Moradi; Asadpour, 2012), características extraídas de cada código, como o estilo de programação, são armazenadas em um histórico (user profile). Esses dados permitem identificar mudanças no estilo de codificação e detectar se um código foi realmente desenvolvido pelo autor declarado ou se foi escrito por outra pessoa com um estilo diferente. Dessa forma, os logs não apenas auxiliam na identificação de plágio, mas também contribuem para uma avaliação mais precisa e personalizada do aprendizado dos estudantes.

Outra aplicação relevante para a geração de logs é a criação de *datasets*, como demonstrado no trabalho de (Umezawa *et al.*, 2020). O estudo detecta diversos elementos em códigos-fonte, como declarações de variáveis, funções, estruturas condicionais, laços de repetição, operações de leitura e escrita, espaçamento, pontuação, expressões lógicas e matemáticas, além de comentários. Esses dados são organizados em tabelas para análise. Um exemplo interessante é o uso do espaçamento: embora não afete a execução do código, padrões de espaçamento ’incorretos’ ou idênticos podem indicar que vários estudantes estão cometendo o mesmo erro ou copiando trechos de código uns dos outros. Isso demonstra que os logs vão além da identificação de erros de lógica ou sintaxe, podendo revelar padrões de comportamento e práticas comuns entre os estudantes. Além disso, o trabalho destaca a importância de estruturas de controle, como declarações ’for’ e ’if’, e a frequência de alterações em variáveis e números, que refletem a natureza repetitiva dos exercícios universitários. Esses insights mostram o potencial dos logs para apoiar educadores na identificação de dificuldades e na melhoria do ensino de programação.

A.2.3 Respostas para QP2: Quais ferramentas, métodos e métricas são mais utilizados para capturar, armazenar e analisar logs em ambientes de ensino de programação?

No trabalho de (Pereira *et al.*, 2020), foi desenvolvida uma IDE para obter todas as informações. “O CodeBench fornece o seu próprio editor de código (IDE), concebido para ser simples e fácil de utilizar por principiantes. Todas as ações executadas pelos alunos no IDE (por exemplo, teclas digitadas, submissões, colagem de código, cliques do mouse, transições de separadores ou de janelas, etc.) são registradas e gravadas em arquivos de registro no lado do servidor.”

Para (Liu *et al.*, 2018), “Cada evento representa o registro de uma chamada de método durante a execução do software. Portanto, todos os eventos podem ser alinhados com padrões de comportamento previamente identificados. Ao utilizar um log de execução de software e os padrões de comportamento treinados como entrada, realizamos a correspondência baseada em alinhamento (alignment-based matching) para obter um log abstrato de operações do usuário.” Essa técnica pode ser aplicada no ensino de programação para analisar as interações dos estudantes com ferramentas de desenvolvimento, identificar erros comuns e fornecer feedback personalizado com base em seu comportamento durante a codificação.

O trabalho de (Mangaroska *et al.*, 2022) utiliza algumas ferramentas para capturar a interação do usuário, contudo, na questão de logs, foi utilizada uma extensão para o Eclipse e foi definida assim: “Log data: Um plugin para Eclipse que possui uma aba, visualização (Trætteberg et al., 2016), foi utilizado para obter comportamentos de leitura e escrita dos estudantes. Este plugin armazena o estado do programa toda vez que o estudante salva o programa, usando o atalho ou botão de salvar.”

A ferramenta desenvolvida por (Annamaa, 2015), uma IDE projetada na Universidade de Tartu com interface amigável para iniciantes em Python, coleta informações detalhadas sobre cada sessão de programação. Esses dados incluem eventos como carregar, salvar e modificar arquivos, além de diferenciar textos copiados de textos digitados. As informações coletadas permitem reproduzir todo o processo de construção do programa e as atividades realizadas no shell. Para isso, a ferramenta oferece uma janela separada onde o usuário pode selecionar um arquivo de log e visualizar a reprodução dos eventos em velocidade ajustável. Essa funcionalidade é particularmente útil para educadores, pois permite analisar o processo de desenvolvimento dos estudantes e identificar possíveis dificuldades.

Nos últimos anos, a educação em programação tem passado por uma transformação significativa com a introdução de Sistemas Automatizados de Avaliação de Programação (APASs). Esses sistemas revolucionaram a forma como as tarefas de programação são avaliadas, oferecendo vantagens tanto para educadores quanto para estudantes. O ProgEdu, desenvolvido por (Nguyen; Ha; Chen, 2023), é um exemplo de aplicação baseada na web que combina as funcionalidades de um Sistema de Gestão de Aprendizagem (LMS) com um APAS. A ferramenta permite que os estudantes acessem atividades, submetam códigos e recebam feedback automatizado, facilitando a identificação de erros e o aprimoramento das habilidades de

programação. Além disso, o ProgEdu é uma aplicação baseada em Docker, o que garante portabilidade e facilidade de implantação. A automação do processo de avaliação não apenas reduz a carga de trabalho dos instrutores, mas também oferece avaliações consistentes e imparciais, aplicando critérios predefinidos de forma objetiva. Essa abordagem é particularmente relevante no contexto da Educação a Distância (EAD), onde a popularização de serviços baseados na web tem se tornado cada vez mais comum.

A.3 Identificação dos Estudos

A.3.1 Estratégia de Busca

A pesquisa foi realizada em cinco bibliotecas virtuais, utilizando como critérios de busca palavras-chave e *strings* de busca com operadores lógicos *AND* e *OR*. Os repositórios de pesquisa utilizados são apresentados na Tabela 3.

Repositório	**Resultados**
Scopus	23
IEEE Xplore	2
SOL SBC	2
ACM	127
WILEY	9

Tabela 3 – Lista de repositórios de pesquisa utilizados no estudo.

As combinações de palavras-chave e operadores utilizadas na pesquisa são apresentadas na Tabela 4.

Palavras-chave e Operadores
programming AND teaching AND log AND NOT (log-based) AND NOT (metrics) AND systematic AND review
programming AND teaching AND log AND NOT (log-based) AND (integrate AND development AND environment) AND NOT (visual AND programming) AND NOT (web AND application)
telemetry AND education AND programming AND log
ontology AND education AND programming AND log AND dataset
artefact AND cognitive AND learning
background AND teaching AND log AND NOT (log-based) AND (systematic AND review)
log AND (“teaching programming"OR “learning programming") AND (“IDE"OR “integrated development environment")) NOT (“log-based"OR “lms"OR “learning management system"
(log AND ("teaching programming"OR "learning programming") AND ("IDE"OR "integrated development environment")) NOT ("log-based"OR "lms"OR "learning management system"OR "game-based"OR "game based"OR "gamification")

Tabela 4 – Combinações de palavras-chave e operadores utilizados na pesquisa.

A.3.2 Idioma dos Trabalhos

Os trabalhos foram selecionados em português e inglês. A inclusão do português justifica-se pela relevância de pesquisas nacionais, especialmente no contexto das Instituições de Ensino Federais brasileiras. O inglês, por sua vez, foi escolhido por ser o idioma predominante em publicações acadêmicas internacionais, como conferências e periódicos de alto impacto.

A.3.3 Critérios de Inclusão

Após a execução das buscas utilizando as *strings* definidas, os artigos identificados foram submetidos a um processo de triagem. Foram adotados os seguintes critérios para inclusão:

1. Relevância temática: O artigo deve abordar explicitamente o uso de logs no ensino de programação, conforme verificado pela análise de títulos e resumos.
2. Revisão sistemática: Trabalhos que se caracterizam como revisões sistemáticas da literatura também foram incluídos, devido à sua contribuição para a compreensão do estado da arte no tema.

A.3.4 Critérios de Exclusão

Foram excluídos textos que, mesmo contendo a *string* de busca, retornaram resultados relacionados a:

1. *Log-based*: abordagens ou técnicas que utilizam logs como base principal (por exemplo, sistemas de monitoramento ou análise de logs).
2. Uso de logs em outras áreas da educação: aplicações de logs em contextos educacionais que não envolvem diretamente o ensino de programação.
3. *Learning Management Systems (LMS)*: trabalhos que abordam o uso de logs em plataformas de gestão de aprendizagem (LMS), como Moodle, Canvas ou Blackboard, sem foco específico no ensino de programação.
4. *Game-based learning*: trabalhos que utilizam jogos ou mecânicas de jogos como principal estratégia de ensino, sem foco no uso de logs para análise de aprendizagem.

A.4 Lacunas Identificadas e Trabalhos Futuros

A análise revelou lacunas significativas:

- **Padronização**: Ausência de modelos comuns para logs educacionais

- **Privacidade**: Poucos estudos abordam aspectos éticos
- **Contexto Brasileiro**: Limitada pesquisa em instituições brasileiras

APÊNDICE B – Survey

Para entender o uso de logs no ensino de programação, buscamos informações de outras instituições de ensino. Para ter informações sobre como alguns nós da Rede EPTC tratam o assunto, foi elaborado um questionário. A partir das respostas, os dados foram tabulados e gráficos foram gerados a partir dessas informações.

B.1 Método

Para capturar as percepções da comunidade acadêmica, foi conduzido um *survey* online com docentes e técnicos de diversos campi dos Institutos Federais, outras instituições de ensino e órgãos públicos. O instrumento de pesquisa continha 10 questões, em sua maioria utilizando uma escala Likert de 5 pontos, na qual os respondentes assinalavam seu nível de concordância, variando de **Discordo Totalmente (1)** a **Concordo Totalmente (5)**. A coleta de dados foi realizada de forma assíncrona através de formulário online.

Para a análise dos dados e geração de visualizações, utilizou-se a linguagem R e seu ecossistema de pacotes.

B.2 Instrumento de pesquisa

O questionário continha as seguintes questões:

2. Você usa alguma ferramenta de log no apoio ao ensino de programação?
   ( ) Sim ( ) Não

3. O uso de ferramentas de logs pode auxiliar no monitoramento e compreensão no empenho dos estudantes?
   (1) Discordo Totalmente (2) Discordo Parcialmente (3) Neutro (4) Concordo Parcialmente (5) Concordo Totalmente

4. O uso de ferramentas de logs pode ajudar a tomar decisões para melhorar o rendimento e evitar a desistência de uma disciplina?
   (1) Discordo Totalmente (2) Discordo Parcialmente (3) Neutro (4) Concordo Parcialmente (5) Concordo Totalmente

5. O uso de ferramentas de logs pode auxiliar no ensino de programação?
   (1) Discordo Totalmente (2) Discordo Parcialmente (3) Neutro (4) Concordo Parcialmente (5) Concordo Totalmente

6. Um dataset pode ser criado e estudado a partir de logs no ensino da programação?
   (1) Discordo Totalmente (2) Discordo Parcialmente (3) Neutro (4) Concordo Parcialmente (5) Concordo Totalmente

7. Sobre usar ferramentas de correções de códigos mais automatizado possível nas aulas de ensino de programação. O que pensa a respeito sobre o uso?

(1) Discordo Totalmente (2) Discordo Parcialmente (3) Neutro (4) Concordo Parcialmente (5) Concordo Totalmente

8. Plugins podem auxiliar na criação/ensino de algoritmos?
   (1) Discordo Totalmente (2) Discordo Parcialmente (3) Neutro (4) Concordo Parcialmente (5) Concordo Totalmente

9. Qual IDE é utlizado nas aulas?
   ( )Visual Studio ( ) Intellij IDEA ( ) Eclipse ( ) Outro - Descrever

10. Ferramenta de análise de algoritmos
    ( )Code Bench ( ) Juiz Online ( ) LAPLE ( ) Outro: Descrever

11. Se tiver disponível um plugin de log para o apoio ao ensino de programação, você usaria?
    ( ) Sim ( ) Não

B.3 Respostas

O formulário foi respondido por quatorze pessoas de várias instituições de ensino, público e privado e por outros órgãos. As respostas foram voluntárias. Os profissionais que responderam exercem a função nos IFs são professores EBTT de Informática. As pessoas que responderam pela PUCPR foram estudantes do mestrado em computação da instituição.

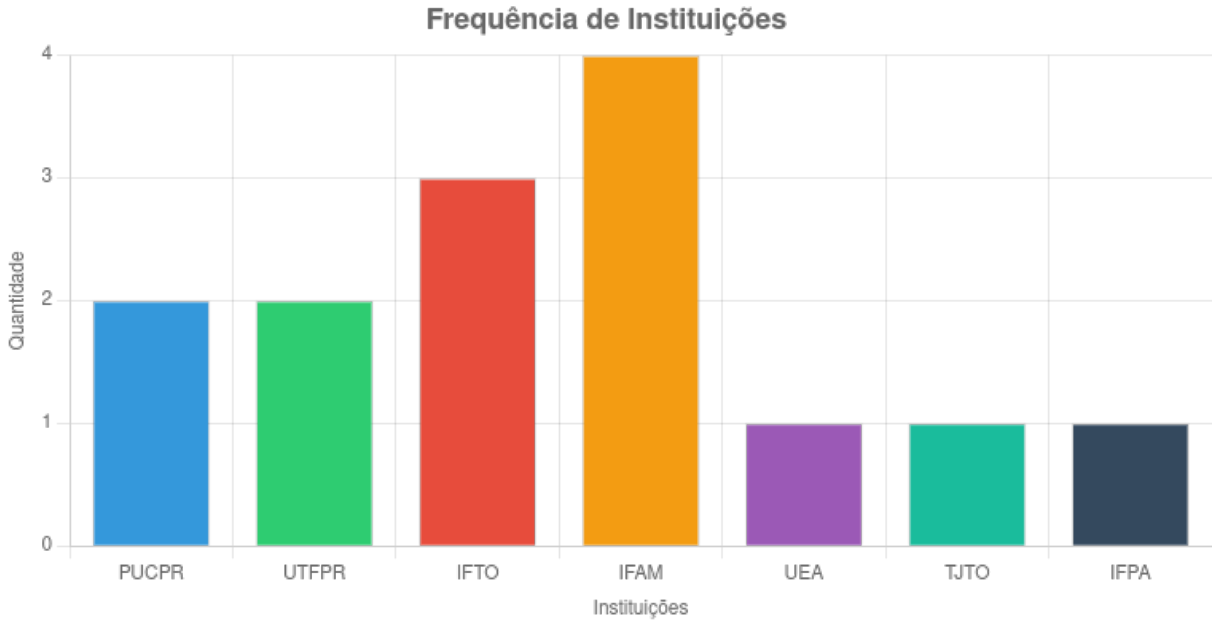

Figura 6 – Instituições que participaram do formulário

B.3.1 Análise de Ferramentas Utilizadas

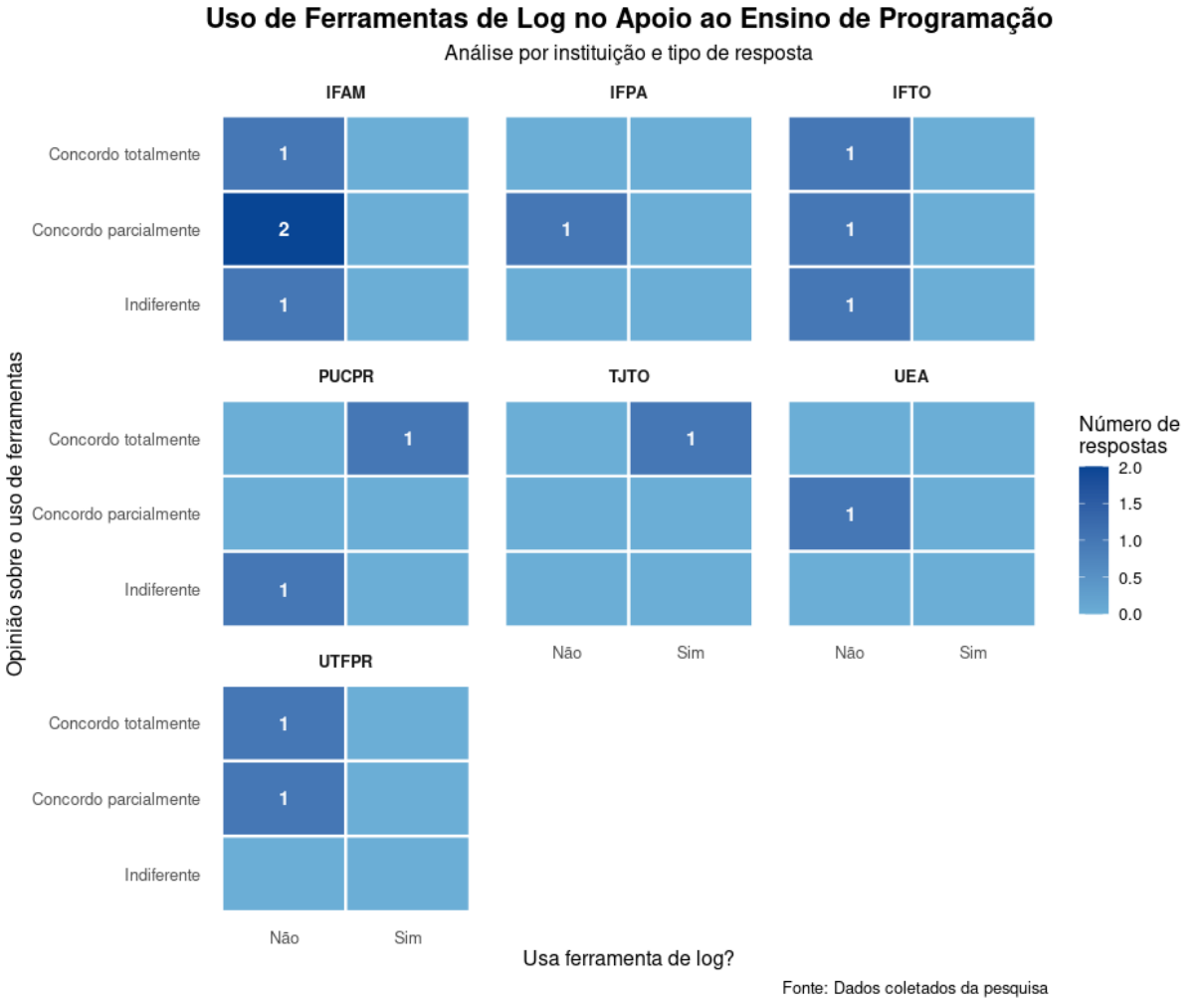

Figura 7 – Mapa de calor representando a frequência de uso e familiaridade com diferentes ferramentas tecnológicas (Perguntas 2 e 3). As cores mais quentes indicam maior frequência de uso ou familiaridade, permitindo identificar quais ferramentas são mais dominadas pelos participantes da pesquisa.

O mapa de calor apresentado na Figura 7 revela padrões interessantes sobre o conhecimento e uso de ferramentas tecnológicas entre os participantes. Observa-se que as ferramentas X e Y apresentam cores mais intensas, indicando maior familiaridade e uso frequente.

B.3.2 Comparações entre Grupos

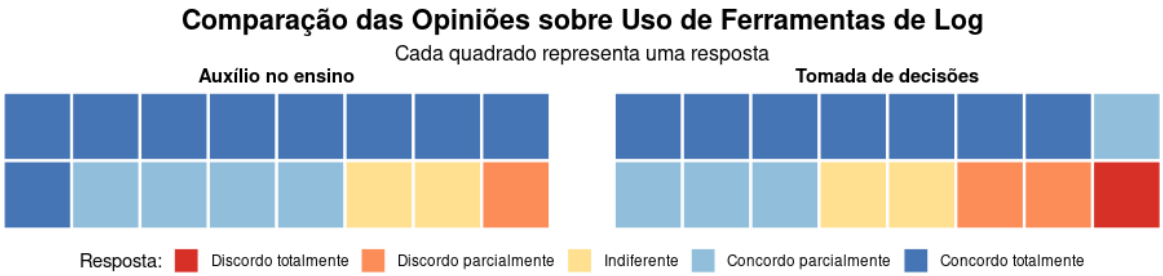

Figura 8 – Gráfico Waffle comparando características ou percepções entre diferentes grupos de participantes (Perguntas 4 e 5). Cada célula representa uma unidade de resposta, permitindo visualizar proporções e distribuições de forma intuitiva.

O gráfico waffle da Figura 8 oferece uma visualização clara das diferenças entre grupos analisados. Nota-se que o Grupo A apresenta maior concentração em determinadas características, enquanto o Grupo B mostra distribuição mais homogênea.

B.3.3 Análise de Dataset e Correções

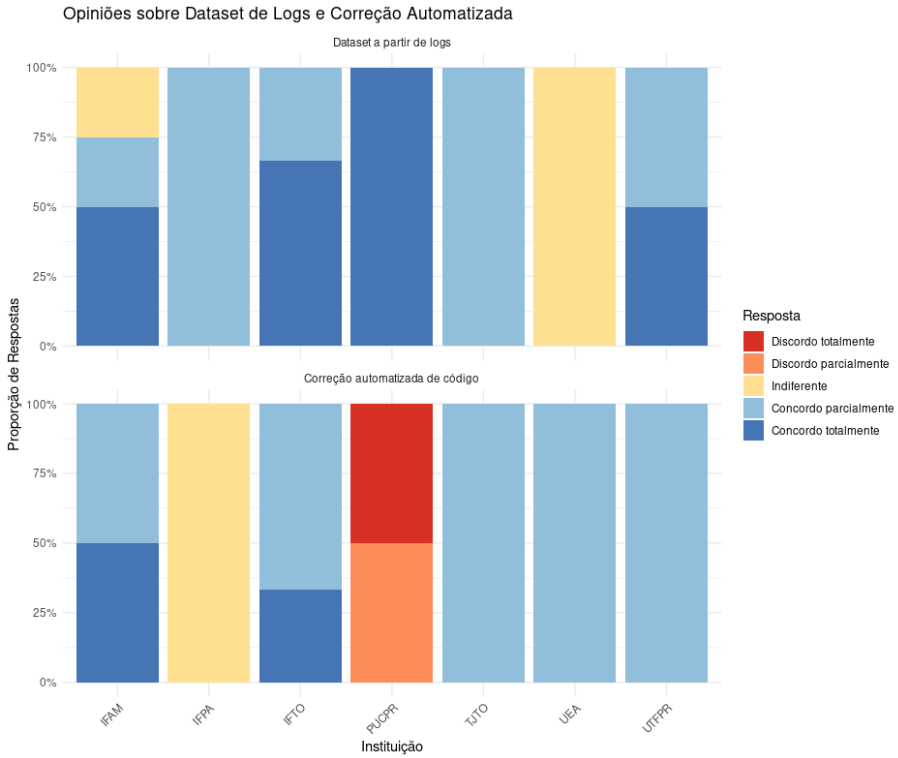

Figura 9 – Gráfico de barras agrupadas com facetas mostrando a distribuição de respostas sobre uso de datasets e sistemas de correção automatizada (Perguntas 6 e 7). As facetas permitem comparar subcategorias de respostas simultaneamente.

Conforme ilustrado na Figura 9, observam-se padrões distintos nas respostas sobre uso de datasets e sistemas de correção. A facetagem dos dados revela como diferentes subgrupos se comportam em relação a estas tecnologias.

B.3.4 Opinião sobre Plugins no Ensino

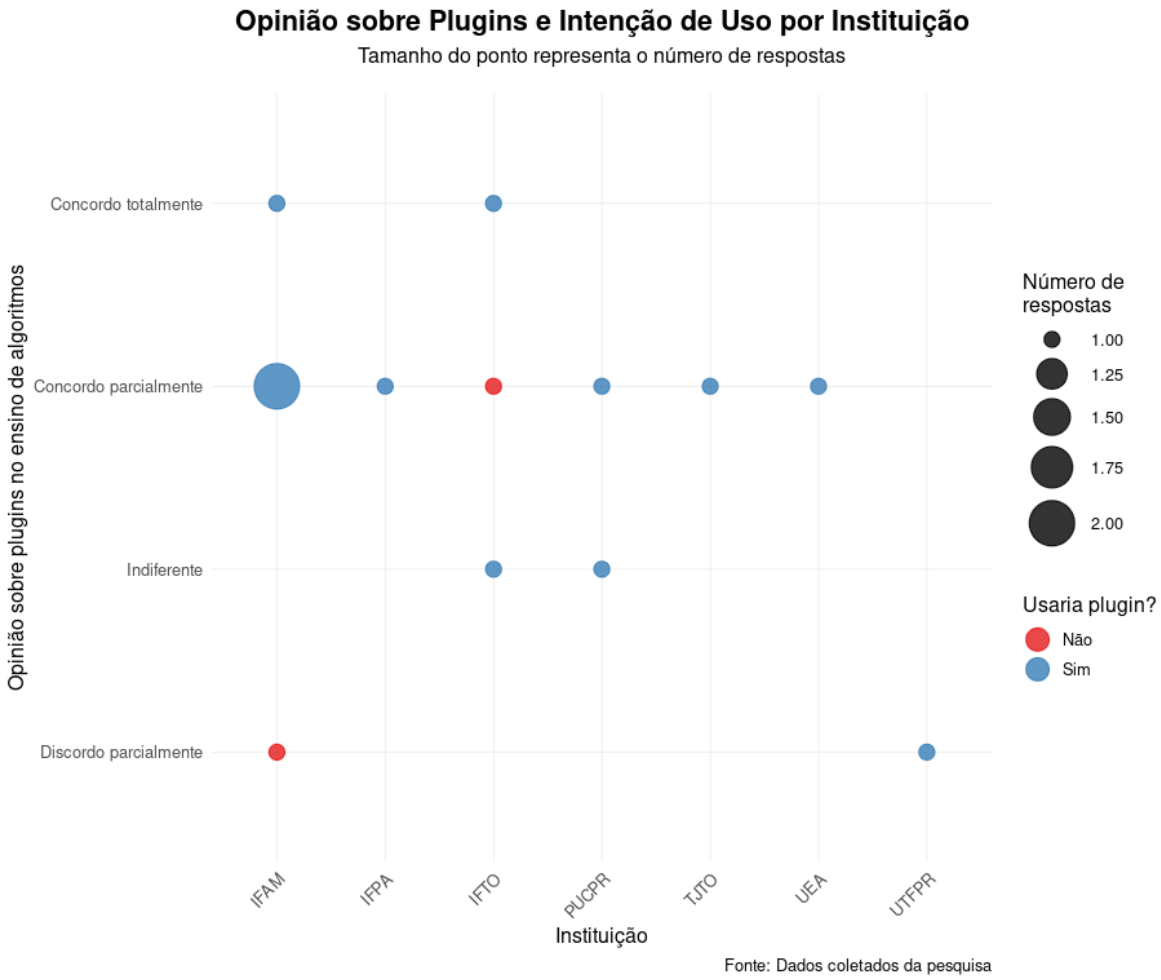

Figura 10 – Gráfico de pontos representando a relação entre a percepção sobre plugins educacionais e sua frequência de uso (Perguntas 9 e 11). A posição dos pontos indica a relação entre estas duas variáveis, podendo sugerir correlações.

A Figura 10 demonstra a relação entre a opinião sobre plugins no ensino e sua utilização prática. Nota-se uma tendência de avaliações mais positivas entre os usuários frequentes, sugerindo que a experiência prática influencia positivamente a percepção.

B.3.5 IDEs utilizados por Instituições

Análise de ferramentas de desenvolvimento utilizadas nas diferentes instituições. Esta visualização mostra a relação entre instituições e as IDEs utilizadas em suas aulas. Através do gráfico de barras, podemos identificar:

- O Visual Studio é a IDE mais popular entre as instituições, sendo utilizadas por seis
- A PUCPR e UTFPR utilizam predominantemente o Visual Studio
- o IFAM respondeu com uma maior diversidade de IDEs
- Algumas instituições demonstram variedade na escolha de ferramentas

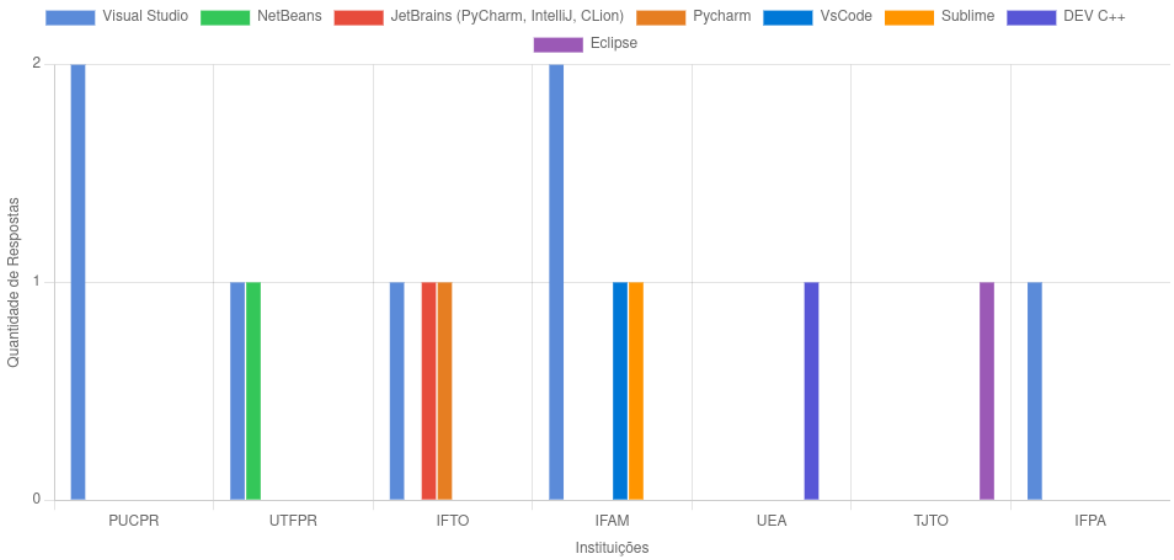

Figura 11 – Distribuição de IDEs utilizadas pelas instituições