Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

ProjPlag: Uma Aplicação para Auxiliar o Processo de Detecção de Plágio em Cursos Introdutórios de Programação

Rodrigo Cardoso Aniceto

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Orientadora
Prof.a Dr.a Maristela Terto de Holanda

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

ProjPlag: Uma Aplicação para Auxiliar o Processo de Detecção de Plágio em Cursos Introdutórios de Programação

Rodrigo Cardoso Aniceto

Dissertação apresentada como requisito parcial para
conclusão do Mestrado em Informática

Prof.a Dr.a Maristela Terto de Holanda (Orientadora)
CIC/UnB

Prof.a Dr.a Dilma Da Silva
Texas A&M University

Prof. Dr. Vinicius Ruela Pereira Borges
CIC/UnB

Prof. Dr. Ricardo Pezzuol Jacobi
Coordenador do Programa de Pós-graduação em Informática

Brasília, 12 de julho de 2023

Dedicatória

Dedico esse trabalho a minha esposa, Ana Rafaela, que esteve ao meu lado quando cada página estava sendo escrita.

Agradecimentos

Agradeço primeiramente a Deus, sem o qual nada do que existe teria sido feito. Agradeço à minha esposa, família, amigos e colegas de trabalho que com muita paciência souberam lidar com a dedicação que esse projeto exigiu. Agradeço à Universidade de Brasília pela oportunidade e ao Departamento de Ciência da Computação que se manteve atento as necessidades dos alunos, especialmente em momentos de pandemia. Agradeço também o apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), por fornecer o acesso ao Portal de Periódicos, tão útil a essa pesquisa. Por fim, agradeço a todos os professores que fizeram parte dessa jornada, em especial a Dr.a Maristela Holanda que orientou esse trabalho com dedicação e cuidado.

Muito obrigado a todos.

Resumo

Este trabalho apresenta a aplicação ProjPlag desenvolvida para auxiliar o professor a identificar alunos suspeitos de plágio em códigos fontes em uma disciplina introdutória de programação. Para criar a aplicação, foi feita uma análise das ferramentas de detecção de plágio Moss e JPlag e dos dados dos alunos extraídos da plataforma de ensino Moodle, isso permitiu aprender mais sobre o perfil de alunos que copiam códigos e, com base nisso, é possível traçar estratégias voltadas a reduzir a ocorrência dessa prática. A aplicação foi então desenvolvida com a funcionalidade de gerar relatórios integrando os dados disponíveis. Esses relatórios contêm percentuais de similaridades entre códigos, notas, padrões de desenvolvimento e padrões de envio dos trabalhos, para uso pelo professor. O ProjPlag também foi testado na automatização da identificação de alunos que plagiaram, apresentando uma taxa de acerto de 90%. Pesquisas futuras podem ser feitas aumentando o volume de dados e aperfeiçoando sua coleta.

Palavras-chave: ensino programação, código fonte, detecção de plágio, Moodle

Abstract

This project presents the ProjPlag application developed to assist instructors in identifying students suspected of source codes plagiarism in an introductory programming course. To make this possible, an analysis was made of the plagiarism detection tools Moss and JPlag and of the students' data extracted from the Moodle teaching platform, this analysis also allowed to learn more about the profile of students who cheated and can be used to develop strategies to reduce the occurrence of this practice. The application was then developed with the functionality of generating reports integrating the available data. These reports contain similarity percentages between codes, grades, development patterns and assignment submission patterns, for use by the instructor. ProjPlag was also tested for automating the identification of students who plagiarized, with a success rate of 90%. Future research can be done by increasing the volume of data and improving the way it is collected.

Keywords: programming teaching, source code, plagiarism detection, Moodle

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Contextualização

Plágio é definido como o processo ou prática de uma pessoa usar as ideias ou o trabalho de outro como sendo de autoria própria [2]. Trata-se de uma prática que pode ser observada em diferentes contextos, como na literatura, artes plásticas e música. Pode ser visto também no meio acadêmico, em publicações científicas com trechos ou conteúdos de outros autores sem os devidos créditos. Em se tratando de ambientes de ensino, o plágio pode ocorrer quando um aluno copia de uma fonte não autorizada as respostas de uma prova ou trabalho [3].

A grande quantidade de informações disponíveis on-line e seu fácil acesso, além da rapidez nas comunicações, tornaram mais simples a tarefa de quem quer plagiar algum conteúdo. No entanto, é importante não ignorar esse problema, primeiramente por questões éticas, mas também por outros motivos, como as possíveis implicações financeiras dependendo do setor envolvido. No caso específico do plágio em ambientes de ensino, pode prejudicar o desenvolvimento do aluno e afetar negativamente a reputação da instituição de ensino onde ocorre [4].

O tipo de plágio estudado nesta dissertação de mestrado é o de códigos fontes, com o foco restrito às disciplinas de programação de nível universitário. Um exemplo disso é o professor passar um trabalho que envolva a implementação de software, e os alunos, sem autorização, copiarem trechos do código da solução entre si ou da internet. Situações como essa são comuns e amplamente pesquisadas ao redor do mundo, a literatura recente mostra o crescimento do número de artigos que buscam lidar com esse problema [1].

Quanto ao enfrentamento da questão, existem diferentes ferramentas de detecção de plágio em códigos fontes já estabelecidas, como SIM, Plaggie, Sherlock e Moss [5], elas se baseiam principalmente na comparação dos códigos para determinar seus níveis de

similaridade. Aqueles que apresentarem uma semelhança acima do que seria classificado como normal são considerados suspeitos e devem ser investigados.

Além das ferramentas tradicionais, existem também outras abordagens com propostas que fazem análises mais complexas. Por exemplo, ferramentas que usam técnicas de inteligência artificial para analisar características do código, tais como tamanho de linhas, frequência de comentários, espaços em branco, etc, com o objetivo de identificar quem é o autor original de um determinado trabalho [6] [7]. Nessa mesma linha foram encontradas soluções que registram todas as alterações no código no momento em que são efetuadas, de modo que essas soluções observam o processo de desenvolvimento, identificando o padrão de codificação comum em comparação com o de quem está copiando [8].

Por fim, existem soluções integradas à plataforma de ensino usada no curso, onde são coletados dados do progresso do aluno na disciplina e também dos códigos de submissões intermediárias efetuadas antes da entrega final do trabalho [9]. Com base nisso, é possível inferir o nível de dedicação e interesse do aluno na disciplina, assim como as características individuais de cada perfil de aluno. Esse tipo de solução pode enriquecer os dados extraídos das ferramentas de comparação de códigos e auxiliar a tomada de decisão por parte do professor.

O estudo dessas ferramentas e pesquisas possibilita observar como está o andamento da área. A partir dessa análise, se pode ter o direcionamento para a criação de novas soluções que sejam relevantes no enfrentamento do problema do plágio em códigos fontes e também que sejam adequadas a realidade local.

1.2 Definição do Problema

Esta pesquisa está inserida no contexto da disciplina de programação Algoritmos e Programação de Computadores (APC) na Universidade de Brasília (UnB) entre os anos de 2020 e 2022. Nesse período, o processo de detecção de plágio nos códigos fontes dos projetos dos alunos estava sendo feito de maneira semiautomatizada.

Primeiramente, uma equipe formada por professores e monitores submetia os códigos dos alunos para serem analisados pelo Moss, a ferramenta de detecção de plágio adotada [10]. Os relatórios gerados por essa ferramenta contêm o percentual de semelhança entre os trabalhos e os trechos onde essas semelhanças acontecem. A partir desses dados, cabia à equipe criar uma planilha informando os códigos com similaridade classificada como alta e eventualmente juntar a ela outros dados que possam apontar para chances de ter havido plágio. Além disso, também eram criadas representações visuais das conexões entre os alunos com trabalhos similares.

Após a análise desses resultados, uma lista era produzida com os alunos suspeitos de plágio. Essa lista não possuía a função de ser taxativa, mas de fornecer indícios para auxiliar na tomada de decisões relacionadas ao aluno e à turma. É importante destacar que todo o processo de coleta dos dados do Moss, criação de planilhas e análise dos resultados era realizado manualmente, demandando tempo e recursos humanos significativos. Além disso, esse processo é sujeito a vieses e possíveis falhas humanas.

Outras dificuldades estariam presentes em eventuais tentativas de aprimorar o processo. Caso se decida enriquecer a análise adicionando novos dados da plataforma de ensino Moodle [11] utilizada, como por exemplo incluir as datas dos envios dos trabalhos nos parâmetros analisados, se aumentaria também a complexidade dessa fase de coletas. Além disso, não seria viável adicionar mais ferramentas de detecção de plágio em conjunto com o Moss. Também se deve ressaltar que o processo não é escalável em caso de um eventual aumento significativo no número de alunos.

1.3 Objetivos

1.3.1 Objetivo Geral

Este trabalho tem como objetivo o desenvolvimento de uma aplicação que utilize ferramentas de detecção automática de plágio e os dados dos alunos disponíveis na plataforma Moodle para gerar relatórios que auxiliem o professor em sua tomada de decisão. Para alcançar esse objetivo, se propõe uma análise dos dados disponíveis a fim de verificar sua relevância na identificação de alunos suspeitos e de que forma os usar para simplificar e aprimorar o processo de detecção.

1.3.2 Objetivos Específicos

Com o fim de alcançar o objetivo geral proposto, foram traçados os seguintes objetivos específicos:

1. Definir uma ferramenta de detecção automática de plágio que possua relevância acadêmica e a testar em conjunto com o Moss, a fim de comparar e avaliar o desempenho de ambas;
2. Verificar entre os dados que podem ser exportados da plataforma Moodle, aqueles que tiverem relevância na identificação de alunos suspeitos de plágio, a fim de se obter *insights* que possam ajudar o ensino de programação em contexto universitário;

3. Implementar a aplicação que automatize a execução das ferramentas de detecção de plágio e a leitura dos dados do Moodle, e que gere relatórios e visualizações contendo informações para auxiliar o professor em suas tomadas de decisão;
4. Implementar e testar uma verificação automática de chances plágio com base nas análises anteriores.

Nesse contexto, as contribuições desta dissertação englobam os resultados da análise dos dados da turma de programação extraídos do Moodle, bem como a aplicação desenvolvida para automatizar a elaboração de relatórios de suspeita de plágio.

1.4 Estrutura do Documento

Tendo em vista os objetivos pretendidos, este trabalho é dividido da seguinte forma:

- O Capítulo 2 apresenta o referencial teórico necessário para embasar a pesquisa. Nele, é feita uma breve revisão na literatura acadêmica sobre o assunto estudado e são dadas informações sobre as principais abordagens relacionadas à detecção de plágio e as ferramentas que auxiliam nesse processo. Em seguida, são apresentadas características gerais do Moodle e, por fim, os trabalhos relacionados na área.
- O Capítulo 3 descreve os dados disponíveis para serem utilizados nos experimentos. São realizadas análises desses dados com o fim de cumprir os objetivos estabelecidos.
- O Capítulo 4 faz a descrição da aplicação proposta, incluindo seu funcionamento e aspectos da sua implementação. Também apresenta os resultados dos experimentos feitos com ela.
- O Capítulo 5 apresenta as conclusões finais, bem como sugestões para possíveis trabalhos futuros na área.

Capítulo 2

Referencial Teórico

Este capítulo apresenta uma revisão teórica sobre os assuntos abordados na pesquisa. Na primeira seção, é descrita uma breve revisão na literatura acadêmica sobre plágio em códigos fontes e dado um panorama geral sobre as principais tecnologias de detecção e as características do combate ao plágio em ambientes de ensino. A segunda seção aborda a plataforma Moodle, ferramenta de ensino usada neste projeto. A terceira e última seção descreve os trabalhos relacionados e como se buscou diferenciar-se deles.

2.1 Literatura sobre Plágio em Códigos Fontes

Segundo Helgesson em [12], plágio se define como "usar um produto intelectual de outra pessoa (como textos, ideias e resultados) de modo que se implique que ele é de autoria própria", tradução livre. A partir desse conceito, é feita uma diferenciação entre plágio e roubo, pois o roubo seria fazer uso não autorizado de um objeto físico ou escasso de outra pessoa, e plágio o uso de algo não tangível como uma ideia. Deve-se destacar que o plágio pode ocorrer de maneira não intencional, e está presente em diferentes contextos, como na música, literatura, artes plásticas, design e produções científicas [12].

Em se tratando de programação, em [1] foi feita uma revisão na literatura envolvendo 96 artigos científicos sobre plágio em códigos fontes em ambientes universitários entre 2015 e 2020 que é estudada aqui. Entre seus resultados, foi possível notar que existem publicações de vários lugares do mundo tratando desse tema, conforme mostra a Figura 2.1. A lista de países e seus respectivos números de artigos estudados é composta por [1]: China(14), Estados Unidos(14), Índia(12), Indonésia(11), Austrália(8), Eslováquia(6), África do Sul(5), Croácia(4), Portugal(4), Alemanha(3), Brasil(3), Reino Unido(3), Rússia(3), Servia(3), Colômbia(2), Espanha(2), Finlândia(2), Irlanda(2), Japão(2), Taiwan(2), Áustria(1), Bangladesh(1), Bélgica(1), Bósnia e Herzegovina(1), Coréia do Sul(1), Eslovênia(1), Grécia(1), Jordânia(1), Macedônia do Norte(1), Malásia(1),

Noruega(1), Paquistão(1), Polônia(1), Romênia(1), Tailândia(1) e Turquia(1). Essa ampla variedade de localizações e, consequentemente, de autores evidencia que se trata de um assunto de interesse global.

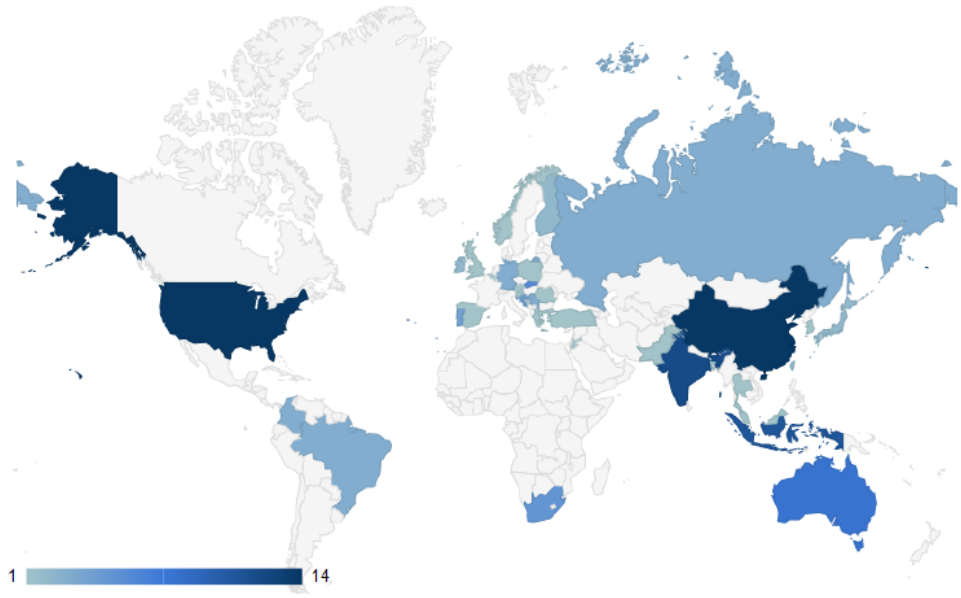


Figura 2.1: Distribuição de artigos por país [1].

Outra constatação obtida é que o número de publicações aumentou ao longo dos anos estudados. A Figura 2.2 mostra um gráfico com o número de artigos que tratam do enfrentamento ao plágio em códigos fontes, encontrados para cada ano do período estudado [1], sendo visível que o interesse nesse assunto tem crescido de forma quase contínua.

Ainda se tratando do número de artigos por ano, neste trabalho de mestrado, foi decidido por repetir a busca realizada em [1] utilizando os mesmos critérios de inclusão e exclusão de artigos, a fim de verificar quantas publicações foram feitas nos anos de 2021 e 2022 que não haviam sido abordadas. Essa busca identificou 37 artigos de 2021 e 36 de 2022. Isso indica que o interesse no assunto teve um novo aumento recente, causado principalmente devido às adaptações ao ensino causadas pela pandemia de COVID-19 [13].

No que se refere às linguagens de programação, há interesse em saber quais são as adotadas nos cursos em que foram realizadas pesquisas sobre plágio. Esse aspecto também foi verificado na revisão [1] que compilou as linguagens mais mencionadas nos artigos analisados. A Tabela 2.1 apresenta essa lista de linguagens, juntamente com o número de publicações que tratou delas em suas pesquisas. Observa-se que Java, C++ e C receberam maior destaque. No entanto, é importante ressaltar que várias soluções são genéricas e independentes da linguagem de programação utilizada pelos alunos.

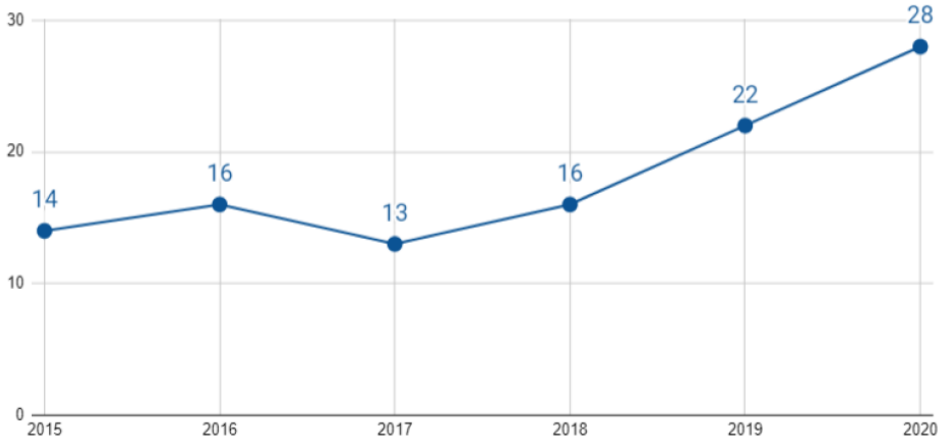


Figura 2.2: Número de artigos publicados por ano [1].

Tabela 2.1: Linguagens de programação mais abordadas nos artigos sobre plágio [1].

Linguagem	**Número artigos**
Java	30
C++	19
C	18
Python	8
C#	6
PHP	5
JavaScript	2
Assembly	1
Go	1
Pearl	1

Os demais aspectos ligados às ferramentas de detecção e abordagens sobre como lidar com o plágio nos contextos educacionais são tratados nas subseções a seguir. As informações apresentadas foram extraídas dos artigos estudados na revisão [1] e também de outras pesquisas posteriores.

2.1.1 Ferramentas de Detecção de Plágio

Existem várias abordagens para a detecção automática de plágio em códigos fontes. A principal delas se baseia na comparação direta entre os códigos para determinar seu nível de similaridade. Isso é feito por meio do uso de variantes do algoritmo de "tokenização", no qual os códigos são divididos em segmentos menores chamados *tokens* e comparados entre si [1]. Além disso, espaços em branco e comentários são removidos para efetuar a

análise. Essa abordagem é usada pelas ferramentas mais citadas na literatura científica recente, dentre elas, destacam-se [1]:

- Moss[10]: É a ferramenta mais mencionada na literatura recente e compatível com 24 linguagens de programação, incluindo C, C++, Java, C#, Python, JavaScript, Haskell, Perl, Matlab e MIPS Assembly. O processo de comparações de códigos do Moss é executado em um servidor remoto hospedado na Universidade de Stanford. Para utilizar a ferramenta, o professor deve submeter os códigos por meio de um script de envio que lhe é fornecido. Os resultados são então disponibilizados em uma página da web, acessível por até 14 dias, permitindo ao professor fazer o download dos resultados ou executar novamente a ferramenta, se necessário [14]. O Moss apresenta os percentuais de similaridade entre os códigos e destaca os trechos iguais. Ele é a ferramenta que era utilizada na disciplina que é contexto deste trabalho.
- JPlag[15]: A segunda ferramenta mais citada na literatura, foi desenvolvido em Java e roda localmente, necessitando apenas de uma *Java Virtual Machine* (JVM) para funcionar. A documentação oficial informa que é compatível com 5 linguagens de programação (Java, C#, C, C++ e Scheme) e linguagem natural, porém uma análise em seu código fonte [15] seguida de testes mostrou ser compatível também com Python. Os resultados são gerados no formato HTML similar à interface do Moss.
- SIM[16]: Ferramenta que ocupa a terceira posição entre as mais citadas, foi programada em linguagem C e roda localmente. Desenvolvida com o código aberto e que não possui interface gráfica para execução ou para exibição de resultados. É compatível com as linguagens de programação C, Java, Pascal, Modula-2, Lisp e Miranda, além de linguagem natural.
- Plaggie[17]: É um projeto *open-source* baseado em uma versão antiga do JPlag, e apenas compatível com a linguagem Java. Existem poucas documentações detalhando seu funcionamento. Possui interface gráfica para exibição dos resultados em formato HTML.
- Sherlock[18]: É a quinta ferramenta mais citada, se trata de uma solução Java em que os resultados são exibidos na forma de um grafo interativo mostrando conexões entre os códigos e o percentual de similaridade. É compatível com C++ e Java.

Conforme detalhado no Capítulo 3, a ferramenta escolhida para ser utilizada neste projeto, em conjunto com o Moss, é o JPlag. As interfaces de resultados delas são exibidas nas Figura 2.3 e Figura 2.4, respectivamente. Em ambas é possível ver os dois trabalhos que estão sendo comparados, o uso de cores indica os trechos classificados como iguais,

Figura 2.3: Interface do Moss.

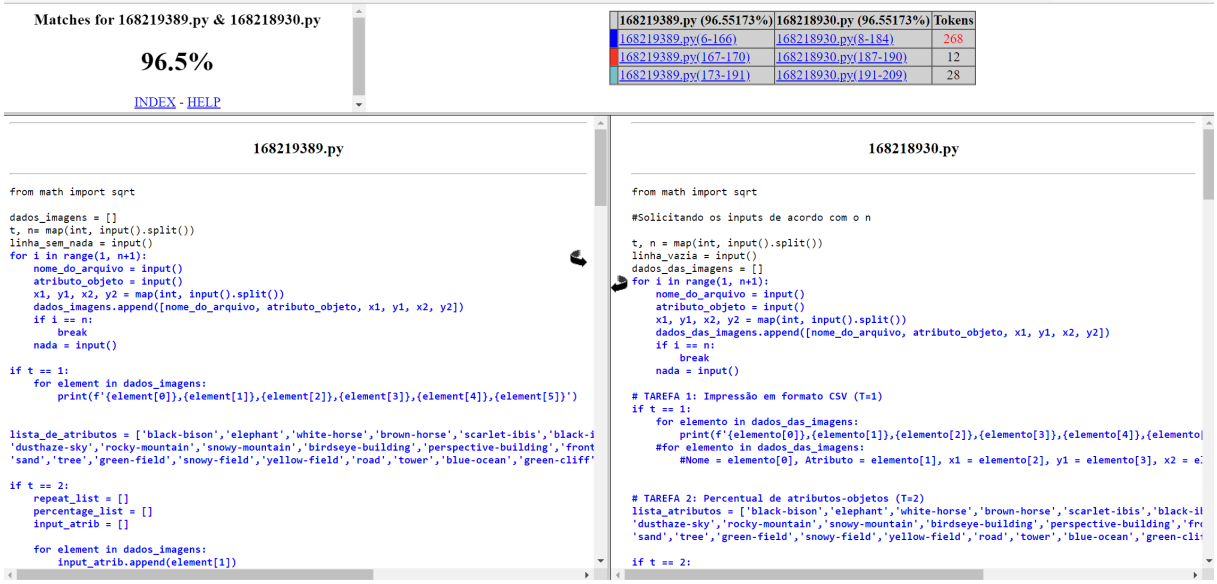


Figura 2.4: Interface do JPlag.

também se nota um painel superior em cinza que mostra os números das linhas onde essas semelhanças ocorrem, juntamente com os percentuais de similaridade. É importante observar que nesses testes foi usado o mesmo par de códigos, evidenciando a diferença entre os resultados das duas ferramentas. No caso do Moss, o trabalho da esquerda foi classificado como 72% semelhante ao da direita, enquanto o da direita 75% semelhante ao da esquerda. Para o JPlag, os dois trabalhos possuem 96,5% de semelhança um com o outro. Acredita-se que essa disparidade nas duas análises possa ser usada experimentalmente

para identificar possíveis falsos negativos ou falsos positivos nos resultados.

Propostas Alternativas

Além das ferramentas mencionadas anteriormente, é importante ressaltar a existência de várias soluções que empregam diferentes tecnologias para realizar comparações [19] [20] [21]. Sobre abordagens alternativas para a detecção de plágio, se destacam ferramentas que atuam antes da submissão final do trabalho. Essas soluções acompanham o progresso do código enquanto está sendo escrito e fazem comparações entre as submissões intermediárias [22] [23]. Dentro dessa abordagem, há soluções que não necessitam sequer de comparações entre diferentes alunos, pois capturam em tempo real o código sendo desenvolvido e identificam qualquer padrão de desenvolvimento suspeito [8].

Uma consideração importante ao implementar soluções personalizadas é a necessidade de adaptar o modelo de ensino, exigindo que os alunos utilizem um repositório para armazenar o código em desenvolvimento ou até mesmo uma plataforma de programação on-line fornecida pela instituição.

Outra abordagem que pode aprimorar a qualidade das buscas por plágio é a observação do estilo de programação do aluno. Um software coleta padrões presentes no código dos estudantes, como o tamanho de linhas, a frequência de comentários, os nomes de variáveis, os espaços em branco, etc. Em seguida, usa técnicas de inteligência artificial para aprender os estilos de programação de cada aluno e identificar eventuais códigos destoantes. Essa abordagem é complementar às comparações tradicionais e também auxilia na identificação do autor original de um código em meio as cópias [6] [7].

A escolha da ferramenta e metodologia para lidar com casos de plágio em uma disciplina de programação é de grande importância e deve ser feita visando a que melhor se adequa a realidade e objetivos da instituição de ensino.

2.1.2 Aspectos Ligados à Detecção Automática de Plágio

Segundo Pierce e Zilles em [24], existem evidências indicando que em disciplinas que envolvem provas e trabalhos, alunos que praticam o plágio tendem a obter notas mais altas nos trabalhos copiados, porém, notas menores nas provas e na disciplina como um todo. A diferença de pontuação é ainda mais significativa quando o aluno copia múltiplos trabalhos, impactando também seu desempenho em disciplinas futuras do curso [24]. Essa correlação pode sugerir que o plágio resulta em notas mais baixas em tarefas posteriores, pois os alunos não tiveram a oportunidade de praticar e desenvolver seu aprendizado, ou pode indicar que eles já apresentavam dificuldades de aprendizado, o que os levou a recorrer à cópia [24]. Independentemente disso, o uso das ferramentas de detecção de

plágio surge como uma alternativa para identificar essas situações, a fim de avaliar o aprendizado dos alunos e desenvolver estratégias para evitar essa prática.

Entretanto, ao decidir utilizar uma ferramenta de detecção, é importante considerar alguns aspectos. O primeiro elemento a ser discutido é a recente popularização de ferramentas de Inteligência Artificial, como o GitHub Copilot [25], que podem ser usadas para facilitar o plágio em códigos fontes e são capazes de burlar muitas das tecnologias de detecção atual [26]. Essas ferramentas devem receber atenção especial em pesquisas futuras na área, com o fim de adaptar o ensino de programação às novas mudanças tecnológicas.

O nível aceitável de similaridade entre os códigos é um elemento que também deve ser discutido previamente, levando em conta o grau de colaboração permitido entre os alunos. Em alguns casos, optou-se por não utilizar mais ferramentas de detecção de plágio, devido à ênfase da instituição de ensino em trabalhos mais colaborativos [27].

Também pode-se observar que, em determinados contextos, os professores podem encorajar os alunos a buscar algoritmos prontos na internet para serem incorporados em seus trabalhos, pois essa é uma prática comum no ambiente profissional. Além disso, a análise de códigos de exemplo desempenha um papel importante no processo de aprendizagem da programação.

O professor também deve levar em consideração que os códigos que apresenta à turma servirão de exemplos para a codificação dos alunos, o que pode resultar em trabalhos semelhantes, uma vez que eles adotam o mesmo estilo de programação visto em sala de aula [28]. Esse efeito é intensificado em tarefas simples, pois possuem um número menor de soluções possíveis.

Um outro aspecto que deve ser observado é a maneira como a turma é informada sobre as ferramentas de detecção. Existe na literatura o registro de que alunos que não copiaram podem ficar com medo de serem pegos em uma ferramenta de detecção de plágio, a depender de como foram informados sobre a ferramenta, e isso pode levá-los a fazer alterações desnecessárias no código, tornando-o potencialmente mal escrito e menos eficiente [28].

Por fim, é importante ter em mente que as ferramentas são auxiliares no processo de identificação de plágios, porém não devem ter autonomia para decidir quem será responsabilizado ou não. Além disso, não se pode esperar que seu funcionamento seja livre de erros ou vícios [29], cabendo ao professor e sua equipe as definições finais. Uma possível maneira de verificar a autoria de um trabalho suspeito é pedir para o aluno explicar oralmente o código que alega ter escrito [8].

Reduzir as Ocorrências de Plágio

Sobre maneiras de reduzir a ocorrência de plágio em códigos fontes, a fim de diminuir a dependência de ferramentas, sabe-se que a não repetição de trabalhos de semestres anteriores pode ser positiva, pois a literatura fala da ampla presença de trabalhos copiados de turmas anteriores na mesma disciplina [24]. Os alunos podem criar uma base de dados com as soluções de tarefas recorrentes no curso.

Outra maneira menos óbvia de reduzir o plágio está relacionada ao modo como são feitas as correções. Foi notado em [30] que os alunos podem perder a motivação nas atividades da disciplina a depender de como receberam o feedback automático com a avaliação de sua tarefa. Um sistema que dá as notas de forma imediata poderia levar os alunos a copiarem mais ao verem seu desempenho eventualmente estar abaixo do esperado [30].

Os esforços educacionais voltados à conscientização dos alunos a respeito desse tema também podem ser sugeridos como uma maneira complementar de reduzir a ocorrência de plágio, independente da metodologia de ensino adotada [31].

Essas informações mostram a relação entre a metodologia em que as atividades da turma são desempenhadas e a incidência de plágios entre os alunos, cabendo novos experimentos práticos a fim de se continuar aperfeiçoando as maneiras de lidar com o tema.

2.2 Plataforma Moodle

O Moodle é um ambiente virtual de aprendizagem desenvolvido inicialmente por Martin Douglas em 1999, se baseia em princípios de colaboração e construção do conhecimento, permitindo a interação dos estudantes entre si e com o professor. Ele pode ser usado no ensino a distância, semipresencial ou presencial como uma ferramenta de suporte à disciplina. O Moodle tem como lema a ideia de que o aluno é responsável pela aquisição do conhecimento, cabendo a este a necessidade de ter autonomia na busca por materiais e a troca de informações, características essenciais na era digital [32].

Trata-se de uma plataforma aberta e usada amplamente ao redor do mundo. Por meio do Moodle, o professor cria e administra um site privado onde pode postar materiais e recursos a serem acessados pela turma, também usa ferramentas administrativas para gerenciar o andamento das atividades. É possível compartilhar vídeos, apostilas, criar questionários e calcular notas de forma automática, entre outras funcionalidades. O Moodle possui fóruns internos para discussão e tirar dúvidas, é customizável, permitindo a criação de extensões que adicionam novas funcionalidades de acordo com as necessidades de cada disciplina [11].

No que se refere aos aspectos técnicos, o Moodle foi desenvolvido em PHP e tem o código aberto. Além disso, faz uso de um sistema gerenciador de banco de dados (SGBD) PostgreSQL ou MySQL, e de um servidor web como o Apache, devendo ser preferencialmente instalado em um servidor Linux [33]. A Figura 2.5 ilustra a interface padrão da página inicial de um administrador no Moodle, em que é possível ver a lista de cursos disponíveis ao centro e as opções administrativas à esquerda.

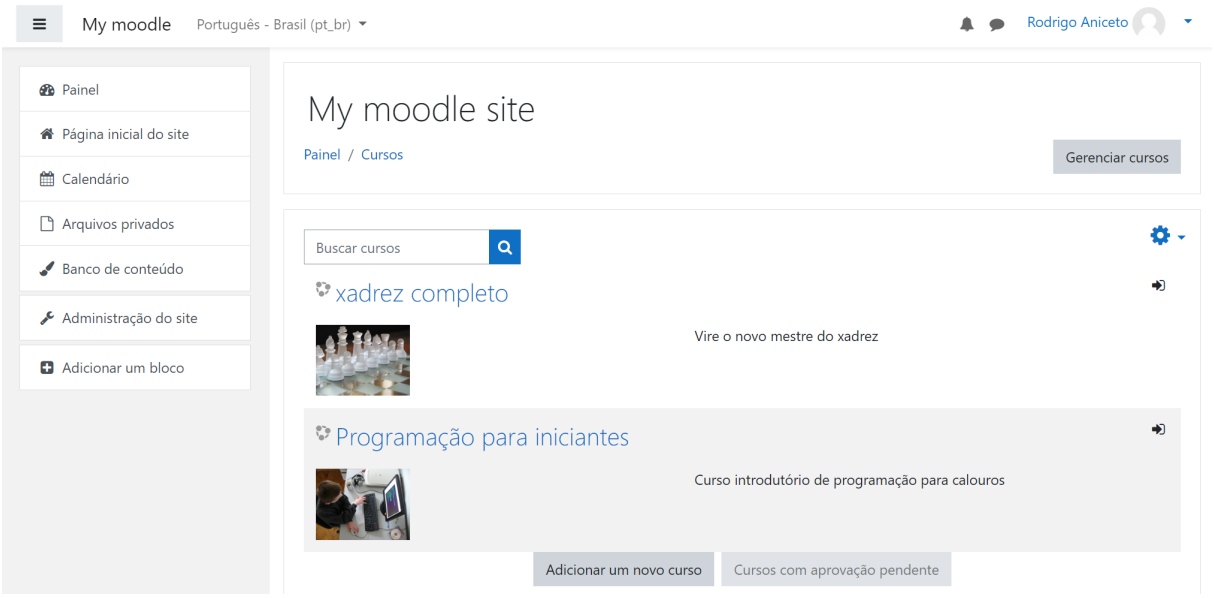


Figura 2.5: Interface do Moodle.

O Moodle é a plataforma usada na disciplina introdutória de programação que este trabalho se foca. Os alunos se cadastravam e tinham acesso a páginas com vídeos e materiais diversos. Haviam áreas do site para preenchimento de questionários e envio de códigos fontes de trabalhos, que eram corrigidos automaticamente por meio da extensão CodeRunner que foi instalada [34]. As tarefas dos alunos ficavam disponíveis para professores e monitores acessarem, e planilhas com informações sobre a participação do aluno, notas, dados dos acessos e envios de trabalhos também estavam disponíveis.

2.3 Trabalhos Relacionados

Para se fazer uma lista de trabalhos relacionados, foi analisada primeiramente a lista de artigos observados na revisão de literatura [1]. Foi possível perceber que a grande maioria deles não faz uso de dados de desenvolvimento de códigos pelo aluno ou dados da disciplina, mas focam majoritariamente em algoritmos que aperfeiçoam a comparação entre códigos fontes [1]. Existem, porém, alguns artigos que se diferenciam por se aproximar

desses temas. As publicações que são analisadas ao longo desta seção foram extraídas dessa revisão de literatura ou de novas pesquisas no assunto feitas posteriormente.

A Toolset for Mining Github Repositories in Educational Software Projects

Smith [9] se baseia em minerar dados de repositórios de estudantes para fazer análises do processo de desenvolvimento de software. É afirmado que na Universidade Texas A&M, nos Estados Unidos da América, foi disponibilizado aos alunos o GitHub [35] como plataforma de armazenamento e versionamento de códigos. A partir disso, foi desenvolvida uma aplicação que extrai *logs* de desenvolvimento dos alunos do GitHub. Entre os dados que foram coletados está o número de submissões, número de linhas alteradas e tempo que o aluno gastou em determinado projeto.

Os dados obtidos foram apresentados em forma de planilhas e gráficos para análise. Isso permitiu que fossem formuladas hipóteses sobre a relação entre o número de submissões e o grau de dificuldade que os alunos estão tendo na disciplina, além da possibilidade de estarem realizando atividades suspeitas, entre outros [9]. Este trabalho mostrou a possibilidade do uso de estatísticas ligadas ao desenvolvimento de códigos como um material útil para o andamento da disciplina.

Mining Student Submission Information to Refine Plagiarism Detection

Catalena [36], também da Texas A&M, criou uma solução que se integra a uma plataforma de avaliação automática e a uma ferramenta de detecção de plágio. As tarefas dos alunos eram corrigidas por uma ferramenta on-line chamada Mimir Classroom [37]. Foi então criada uma solução que coleta dados do Mimir, os analisa e produz *datasets* e visualizações que ajudam o professor a refinar resultados de detecção de plágio.

Entre os dados que a solução coletou estão os códigos de cada submissão (foi usado um software que envia os códigos para um servidor quando são compilados), o número de linhas alteradas e os números e horários das submissões. A solução gerou gráficos analisando o comportamento da turma a partir do qual foram tiradas conclusões, como a de que os alunos que enviaram os trabalhos com antecedência tiraram notas maiores.

A ferramenta de detecção de plágio usada era o Moss, foi possível usá-lo para identificar quem é o autor original de um código, observando que seu código seguiu um padrão de crescimento normal, em comparação com de outro aluno que teve um crescimento repentino logo após seu colega finalizar [36].

Integrating an Online Compiler and a Plagiarism Detection Tool into the Moodle Distance Education System for Easy Assessment of Programming Assignments

Kaya [38] tem como foco a plataforma Moodle na Universidade Çukurova, na Turquia. Foi proposto adicionar recursos ao Moodle para simplificar as atividades de uma disciplina de programação. A pesquisa notou que existem *plugins* voltados à detecção de plágio em códigos fontes, porém muitos destes tem seu uso limitado a versões específicas do Moodle ou foram descontinuados.

Para contornar isso, se atuou diretamente no código fonte do Moodle, fazendo alterações para acrescentar novas funcionalidades. Foi adicionado um compilador GCC para compilar os códigos fontes e a detecção de plágio foi automatizada usando o Moss. Assim, todas as tarefas ligadas a correção puderam ser feitas dentro do próprio Moodle, sem que seja necessário ao professor baixar arquivos em sua máquina.

Ao fim, se pôde demonstrar que a compilação e execução remota dos códigos reduziu significativamente o tempo gasto com correções. Também se observou que o uso de ferramentas de detecção de plágio evitou que alunos copiassem e também melhorou o desempenho da turma [38].

Automated Plagiarism Detection for Computer Programming Exercises Based on Patterns of Resubmission

Tahaei [22] da Universidade da Califórnia, Estados Unidos da América, foca no problema relacionado a alunos que encontram soluções na internet em vez de plagiar de um colega, dificultando o uso de ferramentas de detecção de similaridade. Foi apresentada uma solução que acompanha o desenvolvimento dos códigos sem a necessidade de comparações entre alunos.

No contexto dessa pesquisa, os alunos foram incentivados a enviarem o código várias vezes para receberem feedbacks, que eram baseados em casos de teste. A ferramenta proposta então coletava as diferenças entre os códigos intermediários dos alunos e gerava métricas para se fazer uma análise estatística que apontava a probabilidade de haver um comportamento suspeito. Os códigos testados também foram analisados por um professor que ajudou no treinamento da ferramenta.

Por fim, foi feita uma comparação entre os resultados obtidos pela solução proposta e os do Moss, mostrando uma taxa de acerto maior para a solução proposta, dentro os dados testados. A principal fraqueza da ferramenta foram os alunos que submetem apenas uma vez, necessitando do uso de outras abordagens. Os outros pontos fracos estão ligadas ao

aluno conhecer as métricas que são coletadas, esse risco, porém, está presente em outros tipos de solução [22].

Using Early Plagiarism Detection in Programming Classes to Address the Student's Difficulties

Fonseca et al. [23] da Universidade de Coimbra, Portugal, seguiu uma abordagem semelhante à do trabalho anterior, mas usando uma plataforma chamada CodeInsights, que envia para um servidor uma cópia do código fonte cada vez que o aluno o salva ou o executa.

A detecção de plágio foi feita de maneira tradicional, coletando o nível de similaridade entre os alunos. Ao detectar um caso suspeito, a plataforma enviava uma notificação ao professor. A intenção dessa pesquisa foi mais focada no professor poder ajudar os alunos com dificuldades na disciplina do que fazer uma abordagem punitiva [23].

2.3.1 Resumo dos Trabalhos

A Tabela 2.2 resume as principais características dos trabalhos relacionados em comparação com o projeto apresentado nesta dissertação. Para cada publicação, é apresentada a plataforma utilizada para a submissão dos códigos, a linguagem de programação adotada, as ferramentas de detecção de plágio usadas pelo professor, os dados coletados para análise e o momento da coleta desses dados. Embora os trabalhos listados tenham feito escolhas por tecnologias distintas, eles majoritariamente compartilham o foco em dados relacionados ao processo de desenvolvimento de código, como as linhas alteradas e tempo de programação dos alunos. A coleta de dados ocorre principalmente enquanto as atividades estão sendo feitas.

A solução proposta neste mestrado adota uma abordagem diferente, utilizando os dados dos alunos disponíveis na plataforma Moodle local, como notas e algumas informações de desenvolvimento. O foco é explorar esses dados para avaliar sua utilidade como auxiliar às ferramentas de detecção de plágio e buscar informações relevantes para o ensino de programação. A linguagem de programação usada pelos alunos é o Python, e as ferramentas de detecção de plágio utilizadas são o Moss e o JPlag. Uma outra característica deste projeto é a coleta de dados realizada posteriormente às submissões dos alunos, devido ao modo de operação do Moodle.

Tabela 2.2: Comparativo de trabalhos relacionados.

Trabalho	**Plataforma**	**Linguagem adotada**	**Ferramenta plágio**	**Dados adicionais**	**Momento da coleta**
Smith[9]	GitHub	não informou	não usa	processo de desenvolvimento	durante atividades
Catalena[36]	Mimir	C++	Moss, Mimir	processo de desenvolvimento	durante atividades
Kaya[38]	Moodle	C	Moss	não coleta	-
Tahaei[22]	não informou	C++	não usa	processo de desenvolvimento	durante atividades
Fonseca[23]	CodeInsights	PHP	CodeInsights	processo de desenvolvimento	durante atividades
Solução proposta	Moodle	Python	Moss, JPlag	métricas do Moodle	posterior atividades

Capítulo 3

Análise de Dados dos Alunos

Neste capítulo é feita uma descrição dos dados acadêmicos de alunos de programação coletados nesta pesquisa, explicando como foram produzidos e tratados. Em seguida, são detalhadas as análises efetuadas nesses dados em busca de descobrir maneiras de usá-los para a identificação de alunos suspeitos de plágio e de obter informações que possam ser úteis no ensino universitário de programação.

3.1 Dados Utilizados

Os dados utilizados para fazer os testes aqui descritos são provenientes de duas turmas da disciplina introdutória de programação da Universidade de Brasília que, para fins de simplificação, serão chamadas de Turma A e Turma B a partir deste ponto. A Turma A foi lecionada no segundo semestre acadêmico de 2020 e possuía 234 alunos, a Turma B ocorreu no segundo semestre de 2021 contendo 284 alunos. As atividades de ambas as turmas eram compostas por questionários, que são perguntas teóricas ou pequenas tarefas de programação sobre os assuntos dados em aula, e por projetos que são trabalhos de programação de maior complexidade e com maior prazo de entrega. As tarefas que envolvem programação foram feitas em Python e a detecção de plágio era aplicada apenas nos projetos, por serem códigos maiores e com maior importância na nota da disciplina.

A extração dos dados das turmas foi feita por meio da restauração, em ambiente simulado, de backups do Moodle que foram gerados ao longo das aulas. Esses backups permitem exportar planilhas contendo notas, dados pessoais dos alunos, horário em que eles visualizaram a descrição de uma tarefa, horário de envio da tarefa e o código fonte no caso de tarefas de programação. Juntamente com elas, também foi usada uma lista com códigos fontes de projetos que podem ser afirmados como plágio, a fim de validação dos testes. Esses dados de plágio foram obtidos a partir de listas de alunos suspeitos geradas

pelos professores da disciplina ao longo do curso, seguida de uma conferência manual dos códigos fontes em busca de casos que podem ter sido deixados de fora.

Outro tipo de dado utilizado neste experimento são os resultados da execução das ferramentas de detecção de plágio Moss e JPlag nos projetos. Trata-se de percentuais de similaridade entre cada par de códigos gerados pelas ferramentas de detecção em suas configurações padrão. Portanto, segue a lista final com os dados disponíveis para uso nesta pesquisa:

- Códigos fontes dos projetos;
- Resultados de similaridade dos projetos obtidos via Moss e JPlag;
- Lista de códigos com evidente ocorrência de plágio;
- Nome, matrícula e e-mail de cada aluno;
- Notas dos projetos e questionários;
- Horário que os alunos visualizaram as especificações dos projetos;
- Horário de envio dos projetos;
- Prazo limite de entrega dos projetos;

Esses dados foram coletados da Turma A e da Turma B. Porém, as análises descritas neste capítulo foram feitas apenas na Turma A, pois se optou por usar essa turma como fonte de conhecimentos a serem incorporados na aplicação criada posteriormente. Em seguida, a aplicação é testada na Turma B buscando validar sua eficácia em uma turma diferente da que havia sido estudada.

É importante informar que esta pesquisa recebeu parecer favorável do Comitê de Ética em Pesquisa (CEP) com parecer número 4.283.719, para a coleta e análise de dados educacionais, sendo registrada na Plataforma Brasil [39].

3.2 Resultados da Análise de Dados

Esta seção tem como objetivo apresentar os resultados obtidos a partir dos dados da Turma A de 2020-2. As variáveis estudadas são: resultados das ferramentas de detecção de plágio, prazo restante quando os alunos entregaram os projetos, tempo gasto entre a visualização e o envio dos projetos, e notas dos projetos e questionários. Para cada variável, é feita uma comparação entre os trabalhos identificados como plágio dos demais, a fim de se identificar diferenças de padrões entre eles. Em alguns casos, também foram comparadas as notas dos alunos visando obter informações adicionais sobre as características da turma.

Tabela 3.1: Percentuais de acertos no Moss.

Faixa percentual apontada pelo Moss	**Quantidade efetiva de plágios**
Acima de 50%	Todos (100%)
40% a 49%	85,7%
30% a 39%	42,8%
20% a 29%	38,4%
10% a 19%	3,2%
1% a 9%	1,3%

Tabela 3.2: Percentuais de acertos no JPlag.

Faixa percentual apontada pelo Jplag	**Quantidade efetiva de plágios**
Acima de 90%	Todos (100%)
80% a 89%	88,8%
70% a 79%	50%
60% a 69%	43,7%
50% a 59%	26,6%
40% a 49%	5,8%
30% a 39%	2,3%
20% a 29%	3,8%
10% a 19%	0%
1% a 9%	4,3%

3.2.1 Ferramentas de Detecção de Plágio

Inicialmente, foram coletados os resultados das ferramentas Moss e JPlag em suas comparações dos códigos dos trabalhos dos alunos. No teste aqui apresentado, a metodologia definida foi agrupar os resultados de similaridades em faixas percentuais de 10%, por exemplo: 10% a 19%, 20% a 29%, etc. Em seguida, registrar quantos acertos as ferramentas tiveram para cada faixa percentual em comparação com os registros de plágio efetivo. As Tabelas 3.1 e 3.2 mostram esse percentual de acerto para cada umas das faixas percentuais no Moss e no Jplag respectivamente.

A primeira coisa a se observar nessas tabelas é a tendência do JPlag exibir maiores índices de similaridade por trabalho. Isso se deve a maneira como ele analisa os códigos, são tratados como iguais trechos em que houve mudanças na formatação e nos nomes de variáveis, por exemplo. Isso pode, por consequência levar a mais falsos positivos. Essa tendência fica evidente ao comparar as Tabelas 3.1 e 3.2, em que só foi possível ter certeza que houve plágio quando o JPlag apontou mais de 90% de similaridade, em comparação com os 50% do Moss.

Outra razão para o JPlag exibir mais trabalhos com altos índices de similaridade foi porque ele contabiliza com mais frequência códigos incompletos (com número de linhas muto inferior aos outros) que foram enviados por alunos que desistiram no meio da tarefa.

A Figura 3.1 mostra um exemplo dessa situação, se trata de um código com 8 linhas que o JPlag havia apontado possuir 85% de similaridade com um outro que possuía 179 linhas, pela razão de ambos possuírem um loop *for* semelhante. Essa mesma análise quando feita pelo Moss apontou 0% de similaridade.

```
1  t, n = map(int, input().split())
2
3  for i in range(0, n):
4      input()
5      arquivo = input()
6      objeto = input()
7      x1, y1, x2, y2 = map(int, input().split())
8      print(arquivo, ',', objeto, ',', x1, ',', y1, ',', x2, ',', y2, sep='')
```

Figura 3.1: Exemplo de código incompleto.

Alguns cuidados devem ser tomados ao se analisar os resultados obtidos das ferramentas de detecção de plágio. Primeiramente, por se tratar de uma disciplina de introdução a programação, não se pode descartar a hipótese de que os padrões de plágio mudem ao longo do curso dos alunos. Também se deve considerar o tamanho das soluções em si. O número médio de linhas de código dos alunos que tiraram nota máxima nos Projeto 1, Projeto 2 e Projeto 3 foram 183, 264 e 163 respectivamente. Conforme se aumenta o número de linhas de um trabalho, se espera que o percentual mínimo a ser tratado como plágio venha a reduzir. Isso se deve ao fato de que muitos dos plágios detectados foram de alunos que copiaram apenas um caso de uso, o que representa apenas parte da solução final.

Esta análise apontou uma confiabilidade maior por parte do Moss, caso se decida usar apenas uma das ferramentas de detecção. Porém, o uso de ambas as ferramentas se mostra proveitoso a fim do professor poder fazer uma comparação de resultados. Se garante mais confiança quando as duas ferramentas apontarem índices de similaridade altos e, em cenários que as ferramentas apontarem resultados discrepantes, a necessidade de maior atenção aos códigos.

3.2.2 Prazo Final de Entrega

O teste seguinte foi a análise do quão distante do prazo final foram as entregas de projetos dos alunos. Se buscou por diferenças consistentes dos que plagiaram em relação aos demais. Os dados coletados estão resumidos nas Figuras 3.2 e 3.3 que apresentam gráficos contendo o número total de entregas de projetos, em relação ao número de dias que faltava para o fim do prazo, para os grupos de alunos que não plagiaram e que plagiaram.

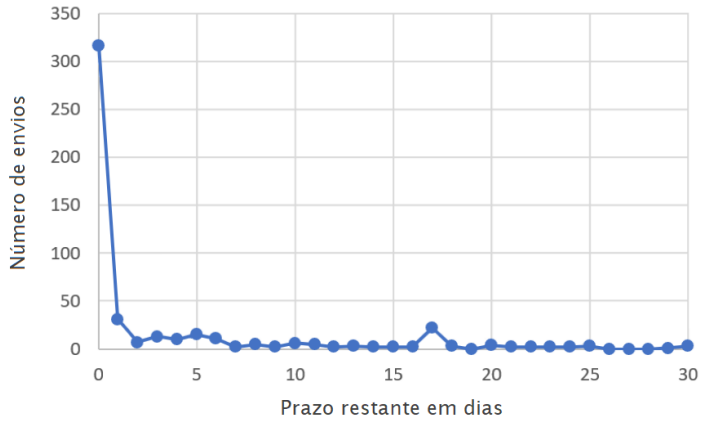


Figura 3.2: Padrão de entregas em projetos sem plágio.

Figura 3.3: Padrão de entregas em projetos plagiados.

A observação dos resultados mostrou que não houve diferença significativa entre os dois grupos de alunos. A grande maioria das entregas foram feitas no último dia do prazo previsto, tal fato pode ser atribuído a diferentes motivos, como os alunos buscando aprimorar o trabalho até os momentos finais ou uma tendência de postergar as atividades para o último momento.

A Tabela 3.3 traz um detalhamento sobre o assunto, mostrando para cada projeto, o percentual dos códigos normais e dos códigos plagiados que foram entregues no último dia. Se nota uma predileção ligeiramente maior dos plagiadores entregarem na data limite. No entanto, isso não pode ser visto como um forte indício de plágio, devido à reduzida diferença entre os dois grupos e à dificuldade de classificar objetivamente os alunos com base nesse critério.

Ainda no assunto prazo de entrega, foram realizados testes para verificar se há alguma

Tabela 3.3: Percentuais de projetos entregues no último dia.

Trabalho	Trabalhos plagiados	Trabalhos normais
Projeto 1	63%	53%
Projeto 2	75%	64%
Projeto 3	100%	86%

relação entre a nota e o número de dias restante no envio. A Figura 3.4 mostra um gráfico de dispersão que une as entregas de todos os projetos, a escala vertical se refere as notas tiradas pelos alunos (entre 0 e 10), e a escala horizontal se refere ao número de dias que faltava para o fim do prazo. É possível observar que a maioria dos alunos que entregaram com mais de 5 dias de antecedência obteve notas acima de 8, o que confirma uma expectativa comum de que os alunos que não adiam suas tarefas tendem a ter um desempenho melhor.

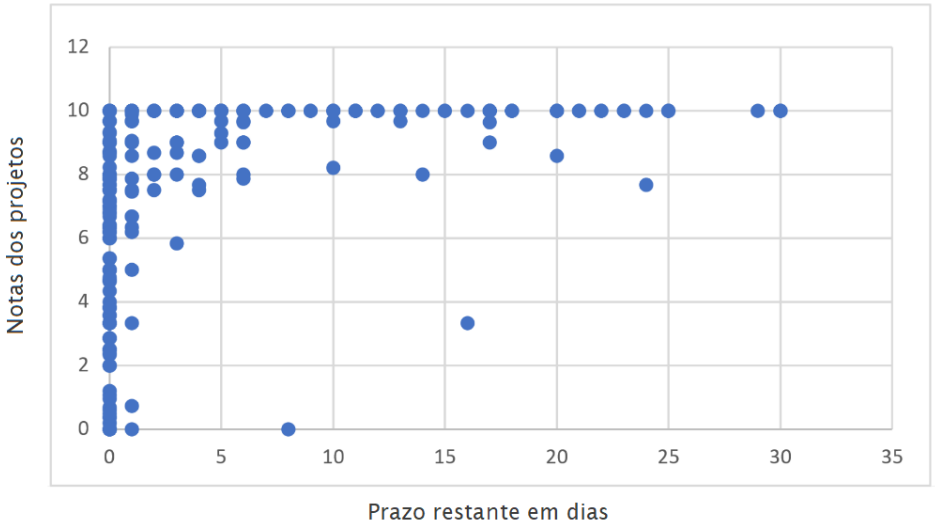


Figura 3.4: Notas dos projetos por dias restantes para entrega.

3.2.3 Duração dos Projetos

O teste seguinte foi a análise da duração que os alunos levaram para fazer cada projeto, segundo o Moodle. A Figura 3.5 mostra três gráficos de linha onde a escala vertical representa o número de projetos submetidos na plataforma, e a escala horizontal representa o número de dias que os alunos gastaram entre a visualização da descrição da tarefa e a entrega efetiva dela. As linhas azuis são os trabalhos plagiados e as linhas laranjas as entregas normais, os gráficos se referem aos 3 projetos aplicados.

A análise da Figura 3.5 mostrou que existe uma aparente aleatoriedade nos gráficos de duração das implementações. A expectativa era de que cada gráfico fosse próximo a uma curva Gaussiana, onde no centro está a duração média dos trabalhos e conforme se afasta dele se reduz o número de alunos [40]. As linhas de plágios e entregas normais tem comportamento difuso e não possuem semelhanças perceptíveis entre si ou entre um projeto e outro.

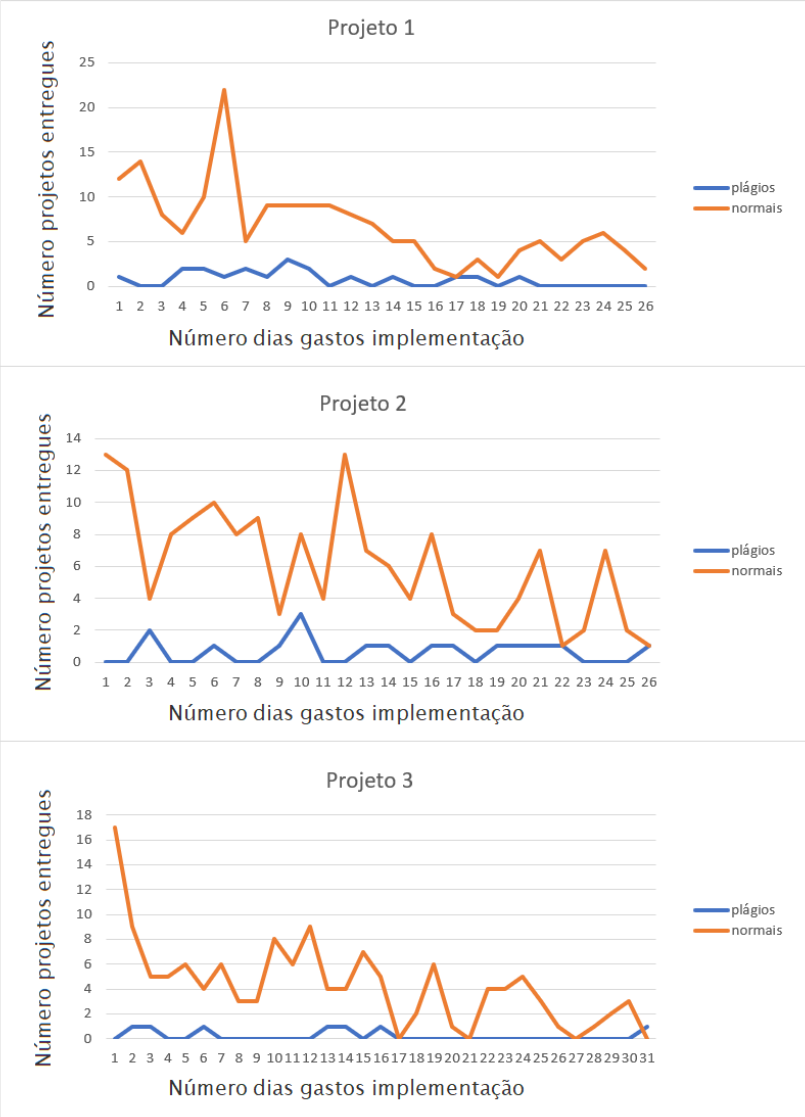


Figura 3.5: Gráficos das durações mais frequentes em projetos normais e plagiados.

Também se buscou por relações entre essas durações e as notas dos trabalhos. Foi criado então o gráfico de dispersão exibido na Figura 3.6, onde a escala vertical representa as notas dos projetos e a escala horizontal o número de dias gastos em sua implementação. É possível observar que existem trabalhos com variadas notas para cada duração possível.

O índice de correção linear entre essas duas variáveis foi de –0,21, evidenciando uma correlação fraca [41].

Figura 3.6: Disperção notas por duração dos projetos.

Os resultados inconclusivos destes testes poderiam ser usados para dizer que não existe relação entre plágio e duração das implementações, entretanto, é importante notar que o Moodle coleta o momento que o aluno primeiro visualiza a descrição da tarefa, podendo não representar a realidade de quando ele começou seu desenvolvimento. Também não se pode medir o tempo que o aluno esteve de fato escrevendo códigos usando apenas o Moodle sem a presença de *plugins* específicos para isso.

3.2.4 Notas Projetos

As notas que os alunos tiraram nos trabalhos foram verificadas em conjunto com a lista de plágios a fim de observar se existe algum padrão no desempenho desses alunos. A Tabela 3.4 mostra o comparativo entre as notas médias dos dois grupos de alunos para cada projeto e as médias gerais para todos os projetos. É possível observar uma diferença significativa entre os dois grupos, os alunos que plagiaram tiraram notas inferiores em todos os projetos e na média total. Essa diferença apontou uma vantagem dos alunos que não plagiam ao longo das atividades. Deve-se notar, entretanto, que só foram analisados os projetos que foram enviados. Os alunos que não fizeram as tarefas receberam nota 0 e não foram contabilizados como plágio ou não.

Se decidiu por aprofundar essa análise de notas comparando trabalhos entre si. Para isso, primeiro foi verificado se os alunos que plagiam são recorrentes ao longo da disciplina ou se fazem isso apenas esporadicamente. Os resultados mostram que 88,2% dos alunos

Tabela 3.4: Comparativo médias trabalhos plagiados em relação aos demais.

Trabalho	**Média trabalhos plagiados**	**Média trabalhos normais**
Projeto 1	7,28	7,78
Projeto 2	5,89	7,59
Projeto 3	2,44	5,93
Total	5,95	7,63

Tabela 3.5: Comparativo das médias em projetos posteriores dos alunos que plagiaram e dos que não.

Trabalho	**Alunos que haviam plagiado antes**	**Alunos que não plagiaram antes**
Projeto 2	3,09	8,08
Projeto 3	3,75	6,32

que copiaram fizeram isso em apenas um projeto, 8,8% plagiaram em dois projetos e apenas 2,9% o fez nos três. Essa tendência notada de não copiar repetidas vezes motivou uma nova investigação se o desempenho dos alunos que plagiam é melhor ou pior que os demais nas outras tarefas.

O teste seguinte, então, foi comparar as notas no Projeto 2 entre os alunos que não plagiaram no Projeto 1 e os que plagiaram. Se buscou verificar se um aluno que plagia é capaz de fazer um trabalho subsequente de maneira honesta, por isso foi analisado apenas códigos não plagiados no Projeto 2. Os resultados foram desfavoráveis para os que plagiaram. Os alunos que copiaram no Projeto 1 tiraram em média 3,09 no Projeto 2, em comparação aos alunos que não plagiaram no Projeto 1 e tiraram média de 8,08 no Projeto 2. O mesmo teste foi feito para o Projeto 3, foi visto que os alunos que plagiaram no Projeto 1 ou 2 tiraram média de 3,75 nele. Os alunos que não plagiaram em nenhum trabalho tiveram média de 6,32 em comparação. A Tabela 3.5 resume esses dados. Os resultados desse teste mostraram que na turma analisada, o desempenho dos alunos que plagiaram foi significativamente pior quando tentaram fazer novos trabalhos honestamente.

Ainda sobre as notas, o teste final foi o reverso do anterior, se buscou ver o desempenho dos alunos que plagiaram em trabalhos anteriores ao que efetuaram plágio. A intenção é saber se quem copia o faz como forma de tentar compensar um desempenho ruim passado. Se comparou as notas no Projeto 1 dos alunos que plagiaram e dos que não plagiaram no Projeto 2, o resultado foi que os que copiaram tiraram média de 5,45 e os que não copiaram tiraram média de 8,55. A mesma característica foi vista ao se analisar os que plagiaram no Projeto 3, estes tiraram nota média de 1,57 no Projeto 2 em comparação com os que não plagiaram que tiraram nota 8,33 no Projeto 2, conforme exibido na Tabela 3.6. A partir desses resultados, e tendo em visto o reduzido número de alunos que plagiaram em

Tabela 3.6: Comparativo das médias em projetos anteriores dos alunos que plagiaram e dos que não.

Trabalho	**Alunos que vieram a plagiar em seguida**	**Alunos que não plagiaram em seguida**
Projeto 1	5,45	8,55
Projeto 2	1,57	8,33

Tabela 3.7: Comparativo de notas médias no questionário anterior e no posterior a cada projeto entre os alunos que plagiaram e os demais.

Trabalho	**Nota questionário anterior**		**Nota questionário posterior**	
	Plágio	**Normal**	**Plágio**	**Normal**
Projeto 1	6,4	7,6	4,89	7,81
Projeto 2	5,83	7,52	5,15	6,91
Projeto 3	5,25	6,31	7	6,38

mais de um projeto, pode-se inferir que notas baixas não foram apenas uma consequência do plágio, mas também puderam ser uma motivação para ele.

3.2.5 Notas Questionários

Foram realizadas análises das notas dos questionários com o objetivo de verificar se os alunos que obtiveram pontuações baixas neles apresentam alguma tendência a cometer plágio nos projetos. O teste mais relevante consistiu em observar as médias das notas dos alunos nos questionários imediatamente antes e depois do prazo de entrega de um projeto. Como mostrado na Tabela 3.7, em quase todas as situações, os alunos que cometeram plágio obtiveram notas inferiores em comparação aos que não cometeram plágio. Isso pode indicar que existiu um nível de dedicação maior nas atividades da disciplina por parte dos alunos que não copiam.

Uma segunda observação relevante é que os alunos que cometeram plágio, apesar de terem obtido notas inferiores nos questionários analisados, apresentaram uma maior assiduidade no envio desses questionários. A Tabela 3.8 corresponde aos mesmos grupos mostrados na Tabela 3.7, porém exibindo o percentual de alunos que enviaram cada questionário, entre os que cometeram plágio e os que não cometeram. Pode-se observar que, em todos os grupos, os alunos que plagiaram tiveram uma taxa de envio de questionários maior.

Para entender melhor esse comportamento aparentemente contraditório das Tabelas 3.7 e 3.8, foi analisado o número total de envios de questionários de todos os alunos que cometeram plágio e dos que não cometeram. Na Turma A analisada, foi aplicado um total de 11 questionários. Os alunos que plagiaram em pelo menos um projeto, realizaram em

Tabela 3.8: Comparativo do percentual de alunos que enviou o questionário anterior e o posterior a cada trabalho separado por alunos que plagiaram e os demais.

Trabalho	**Percentual de envio do questionário anterior**		**Percentual de envio do questionário posterior**	
	Plágio	**Normal**	**Plágio**	**Normal**
Projeto 1	94%	78%	100%	76%
Projeto 2	100%	73%	80%	66%
Projeto 3	57%	51%	71%	57%

média 9,08 questionários, enquanto aqueles que não cometeram plágio fizeram em média 8,05 questionários. Esses dados sugerem que existe uma tendência maior a desistir das atividades na disciplina entre alunos que não copiam. No entanto, é importante ressaltar que novos testes com um grupo maior de alunos são necessários para confirmar essa conclusão.

Capítulo 4

Aplicação ProjPlag

Neste capítulo é detalhada a aplicação ProjPlag que foi desenvolvida para essa dissertação. Primeiramente, é dada uma descrição geral de seu propósito e objetivos. Em seguida, se fala com mais detalhes sobre suas funcionalidades e aspectos técnicos de sua implementação e arquitetura. Por fim, são apresentados os resultados do teste de automatização da detecção de alunos suspeitos.

4.1 Visão Geral da Aplicação

A proposta de criação de um software para auxiliar o processo de detecção de suspeitas de plágio em códigos fontes surgiu da busca por maneiras de aprimorar esse processo. No contexto universitário em que esta pesquisa foi realizada, existiam restrições que dificultavam fazer modificações diretamente na plataforma de ensino adotada, o Moodle. Por essa razão, se decidiu por implementar a aplicação ProjPlag [42], que permite ao professor manipular e visualizar os dados da turma de modo independente da plataforma. Para viabilizar essa implementação, foi realizado um levantamento dos recursos disponíveis, incluindo dados do Moodle e ferramentas de detecção que poderiam ser úteis para o ProjPlag.

Dentro desse contexto, o primeiro recurso incorporado à aplicação foi a automatização das comunicações com a ferramenta de detecção Moss, que já era utilizada na disciplina de programação. Foi implementada a execução e a leitura dos resultados dessa ferramenta com a finalidade de exibi-los em relatórios sobre a turma. Em seguida, para fins de comparação, se automatizou também a comunicação com a ferramenta JPlag, que é a segunda mais mencionada na literatura e possui um funcionamento semelhante ao do Moss.

O outro recurso disponível que foi avaliado é o conjunto de dados provenientes da plataforma Moodle. Se trata das informações relacionados às submissões dos trabalhos e

as notas dos alunos que foram detalhados no Capítulo 3. O ProjPlag passou a permitir a submissão dessas planilhas para armazenamento e visualização dos dados pelo usuário.

Como saída, foi definido que o ProjPlag gere relatórios sobre as turmas que forem cadastradas, contendo as informações disponíveis para análise. Também se fez necessária a geração de algum tipo de representação visual das relações entre alunos com trabalhos semelhantes. Para isso, foi escolhido que a aplicação gere grafos com as conexões entre os alunos, tendo como base as ferramentas de detecção.

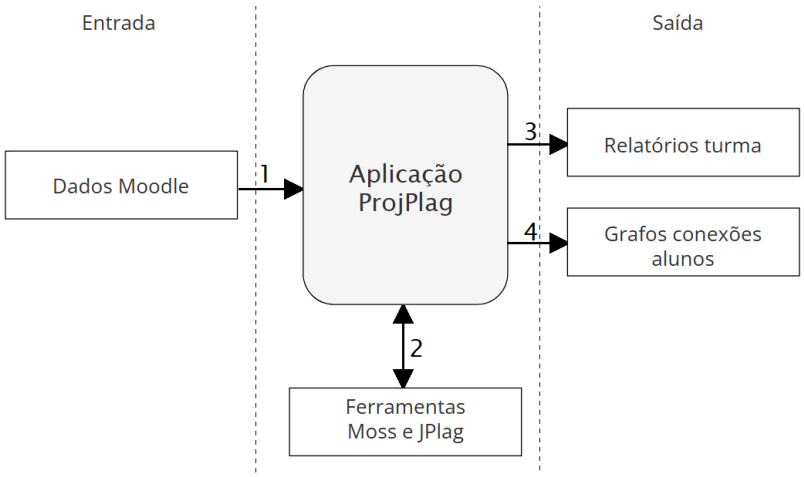


Figura 4.1: Visão geral dos relacionamentos das aplicação.

Tendo em vista essas etapas do projeto, a Figura 4.1 apresenta de forma resumida os relacionamentos da aplicação ProjPlag com elementos externos. A seta de número 1 se refere a entrada das planilhas do Moodle feitas pelo professor. A seta 2 é a comunicação do ProjPlag com as ferramentas de detecção de plágio, onde ele envia os códigos fontes e recebe os dados de similaridade. As setas 3 e 4 se referem a produção das saídas da aplicação que consistem em relatórios das turmas e grafos de conexões entre os alunos. Semelhantemente, a Figura 4.2 contém os casos de uso da aplicação exibindo as funcionalidades que o professor tem ao seu alcance e em quais delas tem atuação das ferramentas de detecção de plágio.

Por fim, além do ProjPlag agregar dados e visualizações que auxiliem o professor a fazer análises da turma, foi definido o teste de gerar índices de suspeita de plágio com base no conhecimento adquirido durante o estudo dos dados disponíveis. A aplicação recebeu em seus relatórios o campo “Nível de suspeita”. Para cada aluno que enviou um projeto, são apresentadas as chances dele ter cometido plágio, segundo a aplicação. Nas próximas seções são mostradas as telas e detalhes de implementação. Em seguida, é detalhado esse teste de automatização usando a Turma B.

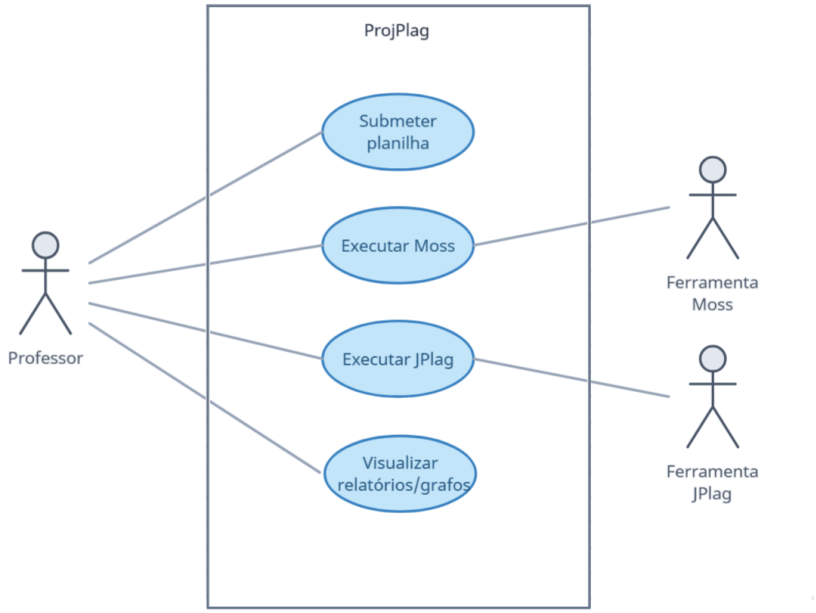


Figura 4.2: Diagrama de casos de uso.

4.2 Descrição das Funcionalidades da Aplicação

O ProjPlag é uma aplicação Web que pode ser visualizada com o uso de qualquer um dos principais navegadores atuais. As suas telas foram construídas visando tornar seu uso intuitivo e que os dados sejam agregados de forma clara para auxiliar as atividades de seus usuários. Seu acesso deve ser restrito à equipe de professores devido à natureza sensível dos dados, portanto, recomenda-se que seja instalada em ambiente controlado. Nas próximas subseções, são mostradas as funcionalidades e telas da aplicação. Deve-se informar que todos os dados de alunos exibidos aqui foram anonimizados por razões de privacidade.

4.2.1 Tela Inicial

A tela inicial é exibida quando o usuário faz o acesso a aplicação. Conforme mostra a Figura 4.3, a tela contém uma tabela com a lista de turmas cadastradas, o número de projetos e a opção de apagá-las, também existe um botão para submeter novas planilhas do Moodle com mais dados para as turmas existentes. Para se criar uma turma nova no ProjPlag, basta submeter uma planilha de uma turma que ainda não exista no sistema.

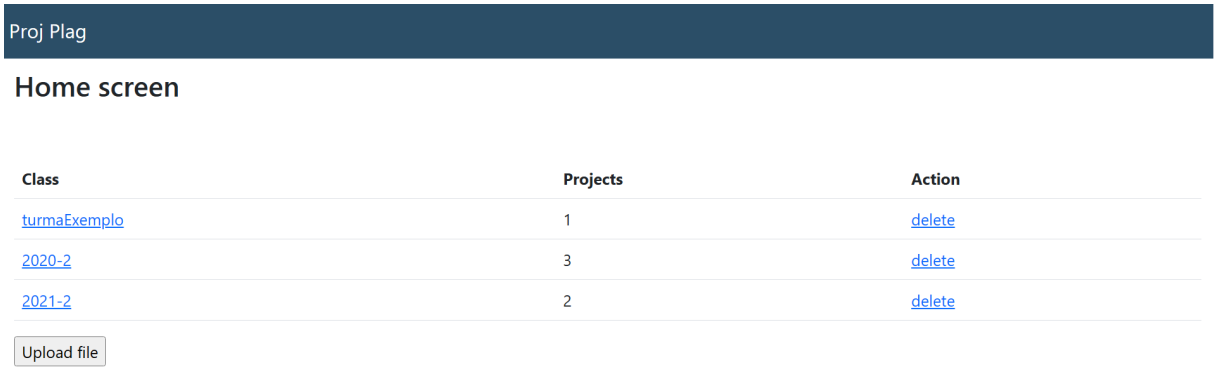


Figura 4.3: Tela inicial ProjPlag.

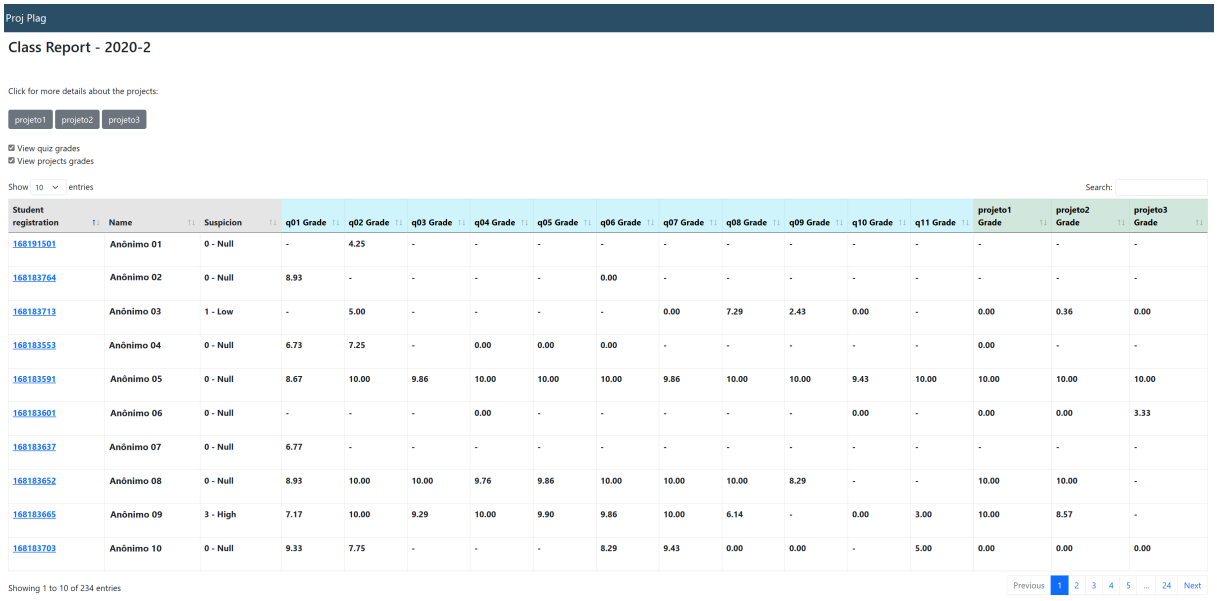


Figura 4.4: Tela do relatório de turma.

4.2.2 Relatório de Turma

Ao selecionar uma turma, o usuário é direcionado a um relatório com informações gerais dessa turma específica. Nessa tela é exibida uma tabela de alunos com os campos de matrícula, nome do aluno, suspeita de plágio, notas de todos os questionários (q01, q02, q03, etc.) e notas dos projetos (Projeto 1, Projeto 2 e Projeto 3). Ainda nessa tela, é possível ocultar a exibição das notas de questionários ou de projetos se clicando nos *checkbox* acima de tabela, também existem botões referentes a cada projeto para ir a um relatório específico deste. A Figura 4.4 exibe a tela com os detalhes explicados aqui.

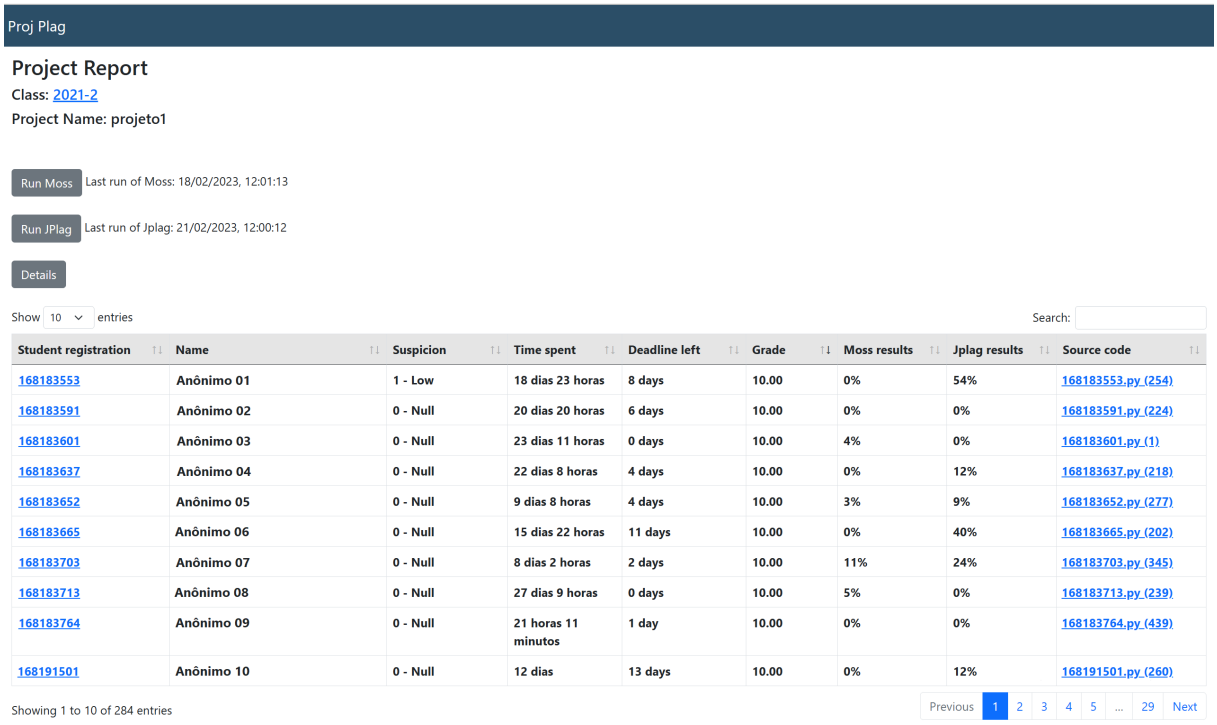


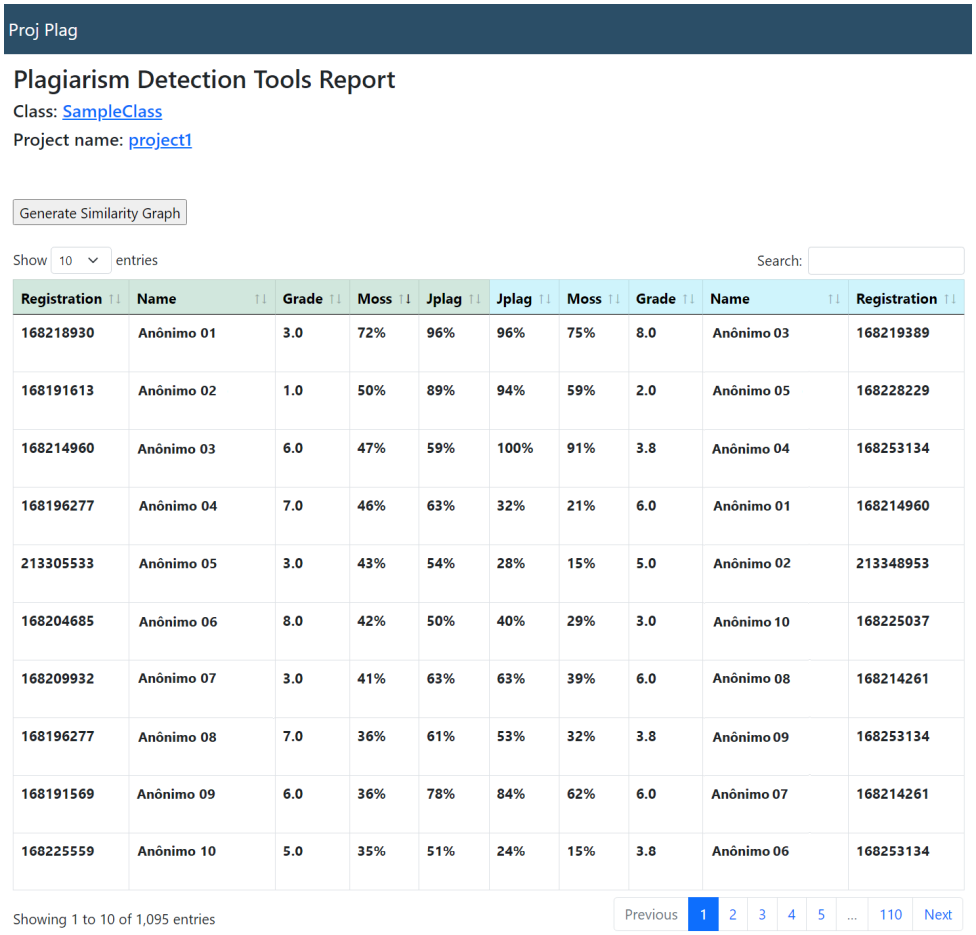
Figura 4.5: Tela do relatório de projeto.

4.2.3 Relatório de Projeto

O relatório de projeto agrega as informações disponíveis sobre um projeto específico. Conforme a Figura 4.5, é exibida uma tabela com matrículas, nomes, suspeitas de plágios, tempo gasto pelos alunos em dias e horas, prazo que restava em dias no momento que os alunos enviaram o projeto, notas, resultados de similaridade do Moss e do Jplag, e um link para a visualização dos códigos fontes dos alunos. Também, é possível executar o Moss e o Jplag desse trabalho a partir dos botões *Execute Moss* e *Execute JPlag* respectivamente, ao lado dos botões é exibido o momento da última execução da ferramenta. O botão *details* direciona para o relatório de ferramentas.

4.2.4 Relatório de Ferramentas

A tela com o relatório de ferramentas foi criada com o intuito de facilitar a visualização dos dados de similaridade ao adotar uma interface semelhante a do Moss e do Jplag. Ela possui uma tabela que exibe os alunos em duplas e apresenta os percentuais gerados pelas ferramentas do aluno da esquerda em relação ao da direita e vice-versa. Também são mostradas as matrículas e notas dos alunos. A Figura 4.6 mostra essa tela, é possível



Registration	Name	Grade	Moss	Jplag	Jplag	Moss	Grade	Name	Registration
168218930	Anônimo 01	3.0	72%	96%	96%	75%	8.0	Anônimo 03	168219389
168191613	Anônimo 02	1.0	50%	89%	94%	59%	2.0	Anônimo 05	168228229
168214960	Anônimo 03	6.0	47%	59%	100%	91%	3.8	Anônimo 04	168253134
168196277	Anônimo 04	7.0	46%	63%	32%	21%	6.0	Anônimo 01	168214960
213305533	Anônimo 05	3.0	43%	54%	28%	15%	5.0	Anônimo 02	213348953
168204685	Anônimo 06	8.0	42%	50%	40%	29%	3.0	Anônimo 10	168225037
168209932	Anônimo 07	3.0	41%	63%	63%	39%	6.0	Anônimo 08	168214261
168196277	Anônimo 08	7.0	36%	61%	53%	32%	3.8	Anônimo 09	168253134
168191569	Anônimo 09	6.0	36%	78%	84%	62%	6.0	Anônimo 07	168214261
168225559	Anônimo 10	5.0	35%	51%	24%	15%	3.8	Anônimo 06	168253134

Figura 4.6: Tela do relatório de ferramentas.

ver acima da tabela um botão que gera o grafo de similaridade, que é uma apresentação visual das conexões entre os alunos.

4.2.5 Grafo de Similaridades

Ao gerar o grafo de similaridades no ProjPlag, o usuário precisa fornecer um percentual mínimo e selecionar uma das duas ferramentas de detecção de plágio disponíveis. O sistema então cria um grafo que exibe as conexões entre os alunos que possuem códigos similares, com base no percentual fornecido e na ferramenta selecionada. Por exemplo, se o usuário escolher a ferramenta Moss e definir o percentual mínimo como 40%, o grafo será gerado apenas com os alunos que o Moss classificou com uma similaridade igual ou superior a esse valor. Cada nó do grafo representa um aluno e exibe seu nome e matrícula, enquanto as setas indicam as conexões entre os alunos. Quanto maior o percentual de similaridade, mais larga será a seta que os conecta. Além disso, a direção de cada seta indica qual aluno possui um percentual maior em relação ao outro. A Figura 4.7 apresenta

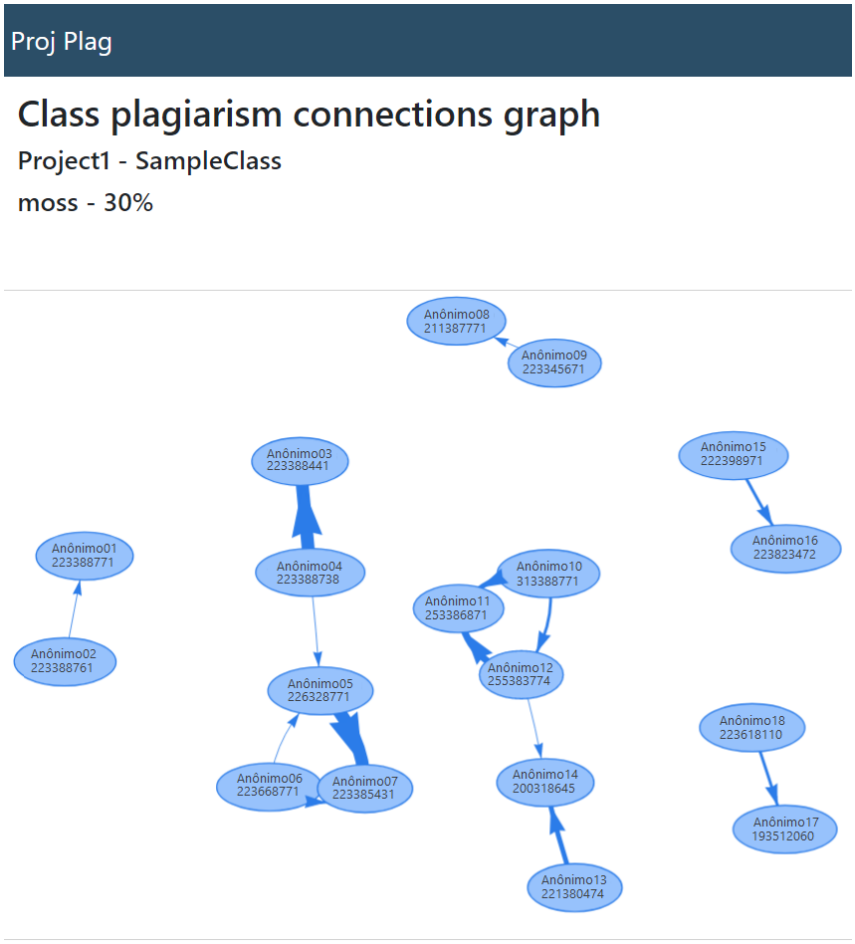


Figura 4.7: Tela do Grafo de Similaridades.

um exemplo dessa tela, que permite mapear grupos de alunos que podem ter trabalhado em conjunto e possíveis caminhos que um código fonte fez.

Para criar essa visualização, utilizou-se a biblioteca VisJs [43], desenvolvida usando JavaScript. A sua função de chamada recebe duas listas como entrada. A primeira lista consiste em todos os nós a serem incorporados ao grafo, identificados por um ID exclusivo e seu conteúdo, que inclui nome e matrícula dos alunos. A segunda lista é composta pelas arestas do grafo, que contêm os IDs dos nós de origem e destino, além de um valor numérico representando o percentual de similaridade. A disposição dos nós no grafo é realizada de forma aleatória, sendo ajustados recursivamente conforme a necessidade de evitar cruzamentos de arestas.

4.3 Aspectos sobre a Implementação

Nesta seção são descritos alguns aspectos técnicos sobre a aplicação ProjPlag, relacionados a sua implementação e tecnologias. Em seguida, são apresentados com mais detalhes tópicos sobre o uso das ferramentas de detecção de plágio e sobre o banco de dados, que possuem maior importância neste trabalho.

O ProjPlag foi idealizado para ser uma aplicação web que professores e equipe podem acessar a partir de um navegador, para maior praticidade. Porém, é necessário que seja instalado em um servidor dedicado onde estará em execução. Existem alguns requisitos que esse ambiente deve cumprir para o seu correto funcionamento, seguem eles:

- Ser um servidor Linux, sistema operacional onde a aplicação foi desenvolvida;
- Python 3 instalado, junto com bibliotecas necessárias como: Flask, SQLAlchemy, Wtforms e Pandas;
- Possuir uma versão do Java instalada para a execução do JPlag;
- Possuir acesso à internet para comunicação com o servidor do Moss;
- Possuir um banco de dados MySQL instalado para a persistência de dados dos alunos;
- Espaço em disco para a armazenamento dos códigos fontes.

O ProjPlag possui acesso à linha de comando do Linux para executar as ferramentas de detecção de plágio e manipular diretórios, razão pela qual foi escolhido esse sistema operacional. Ele foi desenvolvida em Python devido à sua adequação para manipulação de dados e à agilidade no desenvolvimento de códigos [44]. O *framework* Flask foi escolhido para o desenvolvimento web em razão da sua simplicidade [45], não necessitando de uma estrutura rígida de diretórios e de arquitetura para a criação do site.

A tecnologia de banco de dados adotada foi o MySQL, em que sua escolha foi devido ao seu código fonte ser aberto, sua ampla utilização e de ser fácil instalação e configuração [46]. Conforme será visto adiante, não se faz necessário a criação de modelos complexos de dados ou voltados para grandes volumes.

O ProjPlag foi desenvolvido em um computador com Windows 11, usando a plataforma WSL 2 para virtualizar um ambiente Linux Ubuntu 20.04 [47] que foi utilizado como servidor, também foi testado em um computador que possui o sistema Linux Pop!_OS 22.04 [48]. A seguir é detalhada a integração com as ferramentas de detecção de plágio e o armazenamento de dados.

4.3.1 Integração com as Ferramentas de Detecção

Para se fazer a integração do ProjPlag com as ferramentas de detecção de plágio foi necessário que ele tivesse acesso linha de comando do Linux, pois é por meio dela que o Moss e o Jplag são chamados. A Figura 4.8 mostra o comando padrão de execução do Moss usado para os códigos de um trabalho chamado "projeto1" contido na turma "turmaExemplo", é possível ver o log de execução seguido da URL para acesso aos resultados. A aplicação então faz download da página de resultados e a lê como um arquivo.

```
$ ./moss -l python ../codigosAlunos/turmaExemplo/projeto1/*.py
Checking files . . .
OK
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183553.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183591.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183601.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183637.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183652.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183665.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183703.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/168183713.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/260423027.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/260449968.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/260454559.py ...done.
Uploading ../codigosAlunos/turmaExemplo/projeto1/260544095.py ...done.
Query submitted.  Waiting for the server's response.
http://moss.stanford.edu/results/
$
```

Figura 4.8: Comando para execução do Moss.

A execução do JPlag é mais simples que a do Moss, pois não necessita de conexão com a internet. É executado um comando para chamada de uma aplicação Java que irá fazer a comparação entre os códigos e gerar um conjunto de arquivos em formato web para a exibição dos resultados. O ProjPlag lê esses arquivos e em seguida os apaga para economia de espaço.

Os percentuais de similaridade obtidos pelas duas ferramentas são registrados no banco de dados da aplicação. Os códigos dos alunos, por outro lado, são armazenados no próprio sistema de arquivos do servidor onde a aplicação está sendo executada. Essa abordagem foi adotada devido à necessidade das ferramentas de acessar os códigos nesse formato. Extrair os códigos de um banco de dados e gravá-los posteriormente em arquivos demandaria tempo, portanto, optou-se por armazenar os códigos dessa maneira. A Figura 4.9 ilustra como é a estrutura de diretórios onde os arquivos são armazenados, organizados por trabalhos e por turmas. É importante ressaltar que o ProjPlag não realiza a execução de nenhum trabalho de aluno, uma vez que isso poderia representar riscos de segurança para o ambiente.

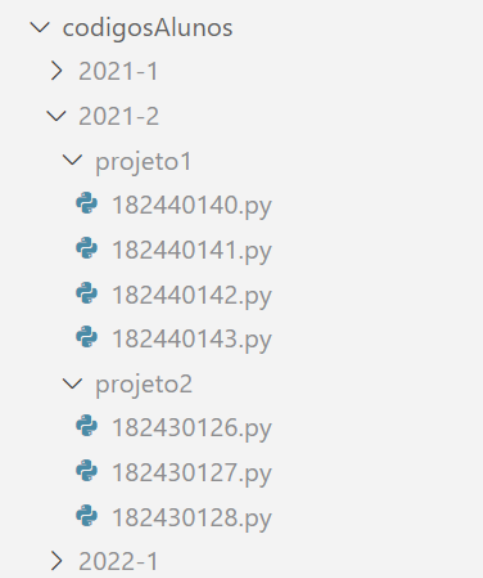


Figura 4.9: Exemplo estrura de diretórios dos arquivos.

4.3.2 Persistência dos Dados

O armazenamento dos dados dos alunos é feito por um banco de dados MySQL rodando no mesmo servidor que a aplicação. Seu acesso é feito usando a biblioteca SQLAlchemy [49] para Flask Python. As inserções são feitas quando o professor submete uma planilha do Moodle na tela inicial ou quando algum processo na aplicação gera dados novos, por exemplo, quando é rodada uma das ferramentas de detecção.

A Figura 4.10 apresenta um diagrama com os relacionamentos entre as tabelas do banco de dados. A lista total de alunos é registrada na tabela "Alunos" onde são registrados os atributos dos estudantes individualmente, como seus nomes, turmas e matrículas. O campo "turma" é preenchido no formato <ano>-<semestre>, por exemplo: 2021-2. O campo "grupo" foi criado para possibilitar registrar divisões internas dentro da turma, mas que não foi usado neste projeto, podendo ser útil em trabalhos futuros. Foi optado por criar uma chave primária "id" em vez de usar o campo matrícula para tal, pois existe a possibilidade de um aluno reprovado fazer a disciplina novamente e ter sua matrícula inserida novamente nessa tabela.

As tabelas "Projetos" e "Questionarios" guardam os dados coletados sobre as atividades que cada aluno desenvolveu, tais como o nome da tarefa e a nota. No caso dos projetos, são armazenadas as informações sobre "tempo gasto" e "prazo restante" que são retirados do Moodle. A aplicação ProjPlag não permite que se crie um projeto ou questionário sem que existam submissões para estes, portanto, não existem tabelas para registrar apenas os nomes dessas atividades sem ao menos um aluno associado.

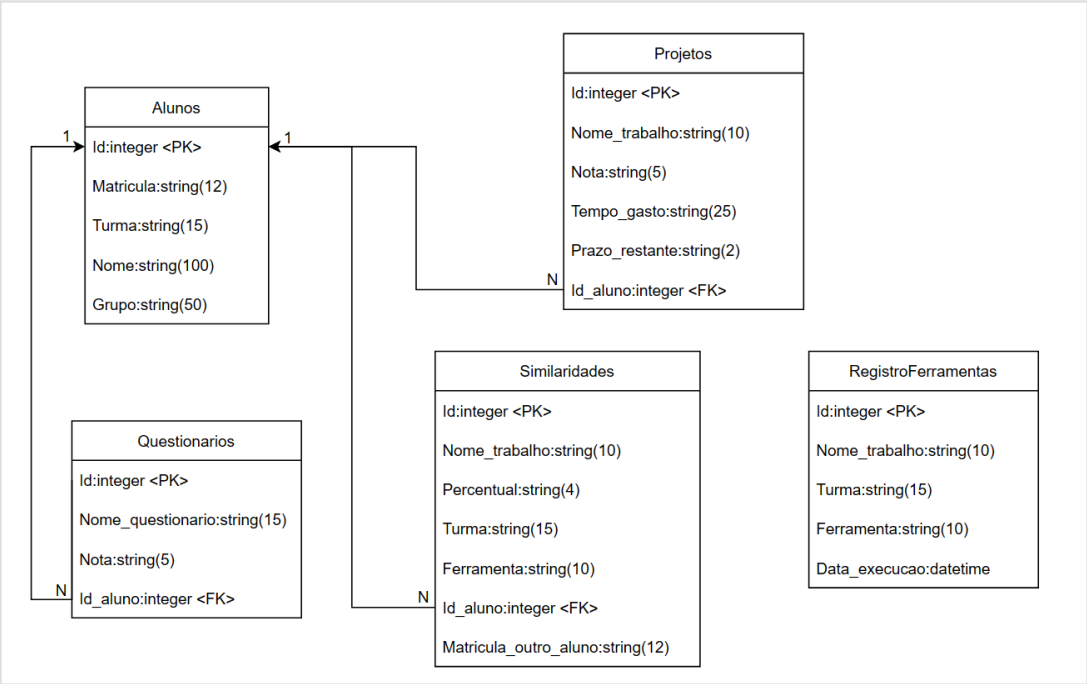


Figura 4.10: Modelo do banco de dados.

A tabela “Similaridades” guarda a lista com as similaridades geradas pelas duas ferramentas de detecção, para todos os projetos e turmas. Os alunos têm percentuais relativos a vários outros alunos (identificados pelo campo “matricula_outro_aluno”) nas suas tarefas. Inicialmente se planejava armazenar apenas o maior percentual de cada aluno por ferramenta, porém, isso impossibilitaria a geração do grafo de conexões, uma vez que existem várias relações entre os alunos que não seriam registradas.

Finalmente, a tabela “RegistroFerramentas” foi criada para armazenar a data e hora das execuções das ferramentas de detecção de plágio. A razão disso é poder informar o professor quando uma ferramenta foi executada pela última vez e permitir que a execute novamente por haver alunos ou trabalho novos adicionados.

As inserções e consultas ao banco são feitas pelos frameworks Flask e SQLAlchemy, e por essa razão, o modelo do banco de dados foi criado visando se adequar a essas ferramentas, em busca de simplicidade no desenvolvimento e performance. Se buscou também permitir que novas ferramentas de detecção de plágio sejam adicionadas a aplicação em eventuais atualizações futuras.

Tabela 4.1: Critérios para o nível de suspeita de um aluno.

Resultado ferramentas de detecção	**Suspeita do aluno**
Acima de 40% no Moss ou acima de 80% no JPlag	Alta
Acima de 30% no Moss ou acima de 60% no JPlag	Média
Acima de 20% no Moss ou acima de 50% no JPlag	Baixa
Outros resultados	Nula

4.4 Resultados do Teste de Identificação de Alunos Suspeitos

O teste final consistiu em usar a aplicação ProjPlag para testar a automatização do processo de detecção de alunos que plagiaram códigos fontes. Entre os dados analisados da Turma A, os que apresentaram maior relevância para a pesquisa foram os resultados das ferramentas de detecção de plágio e as notas de trabalhos e questionários. Foram feitos testes diversos a fim de construir uma fórmula que associe esses dados a possíveis níveis de suspeita de um aluno. Entretanto, não foi possível usar de maneira decisiva os valores das notas, pois, a amostra reduzida de alunos da disciplina e a grande quantidade de comportamentos anômalos (como o de alunos que desistiram do curso, mas continuaram enviando códigos incompletos) tornou inviável usar de maneira objetiva esses dados dentro do escopo deste projeto.

As ferramentas de detecção de plágio apresentaram um comportamento mais preciso e por isso, se decidiu utilizá-las nos testes de automatização do ProjPlag, mantendo-se a possiblidade de incorporar mais dados em pesquisas futuras. Foram estabelecidos critérios para classificar a suspeita de um aluno como alta (*high*), média (*medium*), baixa (*low*) ou nula (*null*), com base nos resultados percentuais das ferramentas Moss e JPlag, conforme mostrado na Tabela 4.1. A Figura 4.11 ilustra a exibição desses índices de suspeita exibidos em um recorte de uma tela da aplicação.

Foram realizados testes com o ProjPlag usando com os códigos da Turma B de 2021-2, composta por 284 alunos. Os resultados são apresentados na Tabela 4.2, mostrando o número de trabalhos classificados em cada nível de suspeita, a quantidade real de casos de plágio e a porcentagem correspondente.

Ao observar os resultados, é possível notar que foram obtidos níveis relativamente satisfatórios de acerto nas categorias de alta e nula. No entanto, as categorias intermediárias foram menos precisas, com poucos trabalhos classificados nelas, o que pode indicar a necessidade de ajustes nos níveis percentuais. Também foram identificadas algumas situações excepcionais dentre os trabalhos dos alunos, como dois projetos com baixa similaridade percentual, mas com trechos de código claramente copiados. Isso resultou em um deles classificado como nulo e outro como baixo, mas ambos eram plágios. Casos

Student registration	Name	Suspicion
168191501	Anônimo 01	0 - Null
168183764	Anônimo 02	0 - Null
168183713	Anônimo 03	1 - Low
168183553	Anônimo 04	0 - Null
168183591	Anônimo 05	0 - Null
168183601	Anônimo 06	0 - Null
168183637	Anônimo 07	0 - Null
168183652	Anônimo 08	0 - Null
168183665	Anônimo 09	3 - High
168183703	Anônimo 10	0 - Null

Figura 4.11: Campo de suspeita de plágio em relatório no ProjPlag.

assim destacam que, embora as ferramentas possam auxiliar os professores na detecção de plágio com bons índices de acerto, elas não podem substituir completamente o julgamento humano em detectar situações excepcionais.

Para fins de comparação, se decidiu também por fazer testes usando apenas o Moss e apenas o JPlag, mantendo os mesmos critérios exibidos na Tabela 4.1. Isso foi feito para se verificar como cada ferramenta de detecção se comporta sozinha e se foi vantajoso o uso das duas em conjunto que foi proposto. A Tabela 4.3 exibe os resultados desses testes, seguindo os mesmos padrões do teste anterior. É possível ver que o Moss parece deslocar os códigos com plágio para níveis de suspeita superiores, pois apontou como chance baixa um trabalho que o JPlag apontou como nula. O JPlag, por outro lado, parece distribuir

Tabela 4.2: Número total de trabalhos e plágios classificados pelo ProjPlag por níveis de suspeita e sua taxa de acerto.

Niveis de suspeita	Num. trabalhos total	Num. plágios	Percentual acerto
Alta	21	19	90%
Média	4	0	0%
Baixa	4	1	25%
Nula	435	1	0.1%

Tabela 4.3: Número de trabalhos e plágios classificados usando apenas Moss e JPlag.

Níveis de suspeita	**Análise Moss**			**Análise JPlag**		
	Num. trabs	**Plágios**	**Acerto**	**Num. trabs**	**Plágios**	**Acerto**
Alta	21	19	90%	16	16	100%
Média	1	0	0%	6	3	50%
Baixa	2	1	50%	2	0	0%
Nula	440	1	0,2%	440	2	0,4%

de maneira ligeiramente mais uniforme os códigos entre as categorias, tendo permitido que 3 trabalhos plagiados recebessem classificação média.

Não é possível afirmar qual das ferramentas foi objetivamente superior, primeiramente devido ao número relativamente baixo de códigos analisados, e também porque os critérios de análise podem variar a depender do professor. O Moss identificou mais plágios, mas com a existência de erros na categoria mais alta. O JPlag, por outro lado, identificou menos plágios, mas teve uma taxa de acerto maior na categoria superior. O uso de ambos em conjunto pela aplicação ProjPlag gerou resultados intermediários.

Capítulo 5

Conclusão e Trabalhos Futuros

Este trabalho buscou investigar como a automatização do processo de identificação de alunos suspeitos de plágio, no contexto de uma disciplina de programação da UnB, pode simplificar e auxiliar esse processo. Para isso, foi desenvolvida a aplicação ProjPlag, que recebe planilhas com dados dos alunos extraídos do Moodle e automatiza a chamada das ferramentas de detecção Moss e JPlag. Em seguida, a aplicação gera relatórios sobre a turma para serem analisados pelo professor. Além disso, os dados também foram analisados em busca de padrões nos alunos que copiam, a fim testar sua eficácia na identificação de suspeitos.

Em se tratando da análise dos dados, muitas das dificuldades encontradas se baseiam no fato das planilhas do Moodle aparentemente não terem sido criadas para esse fim. Pois não foi possível estudar com detalhes os padrões de desenvolvimento de códigos dos alunos a partir delas. No entanto, foi possível observar algumas características da turma e a identificação de certos padrões, como a tendência dos alunos de submeter trabalhos no último dia e de desistir do curso, mas continuar enviando tarefas, além das correlações entre notas inferiores e plágio.

A aplicação ProjPlag pode tornar mais simples o processo de identificação de alunos que possam ter plagiado, pois automatiza várias das etapas feitas de maneira manual nas turmas estudadas. A aplicação permite a visualização fácil dos dados da turma e o uso de duas ferramentas de detecção de plágio adiciona recursos adicionais para auxiliar os professores e equipe.

No que se refere ao campo de nível de suspeita de plágio, o ProjPlag apresentou um desempenho satisfatório, apesar das limitações dos dados disponíveis, e pode ser utilizada como um auxiliar no processo de detecção. Foi visto também que existem situações excepcionais de plágio de difícil identificação automática, como já havia sido mencionado na revisão de literatura.

Em relação aos trabalhos futuros, o uso de um volume maior de dados e turmas, e

também de informações adicionais, como o número de tentativas de envio, pode ser útil para buscar novas correlações. O uso de dados extraídos de outras ferramentas além do Moodle se mostra importante para o aprofundamento de análises. A aplicação ProjPlag também pode ser aprimorada para gerar gráficos automaticamente, além do grafo de conexões. Adicionalmente, é possível fazer novos testes em busca de otimizar os percentuais e critérios de suspeita para cada ferramenta de detecção empregada.

Por fim, as pesquisas futuras devem considerar as ferramentas on-line de geração de código automático, pois a revisão na literatura mostrou que existe a possibilidade de contornar as técnicas de detecção de plágio com o uso de Inteligência Artificial. Devem ser estudadas maneiras de contornar ou se adaptar a essa realidade.

Referências

[1] Aniceto, Rodrigo, Maristela Holanda, Carla Castanho e Dilma Silva: *Source code plagiarism detection in educational context: A literature mapping*. Em *2021 Frontiers in Education Conference (FIE)*. IEEE, outubro 2021. x, xi, 1, 5, 6, 7, 8, 13

[2] *Plagiarism*. https://dictionary.cambridge.org/pt/dicionario/ingles/plagiarism, (acessado em 3 de junho de 2023). 1

[3] Agrawal, Mayank e Dilip Kumar Sharma: *A state of art on source code plagiarism detection*. Em *2016 2nd International Conference on Next Generation Computing Technologies (NGCT)*. IEEE, outubro 2016. `https://doi.org/10.1109/ngct.2016.7877421`. 1

[4] Sheahen, Dana e David Joyner: *TAPS: A MOSS extension for detecting software plagiarism at scale*. Em *L@S 2016 - Proceedings of the 3rd 2016 ACM Conference on Learning at Scale*, páginas 285–288. ACM Press, 2016. `https://doi.org/10.1145/2876034.2893435`. 1

[5] Novak, Matija, Mike Joy e Dragutin Kermek: *Source-code similarity detection and detection tools used in academia*. ACM Transactions on Computing Education, 19(3):1–37, junho 2019. `https://doi.org/10.1145/3313290`. 1

[6] Saoban, Chawalit e Sunisa Rimcharoen: *Identifying an original copy of the source codes in programming assignments*. Em *2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, julho 2019. `https://doi.org/10.1109/jcsse.2019.8864196`. 2, 10

[7] Priya, Sharma R. Mukuntha, Anukul Dixit, Krishanu Das e Ronak Harish Patil: *Plagiarism detection in source code using machine learning*. International Journal of Engineering and Advanced Technology, 8(4):897–901, 2019. 2, 10

[8] Ljubovic, Vedran e Enil Pajic: *Plagiarism detection in computer programming using feature extraction from ultra-fine-grained repositories*. IEEE Access, 8:96505–96514, 2020. 2, 10, 11

[9] Smith, Cameron Maurice: *A toolset for mining github repositories in educational software projects*. Tese de Mestrado, Texas A&M University, 2018. 2, 14, 17

[10] Moss: *A system for detecting software similarity*. https://theory.stanford.edu/ãiken/moss/, (acessado em 14 de abril de 2023). 2, 8

[11] Moodle: *Features*. https://docs.moodle.org/311/en/Features, (acessado em 3 de junho de 2023). 3, 12

[12] Helgesson, Gert e Stefan Eriksson: *Plagiarism in research*. Medicine, Health Care and Philosophy, 18(1):91–101, julho 2014. 5

[13] Goldberg, David M.: *Programming in a pandemic: Attaining academic integrity in online coding courses*. Communications of the Association for Information Systems, 48:47–54, 2021. 6

[14] Mišić, Marko J., Zivojin Siustran e Jelica Ž. Protić: *A comparison of software tools for plagiarism detection in programming assignments*. International Journal of Engineering Education, 32(2):738–748, 2016. 8

[15] JPlag: *Jplag - detecting software plagiarism*. https://github.com/jplag/JPlag, (acessado em 5 de março de 2022). 8

[16] Grune, Dick: *The software and text similarity tester sim*. https://dickgrune.com/Programs/similarity_tester, (acessado em 14 de abril de 2022). 8

[17] Ahtiainen, Aleksi: *Plaggie: download*. https://www.cs.hut.fi/Software/Plaggie, (acessado em 14 de abril de 2022). 8

[18] Warwick: *Sherlock - plagiarism detection software*. https://warwick.ac.uk/fac/sci/dcs/research/ias/software/sherlock, (acessado em 14 de abril de 2022). 8

[19] Rahiman, Nik Faris Aiman Nik, Md. Gapar Md. Johar e Rabab Alayham Abbas Helmi: *Copypoppy - a source code plagiarism detector*. Em *2022 IEEE 10th Conference on Systems, Process & Control (ICSPC)*, páginas 123–128. IEEE, 2022. 10

[20] Ďuračík, Michal, Emil Kršák e Patrik Hrkút: *Searching source code fragments using incremental clustering*. Concurrency and Computation: Practice and Experience, 32(13), junho 2019. `https://doi.org/10.1002/cpe.5416`. 10

[21] Karnalim, Oscar e Simon: *Syntax trees and information retrieval to improve code similarity detection*. Em *Proceedings of the Twenty-Second Australasian Computing Education Conference*. ACM, janeiro 2020. `https://doi.org/10.1145/3373165.3373171`. 10

[22] Tahaei, Narjes e David C. Noelle: *Automated plagiarism detection for computer programming exercises based on patterns of resubmission*. Em *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ACM, agosto 2018. `https://doi.org/10.1145/3230977.3231006`. 10, 15, 16, 17

[23] Fonseca, Nuno Gil, Luis Macedo e Antonio Jose Mendes: *Using early plagiarism detection in programming classes to address the student's difficulties*. Em *2018 International Symposium on Computers in Education (SIIE)*. IEEE, setembro 2018. `https://doi.org/10.1109/siie.2018.8586700`. 10, 16, 17

[24] Pierce, Jonathan e Craig B. Zilles: *Investigating student plagiarism patterns and correlations to grades.* Em *Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE*, páginas 471–476, 2017. 10, 12

[25] Github: *Your ai pair programmer.* https://github.com/features/copilot, (acessado em 3 de julho de 2023). 11

[26] Xiao, Yunkai, Soumyadeep Chatterjee e Edward Gehringer: *A new era of plagiarism the danger of cheating using ai.* Em *2022 20th International Conference on Information Technology Based Higher Education and Training, ITHET 2022*, 2022. 11

[27] Sukhodolsky, Jacob: *New approach to teaching computer programming to freshmen.* Em *EPiC Series in Computing*, volume 69, páginas 197–205, 2020. 11

[28] Pawelczak, Dieter: *Benefits and drawbacks of source code plagiarism detection in engineering education.* Em *IEEE Global Engineering Education Conference, EDUCON*, volume 2018-April, páginas 1048–1056, 2018. 11

[29] Modiba, Phatludi, Vreda Pieterse e Bertram Haskins: *Evaluating plagiarism detection software for introductory programming assignments.* Em *Proceedings CSERC 2016 - Computer Science Education Research Conference*, páginas 37–46. ACM Press, 2016. `https://doi.org/10.1145/2998551.2998558`, Cited By :7. 11

[30] Kyrilov, Angelo e David C. Noelle: *Binary instant feedback on programming exercises can reduce student engagement and promote cheating.* Em *ACM International Conference Proceeding Series*, volume 19-22-Nov-2015, páginas 122–126, 2015. 12

[31] Karnalim, Oscar e Simon: *A review on web scrapping and its applications.* Em *2019 International Conference on Computer Communication and Informatics (ICCCI).* IEEE, 2020. 12

[32] Souza Alencar, Andréia de, Fernanda Cristina de Paula Matias, Fernanda Pereira Guimarães e Rodrigo Sanches de Oliveira: *O moodle como ferramenta didática.* Em *Anais do Congresso Nacional Universidades, EAD e Software Livre*, páginas 1–5. UFMG, 2011. `http://www.periodicos.letras.ufmg.br/index.php/ueadsl/article/viewFile/2919/2878`. 12

[33] Moodle: *Installing moodle.* https://docs.moodle.org/311/en/Installing_Moodle, (acessado em 3 de junho de 2023). 13

[34] Lobb, Richard: *Coderunner.* https://github.com/trampgeek/moodle-qtype_coderunner#code-runner, (acessado em 14 de abril de 2022). 13

[35] GitHub: *Where the world builds software.* https://github.com/, (acessado em 5 de março de 2022). 14

[36] Catalena, Kate Ashley: *Mining student submission information to refine plagiarism detection.* Tese de Mestrado, Texas A&M University, 2020. 14, 17

[37] MimirClassroom: *Scale and automate your computer science classroom.* https://www.mimirhq.com/, (acessado em 5 de março de 2022). 14

[38] Kaya, Mümine Keleş e Selma Ayşe Özel: *Integrating an online compiler and a plagiarism detection tool into the moodle distance education system for easy assessment of programming assignments.* Computer Applications in Engineering Education, 23(3):363–373, julho 2014. https://doi.org/10.1002/cae.21606. 15, 17

[39] Brasil: *Pltaforma brasil.* https://plataformabrasil.saude.gov.br/, (acessado em 11 de maio de 2022). 19

[40] James Chen, Khadija Khartit, Suzanne Kvilhaug: *Normal distribution: What it is, properties, uses, and formula.* https://www.investopedia.com/terms/n/normaldistribution.asp, (acessado em 29 de junho de 2023). 24

[41] McDonald, John H.: *Linear regression and correlation.* https://stats.libretexts.org/Bookshelves/Applied_Statistics/Biological_ Statistics_(McDonald)/05%3A_Tests_for_Multiple_Measurement_Variables/ 5.01%3A_Linear_Regression_and_Correlation, (acessado em 3 de junho de 2023). 25

[42] Aniceto, Rodrigo: *Projplag.* https://github.com/rodrigo-aniceto/ProjPlag, (acessado em 20 de maio de 2023). 29

[43] visjs: *Network.* https://visjs.github.io/vis-network/docs/network/, (acessado em 3 de maio de 2023). 35

[44] Mitchell, Ryan: *Web Scraping with Python: Collecting More Data from the Modern Web.* O'Reilly Media, 2018, ISBN 9781491985526. https://books.google.com.br/books?id=TYtSDwAAQBAJ. 36

[45] Grinberg, Miguel: *Flask Web Development: Developing Web Applications with Python.* O'Reilly Media, 2018, ISBN 9781491991695. https://books.google.com.br/books?id=cVlPDwAAQBAJ. 36

[46] MySQL: *The world's most popular open source database.* https://www.mysql.com/, (acessado em 17 de março de 2022). 36

[47] Microsoft: *Instalar o linux no windows com o wsl.* https://learn.microsoft.com/pt-br/windows/wsl/install/, (acessado em 3 de junho de 2023). 36

[48] System76: *Welcome to pop!_os.* https://pop.system76.com/, (acessado em 3 de junho de 2023). 36

[49] Pallets: *Flask-sqlalchemy.* https://flask-sqlalchemy.palletsprojects.com/en/3.0.x/, (acessado em 3 de junho de 2023). 38