Building and Analyzing an Open Repository of Programming Exercises in Portuguese

Vinicius Kuster Lodi
Deparment of Informatics
Universidade Federal de Viçosa, Brazil
vinicius.lodi@ufv.br

João Pedro M. Sena
Institute of Mathematics and Computer Science
Universidade de São Paulo, Brazil
joao.sena@usp.br

Julio C. S. Reis
Deparment of Informatics
Universidade Federal de Viçosa, Brazil
jreis@ufv.br

ABSTRACT

Practical exercises are an important part of learning programming. For Brazilian Portuguese, however, there are few data sets with this availability. This work aims to collect and model a data repository for programming exercises, containing instructions, test cases, and exercise metadata. First, platforms are defined as data sources, and Web Scraping techniques are applied to extract the desired content. From the collection, the data is modeled and made available in a public repository called ProgramExBR. A characterization of the data is also presented, revealing a number of interesting discoveries about the text patterns it contains. We hope that this can be useful in various contexts, be it for supporting teachers and students, for training inference models and generating new exercises, or even for an application of personalized recommendation methods in education.

KEYWORDS

repository, programming exercise, Portuguese, Web scraping

1 INTRODUÇÃO

As disciplinas de programação de computadores desempenham um papel fundamental na formação de profissionais de diversas áreas, especialmente Ciência da Computação, por desenvolverem o raciocínio lógico e a capacidade de resolução de problemas [12]. No entanto, elas costumam ser desafiadoras para os estudantes, que geralmente possuem um conhecimento bastante heterogêneo de programação, resultando em altos índices de abandono e reprovação [13, 14]. Neste contexto, uma das estratégias para mitigação do problema está relacionada à personalização do ensino considerando as diferentes particularidades de aprendizado dos alunos, principalmente nas atividades práticas [6].

Segundo Pereira *et al.* (2020) [9] e Becker e Quille (2019) [2], aprender programação exige muita prática, especialmente por meio de exercícios. Uma importante ferramenta nesse contexto são os Ambientes de Correção Automática de Códigos (ACACs), usados por professores para trabalhos práticos, exames presenciais, ou até para a criação de listas extraclasse [8]. Nesses ambientes, os alunos recebem uma lista de exercícios, onde são especificados os formatos de entrada e saída esperados para cada problema. Quando um aluno submete uma solução seguindo corretamente a especificação, o ACAC é capaz de fazer a correção automática dessa submissão. Esse alívio de carga com a correção permite que os professores criem mais exercícios para os alunos praticarem os diversos tópicos que compõem a disciplina.

No entanto, mesmo com o uso de ACACs, um dos principais desafios para os professores é criar listas de exercícios adequadas para os diferentes perfis de alunos. Selecionar esses exercícios manualmente para os variados perfis de estudantes acaba se tornando uma solução inviável, uma vez que, além de diversas, as disciplinas de programação costumam ter uma grande quantidade de alunos [3, 10]. Logo, no seu âmbito mais geral, este projeto visa o desenvolvimento futuro de sistemas de recomendação de exercícios de programação com foco em atividades extraclasse. Nesse sentido, uma base de dados de exercícios é um requisito fundamental, independentemente da abordagem de recomendação escolhida. Embora existam algumas iniciativas nesta direção [8], ainda são escassos os esforços considerando exercícios em português do Brasil que sejam de domínio público. É sobre esta lacuna de pesquisa que se estabelece o objetivo deste trabalho.

Particularmente, apresentamos o ProgramExBR, um repositório de dados composto por 2.102 exercícios de programação em português do Brasil coletados a partir de plataformas distintas. Neste contexto, implementamos coletores especializados que acessam páginas *Web* com exercícios de programação publicamente disponíveis e obtêm seus dados. O ProgramExBR está publicamente disponível em: https://doi.org/10.5281/zenodo.15724871. Adicionalmente, conduzimos uma caracterização dos dados que revelou uma série de descobertas interessantes relativas ao padrão textual dos exercícios. Esperamos que este repositório possa ser útil para complementar o material de professores das matérias introdutórias das disciplinas de programação, servindo como base de exercícios para uso direto ou adaptado pelos docentes, contribuindo para a prática dos alunos e, consequentemente, para o aumento do conhecimento técnico, bem como para pesquisas futuras neste contexto.

Na próxima seção, apresentamos conceitos e repositórios relacionados. Em seguida, na Seção 3, detalhamos o processo de construção do ProgramExBR. Uma descrição geral do repositório é apresentada na Seção 4. Por fim, na Seção 5, concluímos o estudo.

2 CONCEITOS E REPOSITÓRIOS DE DADOS RELACIONADOS

Um ACAC, comumente referenciado apenas como juiz *online* em programação (*online judge*, ou *OJ*) é um sistema que permite testar e avaliar código de programação online. Ele funciona compilando, executando e testando o código com dados de entrada pré-definidos para verificar se a saída é a esperada. É amplamente utilizado em

In: V Concurso de Trabalhos de Iniciação Científica (CTIC 2025). Anais Estendidos do XXXI Simpósio Brasileiro de Sistemas Multimídia e Web (CTIC'2025). Rio de Janeiro/RJ, Brasil. Porto Alegre: Brazilian Computer Society, 2025.

ISSN 2596-1683

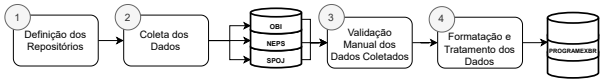

Figura 1: Metodologia para construção do ProgramExBR.

competições de programação e em contextos educacionais [1]. Em outras palavras, ACACs são plataformas que automatizam o processo de verificação e correção de erros em códigos de programação. Estes sistemas são muito úteis pois permitem a identificação e correção de erros rapidamente, economizando tempo e esforço.

De forma geral, os ACACs podem ser classificados em dois tipos principais: (i) de propósito geral e (ii) dedicados. Os juízes *online* de propósito geral possuem milhares de exercícios cadastrados, de diferentes níveis e origens, que podem ser usados de forma autônoma pelos próprios estudantes como ambiente de treinamento extraclasse. Exemplos desse tipo de juiz *online* incluem o *beecrowd*[1], o *UVa Online Judge*[2], o *Codeforces*[3], o *SPOJ*[4], o *SPOJ Brasil*[5], o *Neps Academy*[6] e o *CodeBench*[7] [5]. Neste contexto, é importante destacar que essas plataformas possuem, majoritariamente, exercícios de programação disponíveis em inglês. Já os juízes dedicados podem ser instalados e gerenciados localmente pelos docentes, com fácil cadastro de exercícios próprios e personalizados para a disciplina ministrada, oferecendo mais controle ao docente. Exemplos desse tipo de juiz *online* incluem o *BOCA* [4] e o *Submitty* [11].

No entanto, existem algumas plataformas, como o *CodeBench*[7], que armazenam exercícios de programação em português, mas não os disponibilizam publicamente em formatos editáveis. Além disso, exigem que o usuário faça um cadastro prévio na plataforma para uso do seu banco interno e não permitem a exportação direta desse banco para uso em outras plataformas. Com isso, professores podem ter dificuldade em utilizar esses exercícios em juízes *online* dedicados, pois teriam que converter o enunciado em texto, além de considerar as imagens e tabelas que possivelmente fazem parte da descrição do exercício.

Para preencher esta lacuna, o objetivo deste trabalho é apresentar uma metodologia de coleta de enunciados de exercícios de programação em português brasileiro provenientes de diferentes plataformas e disponibilizá-los em um repositório unificado. A ideia é que este repositório possa servir de fonte para professores e alunos de programação, mas também de suporte para pesquisas futuras no contexto educacional.

3 METODOLOGIA

Na Figura 1, são apresentadas as etapas adotadas para construção do ProgramExBR. Cada uma delas é detalhada a seguir.

1) Definição dos Repositórios. Primeiramente, as plataformas exploradas na pesquisa foram definidas a partir de buscas realizadas em mecanismos de pesquisa como o Google[8] com a inclusão de sistemas amplamente difundidos na comunidade acadêmica (e.g., *CodeBench*). Exemplos de termos utilizados para pesquisa incluem: "Exercícios de programação", "Praticar programação", "Programação competitiva", com buscas realizadas em português e inglês.

Depois, foram estabelecidos três critérios de seleção, considerando o contexto brasileiro, a necessidade de direito de acesso e a disponibilidade dos dados. Os critérios foram: 1) A plataforma possui exercícios de programação disponíveis em português? 2) A plataforma disponibiliza os exercícios publicamente (não exige *login*)? 3) A plataforma não possui Termos de Uso ou não foram encontradas restrições relacionadas à coleta e/ou disponibilização dos dados por terceiros? A Tabela 1 também descreve a avaliação de cada critério nas plataformas analisadas, utilizando ✓ para indicar que o critério foi atendido, ✗ para indicar que foi rejeitado e um hífen (-) quando o critério não foi verificado devido à reprovação em algum critério anterior. Com isso, as plataformas que não atendessem a pelo menos um dos critérios estabelecidos foram desconsideradas do processo de coleta. Por fim, as plataformas selecionadas foram: *SPOJ Brasil*, *Pratique OBI*, e *Neps Academy*.

Tabela 1: Verificação dos critérios nas plataformas.

Plataforma	Critério 1	Critério 2	Critério 3
Beecrowd[1]	✓	✗	-
UVa Online Judge[2]	✗	-	-
Codeforces[3]	✗	-	-
Sphere Online Judge (SPOJ)[4]	✗	-	-
Sphere Online Judge Brasil (SPOJ Brasil)[5]	✓	✓	✓
Neps Academy[6]	✓	✓	✓
CodeBench[7]	✓	✗	-
CodeChef[9]	✗	-	-
Hackerearth[10]	✗	-	-
LeetCode[11]	✗	-	-
Pratique OBI[12]	✓	✓	✓

2) Coleta. A coleta de dados dos exercícios (i.e., descrição, casos de teste, metadados, entre outros) foi conduzida a partir de ferramentas de *Web Scraping*. Os coletores implementados navegaram pelas páginas das fontes definidas e obtiveram o endereço das páginas contendo as informações dos exercícios. Então, a partir do *download* das páginas com as informações, o conteúdo passou por um processo de limpeza e tratamento dos dados. Como cada plataforma possui sua própria estrutura e disponibilização dos exercícios, foi necessário implementar um coletor personalizado para cada plataforma[13]. Para este fim, utilizamos o *framework Playwright*[14].

3) Validação Manual dos Dados Coletados. Os dados coletados de cada um dos repositórios, foram submetidos a uma etapa de validação manual. Especificamente, foi inspecionada uma amostra aleatória de cada repositório. O objetivo desta etapa é verificar alguma inconsistência durante o processo de coleta, como caracteres indesejados e/ou em duplicidade, ou até mesmo erros de formatação como caracteres "\n" escritos explicitamente no texto da questão. Como resultado, foram delineadas as estratégias para tratamento e formatação dos dados.

4) Formatação e Tratamento dos Dados. Finalmente, os dados coletados foram padronizados em formato JSON (*JavaScript Object Notation*). Para cada exercício, disponibilizamos: um identificador único, um nome (a partir da plataforma de origem), a URL original, bem como o conteúdo que inclui uma descrição do problema, definição de entrada, definição de saída, casos de teste e metadados (e.g., tempo máximo de execução, limite de uso de memória e dificuldade), quando disponíveis, visto que há um subconjunto das plataformas selecionadas que não disponibilizam alguns ou nenhum dos metadados. Um exemplo desse arquivo é apresentado na Figura 2.

[1] www.beecrowd.com.br [2] onlinejudge.org [3] codeforces.com [4] www.spoj.com/ [5] br.spoj.com/ [6] neps.academy/ [7] codebench.icomp.ufam.edu.br [8] www.google.com.br/

[9] www.codechef.com/ [10] www.hackerearth.com/ [11] leetcode.com/ [12] olimpiada.ic.unicamp.br/pratique/ [13] Os coletores implementados estão publicamente disponíveis em: github.com/VKusterL/Coletores---WebMedia2025 [14] playwright.dev/python/

```
{
    "ID": 1778,
    "Name": "Rotacionando na Colina",
    "URL": "https://neps.academy/br/exercise/999",
    "Problem_Description": "Voce foi contratado para transportar um pacote
        entre duas cidades. ...",
    "Input": "Na primeira linha haverao tres inteiros ...",
    "Output": "Imprima uma linha contendo N ou M inteiros ...",
    "Test_Case":
      [ {"input": "4 4 1 2 3 1 1",
         "output": "4 2 1 0"},
        {"input": "4 4 2 2 3 1 1",
         "output": "3 2 1 1"},
        {"input": "4 3 1 2 3 1 1",
         "output": "4 2 1"} ],
    "metadata":
       {"Difficulty": "Dificil",
        "Memory_Limit": "256 mb",
        "CPU_Time_Limit": "1 second(s)"}
}
```

Figura 2: Exemplo de JSON para um exercício de programação.

4 CARACTERÍSTICAS DO PROGRAMEXBR

Nesta seção, apresentamos características gerais relacionadas ao repositório de dados apresentado neste estudo, com um sumário na Tabela 2. De forma geral, nota-se características distintas entre as plataformas. *Neps Academy* destaca-se como a plataforma predominante, com 1.448 exercícios, superando significativamente a *OBI* (353) e o *SPOJ-BR* (301). Em relação à quantidade de casos de teste, o *SPOJ-BR* apresenta menos casos (média de 1,19), enquanto *Neps Academy* e *OBI* oferecem aproximadamente o dobro de casos de teste (2,38 e 2,22, respectivamente). Nas descrições de entrada, o *SPOJ-BR* lidera com uma média de 114 caracteres, indicando casos de teste mais elaborados e extensos. Além disso, a *OBI* apresenta as descrições de saída mais longas (67 caracteres), possivelmente refletindo, também, em respostas mais complexas ou descritivas. Na análise de problemas com múltiplos casos de teste, a *OBI* se destaca, com 97,2% dos seus exercícios possuindo mais de um caso de teste, seguida de *Neps Academy* em 82,2%, enquanto o *SPOJ-BR* adota essa abordagem em apenas 14% dos problemas. Esses dados sugerem que a *OBI* e *Neps Academy* priorizam múltiplos casos de teste, tendo, provavelmente, mais complicações que necessitam de testes específicos para serem descobertas.

A Figura 3, apresenta uma comparação do tamanho dos enunciados (em quantidade de caracteres) entre as três plataformas coletadas. A plataforma *Neps Academy* apresenta enunciados com tamanho medianos de aproximadamente 300 caracteres, sendo a plataforma com textos mais diretos, porém com a maior variabilidade, evidenciada pela presença de numerosos *outliers*, que chegam a ultrapassar 1500 caracteres. Ademais, as plataformas *OBI* e *SPOJ-BR* demonstram padrões similares entre si, com medianas próximas a 400 caracteres e distribuições mais homogêneas. A *OBI* destaca-se por tamanhos dos enunciados mais homogêneos, apresentando a menor dispersão entre as três plataformas e poucos outliers. Por sua vez, *SPOJ-BR* exibe uma variabilidade intermediária, com alguns pontos extremos, alcançando cerca de 1100 caracteres. Estes resultados sugerem que *Neps Academy* adota uma abordagem mais

Tabela 2: Sumário dos dados em cada repositório.

Repositórios	Neps Academy	OBI	SPOJ-BR
Nro. de exercícios	1448	353	301
Nro. médio de casos de teste	2,38 ± 1,04	2,62 ± 0,75	1,19 ± 0,68
Tamanho médio da entrada	80,63 ± 72,65	61,91 ± 39,15	114,56 ± 53,55
Tamanho médio da saída	40,03 ± 54,39	67,25 ± 34,12	54,44 ± 36,92
Tamanho médio dos testes	26,41 ± 26,42	30,79 ± 33,22	45,87 ± 46,07
Proporção de multi-casos	82,2%	97,2%	14,0%

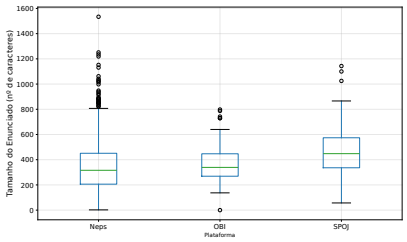

Figura 3: Análise de tamanho dos enunciados.

flexível quanto ao tamanho dos enunciados, enquanto *OBI* e *SPOJ-BR* mantêm textos mais padronizados, o que pode refletir diferentes fins ou públicos-alvo entre elas.

Por fim, apresentamos uma análise dos tópicos identificados pelo BERTopic [7][15], um método que incorpora algoritmos para pesquisar automaticamente tópicos densos em nos enunciados dos exercícios de programação, assumindo que textos semanticamente semelhantes formam tópicos. Na Tabela 3, apresentamos os 10 principais tópicos identificados, que revelam padrões interessantes dos enunciados, com frequências variando entre 136 e 348 exercícios. Pode-se notar que todos os tópicos compartilham um vocabulário relacionado à estrutura de exercícios de programação, com termos recorrentes como "linha", "número", "entrada" e "cada", indicando uma forte presença de instruções sobre o formato de entrada ou saída de dados. O tópico mais frequente (i.e., #6), com 348 exercícios, apresenta os termos "cada", "número", "linha", "contém", "entrada", sugerindo enunciados que descrevem um padrão comum em problemas de programação, onde os dados são apresentados em sequência, um após o outro. Além disso, notamos palavras relacionadas a jogos como "jogador", "jogo", "cartas" e "tabuleiro" aparecendo em vários tópicos (e.g., #1, #7 e #8), o que mostra o uso de histórias de jogos para apresentar problemas de programação, tornando os exercícios mais interessantes para quem está aprendendo. O tópico #9 se destaca por incluir termos mais específicos de programação como "palavra" e "programa", enquanto o tópico #7, aparecendo em menor número (136 exercícios), sugere problemas envolvendo valores e sequências numéricas. Esta distribuição indica que os enunciados seguem padrões estruturais bem definidos e algumas aparições de termos pontuais mas diferentes pelos tópicos listados. Vale ressaltar que o número de exercícios apresentados na Tabela 3 (2.093) não corresponde ao total de exercícios do repositório ProgramExBR (2.102) pois o mesmo exercício pode ser associado a mais de um tópico, gerando, ao final, um número de exercícios que se difere do do número de exercícios processados considerando os 10 tópicos mais frequentes.

5 CONSIDERAÇÕES FINAIS

Neste trabalho, apresentamos o ProgramExBR, um repositório composto por 2.102 enunciados de exercícios de programação e que está publicamente disponível em: https://doi.org/10.5281/zenodo.15724871, contendo estruturas de dados (JSONs) que armazenam os dados dos exercícios, isto é, nome, URL de origem, descrição, definição de entrada, definição de saída, casos de teste e metadados (quando disponíveis), sendo eles: limite de memória, tempo limite de execução e dificuldade. Além disso, realizamos uma caracterização

[15] https://maartengr.github.io/BERTopic/index.html, Modelo: all-MiniLM-L6-v2

Tabela 3: Top-10 tópicos mais representativos identificados a partir do BERTopic.

#	Palavras	#Exercícios
0	estrada, cidades, cada, linha, número	207
1	número, cada, linha, jogo, jogador	178
2	linha, inteiros, número, inteiro, valor	242
3	cada, número, linha, entrada, contém	284
4	linha, pontos, número, inteiros, entrada	140
5	linha, número, cada, figura, sala	164
6	cada, número, linha, contém, entrada	348
7	números, inteiro, linha, valor, sequência	142
8	cartas, jogo, jogador, tabuleiro, número	146
9	linha, cada, palavra, programa, entrada	242

dos dados que revelou características relacionadas ao padrão textual dos exercícios.

Em resumo, os exercícios incluídos no repositório disponibilizado neste artigo cobrem uma uma ampla diversidade de assuntos, incluindo exercícios básicos, como operações básicas, estruturas de decisão e repetição, arrays e matrizes, técnicas de ordenação e busca, programação dinâmica, estruturas de dados, matemática, grafos, técnicas de aplicação em strings, dentre outros. Além disso, acreditamos que algumas direções de pesquisa que podem se beneficiar dos dados disponibilizados no repositório ProgramExBR, incluindo:

Ensino e Prática de Programação. Os dados disponibilizados podem ser úteis no contexto educacional, uma vez que oferecem um repositório variado de exercícios de programação em português que podem ser usados diretamente, por professores ou estudantes, ou servir como base para a criação de novos. Por exemplo, um professor pode selecionar questões para compor listas de exercícios personalizadas para sua turma, organizando-as de acordo com os tópicos abordados em aula (e.g., estruturas condicionais, laços de repetição, etc), uma pré-filtragem com essa finalidade pode ser obtida, inclusive, com o auxilio dos tópicos gerados pelo BERTopic (Tabela 3). Além disso, esses dados podem alimentar sistemas de recomendação automática de exercícios, que sugerem automaticamente ao estudante novos exercícios relacionados às suas dificuldades ou progressos, garantindo uma trilha de aprendizagem mais adaptada e direcionada.

Classificação Automática de Exercícios de Programação. A partir das características extraídas dos exercícios, como tamanho do enunciado, número de casos de teste, presença de restrições específicas ou tipo de lógica necessária para a resolução do exercício, é possível aplicar modelos de aprendizado de máquina e processamento de linguagem natural para estimar automaticamente o nível de dificuldade de cada exercício.

Geração Automática de Exercícios de Programação. Os dados estruturados do ProgramExBR, podem servir como base para o desenvolvimento de sistemas de geração automática de exercícios, incluindo a proposição de novos (e.g., por meio de uso de modelos generativos) bem como a geração de variações de problemas existentes (e.g., modificação de valores, contextos ou restrições).

Vale ressaltar que este trabalho apresenta algumas limitações que devem ser consideradas na análise dos dados extraídos. Primeiramente, um mesmo exercício pode estar disponível em mais de uma plataforma. Como não foi implementado um mecanismo de detecção e remoção de duplicatas, essa redundância pode se fazer presente no repositório devido ao fato de que as plataformas selecionadas podem possuir uma interseção quanto à fonte dos exercícios. Explorar a remoção de duplicatas é algo que pretendemos explorar em etapas futuras deste estudo.

Além disso, a conversão dos exercícios para o formato JSON pode ocasionar a perda de parte da formatação original do conteúdo, como elementos de marcação e imagens. Embora essa transformação simplifique o armazenamento e o processamento automatizado, ela pode comprometer a fidelidade do conteúdo. Para solucionar esse problema, a URL original de cada exercício é preservada no ProgramExBR, permitindo que o usuário consulte o site de origem sempre que necessário. Por fim, é importante destacar que certos elementos visuais, como gráficos ou diagramas, não são incorporados diretamente ao ProgramExBR. Esses elementos, embora possam ser essenciais para a compreensão plena do enunciado, permanecem acessíveis apenas por meio da URL original do exercício, que está incluída tanto no JSON que armazena o exercício, quanto também no JSON de metadados.

Como trabalhos futuros, pretendemos explorar a agregação de outros metadados ao repositório, como por exemplo, código-fonte da solução e aplicação de técnicas de processamento de linguagem natural para identificação dos tópicos específicos de cada exercício bem como indicadores de dificuldade.

Agradecimentos. CAPES, FAPEMIG, INCT-TILD-IAR e PIBEN/UFV–FUNARBEN.

REFERÊNCIAS

[1] Arthur Alves, Leandro Silva Galvão de Carvalho, Elaine Oliveira, and David Fernandes. 2019. Análise comportamental em juízes online para predição do desempenho final de alunos em disciplinas de computação. In *SBIE*.
[2] Brett A Becker and Keith Quille. 2019. 50 years of CS1 at SIGCSE: A review of the evolution of introductory programming education research. In *ACM SIGCSE TS*.
[3] Jean Luca Bez, Carlos E Ferreira, and Neilor Tonin. 2013. Uri online judge academic: A tool for professors. In *ICAICTE*.
[4] Cassio P. Campos and Carlos E. Ferreira. 2004. BOCA: um sistema de apoio a competições de programação. In *WEI*.
[5] Leandro Galvão, David Fernandes, and Bruno Gadelha. 2016. Juiz online como ferramenta de apoio a uma metodologia de ensino híbrido em programação. In *SBIE*.
[6] Dragan Gašević and Agathe Merceron. 2022. *The Handbook of Learning Analytics* (2 ed.). SOLAR. https://www.solaresearch.org/publications/hla-22/
[7] Maarten Grootendorst. 2022. BERTopic: Neural topic modeling with a class-based TF-IDF procedure. arXiv:2203.05794 [cs.CL]
[8] Dion Ribeiro Laranjeira. 2020. *Recomendação de exercícios para alunos de programação em um ambiente de correção automática de códigos.* Mestrado em Informática. Universidade Federal do Amazonas, Manaus. 110 f..
[9] Rodrigo Pessoa Medeiros, Geber Lisboa Ramalho, and Taciana Pontual Falcão. 2018. A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education* 62, 2 (2018).
[10] Filipe D Pereira, Elaine HT Oliveira, David BF Oliveira, Alexandra I Cristea, Leandro SG Carvalho, Samuel C Fonseca, Armando Toda, and Seiji Isotani. 2020. Using learning analytics in the Amazonas: understanding students' behaviour in introductory programming. *British journal of educational technology* 51, 4 (2020), 955–972.
[11] Matthew Peveler, Jeramey Tyler, Samuel Breese, Barbara Cutler, and Ana Milanova. 2017. Submitty: An Open Source, Highly-Configurable Platform for Grading of Programming Assignments. In *ACM SIGCSE TS*.
[12] Gabryella Rodrigues, Ana Francisca Monteiro, and António Osório. 2022. Introductory Programming in Higher Education: A Systematic Literature Review. *OASIcs, Volume 102, ICPEC 2022* 102 (2022), 4:1–4:17. https://doi.org/10.4230/OASICS.ICPEC.2022.4
[13] Carlos Silva, João Solano, André Santos, and Julio Reis. 2023. Previsão de Reprovações em Disciplinas Introdutórias de Programação: Um Estudo em um Ambiente de Correção Automática de Códigos. In *SBIE*.
[14] Ícaro Alvim, Roberto Bittencourt, and Rodrigo Duran. 2024. Evasão nos Cursos de Graduação em Computação no Brasil. In *SBIE*.