ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
ENGENHARIA DE COMPUTAÇÃO

MARIA EDUARDA CORRADINI TOLINO

PROJETO DE FORMATURA - C16

CLASSIFICAÇÃO DE DIFICULDADES NO APRENDIZADO INTRODUTÓRIO DE PROGRAMAÇÃO: uma abordagem analítica

SÃO PAULO
2022

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
ENGENHARIA DE COMPUTAÇÃO

MARIA EDUARDA CORRADINI TOLINO

PROJETO DE FORMATURA - C16

CLASSIFICAÇÃO DE DIFICULDADES NO APRENDIZADO INTRODUTÓRIO DE PROGRAMAÇÃO: uma abordagem analítica

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Bacharel em Engenharia.

SÃO PAULO
2022

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
ENGENHARIA DE COMPUTAÇÃO

MARIA EDUARDA CORRADINI TOLINO

PROJETO DE FORMATURA - C16

CLASSIFICAÇÃO DE DIFICULDADES NO APRENDIZADO INTRODUTÓRIO DE PROGRAMAÇÃO: uma abordagem analítica

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Bacharel em Engenharia.

Área de concentração: Engenharia de Computação

Orientadora: Prof.ª Dra. Anarosa Alves Franco Brandão
Coorientador: Prof. Dr. Leônidas de Oliveira Brandão

SÃO PAULO
2022

AGRADECIMENTOS

A todos que me acompanharam até aqui e me deram todo o suporte que precisei para chegar a esse momento, agradeço com todo meu amor, principalmente à minha família, meus pais e meu irmão, que acompanharam toda a minha trajetória, aos meus amigos do Brasil e do intercâmbio, que foram essenciais para minha perseverança, e aos meus amigos do trabalho, que me deram apoio e compreensão em todos os momentos que precisei. Aos meus orientadores, meu muito obrigado por tanto terem me ensinado.

À Escola Politécnica e à Universidade de São Paulo, pela educação de qualidade e consciência social que me proporcionaram pelo ensino gratuito e de qualidade.

RESUMO

Com o avanço da tecnologia, o uso de plataformas para apoio ao aprendizado de alunos para o acompanhamento de seus cursos tornou-se uma ferramenta muito importante, principalmente após a transferência das atividades escolares e universitárias do ambiente físico para o remoto que se deu a partir do início da pandemia de Covid-19. Programar é uma tarefa desafiadora, principalmente quando se está começando a aprender. As estatísticas de reprovação na disciplina de Introdução à Programação em toda a Universidade de São Paulo mostram que essa é uma das disciplinas com maiores taxas de reprovação. A plataforma utilizada pela Universidade de São Paulo, o Moodle, disponibiliza dados sobre a interação do aluno com as suas ferramentas de aprendizado e avaliação. No projeto, os dados de desempenho dos alunos na disciplina de Introdução à Computação serão utilizados em análises qualitativas e quantitativas a fim de entender as dificuldades dos exercícios propostos ao longo do curso e então poder melhorar a qualidade do ensino e o desempenho dos alunos de disciplinas de programação.

Palavras chave: Moodle, aprendizado, programação, educação.

ABSTRACT

As technology advances, the use of learning support platforms for students to follow their courses has become a very important tool, especially after the transfer of school and university activities from the physical to the remote environment that has taken place since the onset of the Covid-19 pandemic. Programming is a challenging task, especially when you are just starting to learn. Statistics from Universidade de São Paulo show that Introduction to Programming is one of the subjects with the highest failure rates. The platform used by the Universidade de São Paulo, Moodle, provides data on student interaction with its learning and assessment tools. In this project, the performance data from students in Introduction to Computing will be used in qualitative and quantitative analysis in order to understand the difficulties of the exercises proposed throughout the course and then be able to improve the quality of teaching and the performance of students in programming courses.

Key words: Moodle, learning, programming, education.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

Moodle - *Modular Object-Oriented Dynamic Learning Environment*

USP - Universidade de São Paulo

IME - Instituto de Matemática e Estatística

POLI - Escola Politécnica da Universidade de São Paulo

MAC 110 - Código da disciplina

MAC 2166 - Código da disciplina

VPL - *Virtual Programming Lab*

EDM - *Educational Data Mining*

LA - *Learning Analytics*

PHP - *Hypertext Preprocessor*

HTML - *HyperText Markup Language*

SQL - *Structured Language Query*

CONEP - Comitê Nacional de Ética em Pesquisa

SUMÁRIO

1. INTRODUÇÃO

1.1. CONTEXTO

Com o advento da tecnologia, o uso de plataformas para apoio ao aprendizado de alunos para o acompanhamento de cursos se tornou uma ferramenta muito importante, não apenas para o compartilhamento de conteúdo e para a interação entre professor e aluno, mas também para a avaliação de seus conhecimentos. Devido à transferência das atividades escolares do ambiente físico para o remoto que se deu a partir do início da pandemia de Covid-19 em 2020, plataformas como o Moodle - sistema utilizado por diversas universidades, dentre elas, a Universidade de São Paulo - adquiriram papel crucial para a continuidade das atividades de aprendizado.

1.2. PROBLEMÁTICA

Programar é uma tarefa desafiadora, principalmente quando se está começando a aprender. Conforme este ramo se expande e é mais procurado profissionalmente, torna-se cada vez mais importante garantir que o ensino de programação seja feito de forma a facilitar o aprendizado e a fixação dos conteúdos pelos alunos. As estatísticas de reprovação na disciplina de Introdução à Programação não apenas na Escola Politécnica, mas em toda a Universidade de São Paulo, mostram que existe dificuldade no aprendizado dos alunos, pois sua taxa de reprovação foi de cerca de 26,7% entre 2010 e 2014, como relatado em Bosse (2015).

1.3. OBJETIVO

A plataforma Moodle disponibiliza múltiplos dados a respeito da interação do aluno com as suas ferramentas de aprendizado e avaliação que podem ser adicionadas aos cursos dada a liberdade proporcionada pela modularidade da plataforma. Ao longo deste projeto, serão explorados os dados relacionados à resolução de exercícios de produção de códigos de programação, nas disciplinas de Introdução à Computação, MAC 2166, em oferecimento *online* aos alunos de

Engenharia na Escola Politécnica que já reprovaram a disciplina pelo menos uma vez, e MAC 110, em oferecimento presencial, mas com atividades *online* para os alunos do Instituto de Matemática e Estatística, ambas da Universidade de São Paulo (USP). Nessas disciplinas, junto ao *Moodle* é utilizado o módulo VPL - *Virtual Programming Lab*[1], em que é possível criar e editar programas, executá-los e obter avaliação automática das atividades. Sua avaliação é feita por meio de casos de teste adicionados junto ao enunciado pelo criador da atividade.

O objetivo deste projeto é analisar os dados disponibilizados pela plataforma para entender as dificuldades dos exercícios propostos ao longo do curso de Introdução à Computação. Então, serão realizadas análises quantitativas e qualitativas a respeito da evolução dos alunos e do conteúdo a eles ministrado e assim melhorar a qualidade do ensino e o desempenho dos alunos de disciplinas de programação.

Além disso, também serão analisados os exercícios propriamente ditos, considerando se as dificuldades estão relacionadas à sua especificação ou ao conteúdo de programação necessário para realizá-lo com sucesso.

Com os dados disponíveis no módulo VPL, que estão armazenados no Moodle, é possível extrair a informação de quanto tempo o aluno demorou para construir sua solução, quantas vezes as soluções foram submetidas até o envio da resposta final e quantas modificações foram realizadas em cada submissão. Com um sistema de correção automática, ao submeter a solução, o aluno possui acesso à nota atribuída instantaneamente. Isto proporciona a oportunidade de correção e modificação do código antes enviado e a verificação da validade da solução.

A ideia inicial deste projeto era utilizar os dados existentes de forma a desenvolver um algoritmo de *Machine Learning* para prever quando um aluno seria reprovado na disciplina de acordo com as suas interações com o Moodle e com o seu desempenho nas atividades realizadas no VPL ao longo da matéria. Porém, para desenvolver um bom algoritmo de aprendizagem é necessário um extenso *dataset* para teste e para treinamento, e como apenas estão disponíveis os dados dos alunos dos anos de 2021 e 2022, sendo em média 80 alunos por ano no reoferecimento, o espaço amostral se mostra muito reduzido para a aplicação de *Machine Learning*. Em razão disto, um dos objetivos deste projeto é desenvolver

[1] https://vpl.dis.ulpgc.es/index.php/home/presentation

então um módulo que será adicionado ao Moodle, a partir do qual será possível visualizar os dados levantados como essenciais para o entendimento do desempenho dos alunos na disciplina, utilizando as informações provenientes do iTarefa, em que os alunos realizam exercícios de programação em bloco no início do aprendizado. O módulo também poderá ser utilizado para outras disciplinas que realizam atividades a partir do Moodle utilizando o iTarefa, para que esta técnica possa ser aplicada para evitar reprovações. Também serão relatados estudos de casos de alunos e métodos de intervenção utilizados para auxiliar no seu aprendizado, que têm como recomendação serem realizados de acordo com seu desempenho.

Outros dados foram adquiridos por meio de questionários e avaliações feitas pelos alunos, e estes serão melhor explicados no capítulo três.

A partir disso, pretende-se identificar as atividades em que os alunos possuem maior dificuldade, assim como o conteúdo exigido por elas, para que seja possível reconhecer as seções do curso em que os alunos possuem maiores problemas de aprendizagem.

Pelas análises de dados realizadas, será possível identificar com antecedência alunos com maior chance de reprovação e, então, construir um método de prevenção de reprovações, para diminuir a taxa de alunos que precisam cursar a matéria novamente em outros oferecimentos.

Somado a essas análises, este projeto proporcionará aos docentes que ministram a disciplina entender se a dificuldade do aluno está em aprender o conteúdo ou se está relacionada a como a questão foi redigida. Além disso, serão estudadas outras ações necessárias para melhorar o aprendizado dos alunos.

Esta monografia apresentará no capítulo dois a fundamentação teórica e trabalhos relacionados, no terceiro, as especificações do projeto, no quarto, o desenvolvimento do módulo para visibilidade do desempenho dos alunos, no quinto, os métodos de intervenção para auxiliá-los e no sexto as conclusões finais.

2. FUNDAMENTAÇÃO TEÓRICA E TRABALHOS RELACIONADOS

Nesta seção são apresentados os conceitos teóricos que serão utilizados para a realização das análises e as referências de trabalhos relacionados ao aprendizado introdutório de programação.

2.1. FUNDAMENTAÇÃO TEÓRICA

2.1.1. LEARNING ANALYTICS

De acordo com Elias (2011), *Learning Analytics* é produto da interação entre instrutores e tutores com o conteúdo e os alunos. Este campo tem como objetivo utilizar métodos estatísticos para a realização de análises que respondam a questões como: "Quão efetivo é o curso", "O curso está atendendo às necessidades dos alunos", "Como pode-se dar maior suporte às necessidades de quem aprende", "Quão efetivas são as interações" e "Como pode-se melhorar o ensino". A utilização da técnica de *Learning Analytics* se mostra cada vez mais promissora graças ao aumento de dados de cursos advindos da mobilidade acadêmica para o ensino online.

2.1.2. ENSINO DE ALGORITMOS

Em Barcelos (2013), são descritos os passos para a construção de um algoritmo, que devem ser aprendidos pelo aluno: primeiramente, é necessário identificar o problema e, então, seguir o conjunto de procedimentos que levará à sua solução. É pontuado também na pesquisa que o correto para a resolução é subdividi-lo em problemas menores, a fim de que cada problema possa ser tratado como uma tarefa indivisível, que será transformada em código. Além disso, na pesquisa são explicadas as características do processo de construção do conhecimento de algoritmos, sendo elas: aulas expositivas (há preocupação com a variedade e quantidade de conteúdo, em detrimento à reflexão e análise para solução de problemas), peculiaridades dos alunos, peculiaridades dos professores

(como as motivações, as dificuldades e os estilos de aprendizagem abordados por eles, e também a utilização de tecnologias e métodos de avaliação). De acordo com o autor, é necessário motivar os alunos a aprimorarem e praticarem suas habilidades e peculiaridades, e o contato desses com os professores deve ser feito a partir das peculiaridades dos professores.

No mesmo texto, para o ensino de algoritmos, são pontuadas as características do processo de aprendizagem. É necessário que o aluno tenha habilidades de abstração, resolução de problemas, raciocínio lógico e concentração. A interação dos alunos com os professores é feita por meio de motivação, dificuldades, estilos de aprendizagem, utilização de tecnologias e avaliação dos conhecimentos, sendo papel do professor motivar os alunos a aprimorarem e praticarem suas habilidades.

No contexto de uma turma, sabe-se que os estudantes apresentam ritmos diferentes de assimilação de conteúdo e, por isso, são necessárias estratégias educacionais para diminuir as diferenças.

2.1.3. DIFICULDADES ASSOCIADAS

Ainda de acordo com Barcelos (2013), as principais causas para as dificuldades de aprendizagem são: o ensino tradicional (pode ser aprimorada por meio do uso de tecnologia), a falta de motivação, os métodos de avaliação, o material didático, a incompatibilidade do aluno com a área de programação, a linguagem de programação e a abstração. Muitas vezes a opinião de alunos que já tiveram experiências negativas com a disciplina pode causar desmotivação, assim como conceitos muito abstratos e longe da sua realidade. Avaliações pontuais não ajudam no processo contínuo de aprendizagem, assim como materiais didáticos dissociados às aulas, sem exercícios de raciocínio lógico ou até mesmo exemplos do quotidiano. A linguagem de programação pode dificultar o aprendizado, e o mesmo pode ocorrer pela falta de conhecimento na língua inglesa (utilizada nos comandos das linguagens de programação).

É necessário desenvolver no aluno a capacidade de identificar as características essenciais e específicas da programação. A capacidade de abstração deve ser desenvolvida e sua falta pode ser relacionada aos períodos anteriores de

aprendizado, nos Ensinos Fundamental e Médio, caso o aluno não tenha sido provocado a criticar e pensar, apenas a memorizar conteúdos.

2.1.4. FORMAS DE MITIGAR AS DIFICULDADES

Seguindo a pesquisa de Barcelos (2013), a fim de melhorar os métodos de aprendizado de programação, é importante utilizar ferramentas diversas, pois cada aluno possui sua forma de aprender. Utilizar videoaulas, questionários ou até mesmo aplicativos para demonstrar o passo a passo, assim como o funcionamento de exercícios que utilizam algoritmos para sua resolução. Para estimular a abstração, pode-se enviar a resolução dos problemas de algoritmos, ensinar aos alunos como construir um teste de mesa (pois estes possibilitam reflexão e raciocínio lógico, além da abstração); utilizar formas diferentes de representação de algoritmos é um grande auxílio para isso. Também é necessário trabalho pessoal do aluno a respeito de ações mentais que levam à construção do seu próprio conhecimento; isso exige tempo e experimentação, pois não é possível seguir o mesmo método para todos, em razão da preferência de cada um.

2.2. TRABALHOS RELACIONADOS

De Souza (2021) afirma que aprender a programar é um processo complicado para iniciantes, pois é necessário que eles tenham "conhecimento declarativo e procedimental, memorização, compreensão, resolução de problemas, abstração e capacidade de raciocínio", e que "a maior dificuldade encontrada pelos alunos é a compreensão de estruturas de programação". Além disso, é citado no artigo que essa dificuldade pode ser um dos motivos pelos quais os cursos na área de computação possuem um dos maiores índices de evasão (36% entre 2000 e 2005) na USP. Dentre os dados explorados na pesquisa, a disciplina de MAC 2166, ministrada para os alunos durante o primeiro semestre de engenharia na Escola Politécnica da USP, que será utilizada ao longo deste projeto, é uma das que possui maiores índices de reprovação (15% entre 2010 e 2014), e mais de um quarto dos alunos que foram aprovados precisaram cursar a disciplina mais de duas vezes.

Já para Bosse (2015), quando observam-se as dificuldades dos alunos ao aprender a programar, as mais recorrentes são ponteiros, tratamento de erros e recursão, e que o maior problema encontra-se em como aplicar os conceitos na prática. Nesse momento, os maiores impasses são, respectivamente, erros de sintaxe, tipos de variável e complicações com a linguagem, e que a escolha da linguagem impacta no desenvolvimento do aluno, pois uma linguagem como C possui muito mais detalhes sintáticos do que uma como *Python*.

De acordo com Romero (2020), para analisar dados de educação, existem dois métodos que podem ser utilizados: *Educational Data Mining* (EDM), que foca em desenvolver métodos para exploração de dados advindos de plataformas educacionais, e *Learning Analytics* (LA), que divide-se em três elementos importantes: dados, análises e ações.

2.3. TÉCNICAS DE DESENVOLVIMENTO

Para desenvolver o módulo no *Moodle* é necessário acessar o banco de dados com as informações referentes às atividades realizadas no VPL, utilizando a linguagem SQL (*Structured Language Query*). Também é necessário o desenvolvimento em PHP (*Hypertext Preprocessor*, cujo nome original era *Personal Home Page*), uma linguagem de script open source comumente utilizada em desenvolvimento web, em que é possível escrever partes do código utilizando HTML (*HyperText Markup Language*). Com isto, será possível analisar os dados e desenvolver um painel de desempenho dos alunos.

Para isso, foi necessária a criação de um ambiente de teste na plataforma Moodle, que é um servidor à parte da plataforma USP. Com o ambiente de teste pronto, foi desenvolvido então o código para a captura dos dados, para o tratamento desses dados, e para a construção da visualização destes. Com isto, é possível definir métodos a serem realizados quando é necessário intervir no aprendizado de um aluno, dado que seu desempenho mostra que este não está aprendendo os conceitos e sabendo aplicá-los nas atividades.

2.4. ENTREVISTAS

De modo a complementar os dados coletados pelo Moodle, para entender melhor a interação dos alunos com as disciplinas de introdução a computação utilizadas (MAC 2166 e MAC 110), ao longo do projeto foram realizadas entrevistas com alunos que já participaram delas e tiveram um desempenho baixo, mas que foram aprovados na matéria apesar disso, e também com alunos que estavam cursando no presente momento. Foi desenvolvido um roteiro para as entrevistas e este está detalhado no apêndice C, bem como as respostas obtidas até o momento a partir delas.

Somadas às entrevistas com alunos que cursaram oferecimentos anteriores da disciplina, também foram realizadas sessões com os alunos dos oferecimentos de 2022, com a finalidade de entender de forma mais próxima as dificuldades de cada um. Ao longo destas entrevistas, foi pedido para que o aluno apresente a resolução de um exercício, e que tente resolvê-lo em voz alta, para que o entrevistador possa captar todas as etapas da construção da solução.

A partir das notas das atividades realizadas pelos alunos, foi também realizada uma análise de seu desempenho, e foram selecionados os alunos com médias de provas menores que cinco ou média de exercícios menores que cinco.

2.5. PESQUISAS VIA FORMULÁRIO

A fim de embasar os dados referentes às dificuldades dos alunos encontradas por meio das entrevistas e dos exercícios realizados com eles, foi realizada uma pesquisa para entender as dificuldades no aprendizado introdutório de programação. O formato de pesquisa, bem como seus resultados estão disponíveis no apêndice J.

3. ESPECIFICAÇÃO DO PROJETO

Ao longo deste projeto foram elaboradas análises qualitativas e quantitativas utilizando técnicas matemáticas e estatísticas para mineração de dados, para a melhoria do ensino introdutório de programação. Em razão disso, as especificações abordadas a seguir serão a respeito do método empregado para o desenvolvimento da análise.

A priori, foi realizada uma consulta ao Comitê Nacional de Ética em Pesquisa (CONEP) para compreender se seria necessário o registro da pesquisa no sistema do CONEP. De acordo com a resolução 510 de abril de 2016 (apêndice A), artigo 1º, parágrafo único, para Trabalhos de Conclusão de Curso, categoria no qual esta monografia se encaixa, é necessário apresentar o protocolo de pesquisa ao comitê. Então, foi realizado o contato com o comitê para seguir todas as normas éticas para a utilização dos dados de forma a respeitar a privacidade dos alunos.

Aproveitando os oferecimentos das disciplinas MAC 2166 no primeiro semestre de 2022 e MAC 110 no segundo semestre do mesmo ano, foi realizada uma análise de perfil dos alunos matriculados por meio de um questionário (apêndice B) que envolve questões para avaliar o conhecimento do aluno antes do início das aulas em programação, inglês, seu desempenho na última vez em que cursou a disciplina (pois todos já a cursaram pelo menos uma vez anteriormente), sua evolução nos cursos de matemática básica do início da graduação (Cálculo e Álgebra Linear) e também quais as maiores dificuldades enfrentadas para aprender programação. O questionário foi realizado pelo Moodle e teve como tempo limite o final da primeira semana de aulas para ser respondido por todos os alunos.

Em paralelo, foram estudados os dados disponíveis dos oferecimentos da disciplina nos anos de 2020 e 2021. Como em ambos os anos houve a aplicação de exercícios online, foi mais simples coletar informações a respeito do desempenho dos alunos.

A partir dos resultados obtidos pela análise acima, serão sugeridos métodos de intervenção no ensino da disciplina e particularidades nos exercícios que forem identificados como importantes para o aprendizado de programação.

Somado a isso, foi acompanhado o oferecimento das disciplinas ao longo dos semestres, a fim de analisar se as intervenções realizadas foram significativas para o aprendizado.

Uma das ideias iniciais deste projeto era construir um modelo preditivo utilizando técnicas de *Machine Learning* para prever a reprovação de um aluno antes de ela ocorrer, a fim de que o plano de contingência pudesse ser aplicado ao referido aluno. Porém, devido ao baixo volume de dados que poderia ser utilizado para a construção deste modelo atualmente disponível, será desenvolvido um módulo para o Moodle, de forma que nos próximos oferecimentos da disciplina possa ser realizada a coleta de dados de forma a reunir uma extensa coleção, a fim de que estes dados estejam disponíveis no futuro para a construção do modelo preditivo, e também de quaisquer outros modelos que necessitem de um grande conjunto de dados.

Nos subtópicos desta seção estão explicados os procedimentos e a ordem de desenvolvimento que serão necessários para a implementação do módulo no Moodle, bem como para a realização das análises com os dados já disponíveis de outros oferecimentos e do atual oferecimento da disciplina.

3.1. MOODLE E DESENVOLVIMENTO NA PLATAFORMA

O Moodle (*Modular Object-Oriented Dynamic Learning Environment*) é uma plataforma voltada ao aprendizado distribuída gratuitamente. De acordo com as estatísticas disponíveis no próprio domínio do Moodle, em setembro de 2022, o sistema possuía mais de 168 mil sites, com 42 milhões de cursos e 339 milhões de usuários em 242 países. Seu amplo uso se dá principalmente pela possibilidade de adaptação às necessidades de cada curso graças à sua arquitetura modular, sendo possível a programação de módulos e plugins para a plataforma.

Sua linguagem de programação é o PHP, também open source e amplamente utilizado para desenvolvimento *web*, pois permite a utilização de HTML. Para a sua disponibilização é necessária a configuração de um servidor, como por exemplo o *Apache*.

Os dados gerados pelo uso da plataforma são armazenados em um banco de dados relacional. No contexto da USP, o banco utilizado é o MariaDB, porém, ao longo do projeto, foi utilizado o MySQL. Ambos aceitam SQL para a realização de consultas e tratamento de dados. Logo, independente do banco utilizado, o

desenvolvimento realizado é capaz de ser implementado em qualquer Moodle que atenda aos requisitos que serão abordados mais adiante.

Para o desenvolvimento do bloco, é necessário desenvolver um código em PHP. Como o projeto não é desenvolvido diretamente no Moodle da USP, é necessária a criação de um Moodle privado, em um servidor, para que o módulo possa ser testado sem qualquer impacto no ambiente da disciplina. Então, as etapas seguidas para a criação do servidor Moodle em ambiente local são as listadas a seguir:

- Configurar um servidor base: foi utilizado o servidor Apache, que é *open source* (código fonte é disponibilizado abertamente ao público que permite uso e redistribuição gratuitamente) e é amplamente utilizado no mundo todo, com isso, o suporte para o desenvolvimento é benéfico, pois há uma grande documentação a respeito do uso deste servidor;
- Clonar repositório *git* do Moodle no servidor: realizado dentro do servidor;
- Criar uma base de dados: utilizado o *MySQL* para esta etapa do desenvolvimento;
- Instalar o PHP dentro do servidor;
- Criar o diretório de dados do Moodle: um diretório que permite a escrita do servidor;
- Instalar o Moodle;
- Configurar últimos detalhes do Moodle: nesta etapa, foram definidos detalhes como o fuso horário, a fim de evitar problema caso seja necessário o uso de funções de data.

O Moodle disponibiliza uma documentação a respeito das especificidades de sua instalação, bem como um Fórum para a discussão sobre eventuais problemas. Ambos foram utilizados ao longo desta etapa de configuração como referência e como suporte.

Após a instalação do Moodle, foram desenvolvidos códigos para a realização de consultas aos dados referentes ao desempenho dos alunos nas atividades do iTarefa (*query* em SQL) e também o tratamento para a realização das análises de desempenho do aluno para a identificação de casos em que é necessário ter atenção para que este aluno consiga aprender a matéria, dado que no módulo serão identificados alunos com dificuldades de aprendizado.

Para o desenvolvimento em PHP, o suporte às atividades realizadas por outros alunos que também estão desenvolvendo *plugins* para o Moodle e são mestrandos dos orientadores deste projeto.

3.1.1. PLUGINS NO MOODLE

A plataforma Moodle, como já citado anteriormente, é um sistema modular. Isso possibilita a adição de serviços além das funcionalidades principais dela. Por ser *open source*, existem diversas funcionalidades adicionais, chamadas de *plugins*, que podem ser incorporadas em diversos contextos da plataforma, seja no contexto geral, abrangendo todos os cursos do ambiente seja no contexto de um curso, ou até mesmo de uma atividade existente dentro de uma disciplina existente. No próprio site do Moodle, é possível acessar e adicionar estes *plugins* desenvolvidos pelos próprios usuários.

Os *plugins* são divididos em categorias, totalizando em novembro de 2022 mais de 50 categorias, que se diferenciam pela sua variedade de uso. *Plugins* do tipo *Activity Modules*, por exemplo, são essenciais para a existência de funcionalidades essenciais para os cursos, como Fórum de discussão, Quizzes e Tarefas (recebimento de arquivos de tarefa, como relatórios, apresentações, etc.). Já *Admin Tools* permitem o uso de *scripts* importantes para a administração da plataforma e também para a execução de tarefas de manutenção.

Os *plugins* do tipo Bloco permitem a disponibilização de informações no contexto dos cursos, podendo ser disponibilizados dentro das páginas. Este é o tipo de *plugin* que foi escolhido para o desenvolvimento do projeto, que disponibilizará um *dashboard* para que os administradores dos cursos que utilizam o *plugin* do *iTarefa* possam visualizar o desempenho dos alunos inscritos no curso.

3.2. ITAREFA

O *plugin* do Moodle iTarefa, ou *iAssign* em inglês, é um acrônimo para *Interactive Assignment* (tarefa interativa em tradução livre), é um pacote criado por professores do Instituto de Matemática e Estatística da Universidade de São Paulo (IME-USP) que faz parte de um módulo de aprendizagem interativo. A partir da sua

utilização, os professores são capazes de criar blocos de exercícios dentro de suas matérias. Nestes exercícios, os professores escrevem um enunciado, e este é resolvido no ivProg (Visual and Interactive Programming on the Internet), interface que permite ao aluno programar utilizando blocos de programação prontos, facilitando a aprendizagem introdutória de programação.

Aos alunos, é permitido que realizem quantas submissões de programas quiserem, desde que o exercício esteja dentro do prazo permitido para sua realização, e sua nota pode ser visualizada em questão de segundos. Para realizar esta checagem e disponibilizar uma nota ao aluno, ao criar um novo exercício, o administrador deve adicionar *inputs* e seus respectivos *outputs* esperados, para que estes casos sejam testados nos programas realizados pelos alunos. Então, a partir desses testes, o plugin é capaz de atribuir uma nota relacionada ao desempenho do aluno na realização da atividade proposta.

Todas as submissões dos alunos são armazenadas no banco de dados do Moodle, e as tabelas responsáveis pela existência destes dados e que possibilitam a realização do projeto serão melhor detalhadas ao longo do capítulo 3.4 do relatório.

Possuir atividades no iTarefa é o único requisito para a utilização do bloco *Dashboard*, que será detalhado na próxima sessão.

3.3. DASHBOARD

Para facilitar a visualização dos dados referentes ao desempenho dos alunos nas atividades realizadas por eles no *plugin* do iTarefa, um painel de acompanhamento da evolução do curso foi desenvolvido como um bloco no Moodle, para que, caso o professor que ministra o curso queira, ele possa ser adicionado aos recursos presentes na plataforma de ensino.

A escolha do *plugin* do tipo Bloco para o projeto se deu principalmente pelos seguintes motivos:

- Blocos aceitam o contexto de um curso, podendo ser adicionados a cada oferecimento;
- O recurso estará disponível para os administradores que optem por utilizá-lo e que utilizem o *plugin* iTarefa, tornando o projeto útil e aplicável no contexto estudado de aprendizado introdutório em programação.

3.4. DADOS COLETADOS

Para o desenvolvimento do Dashboard, é importante utilizar dados dos alunos que realizaram as atividades propostas. Vários dados são disponibilizados pelo *plugin* do iTarefa. Para a decisão de quais dados são mais importantes para analisar o desempenho dos alunos e suas dificuldades, foi realizada uma análise de todas as tabelas que são criadas pelo iTarefa no banco de dados do Moodle. São elas:

Tabela 1 - Lista de tabelas do banco de dados relacionadas a informações do iTarefa.

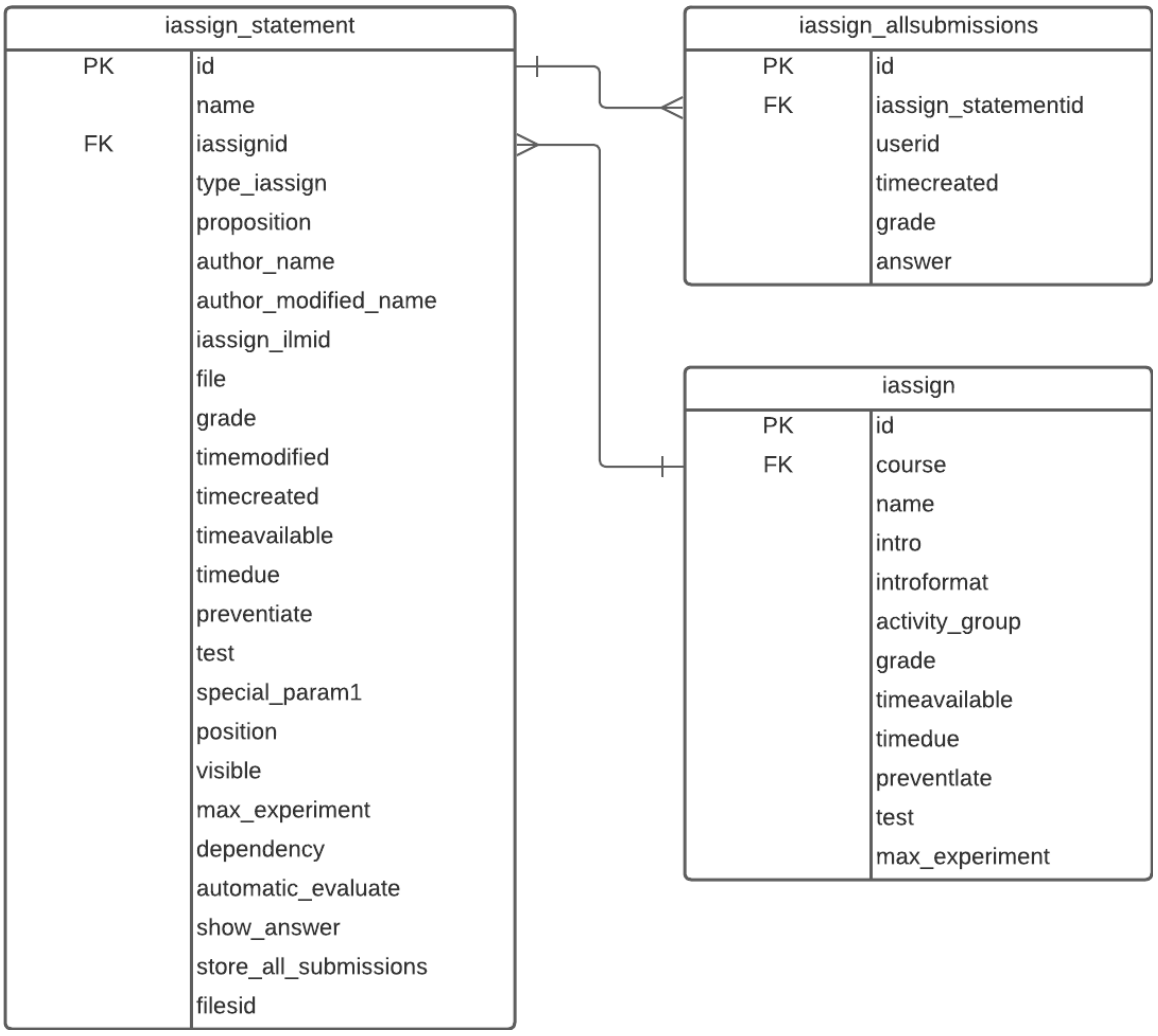
Tabela	Descrição
mdl_iassign	Relaciona a chave de identificação do curso com os blocos de exercício do iTarefa.
mdl_iassign_allsubmissions	Relaciona a chave de identificação das submissões das tarefas, dos alunos, do exercício, além da nota dada pelos testes realizados pelo iTarefa.
mdl_iassing_ilm	Sob investigação.
mdl_iassing_ilm_config	Sem registros.
mdl_iassing_log	Armazena informações a respeito de atualizações realizadas no curso (ex.: instalação, adição de plugins, exercícios).
mdl_iassing_security	Sem registros.
mdl_iassing_statement	Sem registros.
mdl_iassing_submissions	Nota da última submissão do aluno em exercício do iTarefa.
mdl_iassing_submissions_comment	Sem registros.

Fonte: produzida pela autora.

Ao longo do projeto, dentre todas as tabelas que armazenam dados referentes ao iTarefa, apenas as tabelas *mdl_iassign*, *mdl_iassign_allsubmissions* e *mdl_iassing_statement* serão utilizadas, pois são as que contêm as informações

mais importantes a respeito do desempenho dos alunos, além dos métodos para associar os exercícios com os respectivos cursos em que estão. A relação existente entre as tabelas está especificada no seguinte diagrama entidade relacionamento.

Figura 1 - Diagrama entidade relacionamento que define a relação entre as tabelas do iTarefa utilizadas.



Fonte: produzida pela autora.

Além disso, diversas outras tabelas também são disponibilizadas pelo próprio Moodle, contendo as mais diversas informações, tanto de segurança e administração, quanto referentes ao uso da plataforma pelos usuários. Dentre elas, foram escolhidas as seguintes tabelas para adquirir informações referentes aos dados pessoais dos usuários e para relacioná-los aos cursos em que estes se encontram inscritos:

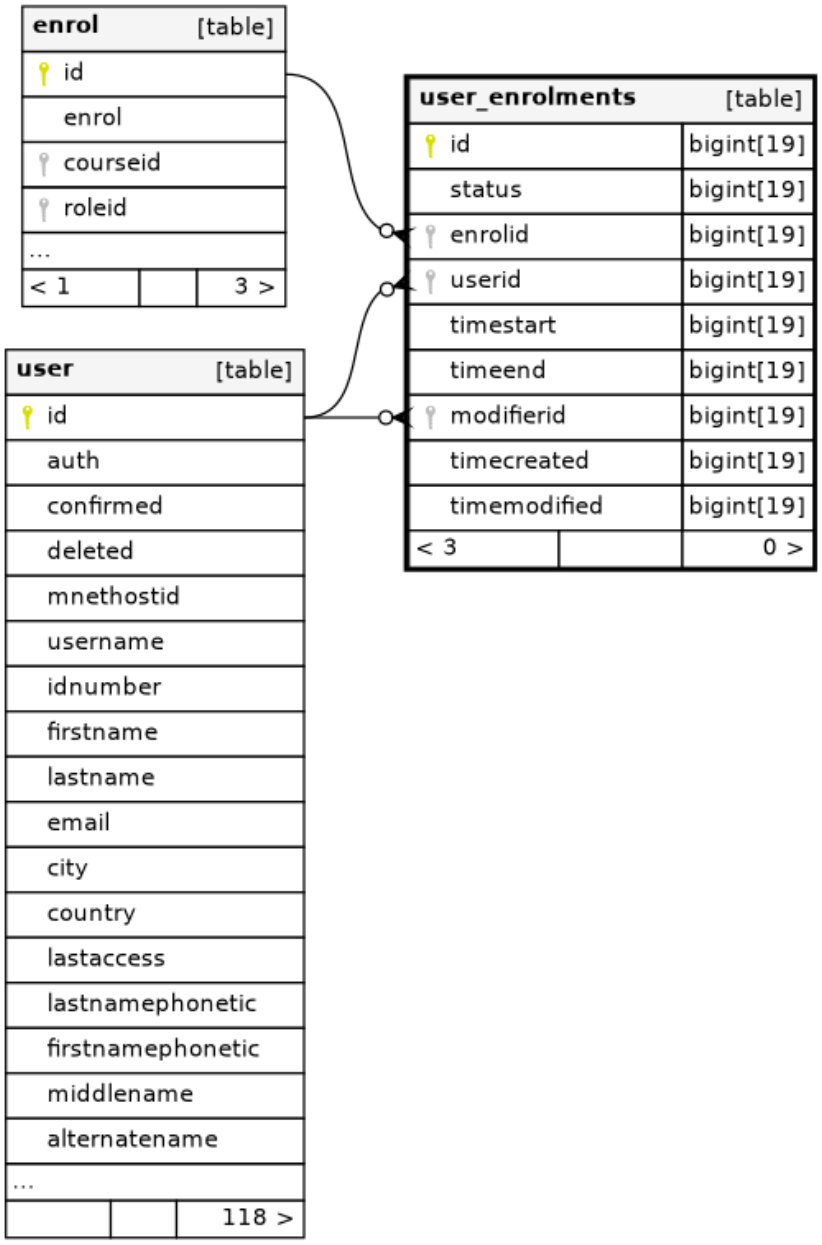
Tabela 2 - Lista de tabelas do banco de dados relacionadas a informações dos usuários, disponibilizadas pelo Moodle.

Tabela	Descrição
mdl_user	Chave de identificação do usuário, com suas informações pessoais como nome, sobrenome, e-mail.
mdl_enrol	Relaciona a chave de identificação do curso com uma chave única do registro de inscrição de um curso (*enrolment*)
mdl_user_enrolments	Relaciona a chave de identificação do usuário com o registro de inscrição de um curso (*enrolment*)

Fonte: produzida pela autora.

As relações entre as tabelas acima encontram-se explicadas no diagrama de entidade-relacionamento abaixo.

Figura 2 - Diagrama entidade relacionamento que define a relação entre usuários e cursos.



Fonte: *User Enrolments Table Schema* em *Moodle Schema* de *Zoola Analytics* [2].

A partir das informações acima citadas, realizou-se um estudo a respeito dos dados armazenados no banco de dados pelo iTarefa, a fim de entender os principais *insights* que podem ser trazidos pelo *dashboard* que será construído, para que fosse tomada a decisão de quais tabelas deveriam ser utilizadas, quais cruzamentos deveriam ser realizados e quais seriam os melhores dados para a construção de informações valiosas para os potenciais utilizadores do bloco (professores). Com

[2] Disponível em: <https://moodleschema.zoola.io/tables/user_enrolments.html>. Acesso em: nov. 2022.

isso, foi feita uma priorização de dados, e os dados considerados mais relevantes para o desenvolvimento do dashboard foram:

- Identificação do curso;
- Identificação do aluno;
- Nome do aluno;
- Identificação do bloco do exercício;
- Identificação do exercício;
- Número de submissões;
- Nota da última submissão;
- Data de envio da última solução.

Além da submissão de atividades, os alunos também respondem questionários avaliando o nível de dificuldade da questão realizada. Então, é possível coletar também os dados referentes à avaliação de dificuldade de cada aluno, tanto no enunciado quanto na execução de cada um dos exercícios que foram realizados. Os dados a serem coletados são:

- Identificação do aluno;
- Identificação do exercício;
- Dificuldade do enunciado;
- Dificuldade do exercício.

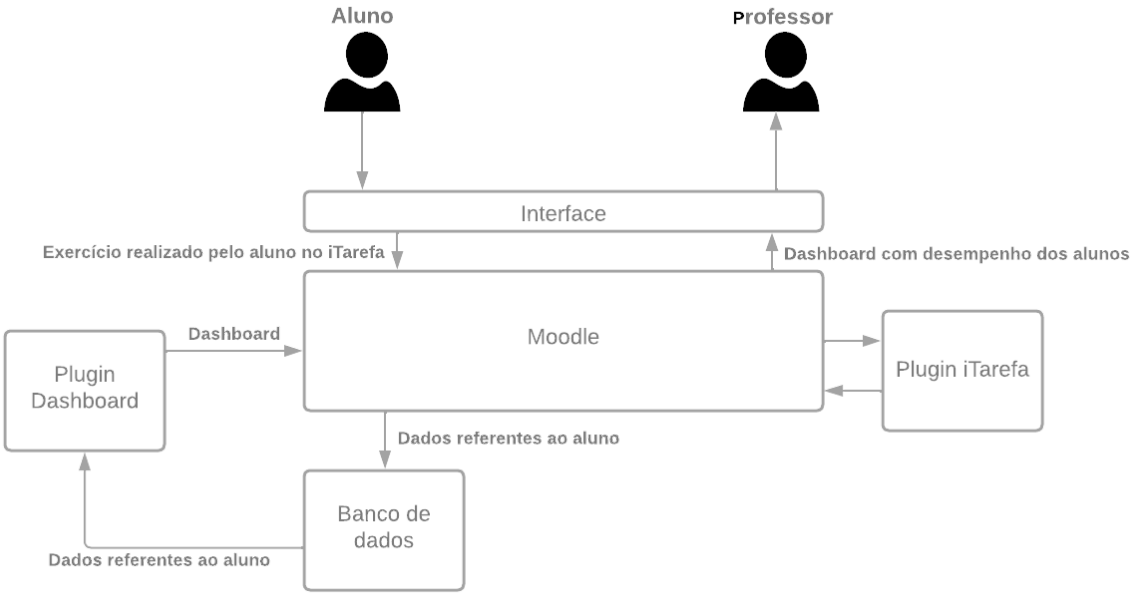
3.5. ARQUITETURA MOODLE

A arquitetura do Moodle precisou ser estudada para que houvesse entendimento do encaixe do bloco Dashboard que foi desenvolvido ao longo do projeto. Nela, existem em suma dois tipos de usuários: alunos e professores. Enquanto os professores administram seus cursos, adicionando conteúdos para os alunos e também atividades que devem ser cumpridas por esses, os alunos não possuem permissão para alterar os conteúdos da plataforma, apenas podendo adicionar suas entregas e realizações de tarefas propostas, ou interagirem com discussões no fórum.

Ambos os usuários possuem permissão para participar de cursos dentro da plataforma. No contexto mais específico do iTarefa, dentro dos cursos, as tarefas são criadas pelos professores conforme explicado no capítulo 3.2, e os dados dos alunos são adquiridos pelo plugin e armazenados no banco de dados. Então, para o projeto, estes dados são acessados pelo *plugin* Dashboard, que os trata e exibe em retorno aos professores informações referentes ao desempenho dos alunos nas atividades propostas.

A figura abaixo ilustra a arquitetura do Moodle explicada e sua relação com os *plugins* que foram utilizados ou desenvolvidos no projeto.

Figura 3 - Arquitetura do Moodle e *plugins*.



Fonte: produzida pela autora.

4. DESENVOLVIMENTO DO BLOCO DASHBOARD NO MOODLE

Para realizar o desenvolvimento do *Dashboard* específico para o iTarefa, foi necessária a instalação de um servidor Apache, do PHP e de um banco de dados. Algumas dificuldades foram encontradas neste ponto do projeto, pois o sistema operacional em que ele foi realizado era o MacOS. Então, utilizou-se a solução MAMP (MacOS Apache, MySQL and PHP) para configurar o servidor local que serviu como suporte essencial para o projeto. Após sua instalação, foi realizada a instalação do Moodle em uma versão anterior (versão 3), pois existiam até o momento de início de desenvolvimento alguns impeditivos para a compatibilidade do iTarefa na versão mais atual (versão 4). Então, com o ambiente pronto, realizou-se a criação de um Moodle local, um curso de teste, usuários de teste (alunos) e um módulo de exercícios no iTarefa. Realizando os exercícios pelo *login* dos usuários com permissão de alunos, as tabelas do banco de dados explicadas na tabela 1 foram preenchidas com dados fictícios, a fim de criar um ambiente de teste para auxiliar no processo de desenvolvimento.

4.1. BLOCO DE RESUMO DO DASHBOARD

Para que fosse disponibilizado um resumo sobre a interação dos alunos com o iTarefa para o professor, foi desenvolvido um bloco que pode ser visualizado na página inicial do curso. Por ser apenas um resumo, o bloco possui informações a respeito da média dos alunos e o número de submissões do iTarefa realizadas por esses até o presente momento.

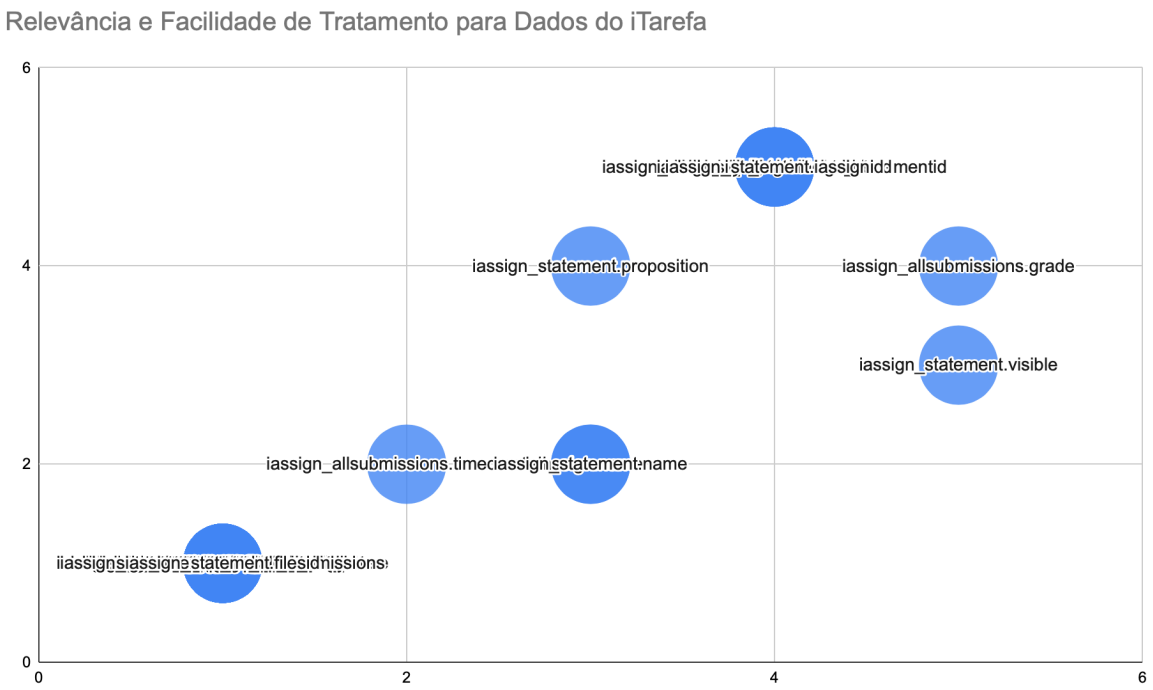
Ao final, o bloco possui um botão que permite o redirecionamento do professor a uma página em que as demais informações dos alunos estão presentes. As explicações a respeito da página serão dadas no seguinte item (4.2).

4.2. VISUALIZAÇÃO DA PÁGINA DO DASHBOARD

Esta página possui as informações escolhidas no projeto para serem disponibilizadas aos professores. Estas foram abordadas no apêndice G. A

priorização foi feita a partir da relevância da informação e da facilidade de tratamento do dado. De forma ilustrativa, os dados escolhidos localizam-se no primeiro quadrante do gráfico abaixo.

Gráfico 1 - Relevância vs. Facilidade do tratamento dos dados disponibilizados pelo iTarefa



Fonte: produzido pela autora.

Para a criação da visualização, foi necessário realizar consultas ao banco de dados do Moodle, utilizando as tabelas do capítulo 3.4. Nenhuma tabela adicional foi criada, e a reunião destes dados foi realizada no arquivo *view.php*.

As informações presentes no dashboard, bem como a área de conclusões que podem ser obtidas a partir delas são as seguintes:

Tabela 3 - Informações presentes no dashboard.

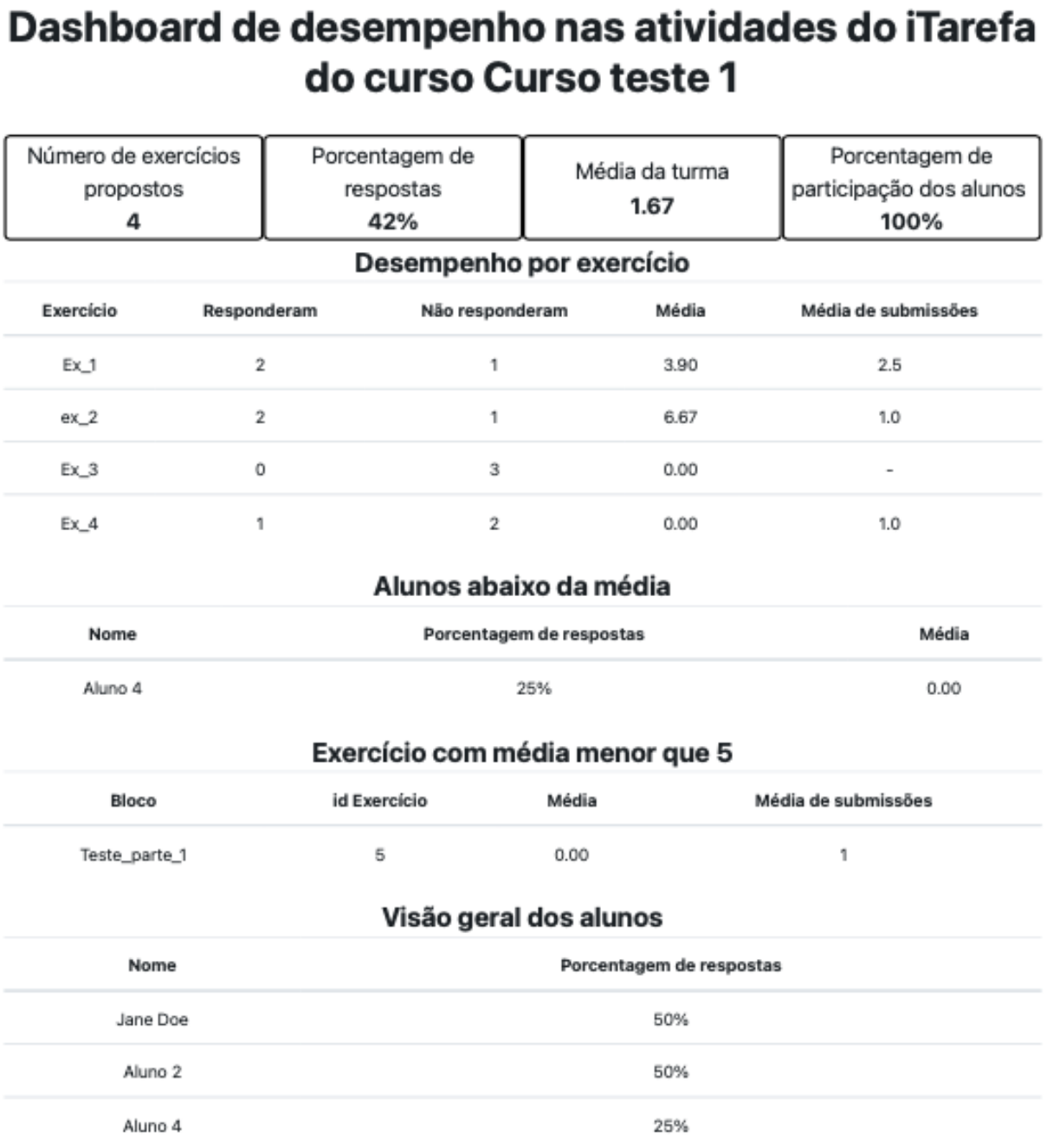
Informação	*Insight* relacionado
Número de exercícios propostos na disciplina	Evolução da disciplina.
Porcentagem de respostas	Participação dos alunos na disciplina.

Média da turma	Desempenho geral da turma.
Porcentagem de participação	Envolvimento geral dos alunos na disciplina.
Para cada exercício, o número de alunos que já submeteram e quantos ainda precisam submeter, a média das submissões e o número médio de vezes que os alunos submeteram respostas nele.	Visibilidade a respeito da participação dos alunos.
Sessão com o nome dos alunos com média abaixo de cinco (média mínima no IME e na POLI para aprovação no curso), porcentagem de respostas em relação ao número de exercícios propostos e sua média nos exercícios.	Visibilidade dos alunos que estão abaixo da média e que precisam de intervenção.
Exercícios que apresentaram as médias de submissão mais baixas, com a média e o número médio de vezes que os alunos submeteram.	Visibilidade de conceitos potenciais para revisão, pois podem ser de dificuldade geral da turma. Exemplo: exercícios em que os alunos precisaram subter diversas vezes e mesmo assim não obtiveram média alta podem mostrar o alto engajamento dos alunos ao tentar realizá-los; uma ação relacionada possível seria revisar o conteúdo relacionado ao exercício em aula.
Para cada aluno, a porcentagem de quantos exercícios realizaram referentes aos exercícios que devem ser entregues (visíveis para os alunos).	Visibilidade a respeito dos alunos que podem estar desmotivados ou com grandes dificuldades no aprendizado, e que necessitam de intervenção.

Fonte: produzida pela autora.

O resultado do Dashboard construído encontra-se na figura a seguir.

Figura 4 - Dashboard desenvolvido para visibilidade do desempenho dos alunos.



Dashboard de desempenho nas atividades do iTarefa do curso Curso teste 1

Número de exercícios propostos	Porcentagem de respostas	Média da turma	Porcentagem de participação dos alunos
4	**42%**	**1.67**	**100%**

Desempenho por exercício

Exercício	Responderam	Não responderam	Média	Média de submissões
Ex_1	2	1	3.90	2.5
ex_2	2	1	6.67	1.0
Ex_3	0	3	0.00	-
Ex_4	1	2	0.00	1.0

Alunos abaixo da média

Nome	Porcentagem de respostas	Média
Aluno 4	25%	0.00

Exercício com média menor que 5

Bloco	id Exercício	Média	Média de submissões
Teste_parte_1	5	0.00	1

Visão geral dos alunos

Nome	Porcentagem de respostas
Jane Doe	50%
Aluno 2	50%
Aluno 4	25%

Fonte: produzido pela autora.

4.3. INSTALAÇÃO DO BLOCO NO MOODLE

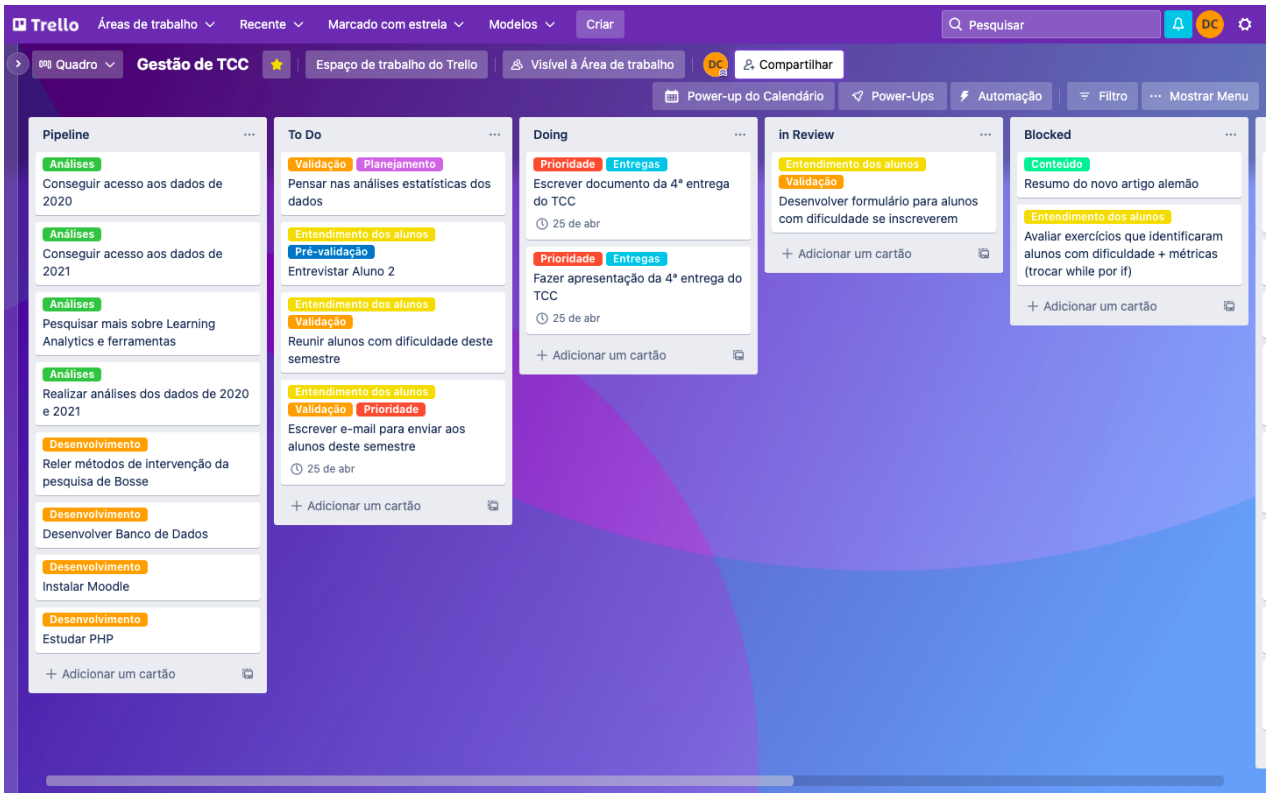
Para a instalação do *Dashboard*, é necessário clonar o código relacionado no caminho *Nome do Moodle/Blocks/*. Vale ressaltar que é importante mudar a visibilidade do bloco após instalação no Moodle, para que ele seja visível apenas para administradores do curso, de modo que não haja exposição de desempenho entre os alunos.

4.4. ROTEIRO DE DESENVOLVIMENTO

Para acompanhar as entregas ao longo do projeto, foi utilizada a ferramenta de gestão *Trello*[3]. Utilizando metodologia ágil, foi realizado o acompanhamento de cada atividade necessária para a que o projeto estivesse em constante evolução: as atividades eram colocadas em faixas que especificavam aquelas que precisam ser desenvolvidas ao longo de todo o projeto e de que já se possuía visibilidade (*Pipeline*), as atividades a serem desenvolvidas ao longo da etapa atual do projeto (*To Do*), as que estavam sendo realizadas (*Doing*), aquelas que estavam em revisão externa dos orientadores (*In Review*), as que estavam bloqueadas pela dependência a outras ações (*Blocked*) e, por fim, as que já haviam sido concluídas (*Done*). Abaixo, está uma figura da organização do Trello do projeto. Nas figuras abaixo está o formato adotado da ferramenta.

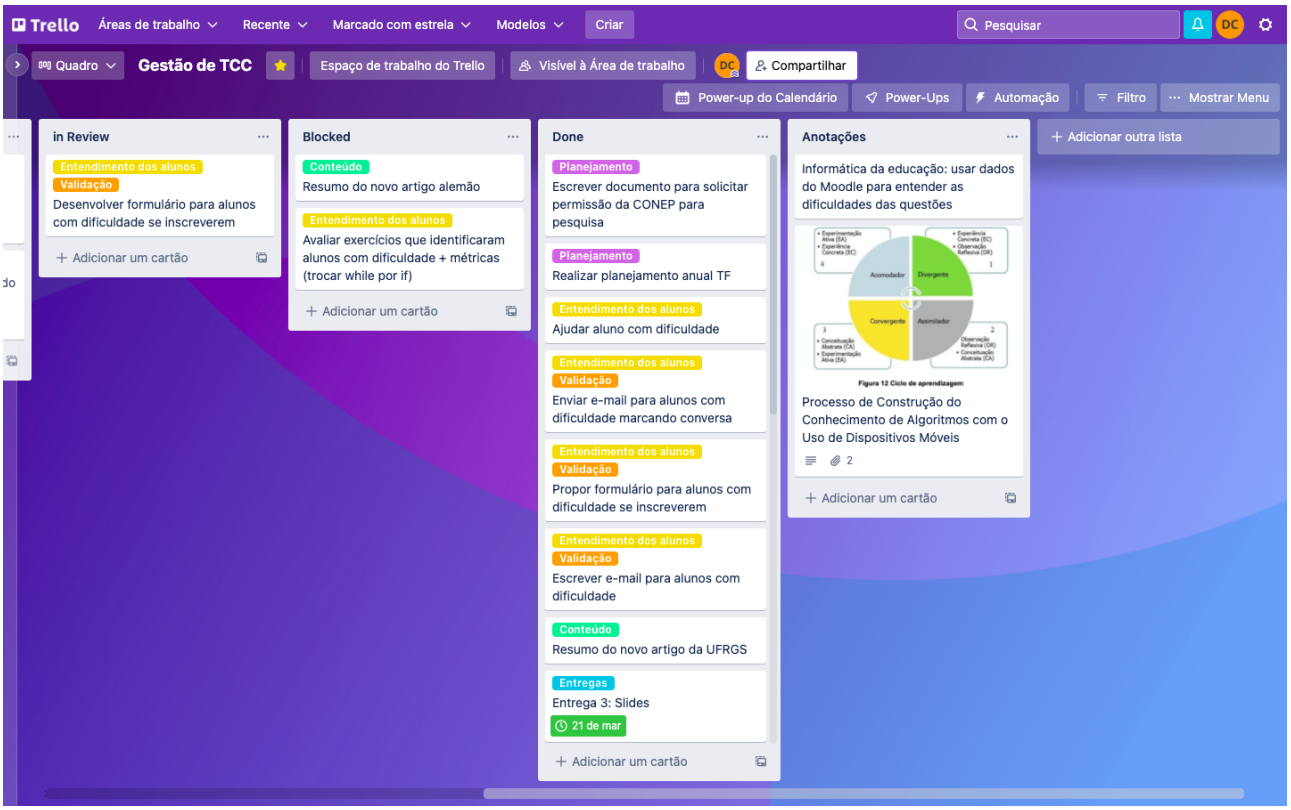
[3] Disponível em: <https://trello.com>.

Figura 5 - Espaço da esquerda do *Trello* do projeto.



Fonte: produzida pela autora.

Figura 6 - Espaço da direita do *Trello* do projeto.



Fonte: produzida pela autora.

4.4.1. ENTREGAS

A fim de explicitar as entregas do projeto, este foi dividido em sete etapas, que foram agrupadas em quatro grandes partes de foco principal do projeto no determinado momento. Ao longo do decorrer dos quadrimestres, o status referente ao andamento das etapas era apresentado aos professores responsáveis pelo projeto. As etapas são:

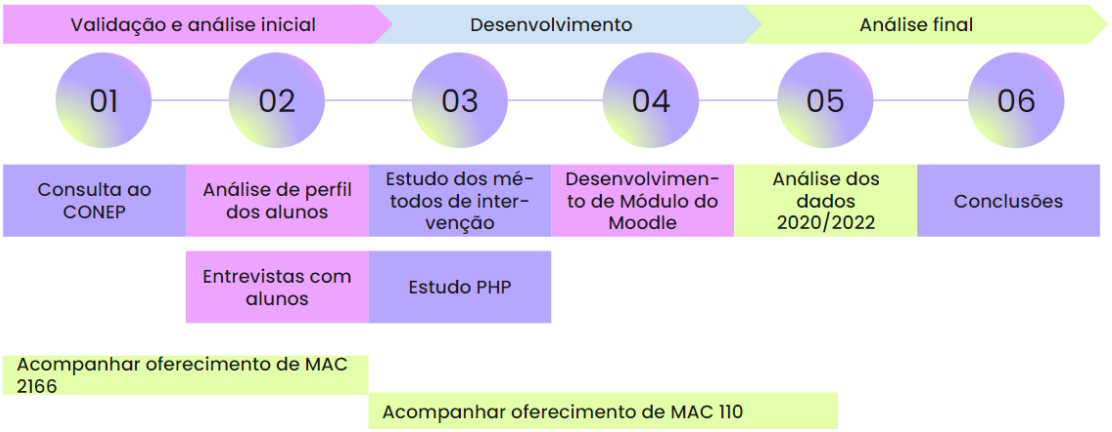
1. Consulta ao CONEP;
2. Análise do perfil dos alunos e entrevistas;
3. Análise dos dados de 2020/2021;
4. Estudos: métodos de intervenção, PHP e Desenvolvimento: banco de dados;
5. Desenvolvimento do Módulo do Moodle em PHP;
6. Análise dos dados de 2022;
7. Conclusões.

Sendo estas etapas agrupadas em quatro grandes partes:

1. Validação;
2. Análise inicial;
3. Desenvolvimento;
4. Análise final.

Na figura a seguir, estão explicitadas todas partes e suas etapas, com as ações principais de cada uma das etapas de desenvolvimento deste projeto, para melhor entendimento.

Figura 7- Divisão das etapas de desenvolvimento do projeto.



Fonte: produzida pela autora.

5. MÉTODOS DE INTERVENÇÃO

Para tornar as aplicações do projeto mais práticas, está sendo acompanhada a turma de MAC110 - Introdução à Programação no Instituto de Matemática e Estatística da Universidade de São Paulo. A intervenção que foi realizada é a de alunos com dificuldade entrarem em contato com a aluna para a realização de uma entrevista e atendimento personalizado nas maiores dificuldades deles. Notou-se que muitos alunos não conseguem entender suas dificuldades, pois estas passam por campos de conceitos mais introdutórios, prejudicando a compreensão de elementos presentes na programação. Um exemplo é a linguagem de programação em bloco ensinada aos alunos no início da disciplina, para que aprendam de forma mais simples antes de passarem para uma linguagem de programação como C ou *Python*: alguns alunos não sabiam a diferença entre os tipos de laço existentes, confundiam a atualização do passo e, por não entender o funcionamento do laço utilizado, não conseguiam chegar à conclusão de qual seria a condição de parada correta para o exercício.

Isso foi notado em dois atendimentos realizados, em que parte da transcrição está presente no apêndice E.

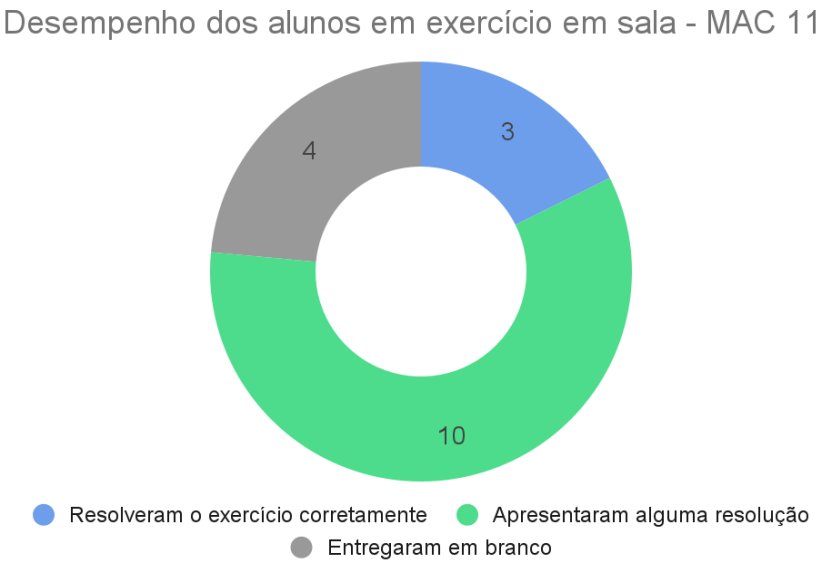
Participando também das aulas, pode-se perceber que existe uma dependência de alguns alunos para a resolução utilizando a ferramenta iVProg, e muitos não conseguiam resolver os problemas escrevendo apenas em uma folha a solução. Este ponto pode ser associado à facilidade de teste quando se está programando no computador, mas que dificulta na depuração da solução desenvolvida, dado que o aluno não analisa de forma mais profunda o código escrito, mas sim a saída que lhe é mostrada ao final de cada execução.

Foram analisados os desempenhos dos alunos na resolução de um exercício em aula e também na primeira prova do semestre, os resultados obtidos pelos alunos no exercício em aula estão disponíveis no apêndice F, enquanto os resultados da prova estão no apêndice G.

Observando enquanto os alunos resolviam o exercício disponibilizado no apêndice F, pode-se notar novamente o fato já citado acima de que existe uma dependência do iVProg para o desenvolvimento da solução, pois muitos não conseguiram escrever o exercício no papel sem antes desenvolvê-lo utilizando a aplicação. Além disso, pode-se perceber também que poucos alunos conseguiram

resolver por completo o exercício no tempo dado (vinte minutos) - três alunos (18%) - número menor que o de alunos que entregaram a resolução em branco - quatro alunos (24%). Este desempenho encontra-se no gráfico 2 a seguir.

Gráfico 2 - Desempenho dos alunos em exercício realizado durante a aula de MAC 110.



Fonte: produzida pela autora.

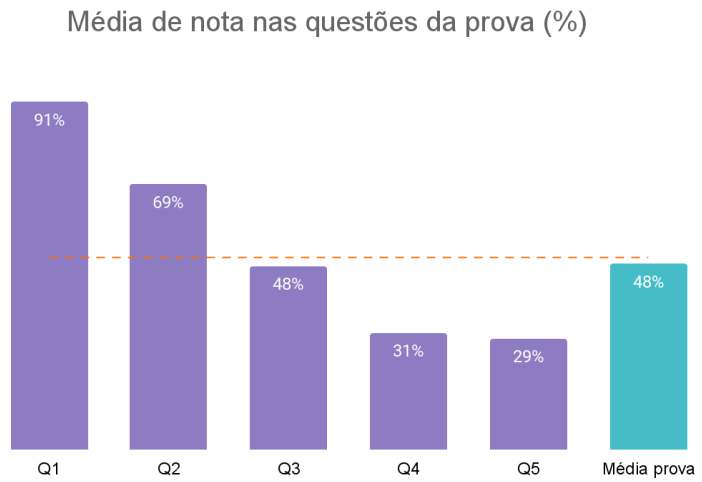
A partir dos atendimentos e os exercícios em sala de aula que foram realizados em sala de aula, pode-se perceber que dentre os conceitos de computação aprendidos, o que despertava mais dúvidas eram os laços de execução, principalmente quando tratava-se da condição de parada para sair desta parte do código e prosseguir no programa.

Analisando o desempenho dos alunos na avaliação, pode-se perceber que os alunos apresentaram maior facilidade na resolução da questão dois, que tratava da realização do teste de um algoritmo colocado na questão. Quando se observa a questão dois, pode-se notar que o desempenho está abaixo do da primeira questão, porém ainda acima da média. A segunda questão explora conhecimentos em relação a condições (if, else), enquanto as demais questões (três, quatro e cinco) exploram também conceitos de laços de repetição e envolvem uma lógica de resolução mais elaborada. Pode-se perceber que, nestas questões com maior complexidade, as médias permaneceram abaixo de 50%.

Analisando o desempenho dos alunos em volume, pode-se perceber a mesma relação: a maioria dos alunos ficou acima da média nas questões um e dois

(que abordaram conceitos mais simples), e também na questão três (porém com diferença de apenas um aluno). Nas questões quatro e cinco, a grande maioria não chegou à média. Isto pode ser visualizado no gráfico 3 abaixo.

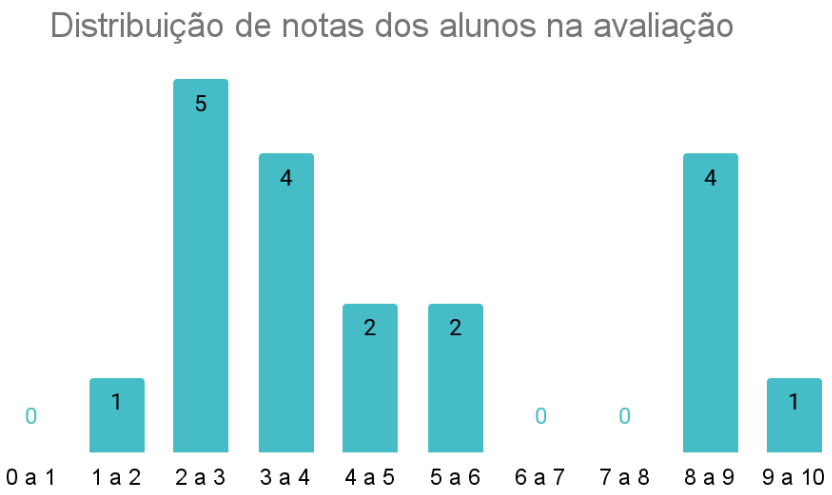
Gráfico 3 - Média do desempenho dos alunos de MAC 110 na avaliação.



Fonte: produzida pela autora.

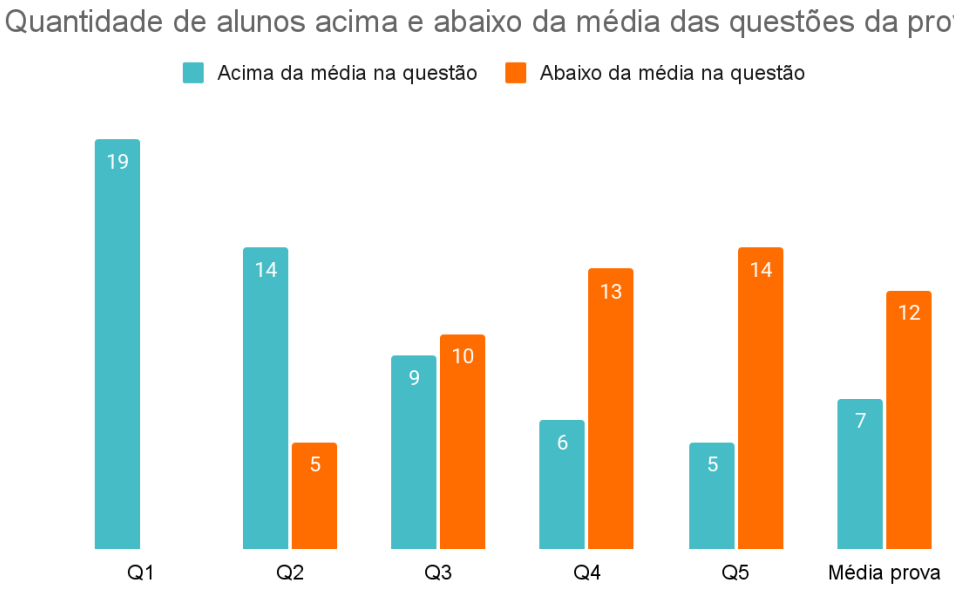
É importante realizar a análise também considerando o número de alunos, pois ao observar apenas a média de notas na avaliação, tem-se a impressão de que a turma apresenta bom desempenho, dado que a média da avaliação está próxima à nota cinco. Porém, observando em quantidade de alunos, percebe-se que na verdade doze dos dezenove alunos (63,2%) que realizaram a avaliação apresentaram desempenho inferior à média cinco, sendo que seis destes (46,2%) não alcançaram nota três. O resultado encontra-se no gráfico 4 a seguir.

Gráfico 4 - Distribuição das notas dos alunos na primeira avaliação de MAC 110.



Fonte: produzida pela autora.

Gráfico 5 - Relação entre alunos acima e abaixo da média nas questões da avaliação.



Fonte: produzida pela autora.

Em consequência dos dados apresentados acima, serão realizadas sessões com os alunos que tiveram média abaixo de cinco, a fim de entender quais foram os principais entraves que dificultaram seu aprendizado, além de reforçar conceitos relacionados a estes entraves, para que o desempenho deles volte a crescer nas avaliações seguintes.

Após a primeira avaliação, também foram realizados encontros com os alunos da disciplina. Um deles foi uma sessão de dúvidas com diversos alunos da

disciplina. O encontro foi comunicado a todos os alunos, porém apenas quatro estiveram presentes.

Para contextualizar, os alunos estavam aprendendo a programar em *Python*, e estavam aprendendo sobre matrizes. Então, de acordo com o programa da disciplina, os conceitos de:

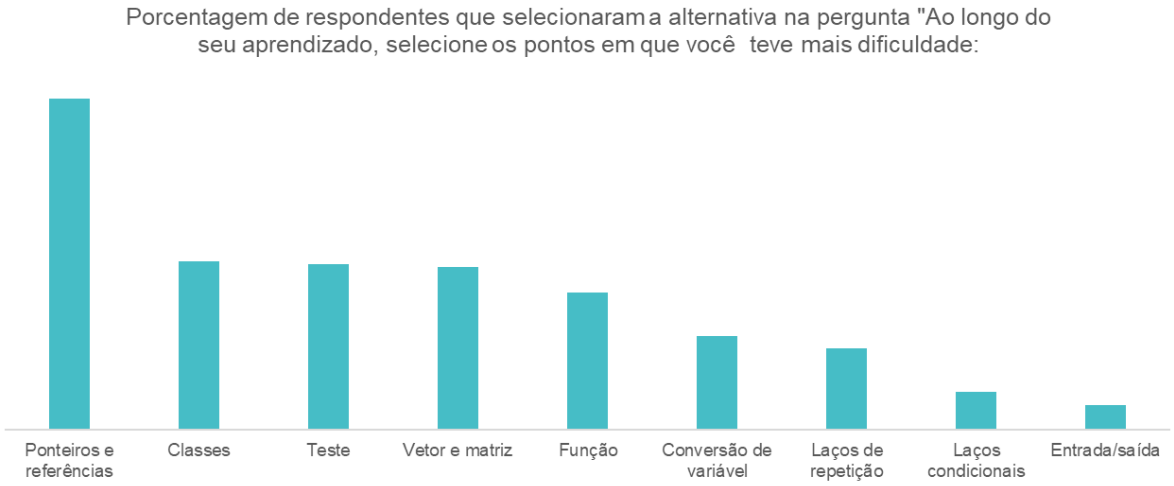
- Breve história da computação;
- Algoritmos: caracterização, notação, estruturas básica;
- Computadores: unidades básicas, instruções, programa armazenado, endereçamento, programas em linguagem de máquina;
- Conceitos de linguagens algorítmicas: expressões; comandos sequenciais, seletivos e repetitivos; entrada/saída;
- Variáveis estruturadas.

Já haviam sido aprendidos. Porém, pode-se perceber que havia dúvidas a respeito de conceitos básicos de como funciona este tipo de estrutura, e também dúvidas de conceitos mais simples, como o funcionamento de laços de repetição (ensinadas ao longo do tópico conceitos de linguagens algorítmicas, comandos repetitivos) presentes na linguagem *Python* (*for* e *while*). As dúvidas estavam principalmente relacionadas a como se realiza a "varredura" da estrutura - por exemplo, para "varrer" a matriz, são necessários dois laços de repetição do tipo *for*.

Além disso, em um atendimento realizado com um aluno surgiu a dúvida de resolução de um exercício, e a aplicação do conceito de funções era necessário para a solução deste. Havia dúvida na diferenciação entre os tipos de função (como a função principal, ou funções com ou sem retorno).

A fim de entender as dificuldades além das turmas que foram acompanhadas para o projeto, foi realizada uma pesquisa via formulário online para entender quais as principais dificuldades no aprendizado introdutório de programação. As perguntas e os resultados completos desta pesquisa estão no apêndice J. As alternativas presentes na questão “Ao longo do seu aprendizado, selecione os pontos em que você teve mais dificuldade” do formulário foram escritas com base no programa da disciplina de MAC2166 - Introdução à Computação, ministrada aos alunos de engenharia da POLI no primeiro semestre da graduação. O gráfico a seguir ilustra em ordem crescente as alternativas com maior número de seleções.

Gráfico 6 - Principais dificuldades dos alunos ao longo do aprendizado introdutório de programação.



Fonte: produzida pela autora.

Como a pesquisa foi realizada com estudantes de diversas universidades e cursos diferentes, algumas das dificuldades encontradas não estão presentes nas interações tidas com os alunos de MAC110, uma vez que os alunos estão aprendendo a linguagem Python, que não apresenta ponteiros e referências e também não apresenta classes em seu aprendizado introdutório. Observando então a pesquisa no contexto da disciplina analisada, pode-se perceber que as principais dificuldades observadas nos alunos são as mesmas com os maiores números de resposta: testes, vetores e matrizes e funções.

5.1. RECOMENDAÇÃO DE INTERVENÇÕES

Dadas as interações realizadas ao longo do projeto com alunos que cursam ou já cursaram introdução a programação, na tabela abaixo encontram-se sugestões de métodos de intervenção, com suas devidas explicações, responsáveis e momentos.

Tabela 4 - Métodos de intervenção sugeridos.

Método	Quando	Responsável	Justificativa
Formulário para conhecer os alunos	Início do curso	Professor	Entender o nível de conhecimento dos alunos e mapear inicialmente

			potenciais alunos que podem necessitar de maior apoio no decorrer da disciplina.
Abertura para ajudar alunos com dúvidas	Ao longo do curso	Professor	Manter o diálogo aberto para que os alunos tenham receptividade quando necessitar de auxílio com suas dúvidas.
Exercícios de acompanhamento	Ao longo do curso	Professor	Desenvolver nos alunos um senso de aprendizado contínuo, gerar insumos para entender seu desempenho.
Checkpoint de conteúdo	Após conclusão do conteúdo	Professor	Revisão de conteúdo e momento para tirar dúvidas.
Plantão de resolução de exercícios	Após *checkpoint* de conteúdo	Monitor	Refazer junto com os alunos os exercícios relativos ao conteúdo que acabou de ser revisado.
Observação do *Dashboard*	Semanalmente/a cada duas semanas	Professor	Entender como está o desempenho e a participação dos alunos na turma.
Lista com alunos que necessitam de atenção	A cada duas semanas	Professor	Conversar com alunos para entender quais estão sendo suas dificuldades, recomendar exercícios e que participem das sessões de dúvidas com o monitor da disciplina.
Sessão individual de dúvidas para alunos com dificuldade	A cada duas semanas	Monitor	Realizar sessão para revisar conteúdo se necessário e ajudar com dúvidas específicas do aluno. Manter uma certa frequência de encontros para observar se o desempenho do aluno está melhorando.
Revisão de conteúdo	Antes de avaliações	Professor	Utilizando o Dashboard, revisar o conteúdo dos

			exercícios em que os alunos estão com média mais baixa.
Feedback de entendimento de dificuldade na prova	Após avaliações	Alunos	Responder ao formulário disponibilizado pelo professor para que este possa entender se a dificuldade da prova era muito maior que a dos exercícios em sala, Caso seja, é importante analisar a possibilidade de incluir exercícios de complexidade parecida nos exercícios de acompanhamento.

Fonte: produzida pela autora.

6. CONCLUSÕES

6.1. CONCLUSÕES SOBRE INTERVENÇÕES

Os tipos de intervenções foram realizados com os alunos:

1. Início do curso: entendimento do contexto dos alunos (questionário disponibilizado no apêndice B);
2. Sessões de dúvidas individuais com alunos com dificuldade (propostos pelos alunos e também pelo professor da disciplina);
3. Sessões de resolução de exercícios em conjunto com os alunos.

A partir das intervenções, pode-se concluir que o contexto da turma, ou seja, quanto os alunos já possuem contato com a programação impactam no desempenho dos alunos. Alunos que já sabiam programar ou já tinham tido contato com algoritmos antes possuíam maior facilidade ao desenvolver o pensamento lógico para resolver os problemas existentes nos exercícios.

Focando nos alunos com dificuldades, nota-se que a transparência da dificuldade com o professor, para que os alunos fossem direcionados para as intervenções, é essencial para que essa identificação seja realizada o mais cedo possível ao longo da disciplina, para que questões de dúvida mais simples sejam resolvidas antes que sejam lecionados conceitos mais complexos (por exemplo, garantir que os alunos entendam como vetores e como se faz para realizar sua declaração, preenchimento com dados e percorrê-los antes de exigir que entendam o conceito de matrizes). Neste ponto, julga-se importante a existência de um programa estruturado de ensino introdutório de programação, e que este programa seja seguindo à risca quanto à ordem em que os conteúdos são passados aos alunos.

Além disso, também é importante a existência de um canal aberto com os alunos, para que estes comuniquem caso haja dúvidas, para que os conceitos sejam revisados e dúvidas sejam sanadas ao longo do processo de aprendizado, não apenas próximo a avaliações, a fim de engajar a participação dos alunos e o aprendizado contínuo.

6.2. CONCLUSÕES SOBRE DIFICULDADES DOS ALUNOS

Com as interações com os alunos e também com os dados coletados na pesquisa, considerando o contexto do conteúdo de introdução à programação, pode-se concluir que as maiores dificuldades estão relacionadas aos conceitos de

testes e depuração, funções, vetores e matrizes. Para mitigar as dificuldades, julga-se necessária a existência de um processo contínuo de aprendizado de programação, sempre dando importância aos assuntos de maior dificuldade, para que estes conceitos não sejam explicados de forma rasa, nem tomados como intuitivos.

Somado a este ponto, pode-se notar pelas interações com os que existe um impacto positivo no processo de aprendizado ao tomar providências utilizando os métodos de intervenção estudados, quando percebe-se que o aluno está apresentando dificuldades, seja pelo desempenho nas atividades ou pela falta de participação deste nas aulas e na realização dos exercícios propostos. O Dashboard desenvolvido torna-se então um método de apoio para os métodos de intervenção, pois, com base no que se é analisado neste, é possível ter visibilidade a respeito do desempenho e do engajamento dos alunos na disciplina, permitindo ao professor clareza para ter conhecimento sobre alunos e quais conteúdos necessitam de maior atenção.

7. REFERÊNCIAS

[1] ELIAS, Tanya. **Learning Analytics: Definitions, Processes and Potential**. *In*: **SEMANTIC SCHOLAR**, 2011. DOI: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.456.7092&rep=rep1&type=pdf.

[2] BARCELOS, Ricardo José dos Santos. **O processo de construção do conhecimento de algoritmos com o uso de dispositivos móveis considerando estilos preferenciais de aprendizagem**. *In*: **REPOSITÓRIO DIGITAL LUME UFRGS**, 2013. DOI: https://lume.ufrgs.br/handle/10183/80524.

[3] DE SOUZA, Lucas Mendonça; FELIX, Igor Moreira; FERREIRA, Bernardo Martins; BRANDÃO, Anarosa Alves Franco; BRANDÃO, Leônidas de Oliveira. **I know what you coded last summer**. *In*: **SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO**, 32. , 2021, Online. Anais do XXXII Simpósio Brasileiro de Informática na Educação. Porto Alegre: Sociedade Brasileira de Computação, 2021. p. 909-920. DOI: https://doi.org/10.5753/sbie.2021.218673.

[4] BOSSE, Yorah; GEROSA, Marco Aurélio. **Reprovações e Trancamentos nas Disciplinas de Introdução à Programação da Universidade de São Paulo**: Um Estudo Preliminar. *In*: **WORKSHOP SOBRE EDUCAÇÃO EM COMPUTAÇÃO**, 2015. DOI: 10.5753/wei.2015.10259.

[5] BOSSE, Yorah; GEROSA, Marco Aurélio. **Why is programming so difficult to learn?**: Patterns of Difficulties Related to Programming Learning Mid-Stage. *In*: **ACM SIGSOFT SOFTWARE ENGINEERING NOTES**, 41, 2016. DOI: 10.1145/3011286.3011301.

[6] ROMERO, Cristobal, VENTURA, Sebastian. **Educational data mining and learning analytics**: An update survey. *In*: **WILEY INTERDISCIPLINARY REVIEWS: DATA MINING AND KNOWLEDGE DISCOVERY**, 2020. DOI: 10.1002/widm.1355.

[7] **PHP: O que é o PHP? - Manual.** PHP: Hypertext Preprocessor, *In*: https://www.php.net/manual/pt_BR/intro-whatis.php, 2022.

[8] **O que é PHP? Guia Básico de Programação PHP**. Hostinger Tutoriais, HOSTINGER, 2019, In: https://www.hostinger.com.br/tutoriais/o-que-e-php-guia-basico.

[9] **PHP FAQ**. MoodleDocs, *In*: https://docs.moodle.org/dev/PHP_FAQ.

[10] **Installing Moodle**. MoodleDocs, *In*: https://docs.moodle.org/400/en/Installing_Mo odle#Requirements.

[11] **Foundation Project**. Welcome to The Apache Software Foundation!, *In*: https://www.apache.org/foundation/.

[12] **Apache for Moodle**. MoodleDocs, *In*: https://docs.moodle.org/400/en/Apache.

[13] **Installing Moodle**. *MoodleDocs*, *In*: https://docs.moodle.org/400/en/Installing_Moodle#Requirements.

[14] **Símbolos e notação de diagramas entidade-relacionamento**. Lucidchart, In: https://www.lucidchart.com/pages/pt/simbolos-de-diagramas-entidade-relacionamento.

[15] **Laboratório de Informática na Educação** (LInE). *In*: http://200.144.254.107/desenvolvimento_moodle/.

[16] **Moodle statistics**. *In*: https://stats.moodle.org/.

[17] **PHP Documentation**. *In*: https://www.php.net/docs.php.

[18] **PHP First Page Tutorial**. *In*: https://www.php.net/manual/pt_BR/tutorial.firstpage.php.

[19] **Apache Documentation**. *In* https://www.apache.org/.

[20] **What is a Moodle Plugin?** *In*: https://moodle.com/faq/what-is-a-moodle-plugin/.

[21] **Plugin Types**. *In*: https://docs.moodle.org/dev/Plugin_types.

[22] **Users Enrolments Table**. *In*: https://moodleschema.zoola.io/tables/user_enrolments.html.

[23] **Moodle Architecture.** 4.0 user documentation. *In*: https://docs.moodle.org/dev/Moodle_architecture.

[24] **Disciplina: MAC2166 - Introdução à Computação.** Instituto de Matemática e Estatística, Ciência da Computação. *In*: **JÚPITER - SISTEMA DE GESTÃO ACADÊMICA DA PRÓ-REITORIA DE GRADUAÇÃO**, https://uspdigital.usp.br/jupiterweb/obterDisciplina?nomdis=&sgldis=MAC2166.

[25] **Disciplina: MAC110 - Introdução à Computação.** Instituto de Matemática e Estatística, Ciência da Computação. *In*: **JÚPITER - SISTEMA DE GESTÃO ACADÊMICA DA PRÓ-REITORIA DE GRADUAÇÃO**, https://uspdigital.usp.br/jupiterweb/obterDisciplina?sgldis=MAC0110&verdis=4.

APÊNDICE A: RESOLUÇÃO 510 DO CONEP DE ABRIL DE 2016

RESOLUÇÃO No 510, DE 07 DE ABRIL DE 2016

O Plenário do Conselho Nacional de Saúde em sua Quinquagésima Nona Reunião Extraordinária, realizada nos dias 06 e 07 de abril de 2016, no uso de suas competências regimentais e atribuições conferidas pela Lei no 8.080, de 19 de setembro de 1990, pela Lei no 8.142, de 28 de dezembro de 1990, pelo Decreto no 5.839, de 11 de julho de 2006, e

Considerando que a ética é uma construção humana, portanto histórica, social e cultural;

Considerando que a ética em pesquisa implica o respeito pela dignidade humana e a proteção devida aos participantes das pesquisas científicas envolvendo seres humanos; Considerando que o agir ético do pesquisador demanda ação consciente e livre do participante;

Considerando que a pesquisa em ciências humanas e sociais exige respeito e garantia do pleno exercício dos direitos dos participantes, devendo ser concebida, avaliada e realizada de modo a prever e evitar possíveis danos aos participantes;

Considerando que as Ciências Humanas e Sociais têm especificidades nas suas concepções e práticas de pesquisa, na medida em que nelas prevalece uma acepção pluralista de ciência da qual decorre a adoção de múltiplas perspectivas teórico- metodológicas, bem como lidam com atribuições de significado, práticas e representações, sem intervenção direta no corpo humano, com natureza e grau de risco específico;

Considerando que a relação pesquisador-participante se constrói continuamente no processo da pesquisa, podendo ser redefinida a qualquer momento no diálogo entre subjetividades, implicando reflexividade e construção de relações não hierárquicas;

Considerando os documentos que constituem os pilares do reconhecimento e da afirmação da dignidade, da liberdade e da autonomia do ser humano, como a Declaração Universal dos Direitos Humanos, de 1948 e a Declaração Interamericana de Direitos e Deveres Humanos, de 1948;

Considerando a existência do sistema dos Comitês de Ética em Pesquisa e da Comissão Nacional de Ética em Pesquisa;

Considerando que a Resolução 466/12, no artigo XIII.3, reconhece as especificidades éticas das pesquisas nas Ciências Humanas e Sociais e de outras que se utilizam de metodologias próprias dessas áreas, dadas suas particularidades;

Considerando que a produção científica deve implicar benefícios atuais ou potenciais para o ser humano, para a comunidade na qual está inserido e para a sociedade, possibilitando a promoção de qualidade digna de vida a partir do respeito aos direitos civis, sociais, culturais e a um meio ambiente ecologicamente equilibrado; e

Considerando a importância de se construir um marco normativo claro, preciso e plenamente compreensível por todos os envolvidos nas atividades de pesquisa em Ciências Humanas e Sociais, resolve:

Art. 1o Esta Resolução dispõe sobre as normas aplicáveis a pesquisas em Ciências Humanas e Sociais cujos procedimentos metodológicos envolvam a utilização de dados diretamente obtidos com os participantes ou de informações identificáveis ou que possam acarretar riscos maiores do que os existentes na vida cotidiana, na forma definida nesta Resolução.

Parágrafo único. Não serão registradas nem avaliadas pelo sistema CEP/CONEP:

I – pesquisa de opinião pública com participantes não identificados;

II – pesquisa que utilize informações de acesso público, nos termos da Lei no 12.527, de 18 de novembro de 2011;

III – pesquisa que utilize informações de domínio público;

IV - pesquisa censitária;

V - pesquisa com bancos de dados, cujas informações são agregadas, sem possibilidade de identificação individual; e

VI - pesquisa realizada exclusivamente com textos científicos para revisão da literatura científica;

VII - pesquisa que objetiva o aprofundamento teórico de situações que emergem espontânea e contingencialmente na prática profissional, desde que não revelem dados que possam identificar o sujeito; e

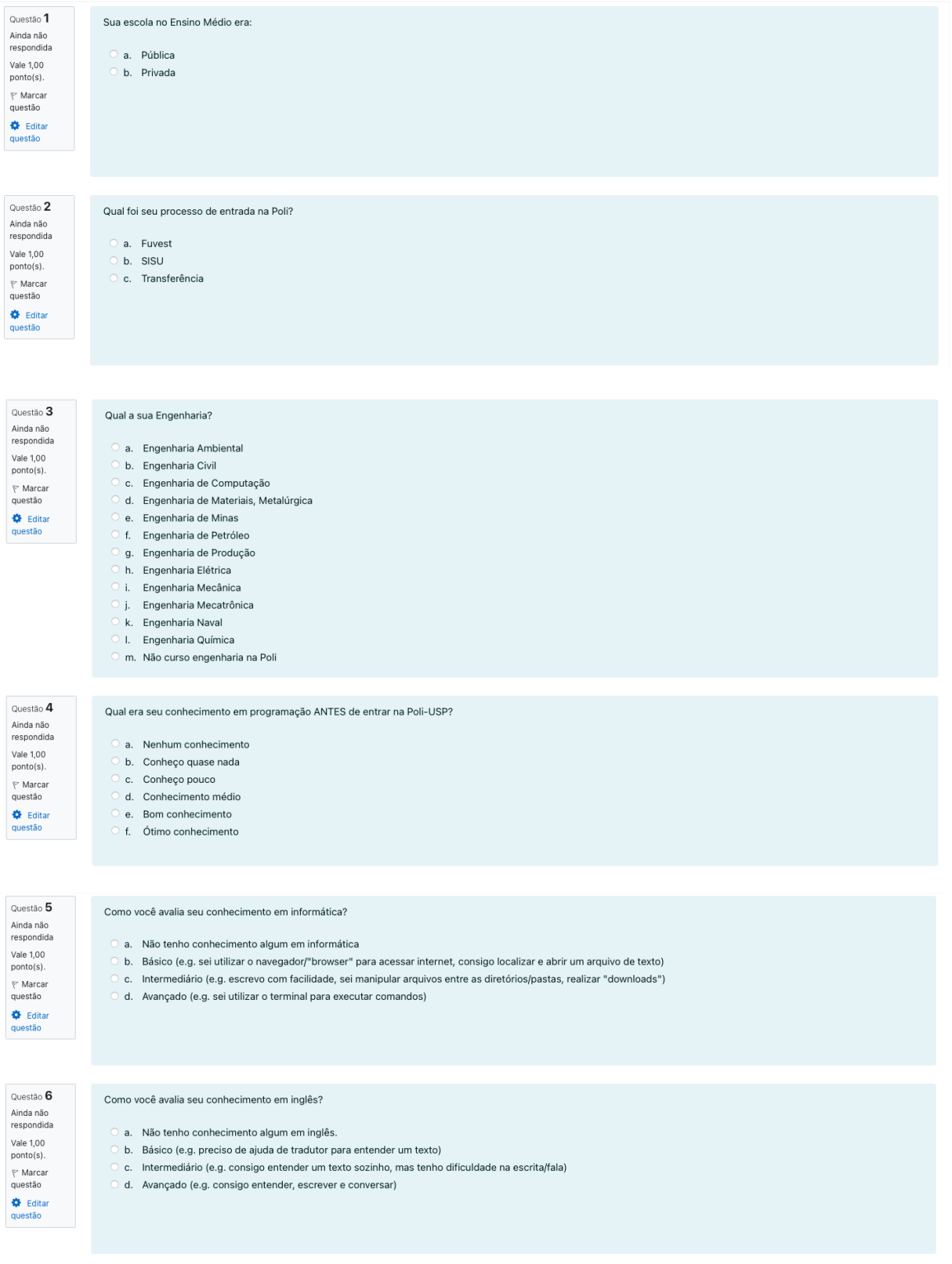
VIII – atividade realizada com o intuito exclusivamente de educação, ensino ou treinamento sem finalidade de pesquisa científica, de alunos de graduação, de curso técnico, ou de profissionais em especialização.

§ 1o Não se enquadram no inciso antecedente os Trabalhos de Conclusão de Curso, monografias e similares, devendo-se, nestes casos, apresentar o protocolo de pesquisa ao sistema CEP/CONEP;

§ 2o Caso, durante o planejamento ou a execução da atividade de educação, ensino ou treinamento surja a intenção de incorporação dos resultados dessas atividades em um projeto de pesquisa, dever-se-á, de forma obrigatória, apresentar o protocolo de pesquisa ao sistema CEP/CONEP.

APÊNDICE B: QUESTIONÁRIO "QUEREMOS CONHECER VOCÊ"

Figura 8 - Formulário “Queremos conhecer você”.



Questão 1
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Sua escola no Ensino Médio era:

a. Pública
b. Privada

Questão 2
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Qual foi seu processo de entrada na Poli?

a. Fuvest
b. SISU
c. Transferência

Questão 3
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Qual a sua Engenharia?

a. Engenharia Ambiental
b. Engenharia Civil
c. Engenharia de Computação
d. Engenharia de Materiais, Metalúrgica
e. Engenharia de Minas
f. Engenharia de Petróleo
g. Engenharia de Produção
h. Engenharia Elétrica
i. Engenharia Mecânica
j. Engenharia Mecatrônica
k. Engenharia Naval
l. Engenharia Química
m. Não curso engenharia na Poli

Questão 4
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Qual era seu conhecimento em programação ANTES de entrar na Poli-USP?

a. Nenhum conhecimento
b. Conheço quase nada
c. Conheço pouco
d. Conhecimento médio
e. Bom conhecimento
f. Ótimo conhecimento

Questão 5
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Como você avalia seu conhecimento em informática?

a. Não tenho conhecimento algum em informática
b. Básico (e.g. sei utilizar o navegador/"browser" para acessar internet, consigo localizar e abrir um arquivo de texto)
c. Intermediário (e.g. escrevo com facilidade, sei manipular arquivos entre as diretórios/pastas, realizar "downloads")
d. Avançado (e.g. sei utilizar o terminal para executar comandos)

Questão 6
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Como você avalia seu conhecimento em inglês?

a. Não tenho conhecimento algum em inglês.
b. Básico (e.g. preciso de ajuda de tradutor para entender um texto)
c. Intermediário (e.g. consigo entender um texto sozinho, mas tenho dificuldade na escrita/fala)
d. Avançado (e.g. consigo entender, escrever e conversar)

Questão **7**
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Quanto você considera saber (hoje) de programação?

a. Nenhum conhecimento
b. Conheço quase nada
c. Conheço pouco
d. Conhecimento médio
e. Bom conhecimento
f. Ótimo conhecimento

Questão **8**
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Você já cursou uma disciplina de programação na USP? (ou seja, se matriculou e assistiu ao menos algumas aulas)

a. Sim
b. Não

Questão **9**
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Código da disciplina

a. MAC2166
b. MAC110
c. MAC115
d. Outra

Questão **10**
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Considerando a última vez que você cursou a disciplina do item anterior, qual foi a frequência de aulas que você realmente assistiu?

a. 0 a 25%
b. 26% a 50%
c. 51% a 75%
d. 76% a 100%

Questão **11**
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Qual foi a sua nota final nas provas?

a. 0.0 <= x < 3.0
b. 3.0 <= x < 5.0
c. 5.0 <= x < 7.5
d. 7.5 <= x <= 10.0
e. No oferecimento que cursei não houve aplicação de prova.

Questão **12**
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Qual foi a sua nota final nos EP (Exercícios-Programa)?

a. 0.0 <= x < 3.0
b. 3.0 <= x < 5.0
c. 5.0 <= x < 7.5
d. 7.5 <= x <= 10.0
e. No oferecimento que cursei não houve aplicação de EP.

Questão 13
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Qual foi a sua nota final nas atividades de programação (caso tenham acontecido na disciplina cursada)?

a. 0.0 <= x < 3.0
b. 3.0 <= x < 5.0
c. 5.0 <= x < 7.5
d. 7.5 <= x <= 10.0
e. Não consigo recuperar esta informação.
f. No oferecimento que cursei não houve aplicação de atividades.

Questão 14
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Por que escolheu fazer o MAC2166 remoto?

Questão 15
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Assinale abaixo as disciplinas que você já cursou:

a. Cálculo 1
b. Cálculo 2
c. Cálculo 3
d. Cálculo 4
e. Álgebra Linear 1
f. Álgebra Linear 2

Questão 16
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Assinale abaixo as disciplinas nas quais você já foi aprovado:

a. Cálculo 1
b. Cálculo 2
c. Cálculo 3
d. Cálculo 4
e. Álgebra Linear 1
f. Álgebra Linear 2

Questão 17
Ainda não respondida
Vale 1,00 ponto(s).
Marcar questão
Editar questão

Considerando a outra vez em que cursou MAC2166 (ou disciplina equivalente), classifique as suas dificuldades em cada um dos itens.
Para cada item as opções são: Nenhuma dificuldade; Pouca dificuldade; Dificuldade razoável; Muita dificuldade; Dificuldade extrema.
Cuidado ao selecionar a opções, pois elas podem aparecer em outra ordem.

Entender o conteúdo da matéria	Escolher...
Entender as aulas que assisti	Escolher...
Memorizar os conteúdos	Escolher...
Ter uma linha de raciocínio lógica para construir um programa	Escolher...
Linguagem de programação (e.g. problemas com a sintaxe ou com o compilador/interpretador)	Escolher...
Estruturas de programação (e.g. de estruturas "if", "else", "while", "for", ...)	Escolher...
Variáveis (e.g. int, float, double, diferenciar as variáveis, fazer conversão de tipos, ...)	Escolher...
Estruturas homogêneas (e.g. matrizes, vetores)	Escolher...
Conseguir resolver os problemas (encontrar solução algoritmica para o problema dado)	Escolher...
Encontrar erros no código sem o compilador (e.g. quando realizo um exercício no papel, consigo identificar meus erros com facilidade)	Escolher...

Finalizar tentativa ...

Fonte: produzido pela autora.

APÊNDICE C: ROTEIRO DE ENTREVISTAS PARA VALIDAÇÃO E RESPOSTAS OBTIDAS

ROTEIRO DE PERGUNTAS:

A princípio, a aluna que realizou a entrevista se apresenta e explica o tema da monografia, explicando que ela tem como o intuito identificar padrões nas dificuldades dos alunos no ensino introdutório de programação e desenvolver formas de mitigá-las. Então, passa-se à sessão de perguntas, sendo elas listadas a seguir:

- Como foi cursar o reoferecimento da disciplina de MAC 2166 para você?
- Qual tática foi utilizada para recuperar seu desempenho na disciplina e ser aprovado?
- Na sua opinião, as aulas das quais você participou te ajudaram a aprender o conteúdo?
- Foram realizadas sessões de reforço e você esteve presente nelas. Este reforço te ajudou nas aulas seguintes? Este reforço te ajudou a entender a matéria que já havia sido ensinada?
- O que você acredita que deve ser feito no oferecimento de matérias introdutórias à programação para melhorar o aprendizado de alunos como você?
- Como você já cursou a disciplina antes, quais foram os seus impeditivos para aprender o conteúdo e ser aprovado nesta outra vez?

RESPOSTAS:

ALUNO 1

Este aluno já havia cursado a disciplina diversas vezes antes, e em sua opinião as atividades recorrentes ajudavam a criar disciplina para que ele estudasse com frequência a matéria. Além disso, o acesso a vídeos de reforço ajudavam na hora de relembrar os conceitos nos momentos de estudo.

O aluno também citou que o fato de a plataforma Moodle estar bem organizada na disciplina, com um guia de estudo por semana de aula e do assunto que seria tratado na semana proporcionaram um ambiente mais intuitivo e simples para o aprendizado.

A respeito dos impeditivos para aprender o conteúdo das outras vezes em que cursou a disciplina, o aluno acredita que o maior impeditivo havia sido não estudar com recorrência o que era necessário, pois apenas com três provas e dois Exercícios Programas, não havia obrigatoriedade de recorrência nos estudos, e que quando havia necessidade de aplicação do que foi ensinado, muitos conceitos já haviam sido esquecidos por falta de aplicação recorrente. Em relação a conceitos, os que o aluno teve maior dificuldade em entender foram a sintaxe e também conseguir criar uma linha de raciocínio para a resolução do problema.

Por fim, o aluno acredita que o formato da disciplina com atividades recorrentes com certeza proporcionou a ele um ambiente favorável para o aprendizado, pois este acredita que é importante exercitar os conceitos de programação para a fixação do conteúdo.

APÊNDICE D: TRANSCRIÇÃO DAS ENTREVISTAS COM ALUNOS DE MAC110 - INTRODUÇÃO À PROGRAMAÇÃO NO INSTITUTO DE MATEMÁTICA E ESTATÍSTICA DA UNIVERSIDADE DE SÃO PAULO

ALUNO 1

ROTEIRO DE PERGUNTAS:

A princípio, a aluna que realizou a entrevista se apresenta e explica o tema da monografia, explicando que ela tem como o intuito identificar padrões nas dificuldades dos alunos no ensino introdutório de programação e desenvolver formas de mitigá-las. Então, passa-se à sessão de perguntas, sendo elas listadas a seguir:

- Você já possuía conhecimento em programação antes de realizar a disciplina?
- Como está sendo cursar a disciplina de MAC 110 para você?
- Qual tática está sendo utilizada para recuperar seu desempenho na disciplina e ser aprovado?
- Na sua opinião, as aulas das quais você participou te ajudam a aprender o conteúdo?
- O que você acredita que deve ser feito no oferecimento de matérias introdutórias à programação para melhorar o aprendizado de alunos como você?
- Como você já cursou a disciplina antes, quais foram os seus impeditivos para aprender o conteúdo e ser aprovado nesta outra vez?
- Quais são as suas dúvidas no atual estágio da disciplina?

RESPOSTAS:

ALUNO 1

Aluno já havia cursado uma disciplina de programação no Instituto de Psicologia da Universidade de São Paulo em 2020, em que era utilizada a linguagem R. Foi aprovado na disciplina e acredita que os principais meios que o ajudaram neste processo de aprendizado estavam no material de apoio presente, além do compartilhamento de resoluções entre os alunos nos fóruns de discussão da disciplina. Nesta disciplina, o aluno relatou que a resolução dos exercícios era feita completamente no computador, até mesmo nas avaliações, sem a utilização de

papel para pensar na solução, e que ficavam disponíveis as resoluções dos exercícios de oferecimentos anteriores da disciplina.

O aluno também cursou Introdução à Programação no Instituto de Matemática e estatística em 2021, porém estava com problemas de saúde que impossibilitaram seu aprendizado e acompanhamento do curso, e então não pode continuar na disciplina, e por isso está cursando MAC 110.

Sobre o oferecimento que está cursando no momento, o aluno tem a percepção de que os conceitos são passados de maneira muito rápida, e muitas vezes por isso não existem muitos momentos de resolução de exercícios e compartilhamento de respostas para posterior estudo. Ele porém ressaltou que a rapidez de resposta do professor no fórum da disciplina ajuda no aprendizado, além da monitoria (sessão de atendimento que estava sendo realizada) auxiliam no entendimento de forma mais especializada. Disse que a melhor forma de auxílio na disciplina tem sido a ajuda de outros alunos na sala nos momentos em que tenta resolver exercícios.

O aluno, então, pediu ajuda para resolver um exercício dos que foram disponibilizados pelo professor que leciona a disciplina:

Enunciado:

Atenção: Nesta atividade é necessário contruir um programa no iVProg que tenha um comando de leitura dentro de um laço de repetição que deve ser do tipo repita_enquanto (ou seja, usar um comando repita_enquanto a partir do botão com fundo laranja </>).

Contexto. A ideia deste programa é implementar um programa do tipo "*encontre uma soma que caiba*", aqui o objetivo é encontrar a maior soma de naturais menores ou iguais a um determinado natural dado.

Construa um programa que lê um natural (digamos na variável *total*) e compute a maior soma dentre os primeiros números naturais (consecutivos), que seja menor ou igual ao valor digitado (<=*total*).

Exemplos: Veja alguns exemplos de entradas e o valor a ser impresso por seu código.

1. Se o usuário digitar 1 (*total*=1), então seu programa deverá imprimir 1, pois 1=0 + 1 e ao tentar somar o natural seguinte extrapola o limite 1 (0 + 1 + 2 > 1).
2. Se o usuário digitar 91 (*total*=91), então seu programa deverá imprimir 91, pois 91=0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 e ao tentar somar o natural seguinte (14) extrapolaria o limite 91 (pois 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 = 105 > 91).
3. Se o usuário digitar 104 (*total*=104), então seu programa deverá imprimir 91, pois 91=0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 e ao tentar somar o natural seguinte (14) extrapolaria o limite 104 (pois 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 = 105 > 104).
4. Se o usuário digitar 105 (*total*=105), então seu programa deverá imprimir 105, pois 105=0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 e ao tentar somar o natural seguinte (15) extrapolaria o limite 105 (pois 0 + 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 + 13 + 14 + 15 = 120 > 105).

Atenção: Cuidado com o último natural a ser somado, no exemplo do *104*, se somar o *14* o resultado seria *105 > 104*! logo errado.

Entrada esperada: um natural *total*.

Saída esperada: um natural correspondente ao somatório dos primeiros naturais (que seja menor que *total*).

Atenção: não imprima textos informativos ao usuário, pois estes atrapalhariam o avaliador automático. Apenas imprima os textos, variáveis ou expressões aritméticas requeridas.

RESOLUÇÃO CONJUNTA:

Passo 1: inicialização de variáveis:

Aluno iniciou a resolução inicializando as variáveis, sendo elas:

- N: o número que será digitado pelo usuário
- Soma: variável para guardar a soma dos números
- Naturais: o valor que será somado

Passo 2: adição do laço para resolução do problema:

O aluno adicionou o laço "repita enquanto" e chegou ao primeiro entrave para a resolução: qual seria a condição de parada?

O aluno pensou inicialmente que a condição deveria ser que os naturais somados fossem menores que o N dado.

Passo 3: condição de parada do laço:

Como a condição de parada não estava correta, foi sugerido pela aluna que estava realizando o atendimento que fosse pensado primeiro na lógica para a resolução do problema (que deveria ser executada a cada laço) ao invés de pensar no "final", que seria a condição de parada.

Passo 4: desenvolvimento do código de solução do exercício (interno ao laço) sem condição de parada verificada:

Então, iniciou-se o desenvolvimento do código dentro do laço, sendo este: adicionar um aos naturais (*naturais = naturais + 1*) para sempre pegar o próximo natural. Também foi adicionada uma linha em que a soma recebia a soma anterior e os naturais (*soma = soma + naturais*).

Passo 5: condição de parada do laço com código de solução do exercício (interno ao laço) resolvido:

Com isso, o aluno pensou em uma nova condição de parada para sair do loop, sendo esta que os naturais fossem menores ou iguais a N. Neste momento, o aluno teve a ideia também de apagar a variável soma, porém a aluna que realizava o atendimento sugeriu que fosse realizado um teste antes de realizar uma alteração tão grande no método de solução, a fim de que, caso fossem necessárias mais alterações, que estas fossem realizadas baseadas no problema concreto que estaria acontecendo no código. Foi perguntado ao aluno se ele tinha conhecimento do que era um teste de mesa, e o aluno respondeu que não conhecia, então a entrevistadora resolveu ensinar como se realiza o teste do algoritmo utilizando uma planilha compartilhada online com o aluno.

Passo 6: explicação do teste de mesa e aplicação prática no contexto do exercício:

Explicação dada: a cada linha diferente do código, serão anotadas as atualizações de valores de todas as variáveis de programa, bem como as saídas (escritas na tela). Abaixo, encontra-se a tabela final do teste de mesa que levou à resolução correta do exercício.

Tabela 5- Teste de mesa realizado para a resolução do exercício.

Linha do código	Variáveis		
	n	soma	naturais
inicialização	0	0	0
leia n	14	0	0
repita	14	0	0
naturais ++	14	0	1
soma = soma + naturais	14	1	1
repita	14	1	1
naturais ++			2
soma = soma + naturais		3	
repita			
naturais ++			3
soma = soma + naturais		6	
repita			
naturais ++			4
soma = soma + naturais		10	
repita			
naturais ++			5
soma = soma + naturais		15	
escreva			

Fonte: produzido pela autora e aluno em atendimento.

Legenda:

__: última execução dentro do laço de execução

Com a utilização do teste de mesa, o aluno pode perceber a importância da utilização de duas variáveis separadas para a soma e para os naturais que deveriam ser somados (estaria pulando diversos naturais na soma), além de adaptar também

a condição de parada que havia sido pensada inicialmente para que o número de execuções não ultrapassasse o necessário.

ALUNO 2:

O aluno, então, pediu ajuda para resolver um exercício dos que foram disponibilizados pelo professor que leciona a disciplina:

Enunciado:

Atenção: Nesta atividade é necessário contruir um programa no iVProg que tenha um comando de leitura dentro de um laço de repetição que deve ser do tipo repita_enquanto (ou seja, usar um comando repita_enquanto a partir do botão com fundo laranja </>).

Contexto. A ideia deste programa é implementar um programa do tipo "*encontre uma soma que caiba*", aqui o objetivo é encontrar a soma dos pares menores ou iguais a um determinado natural dado.

Construa um programa que solicita que o usuário digite um inteiro positivo N, então implemente um laço (obrigatório) que compute a soma dos N primeiro pares, imprimindo a soma final. Considere o inteiro 0 como o primeiro número par.

Mas atenção esse não é um exercício para estimular habilidades de encontrar fórmulas fechadas, mas sim de exercício de construção de laços simples, logo não use propriedade matemática, use algoritmo para computar as somas de modo literal (somando cada um dos valores pares, começando com o valor 0).

Exemplos. Considere as seguintes entradas e respectivas saídas.

Tabela 6 - entradas e saídas esperadas na resolução correta do exercício proposto.

Entradas	Saídas
1	0
2	2
3	6
4	12

5	20
6	30

Fonte: Moodle da disciplina de MAC 110 em 2022.

RESOLUÇÃO CONJUNTA:

O aluno já havia tentado resolver o exercício, porém a reolução não estava correta, então o método do atendimento realizado foi em uma linha diferente do realizado com o aluno 1.

Passo 1: inicialização de variáveis:

Aluno iniciou a resolução inicializando as variáveis, sendo elas:

- N: o número que será digitado pelo usuário
- Soma: variável para guardar a soma dos números
- Pares: o valor que será somado

Passo 2: adição do laço para resolução do problema:

O aluno adicionou o laço "repita enquanto" e chegou ao primeiro entrave para a resolução: qual seria a condição de parada?

O aluno pensou inicialmente que a condição deveria ser que os pares somados fossem menores que o N dado.

Passo 3: condição de parada do laço:

Como a condição de parada não estava correta, foi sugerido pela aluna que estava realizando o atendimento que fosse pensado primeiro na lógica para a resolução do problema (que deveria ser executada a cada laço) ao invés de pensar no "final", que seria a condição de parada, assim como no atendimento com o aluno 1.

Passo 4: desenvolvimento do código de solução do exercício (interno ao laço) sem condição de parada verificada:

O aluno adicionou a linha para realizar a soma dos pares (soma = soma + pares), e atualizou os pares (*pares = pares + 1*). Com isso, o programa estava somando números ímpares também, além de que a condição de parada estava

fazendo com que o *loop* se finalizasse quando os pares alcançavam valor maior que o N dado.

Um problema observado foi o de que o aluno não tinha conhecimento de como funcionava a atualização de passo pré definida no laço, que sempre é executada ao final de cada execução do laço, antes de verificar a condição de parada para que seja tomada a decisão de entrar ou não novamente no código. Então, foi explicado como funcionava a atualização do passo, e com isso o aluno conseguiu entender que a atualização deveria ser realizada com um passo de dois, para que fossem somados apenas números pares. (*pares = pares + 2*).

Passo 5: explicação do teste de mesa e aplicação prática no contexto do exercício:

O aluno disse que o aluno 1 havia explicado um pouco de como funcionava o teste de mesa, e então, da mesma forma, foi construída uma planilha compartilhada para que fosse realizado o teste do exercício até o presente momento de resolução.

Tabela 7 - Teste de mesa realizado para a resolução do exercício.

Linha do código	Variáveis			
	soma	contador	n	pares
ler n	0	0	5	0
repita_para				0
soma = pares + soma	0			
contador + 1		1		
atualização dos pares				2
repita_para				
soma = pares + soma	2			
contador + 1		2		
atualização dos pares				4
repita_para				
soma = pares + soma	6			
contador + 1		3		
atualização dos pares				6
repita_para				

soma = pares + soma	12			
contador + 1		4		
atualização dos pares				8
repita_para				
soma = pares + soma	20			
contador + 1		5		
atualização dos pares				10
repita_para				
soma = pares + soma	30			
contador + 1		6		
atualização dos pares				12
repita_para				

Fonte: produzido pela autora e aluno em atendimento.

Legenda:

__: última execução dentro do laço de execução

Sugeriu-se ao longo do teste mesa adicionar uma nova variável que contasse o número de pares que estavam sendo somados:

- Contador: número de pares já somados

Esta variável foi, então, utilizada para construir a condição de parada que contava o número de pares (*repita_para: contador* <= *N*).

Uma sugestão final foi feita ao aluno, para que ele não se sentisse confuso com a resolução, de nomear as variáveis conforme seu uso ao longo do código (como a soma dos pares foi chama da de soma, os pares que estavam sendo somados, de pares, o número de pares de contador). Sugeriu-se também que, quando o programa em que se está realizando a solução (e também no papel), colocasse comentários para que conseguisse compreender melhor o que as linhas do código significavam (fossem as atribuições de variáveis, ou até mesmo as condições de parada).

APÊNDICE E: ANOTAÇÕES SOBRE O DESEMPENHO DOS ALUNOS DE MAC110 NA RESOLUÇÃO DE UM EXERCÍCIO EM SALA DE AULA

Para a resolução do exercício, o professor apresentou seguinte enunciado na lousa aos alunos:

Dados N (um número natural) e uma sequência de inteiros com N elementos, determinar o número de subsequências de números iguais

Exemplos:

{1} → 1

{1, 2} → 2

{-1, 1, 1, 3} → 3

Foram dados vinte minutos para que os alunos entregassem uma solução em uma folha de papel (da mesma forma como deve ser feito em avaliações).

Os resultados dos alunos encontram-se a seguir.

ALUNO 1

Entregou exercício após já ter sido resolvido na lousa.
Código sem comentário.
Falta: explicar o que é cada variável.

ALUNO 2

Leitura no local errado do código, não entra no loop na primeira vez.
Condição de parada não está correta, programa entraria em loop infinito caso entrasse uma vez.
Falta: somar no contador e no N.

ALUNO 3

Não desenvolveu o loop necessário para a resolução do problema.
Falta: somar no contador e no N.

ALUNO 4

Loop correto.

Falta: inicializar variáveis, leitura do N precisa ser primeiro, contador não acresce quando muda de sequência, não somou no passo.

ALUNO 5

Duas leituras dentro do loop.

Falta: guardar anterior em variável.

ALUNO 6

Entregou exercício em branco.

ALUNO 7

Falta: generalizar exemplo utilizado para resolução.

Loop com range errado.

Não terminou condição de parada.

ALUNO 8

Leitura no local errado do programa e loop com range errado.

Atualizou o contador somando zero e não somou no passo a cada loop.

ALUNO 9

Resolveu o exercício em Python, porém utilizou vetor, método que o professor pediu para que não fosse utilizado.

ALUNO 10

Entregou exercício em branco.

ALUNO 11

Loop com range errado e resolução confusa.

ALUNO 12

Resolução correta, porém sem comentários.

ALUNO 13

Entregou exercício em branco, com anotação de que não está entendendo muito da matéria.

ALUNO 14

Entregou exercício em branco.

ALUNO 15

Fez a resolução no iVProg.

ALUNO 16

Resolução correta.

ALUNO 17

Entregou exercício em branco.

ALUNO 18

Código sem comentário, loop com range errado e com atribuições de variáveis que não são necessárias (variável recebe ela mesma para atualização).
Falta: explicar o que é cada variável.

Aluno apresentou dificuldade na resolução, e então lhe foi ensinado o conceito de teste de mesa, para que pudesse encontrar o problema em seu código. Com isso, conseguiu desenvolver uma solução.

APÊNDICE F: PRIMEIRA AVALIAÇÃO FORMAL DE MAC 110

A primeira avaliação foi composta de seis exercícios, que abordaram conceitos a respeito de atribuição de variável, laços (for, while, …), condições de execução (if, else, …), testes. Para resolução, não era necessário escrever em nenhuma linguagem de programação formal.

Tabela 8 - Enunciado dos exercícios da primeira avaliação de MAC 110.

Número do exercício	Enunciado
1	Dado n e uma sequência de inteiros com n elementos, determinar o número de subsequências de números iguais
2	Simular no espaço abaixo o algoritmo a seguir.
3	Fazer um programa Portugol no qual o usuário digita dois naturais, N1 e N2, então seu programa deve verificar se N1 é múltiplo de N2, se N2 é múltiplo de N1 ou se nenhum é múltiplo um do outro.
4	Fazer um programa Portugol que para um dado N > 1, computa e imprime: 2 n + 3 (n−1) + 4 (n−2) + . . . + 3 Usar um único laço do tipo repita para.
5	Fazer um programa Portugol no qual o usuário digita um natural N, gerando um natural K com os dígitos de N em ordem inversa.
6	Fazer um programa Portugol no qual o usuário digita um natural N composto apenas por dígitos 0 e 1 (portanto representando um número binário), então converter esse número para seu correspondente decimal.

Fonte: produzido pelo professor da disciplina.

O desempenho dos alunos encontra-se na tabela a seguir.

Tabela 9 - Desempenho dos alunos nas questões da avaliação.

Q1	Q2	Q3	Q4	Q5	Soma questões	Nota	Desvio padrão	Duração
1,5	1,3	1,8	2,3	2	8,9	8,9	79.21	01:40

1,5	1,3	1,7	0,5	0	5	5	25.00	01:10
1,3	0,7	0,3	0	2,3	4,6	5,29		01:50
1,5	1,3	0,5	0	0	3,3	3,3	10.89	01:51
1,5	1,5	1,9	0	0	4,9	4,9	24.01	01:50
0,8	1	0	1,8		3,6	3,24		00:45
1,2	0	0	0	1,2	2,4	1,44		00:40
1,5	1	0	1	0	3,5	3,5	12.25	01:10
1,1	0,5	1,5		0,5	3,6	3,6	12.96	01:00
1,5	1,3	1,9		1	5,7	5,7	32.49	01:44
1,3	0,5	1	0	2,8	5,6	7,84		01:51
1,5	1,1	0	0	2,6	5,2	6,76		01:35
1,5	1,3	2	2,5	2,3	9,6	9,6	92.16	01:30
1,5	1,5	0,3	2,5	2,3	8,1	8,1	65.61	01:50
1,5	1,1	1,7	2,4	2,2	8,9	8,9	79.21	01:15
1,5	1,5	1,5	0	0	4,5	4,5	20.25	01:44
1,5	1,4	2	1,5	2,3	8,7	8,7	75.69	01:38
1,2	0,1	0,1	0	1,4	2,8	1,96		00:40
0,9	1,3	0	0	2,2	4,4	4,84		01:10

Fonte: disponibilizada pelo professor da disciplina.

APÊNDICE G: PRIORIZAÇÃO DE DADOS DO ITAREFA PARA SEREM VISUALIZADOS NO DASHBOARD

Tabela 10 - Priorização dos dados disponíveis do iTarefa a partir da relevância e do esforço para tratamento.

Tabela	Dado	Formato	Descrição	Relevância	Esforço
iassign	id	Big int	Identificação do bloco do iTarefa	4	5
iassign	course	Big int	Identificação do curso em que está o bloco de exercícios	4	5
iassign	name	Var char	Nome do bloco	3	2
iassign	intro	Long text	-	1	1
iassign	introformat	Small int	-	1	1
iassign	activity_group	Tiny int	-	1	1
iassign	grade	Double	-	1	1
iassign	timeavailable	Big int	Tempo disponível	1	1
iassign	timedue	Big int	Tempo para conclusão	1	1
iassign	preventiate	Tiny int	-	1	1
iassign	test	Tiny int	-	1	1
iassign	max_experiment	Big int	-	1	1
iassign_allsubmissions	id	Big int	Identificação da submissão	4	5
iassign_allsubmissions	iassign_statementid	Big int	Identificação do exercício	4	5

iassign_allsubmissions	userid	Big int	Identificação do usuário (aluno)	4	5
iassign_allsubmissions	timecreated	Big int	Data de criação	2	2
iassign_allsubmissions	grade	Double	Nota na submissão (entre 0 e 1)	5	4
iassign_allsubmissions	answer	Long text	-	1	1
iassign_statement	id	Big int	Identificação do exercício	4	5
iassign_statement	name	Var char	Nome do exercício	3	2
iassign_statement	iassignid	Big int	Identificação do bloco do iTarefa	4	5
iassign_statement	type_iassign	Tiny int	Tipo de exercício	1	1
iassign_statement	proposition	Long text	Enunciado do exercício	3	4
iassign_statement	author_name	Var char	Nome do criador do exercício	1	1
iassign_statement	author_modified_name	Var char	Nome do usuário que modificou o exercício	1	1
iassign_statement	iassign_limid	Big int	-	1	1
iassign_statement	file	Var char	-	1	1
iassign_state	grade	Double	-	1	1

ment					
iassign_statement	timemodified	Big int	Data de modificação	1	1
iassign_statement	timecreated	Big int	Data de criação	1	1
iassign_statement	timeavailable	Big int	Tempo de disponibilidade	1	1
iassign_statement	timedue	Big int	Tempo para conclusão	1	1
iassign_statement	preventiate	Big int	-	1	1
iassign_statement	test	Tiny int	-	1	1
iassign_statement	special_param1	Tiny int	-	1	1
iassign_statement	position	Big int	-	1	1
iassign_statement	visible	Tiny int	Flag de visibilidade para os alunos	5	3
iassign_statement	max_experiment	Big int	-	1	1
iassign_statement	dependency	Var char	-	1	1
iassign_statement	automatic_evaluate	Tiny int	-	1	1
iassign_statement	show_answer	Tiny int	Flag para mostrar ou não a resposta para os alunos	1	1

iassign_statement	store_all_submissions	Big int	Flag para guardar todas as submissões dos alunos	1	1
iassign_statement	filesid	Var char	Id do arquivo	1	1

Fonte: produzida pela autora.

APÊNDICE H: CÓDIGO PARA BLOCO DASHBOARD

```
<?php
class block_tccdashboard extends block_base {
   public function init() {
       $this->title = get_string('pluginname',
'block_tccdashboard');
   }

   public function get_content() {
       global $USER, $DB, $COURSE, $PAGE, $OUTPUT, $CFG;

       if ($this->content !== null) { return $this->content; }
       $id_course = optional_param('id', 0, PARAM_INT);
       $query_count = 'SELECT count(id) FROM
{iassign_allsubmissions}';
       $query_average =
           "SELECT
               ROUND(AVG(grade), 2) as avg_grade
           FROM(
               SELECT
                   user_grade_fixed.id_user,
                   AVG(grade) as grade
               FROM(
                   SELECT
                       user_ex.userid as id_user,
                       IF(ISNULL(user_grade.grade), 0,
user_grade.grade)*10 as grade
                   FROM(
                       SELECT
                       statement_aux.id as iassign_statementid,
                       all_subm.userid,
                       MAX(all_subm.timecreated) AS timecreated
                       FROM (
                           SELECT
                               statem.id
                           FROM
                               {iassign_statement} as statem,
                               {iassign} as iassign
                           WHERE
                               statem.visible = 1 AND
                               statem.iassignid = iassign.id AND
```

```
                                iassign.course = {$id_course}
                                ) as statement_aux
                        LEFT JOIN
                            {iassign_allsubmissions} as all_subm
                        ON 1 = 1
                        GROUP BY 1, 2
            ) as user_ex
            LEFT JOIN
                {iassign_allsubmissions} as user_grade
            ON
                user_ex.iassign_statementid =
user_grade.iassign_statementid AND
                user_ex.userid = user_grade.userid AND
                user_ex.timecreated = user_grade.timecreated) as
user_grade_fixed
            GROUP BY 1) avg_aux;";

        $count_class = $DB-> get_record_sql($query_count);
        $average_class = $DB->get_record_sql($query_average);

        foreach($count_class as $x) $count = $x;
        foreach($average_class as $x) $average = round($x, 2);

        $this->content =  new stdClass;

        $this->content->text = "
            <br> Entre aqui para ver as análises sobre os alunos
            <br> No curso atual, temos {$count} submissões no
iTarefa.";

        $this->content->text .= "<br> A média atual da turma é de
{$average}";
        //botão para página do dashboard
        $parameters = array('id'=>$COURSE->id);
        $url = new moodle_url('/blocks/tccdashboard/view.php',
$parameters);
        $this->content->text .= "<center>" .
$OUTPUT->single_button($url, "Saiba mais") . "</center>";
        return $this->content;

    }
}
```

APÊNDICE I: CÓDIGO PARA PÁGINA DE VISUALIZAÇÃO DO DASHBOARD

```
<?php

require_once("../../config.php");

$id_course = optional_param('id', 0, PARAM_INT); // Course Module
ID
print $OUTPUT->header();

if ($id_course != 0) {
   print "Aqui: course.id=" . $id_course . "<br>";

   //Título do Dashboard
   $query_course_name =
       "SELECT
           fullname
       FROM
           {course}
       WHERE
           id = {$id_course}
       ";
   $course_name = $DB->get_record_sql($query_course_name);

   foreach($course_name as $x) $name = $x;

   print "<br><h1><div style='text-align:center;
font-weight:bold'>Dashboard de desempenho nas atividades do iTarefa
do curso {$name}</h1></div><br>";


   //
   //
   //Big numbers
   //
   //

   //1:
   //Número de exercícios propostos aos alunos
   $query_num_proposed_ex =
       "SELECT
           count({iassign_statement}.id)
```

```
        FROM
            {iassign_statement}, {iassign}
        WHERE
            visible = 1 AND
            {iassign_statement}.iassignid = {iassign}.id AND
            {iassign}.course = {$id_course}
            ";

    $num_proposed_ex = $DB->get_record_sql($query_num_proposed_ex);

    foreach($num_proposed_ex as $x) $num = $x;

    //2:
    //% de respostas
    //Número de usuários do curso
    $query_num_users_on_course =
        "SELECT
            count(DISTINCT userid)
        FROM
            {user_enrolments} as us,
            {enrol} as en
        WHERE
            en.courseid = {$id_course} AND
            us.enrolid = en.id
        ";
    $num_users_on_course =
$DB->get_record_sql($query_num_users_on_course);
    foreach($num_users_on_course as $x) $num_users = $x;
    //Número de submissões
    $query_actual_num_sub =
        "SELECT
        count(*)
    FROM (
        SELECT
            all_subm.userid,
            all_subm.iassign_statementid,
            max(all_subm.timecreated) as timecreated
        FROM
            {iassign_allsubmissions} as all_subm,
            {iassign_statement} as statem,
            {iassign} as ias
        WHERE
```

```
            statem.visible = 1 AND
            statem.iassignid = ias.id AND
            ias.course = 2  AND
            statem.id = all_subm.iassign_statementid
        GROUP BY 1, 2) as aux
        ";

    $actual_num_sub = $DB->get_record_sql($query_actual_num_sub);
    foreach($actual_num_sub as $x) $num_sub = $x;

    $perc_submissions = round($num_sub /($num * $num_users) * 100,
0, PHP_ROUND_HALF_EVEN);

    //3:
    //Média da turma
    $query_avg_grade_course =
        "SELECT
            ROUND(AVG(grade), 2) as avg_grade
        FROM(
            SELECT
                user_grade_fixed.id_user,
                AVG(grade) as grade
            FROM(
                SELECT
                    user_ex.userid as id_user,
                    IF(ISNULL(user_grade.grade), 0,
user_grade.grade)*10 as grade
                FROM(
                    SELECT
                        statement_aux.id as iassign_statementid,
                        all_subm.userid,
                        MAX(all_subm.timecreated) AS timecreated
                    FROM (
                        SELECT
                            statem.id
                        FROM
                            {iassign_statement} as statem,
                            {iassign} as iassign
                         WHERE
                             statem.visible = 1 AND
                             statem.iassignid = iassign.id AND
                             iassign.course = {$id_course}
```

```
                        ) as statement_aux,
                        {iassign_allsubmissions} as all_subm
                    GROUP BY 1, 2
            ) as user_ex
            LEFT JOIN
                {iassign_allsubmissions} as user_grade
            ON
                user_ex.iassign_statementid =
user_grade.iassign_statementid AND
                user_ex.userid = user_grade.userid AND
                user_ex.timecreated = user_grade.timecreated) as
user_grade_fixed
        GROUP BY 1) avg_aux
            ";

    $avg_grade_course =
$DB->get_record_sql($query_avg_grade_course);

    foreach($avg_grade_course as $x) $course_avg_grade = $x;
    $course_avg_grade = round($course_avg_grade, 2,
PHP_ROUND_HALF_EVEN);

    //4:
    //% de participação da turma no curso
    $query_num_user_participation =
        "SELECT
            count(DISTINCT all_subm.userid) as num_users_part
        FROM
            {iassign_allsubmissions} as all_subm,
            {iassign_statement} as statem,
            {iassign} as iassign
        WHERE
            statem.iassignid = iassign.id AND
            iassign.course = {$id_course}  AND
            statem.id = all_subm.iassign_statementid";
    $num_user_participation =
$DB->get_record_sql($query_num_user_participation);

    foreach($num_user_participation as $x) $num_user_ex = $x;

    $participation = round($num_user_ex /$num_users * 100, 0,
PHP_ROUND_HALF_EVEN);
```

```
    //impressão Big numbers
    print "
        <body>
        <div style='width: 100%;
                    display: table-row;
                    left-margin: 20px;
                    right-margin:20px;
                    '>
            <div style='width: 25%;
                        display: table-cell;
                        text-align: center;
                        font-size: 20px;
                        vertical-align: middle;
                        border: 2px solid black;
                        border-radius: 5px;
                        background: #89D4FF;
                        '>
                Número de exercícios propostos<br><b>{$num}</b>
            </div>
            <div style='width: 25%;
                        display: table-cell;
                        text-align: center;
                        font-size: 20px;
                        vertical-align: middle;
                        border: 2px solid black;
                        border-radius: 5px;
                        background: #6ed4b1;
                        '>
                Porcentagem de
respostas<br><b>{$perc_submissions}%</b>
            </div>
            <div style='width: 25%;
                        display: table-cell;
                        text-align: center;
                        font-size: 20px;
                        vertical-align: middle;
                        border: 2px solid black;
                        border-radius: 5px;
                        background: #6F9AF2;'>
                Média da turma<br><b>{$course_avg_grade}</b>
            </div>
```

```
            <div style='width: 25%;
                        display: table-cell;
                        text-align: center;
                        font-size: 20px;
                        vertical-align: middle;
                        border: 2px solid black;
                        border-radius: 5px;
                        background: #B792D6;'>
                    Porcentagem de participação dos alunos
<br><b>{$participation}%</b>
            </div>
        </div>

            ";

    //Submissões e médias dos alunos por exercício
    $query_table_stat =
        "SELECT
            student.id_ex,
            student.ex_name,
            IF(ISNULL(student.avg_grade), '-', student.avg_grade) as
avg_grade,
            IF(ISNULL(student.avg_number_of_submissions), '-',
student.avg_number_of_submissions) as avg_number_of_submissions,
            IF(ISNULL(all_subm.num_us), 0 ,all_subm.num_us) AS
num_of_users
        FROM
            (SELECT
                student_grade.iassign_statementid as id_ex,
                student_grade.name as ex_name,
                ROUND(AVG(student_grade.grade), 2) AS avg_grade,
                ROUND(AVG(num_subm.number_of_submissions), 1) AS
avg_number_of_submissions
            FROM
                (SELECT
                last_subm.iassign_statementid,
                last_subm.id_user,
                last_subm.name,
                IF(ISNULL(all_subm.grade), 0, all_subm.grade)*10 as
grade
                FROM(
                    SELECT
```

```
                    all_us_ex.iassign_statementid,
                    all_us_ex.id_user,
                    all_us_ex.name,
                    MAX(all_subm.timecreated) as date_time
                    FROM (
                        #todos os alunos do curso
                        SELECT DISTINCT
                            statement_aux.id as iassign_statementid,
                            statement_aux.name,
                            us.userid as id_user #id do aluno
                        FROM
                            (SELECT
                                statem.id,
                                statem.name
                            FROM
                                {iassign_statement} as statem,
                                {iassign} as iassign
                            WHERE
                                statem.visible = 1 AND
                                statem.iassignid = iassign.id AND
                                iassign.course = {$id_course}
                            ) as statement_aux
                            LEFT JOIN
                                (SELECT
                                    userid
                                FROM
                                    {user_enrolments} as us
                                LEFT JOIN
                                    {enrol} as en
                                ON
                                    us.enrolid = en.id
                                WHERE
                                    en.courseid = {$id_course}
                                ) as us
                            ON
                                1 = 1
                        ) AS all_us_ex
                    LEFT JOIN
                        {iassign_allsubmissions} as all_subm
                    ON
                        all_us_ex.iassign_statementid =
all_subm.iassign_statementid AND
```

```
                    all_us_ex.id_user = all_subm.userid
                GROUP BY 1, 2
                ) as last_subm
            LEFT JOIN
                {iassign_allsubmissions} as all_subm
            ON
                last_subm.id_user = all_subm.userid AND #para
cada usuário
                last_subm.date_time = all_subm.timecreated AND
#para a data correta
                last_subm.iassign_statementid =
all_subm.iassign_statementid #para o exercício correto
            ) as student_grade
        LEFT JOIN (
                SELECT
                    userid as id_user,
                    iassign_statementid,
                    IF(ISNULL(COUNT(all_subm.id)), 0,
COUNT(all_subm.id)) as number_of_submissions
                FROM
                    {iassign_allsubmissions} as all_subm,
                    {iassign_statement} as statem,
                    {iassign} as iassign
                WHERE
                    statem.visible = 1 AND
                    statem.iassignid = iassign.id AND
                    iassign.course = {$id_course}  AND
                    statem.id = all_subm.iassign_statementid
            GROUP BY 1, 2
            ) AS num_subm
        ON
            num_subm.id_user = student_grade.id_user AND
            num_subm.iassign_statementid =
student_grade.iassign_statementid
        GROUP BY 1
        ) AS student
    LEFT JOIN
        (SELECT
            iassign_statementid,
            IF(ISNULL(COUNT(DISTINCT userid)), 0, COUNT(DISTINCT
userid)) as num_us
        FROM
```

```
                {iassign_allsubmissions}
            GROUP BY 1
            ) as all_subm
        ON
            student.id_ex = all_subm.iassign_statementid
        ORDER BY 1
        ";

    $class_table_students_status = $DB->
get_records_sql($query_table_stat);
    $table_stat = new html_table();
    $table_stat->head = array('Exercício',  'Responderam', ' Não
responderam', 'Média', 'Média de submissões');

    $linhas = array();
    foreach ($class_table_students_status as $records) {
        $ex_name = $records->ex_name;
        $num_of_users = $records->num_of_users;
        $num_remaining_users = ($num_users -
$records->num_of_users);
        $avg_grade = $records->avg_grade;
        $avg_number_of_submissions =
$records->avg_number_of_submissions;
        $linhas[] = array($ex_name, $num_of_users,
$num_remaining_users, $avg_grade, $avg_number_of_submissions);
    }
    $table_stat->data = $linhas;

    print "<div style='width: 100%;
                        text-align: center;
                        font-size: 24px;
                        '>
                <b>Desempenho por exercício</b>
            </div>";
    print "<div style='width: 100%;
                        text-align: center;
                        '> ";
    print html_writer::table($table_stat);
    print "</div>";

    //Alunos com média menor que 5:
    $query_user_course_information =
```

```
        "SELECT
            user_course_info.*,
            CONCAT(user.firstname, ' ', user.lastname) as user_name
        FROM
            (SELECT
                user_ex.userid as id_user,
                user_ex.user_number_of_submissions,
                ROUND(AVG(user_grade.grade)*10, 2) as avg_grade
            FROM(
                SELECT
                    us_all_subm.*,
                    count_subm.user_number_of_submissions
                FROM(
                    SELECT
                        statement_aux.id as iassign_statementid,
                        all_subm.userid,
                        MAX(all_subm.timecreated) AS timecreated
                    FROM (
                        SELECT
                            statem.id
                        FROM
                            {iassign_statement} as statem,
                            {iassign} as iassign
                        WHERE
                            statem.visible = 1 AND
                            statem.iassignid = iassign.id AND
                            iassign.course = {$id_course}
                        ) as statement_aux,
                            {iassign_allsubmissions} as all_subm
                        GROUP BY 1, 2
                        ) as us_all_subm
                    LEFT JOIN
                        (SELECT
                            userid,
                            COUNT(DISTINCT iassign_statementid) AS
user_number_of_submissions
                        FROM
                            {iassign_allsubmissions}
                        GROUP BY 1) as count_subm
                    ON
                        count_subm.userid = us_all_subm.userid
            ) as user_ex
```

```
            LEFT JOIN
                {iassign_allsubmissions} as user_grade
            ON
                user_ex.iassign_statementid =
user_grade.iassign_statementid AND
                user_ex.userid = user_grade.userid AND
                user_ex.timecreated = user_grade.timecreated
            GROUP BY 1, 2) as user_course_info,
                {user} as user
            WHERE
                user.id = user_course_info.id_user AND
                user_course_info.avg_grade <= 5
            ORDER BY 1
            ";
    $class_table_students_course_info = $DB->
get_records_sql($query_user_course_information);
    $table_students_course_info = new html_table();
    $table_students_course_info->head = array('Nome', 'Porcentagem
de respostas', 'Média');

    $lines_students_course_info = array();
    foreach ($class_table_students_course_info as $records) {
        $user_name = $records->user_name;
        $perc_answer = ($records->user_number_of_submissions / $num
* 100) . "%";
        $avg_grade = $records->avg_grade;
        $lines_students_course_info[] = array($user_name,
$perc_answer, $avg_grade);
    }
    $table_students_course_info->data = $lines_students_course_info;

    print "<div style='width: 100%;
                        text-align: center;
                        font-size: 24px;
                        '>
                <b>Alunos abaixo da média</b>
            </div>";

    print "<div style='width: 100%;
                        text-align: center;
                        '> ";
```

```
    print html_writer::table($table_students_course_info);

    print "</div>";


    //Exercícios com média menor que 5
    $query_exercise_information =
        "SELECT *
        FROM (
            SELECT
                subm_number.*,
                IF(ISNULL(student_grade.grade), '-',
ROUND(AVG(student_grade.grade)*10, 2)) AS avg_grade,
                IF(ISNULL(student_grade.number_of_submissions), '-',
ROUND(AVG(student_grade.number_of_submissions), 0)) AS
avg_number_of_submissions
        FROM
            (SELECT
                course_ex.*,
                COUNT(DISTINCT subm.userid) as num_of_users #número
de alunos que fizeram a tarefa
            FROM
                (SELECT
                    block.id as id_block_iassign, #id de cada bloco
do iassign
                    block.course,
                    block.name as block_name,
                    ex.id as id_ex
                FROM
                    {iassign} as block,
                    {iassign_statement} as ex,
                    {iassign} as iassign
                WHERE
                    block.id = ex.iassignid AND
                    ex.visible = 1 AND
                    ex.iassignid = iassign.id AND
                    iassign.course = {$id_course}
            ) as course_ex
            LEFT JOIN
                {iassign_allsubmissions} as subm
            ON
                course_ex.id_ex = subm.iassign_statementid
```

```
            GROUP BY 1, 2, 3, 4
            ) AS subm_number
        LEFT JOIN
            (
            SELECT
                max_grade_ex.*,
                num_subm.number_of_submissions
            FROM
                (SELECT
                    last_subm.*,
                    all_subm.grade
                FROM
                    (SELECT
                        iassign_statementid,
                        userid as id_user, #id do aluno
                        MAX(all_subm.timecreated) as date_time #data
de submissão, está em bigint --> ver como traduzir para data
                    FROM
                        {iassign_allsubmissions} as all_subm,
                        {iassign_statement} as statem,
                        {iassign} as iassign
                    WHERE
                        statem.visible = 1 AND
                        statem.iassignid = iassign.id AND
                        iassign.course = {$id_course}  AND
                        statem.id = all_subm.iassign_statementid
                    GROUP BY 1, 2
                    ) AS last_subm,
                    {iassign_allsubmissions} as all_subm
                WHERE
                    last_subm.id_user = all_subm.userid AND #para
cada usuário
                    last_subm.date_time = all_subm.timecreated AND
#para a data correta
                    last_subm.iassign_statementid =
all_subm.iassign_statementid #para o exercício correto
                ) AS max_grade_ex
            LEFT JOIN
                (SELECT
                    userid,
                    iassign_statementid,
                    COUNT(all_subm.id) as number_of_submissions
```

```
                FROM
                    {iassign_allsubmissions} as all_subm,
                    {iassign_statement} as statem,
                    {iassign} as iassign
                WHERE
                    statem.visible = 1 AND
                    statem.iassignid = iassign.id AND
                    iassign.course = {$id_course}  AND
                    statem.id = all_subm.iassign_statementid
                GROUP BY 1, 2
                ) AS num_subm
            ON
                max_grade_ex.id_user = num_subm.userid AND
                max_grade_ex.iassign_statementid =
num_subm.iassign_statementid
            ) AS student_grade
        ON
            student_grade.iassign_statementid = subm_number.id_ex
        GROUP BY
            1, 2, 3, 4, 5
        ORDER BY 2, 3, 4) as ex_info
        WHERE
            ex_info.avg_grade <= 5
        ";

   $class_table_exercise_information = $DB->
get_records_sql($query_exercise_information);
   $table_exercise_information = new html_table();
   $table_exercise_information->head = array('Bloco', 'id
Exercício', 'Média', 'Média de submissões');

   $linhas_exercise_information = array();
   foreach ($class_table_exercise_information as $records) {
       $block_name = $records->block_name;
       $id_ex = $records->id_ex;
       $avg_grade = $records->avg_grade;
       $avg_number_of_submissions =
$records->avg_number_of_submissions;
       $linhas_exercise_information[] = array($block_name, $id_ex,
$avg_grade, $avg_number_of_submissions);
   }
   $table_exercise_information->data =
```

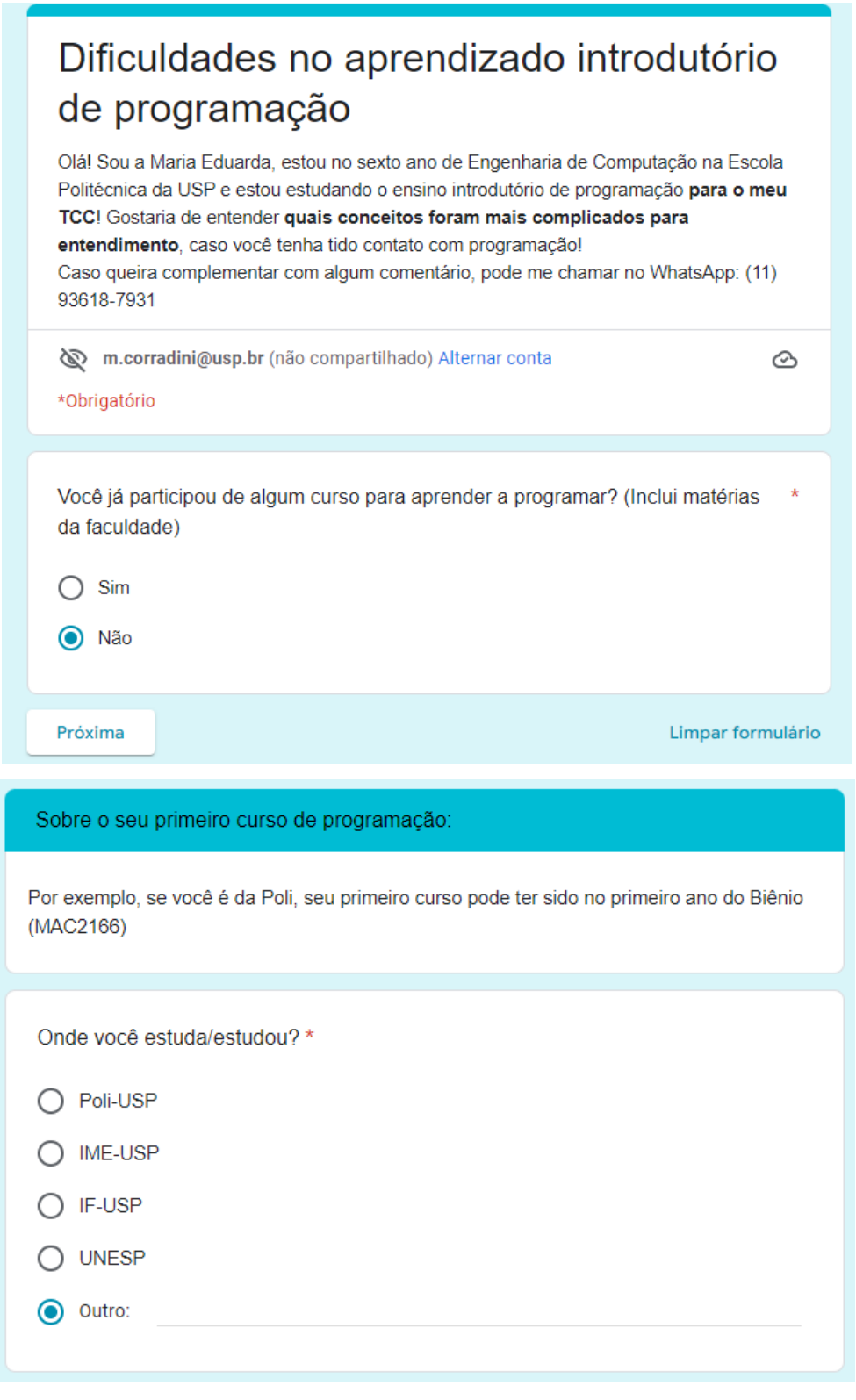
```
$linhas_exercise_information;
    print "<div style='width: 100%;
                        text-align: center;
                        font-size: 24px;
                        '>
                <b>Exercício com média menor que 5</b>
            </div>";

    print "<div style='width: 100%;
                        text-align: center;
                        '> ";
    print html_writer::table($table_exercise_information);
    print "</div>";


    //todos os alunos

    $query_all_user_course_information =
        "SELECT
            user_course_info.*,
            CONCAT(user.firstname, ' ', user.lastname) as user_name
        FROM
            (SELECT
                user_ex.userid as id_user,
                user_ex.user_number_of_submissions,
                ROUND(AVG(user_grade.grade)*10, 2) as avg_grade
            FROM(
                SELECT
                    us_all_subm.*,
                    count_subm.user_number_of_submissions
                FROM(
                    SELECT
                        statement_aux.id as iassign_statementid,
                        all_subm.userid,
                        MAX(all_subm.timecreated) AS timecreated
                    FROM (
                        SELECT
                            statem.id
                        FROM
                            {iassign_statement} as statem,
                            {iassign} as iassign
                        WHERE
```

```
                            statem.visible = 1 AND
                            statem.iassignid = iassign.id AND
                            iassign.course = {$id_course}) as
statement_aux,
                            {iassign_allsubmissions} as all_subm
                        GROUP BY 1, 2
                        ) as us_all_subm
                    LEFT JOIN
                        (SELECT
                            userid,
                            COUNT(DISTINCT iassign_statementid) AS
user_number_of_submissions
                        FROM
                            {iassign_allsubmissions}
                        GROUP BY 1) as count_subm
                    ON
                        count_subm.userid = us_all_subm.userid
            ) as user_ex
            LEFT JOIN
                {iassign_allsubmissions} as user_grade
            ON
                user_ex.iassign_statementid =
user_grade.iassign_statementid AND
                user_ex.userid = user_grade.userid AND
                user_ex.timecreated = user_grade.timecreated
            GROUP BY 1, 2) as user_course_info,
                {user} as user
            WHERE
                user.id = user_course_info.id_user
            ORDER BY 1
            ";

   $class_table_all_user_course_information = $DB->
get_records_sql($query_all_user_course_information);
   $table_all_user_course_information = new html_table();
   $table_all_user_course_information->head = array('Nome',
'Porcentagem de respostas');

   $lines_all_user_course_information = array();
   foreach ($class_table_all_user_course_information as $records) {
       $user_name = $records->user_name;
       $perc_answer = ($records->user_number_of_submissions / $num
```

```
* 100) . "%";
        $lines_all_user_course_information[] = array($user_name,
$perc_answer);
    }
    $table_all_user_course_information->data =
$lines_all_user_course_information;

    print "<div style='width: 100%;
                        text-align: center;
                        font-size: 24px;
                        '>
                <b>Visão geral dos alunos</b>
            </div>";

    print "<div style='width: 100%;
                        text-align: center;
                        '> ";

    print html_writer::table($table_all_user_course_information);

    print "</div>";
    print "</body>";
}
else {
    print "<br>
    <h1>
    <div style='text-align:center; font-weight:bold'>
        Dashboard não disponível
    </div>
    </h1>
    ";
}
print "<div style 'vertical-align: bottom;'>";
    print $OUTPUT->footer();
    print "</div>";


?>
```

APÊNDICE J: FORMULÁRIO PARA PESQUISA SOBRE DIFICULDADES NO APRENDIZADO INTRODUTÓRIO DE PROGRAMAÇÃO

Figura 9 - Formulário “Dificuldades no aprendizado introdutório de programação”.



Qual curso você estudou/estuda? *

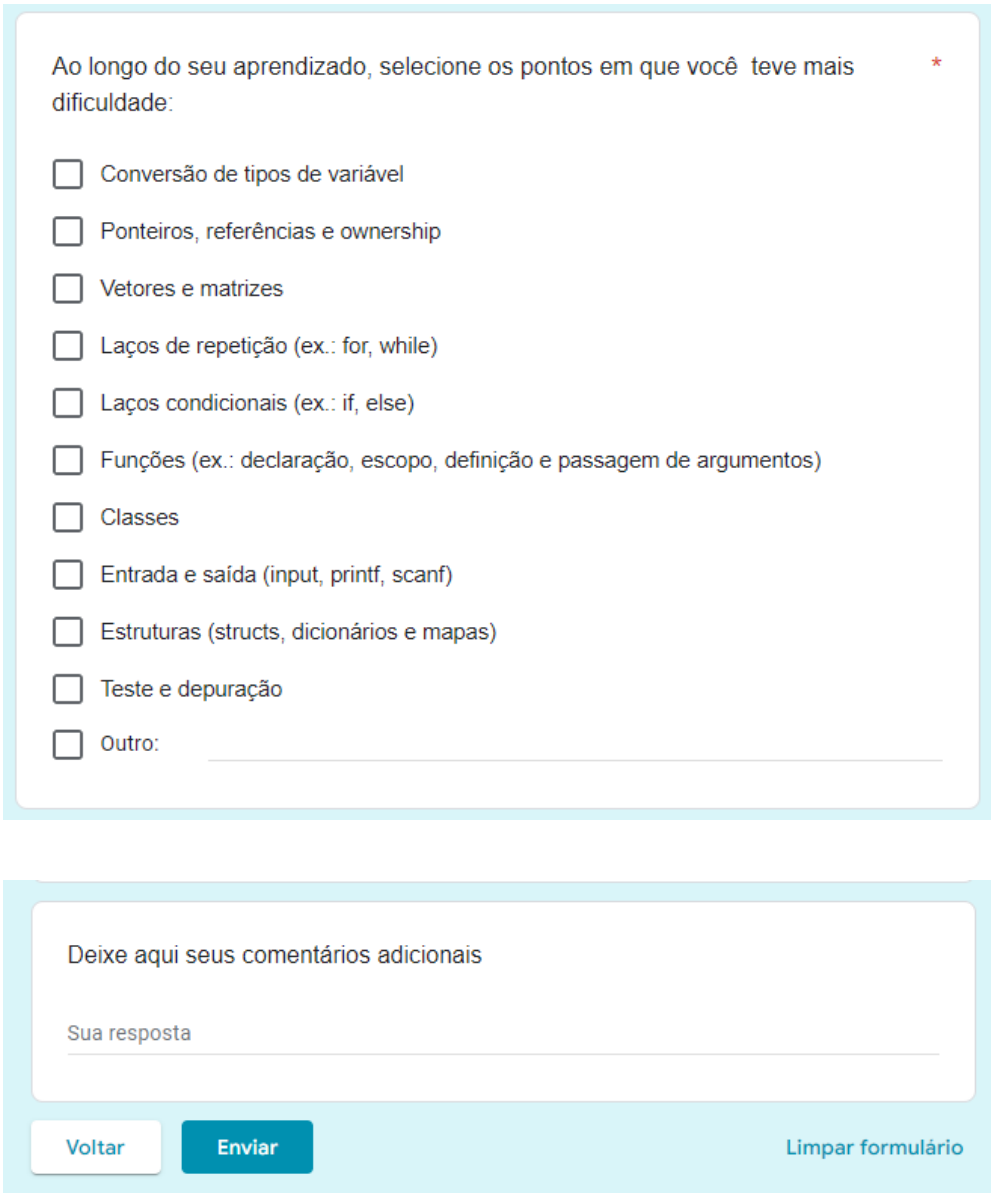
- ◯ Engenharia ou relacionados
- ◯ Matemática, Estatística ou relacionados
- ◯ Física ou relacionados
- ◯ Outro: _____

Onde você teve seu primeiro curso de programação? *

- ◯ Na escola (ensino fundamental, ensino médio, técnico)
- ◯ Na faculdade
- ◯ Online (Ex.: coursera)
- ◯ Outro: _____

Qual linguagem de programação você aprendeu primeiro? *

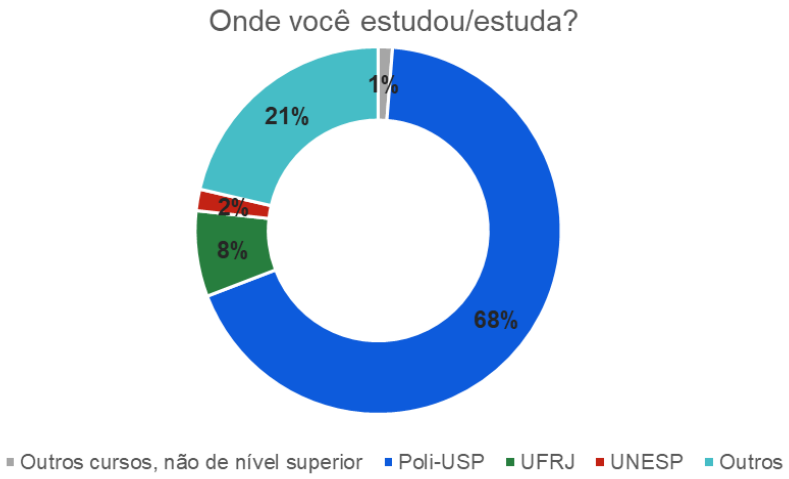
- ◯ C/C++
- ◯ Python
- ◯ Java
- ◯ VBA
- ◯ Javascript
- ◯ Scratch, Portugol ou outra linguagem de programação em bloco
- ◯ Outro: _____



Fonte: produzido pela autora.

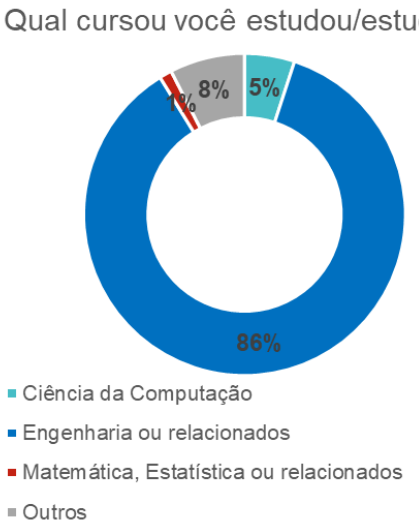
As respostas referentes às perguntas do formulário acima estão ilustradas nos gráficos a seguir.

Gráfico 6 - Porcentagem de respostas no formulário à pergunta “onde você estudou/estuda?”.



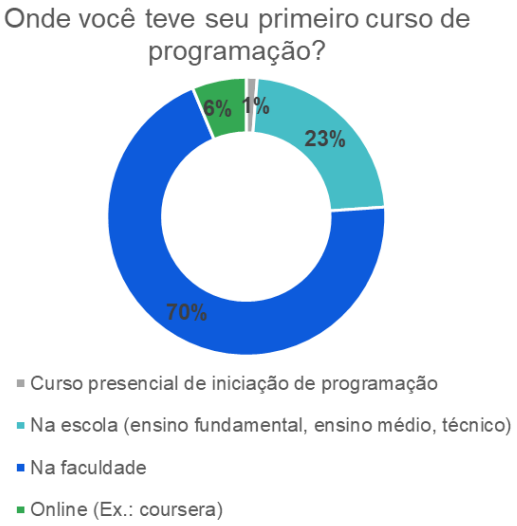
Fonte: produzido pela autora.

Gráfico 7 - Porcentagem de respostas no formulário à pergunta “qual curso você estudou/estuda?”.



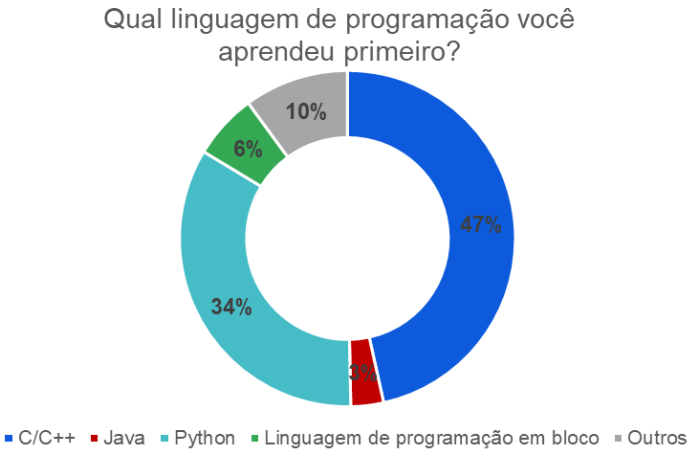
Fonte: produzido pela autora.

Gráfico 8 - Porcentagem de respostas no formulário à pergunta “onde você teve seu primeiro curso de programação?”.



Fonte: produzido pela autora.

Gráfico 9 - Porcentagem de respostas no formulário à pergunta “qual linguagem de programação você aprendeu primeiro?”.



Fonte: produzido pela autora.