UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS**

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

LICENCIATURA EM COMPUTAÇÃO

WELDREY TONETO DE OLIVEIRA

UM ESTUDO SOBRE OS PROBLEMAS E DIFICULDADES RELACIONADOS AO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

BANDEIRANTES - PR

2018

WELDREY TONETO DE OLIVEIRA

UM ESTUDO SOBRE OS PROBLEMAS E DIFICULDADES RELACIONADOS AO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação e Licenciado em Computação.

Orientador: Prof. Dr. Maurício Massaru Arimoto

BANDEIRANTES - PR

2018

WELDREY TONETO DE OLIVEIRA

UM ESTUDO SOBRE OS PROBLEMAS E DIFICULDADES RELACIONADOS AO ENSINO E APRENDIZAGEM DE PROGRAMAÇÃO

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação e Licenciado em Computação.

COMISSÃO EXAMINADORA

Prof. Dr. Maurício Massaru Arimoto
UENP – Campus Luiz Meneghel

Prof. Me. Fábio de Sordi Junior
UENP – Campus Luiz Meneghel

Prof. Me. Fábio Carlos Moreno
UENP – Campus Luiz Meneghel

Bandeirantes, 11 de Julho de 2018.

AGRADECIMENTOS

Agradeço primeiramente a Deus que permitiu que tudo isso acontecesse ao longo de minha vida e não somente nestes anos como universitário.

Aos meus pais Lúcia e Wellington e ao meu irmão Weslley que sempre me apoiaram nas minhas decisões e estiveram comigo em todos os momentos me dando forças para continuar acreditando.

Ao meu orientador Professor Maurício Massaru Arimoto, pela paciência na orientação, pela oportunidade e apoio que tornaram possível a conclusão deste trabalho.

Aos membros da banca examinadora Professor Fábio de Sordi Junior e Professor Fábio Carlos Moreno pelas sugestões apontadas.

A todos os professores que participaram da minha formação acadêmica durante esses anos de curso, por me proporcionarem conhecimento e conselhos.

A todos os meus amigos que me acompanharam durante essa jornada, em especial Aline e Yhago que foram meus companheiros de estudos e irmãos na amizade que fizeram parte da minha formação e que vão continuar presentes em minha vida.

RESUMO

A programação de computadores é uma disciplina que auxilia no desenvolvimento do raciocínio lógico, criatividade e capacidade de abstração, além de ser de suma importância para os estudantes que desejam trabalhar futuramente no desenvolvimento de software. No entanto, muitos estudantes têm dificuldades de compreender os conceitos de programação devido sua inerente complexidade. Essas dificuldades, se não sanadas a tempo, acabam prejudicando a aprendizagem dos estudantes ao longo da disciplina de programação e contribuem com o aumento da taxa de reprovação e evasão dos cursos da área de tecnologia. Este trabalho tem como objetivo identificar e analisar as principais barreiras e dificuldades que os estudantes encontram durante o aprendizado de programação. Para isso, um estudo baseado em survey é realizado com estudantes e recém-formados em cursos de Tecnologia da Informação e Comunicação de diversas regiões do Brasil. Os resultados obtidos evidenciam diversas barreiras e dificuldades para a aprendizagem de programação, que variam desde o entendimento da lógica de programação até a definição e uso de algoritmos recursivos e estrutura de dados. Os problemas levantados neste trabalho apontam a para a necessidade de mudanças e alterações nas práticas atuais de ensino e aprendizagem de programação, aliadas ao uso de ferramentas e ambientes de aprendizagem que facilitem o entendimento dos conceitos abstratos da programação.

Palavras-chave: Programação de computadores, ensino e aprendizagem de programação, dificuldades no aprendizado de programação, reprovação, evasão.

ABSTRACT

The Computer programming is a discipline that assists students in the development of logical thinking, creativity and capability for abstraction, as well as fundamental for students who wish to to work with software development in the future. However, many students have difficulties to understand programming concepts due to their inherent complexity. These difficulties, if not solved in time, impair students learning throughout the programming course and contribute to the increase of rate of disapproval and evasion in technology courses. This work aims to identify and analyze the main barriers and difficulties that students face during programming learning. We conduct a survey-based study with students and recent graduates in Information and Communication Technology courses from different regions of Brazil. The results show several barriers and difficulties for learning programming, ranging from the understanding of programming logic to the definition of recursive algorithms and data structure. The problems raised in this work point to the need for changes in the current practices of teaching and learning programming, combined with the use of tools and learning environments that facilitate the understanding of abstract concepts programming.

Keywords: Computer programming, teaching and learning programming, programming learning difficulties, disapproval, evasion.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ADS	Análise e Desenvolvimento de Sistemas
CC	Ciências da Computação
CLM	Campus Luiz Meneghel
DF	Distrito Federal
IES	Instituições de Ensino Superior
SBC	Sociedade Brasileira de Computação
SI	Sistemas de Informação
TI	Tecnologia da Informação
UENP	Universidade Estadual do Norte do Paraná

SUMÁRIO

1 INTRODUÇÃO

A programação de computadores (e demais disciplinas correlatas) envolve raciocínio lógico, resolução de problemas e capacidade de abstração tanto em uma representação formal quanto em uma linguagem computacional (JESUS e BRITO, 2009), porém, muitos estudantes nos cursos de tecnologia chegam despreparados ao ensino superior, apresentando deficiência relacionada ao raciocínio lógico, criatividade e abstração (MALTEMPI e VALENTE, 2000).

De acordo com Pimentel *et al.* (2003), a programação é uma das disciplinas mais importantes na formação dos futuros projetistas e desenvolvedores de software, sendo também a base para que os estudantes desenvolvam o raciocínio lógico e aprendam a resolver problemas computacionais. No entanto, o aprendizado desta disciplina é complexo e marcado por inúmeros desafios que são alvos de investigação no mundo todo, como por exemplo a baixa capacidade dos estudantes na resolução de problemas e suas dificuldades na abstração do conteúdo (BINI e KOSCIANSKI, 2000).

Outro ponto a ser destacado é que, diante dos desafios que os estudantes enfrentam no aprendizado de programação, estes acabam ficando desmotivados, constituindo um fator preponderante pela desistência do curso e contribuindo com o aumento do índice de evasão (GONÇALVES, *et al.* 2013). Consequentemente, disciplinas de algoritmos e programação possuem um dos maiores índices de reprovação em cursos da área de Tecnologia da Informação e Comunicação (TI) (DETERS, *et al.* 2008).

Este trabalho tem como objetivo investigar e analisar as principais dificuldades enfrentadas pelos estudantes nas disciplinas de programação de computadores e similares. Para isso, um estudo baseado em um survey é conduzido com estudantes e recém-formados nos cursos de TI, abrangendo diferentes regiões do Brasil.

Com base no estudo conduzido, espera-se obter uma melhor compreensão acerca dos problemas relacionados à aprendizagem de programação. A partir disso, evidenciar quais são as medidas que precisam ser tomadas para aprimorar e facilitar o processo de ensino e aprendizagem de programação, e por consequência, diminuir a taxa de evasão e reprovação nos cursos de tecnologia.

1.1 OBJETIVOS

O presente trabalho tem como objetivo geral compreender e analisar os problemas relacionados ao ensino e aprendizagem de disciplinas relacionadas à programação de

computadores, e a partir disso, fornecer subsídios para incentivar e promover mudanças que possam efetivamente melhorar esse processo. Espera-se com isso, contribuir com a melhoria da qualidade do ensino e aprendizagem de programação de computadores e com a diminuição da taxa de evasão em cursos de TI, como Sistemas de Informação e demais cursos correlatos.

1.1.1 Objetivos específicos

Para que o objetivo geral supracitado seja atendido, faz-se necessário o cumprimento dos seguintes objetivos específicos:

- Elaborar e aplicar um survey junto aos estudantes dos cursos de Sistema de Informação e Ciência da Computação da Universidade Estadual do Norte do Paraná (UENP), além de estender a aplicação em outras universidades espalhadas pelo Brasil;
- Identificar quais são as dificuldades enfrentadas pelos estudantes no aprendizado de programação de computadores;
- Descobrir possíveis problemas relacionados ao processo de ensino e aprendizagem de programação de computadores;
- Analisar qualitativa e quantitativamente os dados coletados por meio do survey;
- Identificar quais são as principais necessidades que precisam ser tratadas para a melhoria do processo de ensino e aprendizagem de programação de computadores.

1.2 JUSTIFICATIVA

A programação de computadores é essencial para todas as carreiras ligadas à Informática, sendo considerada uma das disciplinas mais importantes para aqueles estudantes que, após a conclusão do curso, pretendem trabalhar diretamente com o desenvolvimento de produtos de software (PIMENTEL, FRANÇA E OMAR, 2003). Apesar da disciplina de programação ser de extrema importância na formação dos graduandos dos cursos TI, um estudo realizado por Campos (2009) mostrou que mais de 60% dos estudantes são reprovados todos os semestres em tais disciplinas, ou em disciplinas correlatas.

Uma das consequências causadas pelas dificuldades encontradas pelos estudantes nas disciplinas de programação é a desmotivação, levando os cursos de exatas como os cursos de Ciência da Computação e Sistemas de Informação, a terem os mais altos índices de reprovação e evasão (BARBOSA, FERREIRA E COSTA, 2014).

A disciplina de programação é considerada de difícil entendimento para a maioria dos estudantes, por ser um processo que exige muito tempo e esforço de aprendizado (PIMENTEL,

FRANÇA e OMAR, 2003), e exige um alto nível de abstração. Dessa forma, o ensino e aprendizagem de programação ainda é um desafio recorrente em cursos de TI.

Apesar de existirem diversos estudos que tratam da problemática no ensino e aprendizagem de programação de computadores (e disciplinas correlatas), como Bini e Koscianski (2000), Gomes (2008), Campos (2009), Rosa e Giraffa (2011), Viegas *et al.* (2015), Souza, Batista e Barbosa (2016), dentre outros, ainda não está claro quais são os reais obstáculos que dificultam a aprendizagem dessa disciplina e como esses poderiam ser efetivamente solucionados ou minimizados.

Diante do exposto, este trabalho tem como principal objetivo identificar, compreender e analisar quais são os problemas e dificuldades enfrentados pelos estudantes na aprendizagem de disciplinas relacionadas à programação de computadores. A ideia é evidenciar quais são as necessidades que precisam ser tratadas no contexto de ensino e aprendizagem de programação de modo que medidas possam ser tomadas futuramente para mudar o cenário atual, e consequentemente contribuir com a diminuição do alto índice de reprovação e evasão.

1.3 METODOLOGIA

A pesquisa a ser realizada classifica-se como um estudo exploratório, pois tem como objetivo proporcionar maior familiaridade com a problemática e dificuldades encontradas pelos estudantes na aprendizagem de programação de computadores e disciplinas similares, em que o foco principal deste tipo de pesquisa está no aprimoramento de ideias ou de descoberta de intuições (GIL, 2002).

No que se refere aos procedimentos técnicos, este trabalho classifica-se como uma pesquisa bibliográfica (GIL, 2002) que é parte fundamental de qualquer trabalho científico. Tal classificação se deve ao fato de que a ideia é obter uma compreensão acerca das problemáticas relacionadas ao ensino e aprendizagem de programação por meio do estudo de trabalhos relevantes que abordam o tema em questão.

Este trabalho também é classificado como uma pesquisa não-experimental, consistindo de uma pesquisa de levantamento (Wazlawick, 2014) no qual os dados são adquiridos diretamente do ambiente por meio de um questionário. Nesse contexto, a pesquisa é caracterizada como um survey com o propósito de obter dados e informações relevantes por meio de uma amostra representativa de uma população.

A amostra deste survey é composta por estudantes e recém-formados dos cursos de TI,

abrangendo diversas regiões do Brasil. Os dados coletados por meio do survey são analisados qualitativa e quantitativamente. Qualitativa, pois os dados explorados são identificados e analisados para que assim seja possível compreendê-los e interpretá-los segundo a situação dos estudantes, sem se preocupar com representações numéricas. Quantitativa, já que é utilizado como instrumento de coleta de dados um questionário estruturado com questões objetivas, o qual será aplicado a uma população específica, com o objetivo de analisar e apresentar os dados obtidos de forma estatística (TERENCE e FILHO, 2006).

Atualmente, existem diversos softwares/ferramentas que apoiam às atividades de mineração de dados. Um deles é o Weka[1], um software livre que está sob a licença *General Public License* (GPL)[2] e que oferece um conjunto de algoritmos de aprendizado de máquina para a mineração de dados (DAMASCENO, 2010).

Devido às características e facilidades providas pelo Weka, neste trabalho utiliza-se tal ferramenta como apoio, a fim de ajudar na análise consolidada dos dados e alcançar resultados mais relevantes para a pesquisa.

Dentre os diversos algoritmos propostos pelo Weka, neste trabalho adota-se o algoritmo de classificação, em que os dados são apresentados na forma de árvores. Os graus de dificuldades apresentados nessa árvore foram os mais assinalados pelos participantes.

Na Figura 1 apresenta-se a estrutura da pesquisa, a metodologia utilizada e o propósito de cada etapa, conforme descrito a seguir.

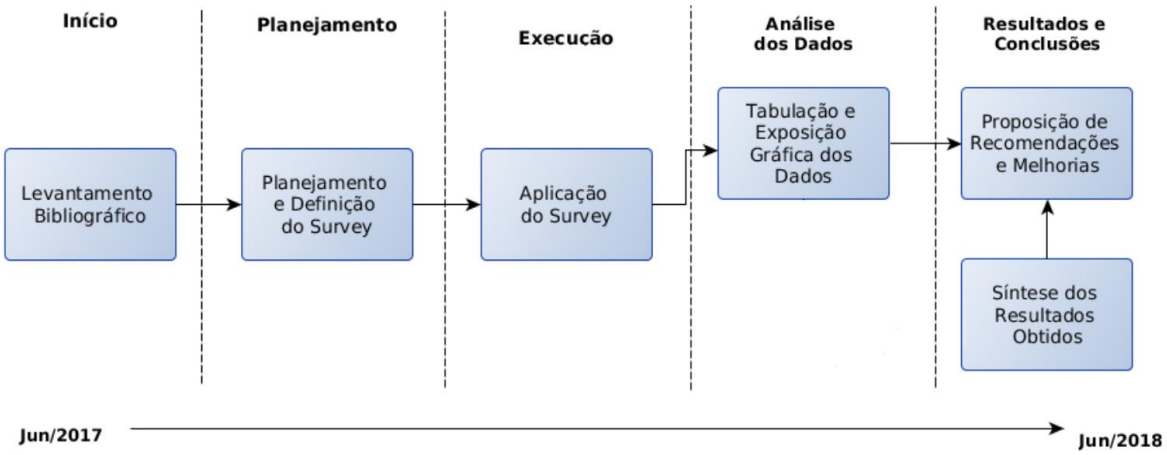


Figura 1: Estrutura da pesquisa.

Fonte: Elaborado pelo autor.

[1] https://www.cs.waikato.ac.nz/ml/weka/

[2] http://www.gnu.org/licenses/gpl.html

1. Início: levantamento bibliográfico de estudos que abordam as problemáticas relacionadas ao ensino e aprendizagem de programação e disciplinas similares;
2. Planejamento: planejamento e definição do survey englobando um questionário com perguntas diretas e objetivas com o intuito de identificar as principais dificuldades enfrentadas pelos estudantes durante o ensino e aprendizagem de programação;
3. Execução: aplicação do survey junto aos estudantes de graduação e cursos de tecnologias em diversas regiões do Brasil;
4. Análise dos dados: tabulação e exposição gráfica dos dados para facilitar a análise qualitativa e quantitativa dos resultados obtidos e melhorar a compreensão sobre o problema investigado;
5. Resultados e conclusões: síntese dos principais resultados obtidos por meio do survey; identificação das principais necessidades de mudanças/melhorias com base na análise realizada que possam contribuir efetivamente com o processo de ensino e aprendizagem de programação.

1.4 ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado da seguinte forma: na Seção 2 é apresentada a fundamentação teórica necessária para a compreensão e desenvolvimento do trabalho. Na Seção 3 apresenta-se o desenvolvimento do trabalho mostrando como foram realizadas as etapas de definição, planejamento e execução do survey. Na Seção 4 apresenta-se as análises realizadas sobre os dados obtidos com a execução do survey. Por fim, na Seção 5, são apresentadas as conclusões sobre o trabalho realizado, as limitações e perspectivas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo apresenta-se o embasamento teórico necessário para o entendimento e desenvolvimento do trabalho. O foco são os estudos que abordam o ensino e aprendizagem de programação (e disciplinas correlatas), sua a importância na formação dos estudantes, bem como a problemática inerente ao ensino e aprendizagem dessa disciplina que contribuem com o alto índice de evasão em cursos de TI.

2.1 Ensino e Aprendizagem de Programação

Sabe-se que o ensino e aprendizagem de programação de computadores é parte integrante dos currículos de referência da Sociedade Brasileira de Computação (SBC)[3].

Disciplinas relacionadas à programação de computadores, como por exemplo, a disciplina de algoritmos, possuem uma grande importância na formação dos estudantes, abordando princípios de lógica de programação que têm como objetivo desenvolver a capacidade de análise e de resolução de problemas (RAPKIEWICZ *et al*. 2006).

Segundo Gomes *et. al.* (2008), os métodos de ensino utilizados em disciplinas de programação não são adequados à maioria dos estudantes por razões que envolvem desde: (1) um ensino que não é personalizado, em que as estratégias dos professores não são capazes de contemplar todos os estudantes, considerando que cada pessoa aprende de uma forma e em ritmos diferentes; (2) e que uma parte dos professores se preocupam mais em ensinar uma linguagem de programação e se esquecem que precisam utilizar a linguagem de programação para promover nos estudantes a capacidade de resolução de problemas.

Santos e Costa (2006) afirmam que ensinar programação não é simplesmente ensinar uma determinada linguagem de programação, mas que o ensino desta envolve, sobretudo, entender problemas e descrever formas de resolução de maneira apropriada, para que então sejam codificadas em uma linguagem de programação. Ou seja, somente depois de haver o aprendizado dos conceitos de algoritmos e fundamentos de lógica, é que o estudante pode travar contato com uma linguagem de programação concreta para utilizar tais conceitos que precisam ser bem enfatizados tais como: atribuição, chamada de procedimento, envio de mensagens, passagem de parâmetros, herança, polimorfismo e encapsulamento.

[3] http://sbc.org.br/

Na programação, os conceitos não precisam ser decorados, mas sim compreendidos, pois não existe uma única forma de resolver um problema em programação, ou seja, não existem passos pré-definidos a serem seguidos, mas várias soluções que podem ser obtidas, e por fim chegar a um mesmo resultado (MALTEMPI e VALENTE, 2000).

De acordo com Maltempi e Valente (2000), o aprendizado de programação precisa passar por um ciclo que envolve descrição-execução-reflexão-depuração, e essas atividades são consideradas árduas e demandam um grande esforço e concentração dos estudantes. Porém, os estudantes falham ao passar por esses processos, e acabam entrando em um ciclo em que escrevem o programa e compilam a procura de erros, fazendo alterações muitas vezes sem sentido, pois não realizam uma reflexão sobre o que precisa ser feito. Assim, os estudantes gastam tempo e esforço, tornando a programação "uma atividade pobre em termos de aprendizagem" (MALTEMPI e VALENTE, 2000).

No estudo conduzido por Campos (2009), docentes de disciplinas de programação de 4 instituições de ensino superior foram entrevistados, com o intuito de identificar a abordagem utilizada por eles no ensino da programação. Como resultado, constatou-se que mais de 90% dos docentes adotam como método de ensino a construção da solução para resolver o problema proposto.

Diante das dificuldades dos estudantes com o método de ensino supracitado, Campos (2009) propôs a criação de uma nova estratégia de ensino para lógica de programação, conhecida como metodologia ERMC² ou ERM2C. Tal metodologia enfatiza que, antes do estudante conseguir desenvolver um algoritmo, ele precisa primeiro ter a capacidade de ler e entender qualquer algoritmo, sendo seu ou de terceiros.

A metodologia ERMC² é dividida em 5 etapas: Entender, Revisar, Melhorar, Complementar e Construir. Essas etapas foram pensadas para que o estudante possa construir o conhecimento passo a passo, entendendo o que precisa ser feito (CAMPOS, 2009).

Amaral *et al.* (2015), chamam a atenção para a importância de inserir o ensino de programação para estudantes do ensino médio, pois dessa maneira, os estudantes terão contato com a programação mais cedo, podendo chegar à universidade mais capacitados. Além de favorecer o desenvolvimento do raciocínio lógico e da criatividade, a programação também pode ser uma estratégia para aumentar o interesse dos estudantes pelas áreas de engenharia e TI.

2.2 Dificuldades para o Aprendizado de Programação

Segundo Gomes e Mendes (2007), para muitos estudantes a dificuldade referente ao aprendizado da programação começa na fase inicial de aprendizagem, quando eles precisam entender e aplicar conceitos abstratos de programação.

Assim, se um conceito não é totalmente compreendido pelo estudante, este não consegue entender os próximos conteúdos abordados, pois, para que um novo conhecimento de programação possa ser aprendido, é necessário haver uma assimilação de conhecimentos anteriores, ou seja, esta disciplina possui um conteúdo cumulativo em que um novo conhecimento necessita de um já existente. Além dessa assimilação de conteúdos, os estudantes precisam abstrair os conceitos de programação conseguindo visualizar o que precisa ser feito antes e durante a construção do código fonte, o que muitas vezes se torna difícil para a maioria deles (VIEGAS, *et al.* 2015).

Essa dificuldade, segundo a afirmação de Martins (2015) dá-se, pelo fato de que os "Algoritmos, geralmente são assuntos de complexo entendimento para os estudantes dos cursos da área de Informática, primeiramente por se tratar de algo nunca visto pelos mesmos [...]".

Seguindo essa linha, a programação é um processo difícil para a maioria dos estudantes, já boa parte destes apresentam grande dificuldade em compreender e aplicar conceitos, como recursão, passagem de parâmetros, estruturas de seleção e repetição aprendidos na disciplina introdutória de programação (PIMENTEL *et al.* 2003).

Gomes *et al.* (2008), apontam que em diversos estudos referentes às dificuldades no aprendizado da programação tem-se como principais motivos a dificuldade de compreensão e aplicação de noções básicas para se conseguir criar algoritmos que resolvam problemas concretos.

Muitos dos estudantes possuem dificuldades em interpretar um texto ou expressar ideias de forma lógica, o que acaba tornando um grande problema no aprendizado de programação, levando em conta que para a resolução de um determinado problema em questão, o estudante necessita interpretar e conseguir compreender o que foi proposto (SCOLARI, BERNARD e CORDENONSI, 2007).

Um fator gerado por essa dificuldade de compreensão, leva os estudantes a decorarem o que precisa ser feito, dessa maneira não se cria uma aprendizagem, mas sim, após a utilização

do conteúdo decorado os estudantes acabam esquecendo tudo que já foi visto, e quando precisam aplicá-los novamente, encontram grande dificuldade e muitas vezes não o conseguem fazer (SCOLARI, BERNARD e CORDENONSI, 2007).

Diante dos fatos citados, pode-se evidenciar a necessidade de estimular o desenvolvimento do raciocínio lógico ainda nas escolas, onde os adolescentes com idade entre 12 e 15 anos estão começando a desenvolver ideias abstratas. E nota-se que a utilização dos jogos educacionais podem contribuir de forma motivadora para desenvolvimento do raciocínio lógico dos estudantes (SCOLARI, BERNARD e CORDENONSI, 2007).

2.3 Problemas e Consequências

Com o rápido avanço da tecnologia, a cada dia mais profissionais capacitados nas áreas de engenharia e tecnologia fazem-se necessários, porém, a formação desses profissionais não segue o mesmo ritmo (VIEL, RAABE e ZEFERINO, 2014).

As disciplinas de programação e correlatas, possuem a reputação de serem difíceis, e essa visão negativa é passada de estudante para estudante, o que leva estes a ficarem desmotivados a estudar e compreender essas disciplinas (GOMES *et. al*. 2008).

Santos e Costas (2006), citam alguns problemas relacionados ao ensino e aprendizagem de programação que vão desde à falta de motivação dos estudantes, podendo ocorrer devido suas dificuldades no aprendizado, até pela didática utilizada no ensino dessa disciplina. Tais dificuldades acabam tornando uma das causas de desinteresse de muitos estudantes pelo curso, além de ser uma das responsáveis pelas altas taxas de desistência já no primeiro semestre (SANTIAGO; KRONBAUER, 2016). Esse grande número de reprovação ocasiona atraso nos estudos e consequentemente aumenta o índice de evasão desses cursos (BARBOSA; FERREIRA; COSTA, 2014).

O número de estudantes que conseguem concluir os cursos de TI são consideravelmente inferior ao número de entrada nesses cursos, isso devido aos altos níveis de reprovação e evasão presentes nas disciplinas de programação e similares (JÚNIOR, *et al.* 2014).

Durante 8 semestres, uma pesquisa foi realizada em 4 instituições de ensino superior no Distrito Federal (DF) envolvendo estudantes de disciplinas de lógica de programação. Pode-se constatar que pelos menos 60% dos estudantes eram reprovados a cada semestre nas disciplinas, e esses 60% eram divididos em 37% para os estudantes novatos e 23% para os

estudantes repetentes (CAMPOS, 2010).

Segundo Raabe e Silva (2014), esse elevado número de reprovações e desistências são muitas vezes ocasionados porque as dificuldades que os estudantes possuem no aprendizado não são descobertas a tempo.

De acordo com estudos realizados por Kantorski *et al.* (2016) e Pascoal *et al.* (2016), foi possível verificar que os altos índices de evasão nas Instituições de Ensino Superior (IES), tanto públicas como privadas são evidentes, e dentre estes, os cursos de Computação são considerados um dos que possuem os mais altos índices de reprovação e desistências, sendo as dificuldades enfrentadas pelos estudantes enfrentam no aprendizado das disciplinas de programação e lógica um dos principais fatores.

A evasão nos cursos de graduação é considerada um problema relevante para a gestão acadêmica e financeira das IES, pois são investidos recursos na atração e captação de estudantes e os resultados têm sido negativos devido às taxas elevadas de evasão (PASCOAL, *et al.* 2016).

No estudo conduzido por Fernandes e Junior (2016) no Instituto Federal Catarinense, foram analisadas as turmas de 2010 a 2015 nas disciplinas de Linguagem de Programação 1 e Linguagem de Programação 2 com o intuito de verificar as taxas de aprovação, desistência e reprovação nas disciplinas de programação. Num total de 410 participantes, os dados obtidos foram que 58% dos estudantes obtiveram êxito, totalizando 239 aprovados, enquanto 104 estudantes desistiram das disciplinas e outros 67 reprovaram, respectivamente (Figura 2).

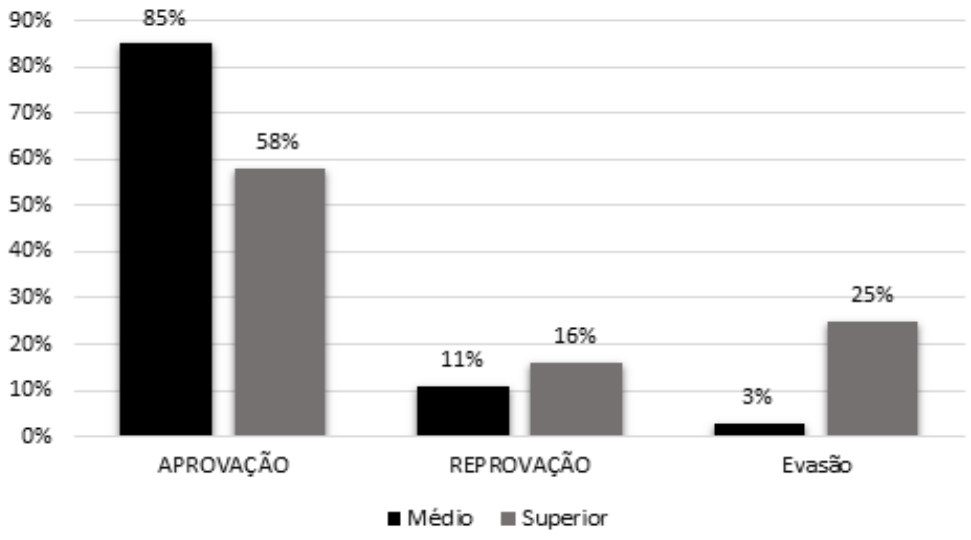


Figura 2: Desempenho dos estudantes.

Fonte: Fernandes e Junior, 2016.

Segundo Silva, Silva, e Santos (2009), as disciplinas de programação e correlatas têm apresentado altos índices de evasão e reprovação. Ainda segundo os autores “a evasão, além do afastamento dos estudantes do seu objetivo principal (a formação intelectual e profissional), gera desconfiança na comunidade acadêmica quanto à qualidade de tais cursos superiores, impedindo a entrada de novos estudantes e retardando o crescimento da área de Computação.”

2.4 Ferramentas de apoio à Aprendizagem de Programação

Ao longo dos anos, diversas ferramentas e ambientes de aprendizagem vêm sendo propostas com o objetivo de facilitar e instigar o aprendizado de lógica e programação de computadores, tais como: mini linguagens, mundos programáveis, ambientes de desenvolvimento controlados, ferramentas de animação, entre outras (PIMENTEL, FRANÇA e OMAR, 2003).

Rosa e Giraffa (2011) afirmam ser essencial o desenvolvimento de ferramentas que busquem despertar interesse nos estudantes como também facilitar o entendimento da lógica de programação, pois o aprendizado de programação é um desafio que grande parte dos estudantes possuem.

Ramos *et al*. (2015) conduziram uma revisão sistemática da literatura, que teve como objetivo verificar artefatos que influenciam nas taxas de sucesso e reprovação nas disciplinas de programação e correlatas. A partir do estudo conduzido, foi possível verificar que a utilização de ferramentas aliadas às metodologias de ensino (por exemplo, o uso de jogos como um fator de motivação para o ensino e aprendizagem) influenciou para que a taxa de reprovação nas disciplinas de programação tivessem uma redução significativa. Os autores citam que essa redução nas reprovações no caso estudado foi de 20% para os estudantes repetentes e 50% para os novatos, após o uso dos jogos como ferramenta de ensino.

No entanto, uma ferramenta computacional precisa ser bem elaborada para que torne o conteúdo abordado mais prático e atraente, a fim de despertar maior interesse nos estudantes (SANTOS e COSTA, 2006).

Viegas *et al*. (2015) apresentam o *Scratch*, uma ferramenta na qual é possível ensinar programação comparando a uma aprendizagem de programação tradicional utilizando a linguagem C. Com esta ferramenta os estudantes conseguem desenvolver seu raciocínio lógico, capacidade de abstração e criatividade mais facilmente por esta ser uma ferramenta

lúdica de ensino.

De forma análoga, *Alice* (VIEGAS, *et al*. 2015) é uma ferramenta que permite criação de animação e mundos virtuais de forma simplificada, em que os comandos são realizados quando arrastados No entanto, segundo os autores essa ferramenta é útil para a introdução de conceitos de programação, mas necessita ser combinada com a programação JAVA para ter uma aprendizagem mais sólida.

Outra ferramenta é o *Greenfoot* que permite tornar a aprendizagem de JAVA mais fácil, pois esta proporciona um ambiente que remove parte da complexidade associada à linguagem de programação orientada a objetos (VIEGAS, *et al*. 2015).

Entretanto, entende-se que essas ferramentas não substituem os modelos atuais de ensino de programação, mas auxiliam no processo de ensino e aprendizagem desta, facilitando a compreensão dos estudantes principalmente na introdução dos conceitos de programação (VIEGAS, *et al*. 2015).

Pode-se citar também o *Robocode*, que é um ambiente lúdico que facilita o processo de ensino aprendizagem de programação, em que os estudantes podem aprender através de um jogo, que é um tipo de metodologia que vem cada vez mais se mostrando promissora (AMARAL, et al. 2015).

Medeiros, Silva e Aranha (2013), citam que os jogos digitais são elementos motivadores no processo de ensino e aprendizagem de programação, visto que as dificuldades que os estudantes enfrentavam no aprendizado podem ser minimizadas com uso desses jogos.

Além disso, a linguagem LOGO pode ser considerada uma forma de facilitar o entendimento de alguns conceitos introdutórios de programação além de auxiliar o estudante na aprendizagem desta disciplina. Os conceitos introdutórios usados em linguagens como PASCAL, JAVA, C e C++, também podem ser implementadas na linguagem LOGO, o que muda é o vocabulário, o ambiente e a forma de representação das ideias, porém esta linguagem não deixa nada a desejar comparada a outras linguagens, pelo contrário, ela possibilita uma maior compreensão e aprendizagem devido a sua simplicidade e seu aspecto gráfico que permite uma rápida visualização de um determinado conceito (LIMA e LEAL, 2010).

A linguagem LOGO "é uma linguagem de programação que foi concebida para ser inteligível e de fácil acesso para iniciantes" (LIMA e LEAL, 2010). Com uma interface simples e com uma janela de comandos o usuário pode escrever os códigos, e observando a tartaruga que se desloca de acordo com os comandos efetuados, é possível aprender de forma simples,

dessa maneira o usuário consegue obter um feedback e refletir sobre os comandos utilizados (LIMA e LEAL, 2010).

Como visto, a programação é uma disciplina que requer prática, além de criatividade e raciocínio lógico, e em uma turma inicial de programação que geralmente possui um grande número de estudantes, o professor tem dificuldade em verificar e acompanhar o aprendizado de todos. A ferramenta *The Huxley* insere-se nesse contexto, tendo como principal objetivo viabilizar a melhoria na formação dos estudantes sem perder o controle de evolução do aprendizado individual (PAES, *et al.* 2013).

A ferramenta *The Huxley* permite que os estudantes submetam códigos em diversas linguagens de programação, e para cada submissão os estudantes recebem um *feedback* da correção automática pelo sistema (PAES, *et al*. 2013). Com o uso da ferramenta, o professor passa a ter uma visão mais autêntica do desempenho de cada estudante, pois ele consegue saber a quantidade de problemas resolvidos, a porcentagem de acertos e erros, os tipos de problemas com mais erros, além de detectar plágio e erros específicos de cada estudante (PAES, *et al*. 2013).

Ainda segundo Paes, *et al*. (2013), os resultados da aplicação desta ferramenta indicou que ela é efetiva no auxílio ao aprendizado, enquanto viabiliza o acompanhamento individualizado do desempenho de muitos estudantes por parte do professor.

Run.codes é uma ferramenta desenvolvida por Felipe Duarte e Fábio Sikansi, alunos do Instituto de Ciências Matemáticas e de Computação (ICMC) da Universidade de São Paulo (USP), em São Carlos. Esta ferramenta possibilita que os estudantes cadastrem on-line os trabalhos de programação realizados e aguarde alguns segundos até que o resultado da correção apareça. “Nossa ideia foi fazer um sistema eficaz para que o professor pudesse gerir efetivamente o trabalho realizado em uma sala de aula”, explicou Duarte, doutorando do ICMC (CAVALCANTI, 2015).

3 DESENVOLVIMENTO

Neste capítulo descrevem-se os procedimentos realizados para o planejamento e definição do survey e sua respectiva aplicação junto ao público-alvo da pesquisa.

3.1 Planejamento

Neste trabalho, a escolha pela condução de um estudo baseado em survey se deu pelo fato de ser uma pesquisa em grande escala, que abrange estudantes de diversos cursos da área de TI dispersos em várias regiões do Brasil. Com isso, o propósito foi o de obter o maior número possível de respostas acerca dos problemas e dificuldades que os estudantes enfrentam nesses cursos no que tange às disciplinas de programação de computadores.

Assim, durante a delimitação do público-alvo da pesquisa foram escolhidos os cursos técnicos/superiores da área de TI que possuem em sua grade curricular, disciplinas de programação de computadores e similares como algoritmos e estrutura de dados. A amostra da população foi composta por os estudantes e recém-formados na área em questão.

Como instrumento de coleta de dados foi proposto um questionário on-line com 38 questões estruturadas (APÊNDICE A). O questionário foi composto por questões “fechadas” e “abertas”. A maioria das questões fechadas foram definidas de acordo com a escala de Likert (LIKERT, 1932) no qual cada participante do survey específica o grau de dificuldade ou nível de concordância em relação a uma determinada questão ou item investigado.

Ainda sobre as questões fechadas, também foram definidas questões objetivas no qual cada participante escolhe uma ou mais opções dentre duas ou várias opções existentes, tais como tipo de instituição (pública ou privada), tipo de curso (bacharelado, licenciatura, técnico ou tecnólogo), período do curso (diurno, noturno ou integral), dentre outras.

Já em relação às questões abertas, elas foram definidas sobretudo para coletar dados adicionais e de suma importância para a pesquisa, tais como as linguagens de programação mais conhecidas pelos participantes, linguagens de programação que apresentam mais dificuldade de aprendizado, etc. Além disso, foram definidas questões abertas, consistindo de um espaço para que os participantes pudessem sugerir mudanças e melhorias necessárias ao processo de ensino e aprendizagem de programação.

Para apoiar na elaboração do questionário, foi utilizado o *Google Forms*, uma ferramenta gratuita para criação de questionários on-line. Utilizou-se dessa ferramenta on-line como forma de facilitar a distribuição do questionário e alcançar o maior número de participantes nas diversas regiões do Brasil.

Inicialmente, foi elaborado uma versão preliminar do questionário, o qual foi enviado para professores e especialistas da área para análise e validação. O feedback obtido e as sugestões de correções e melhorias identificadas foram levadas em consideração na elaboração da versão final do questionário, submetida aos participantes da pesquisa.

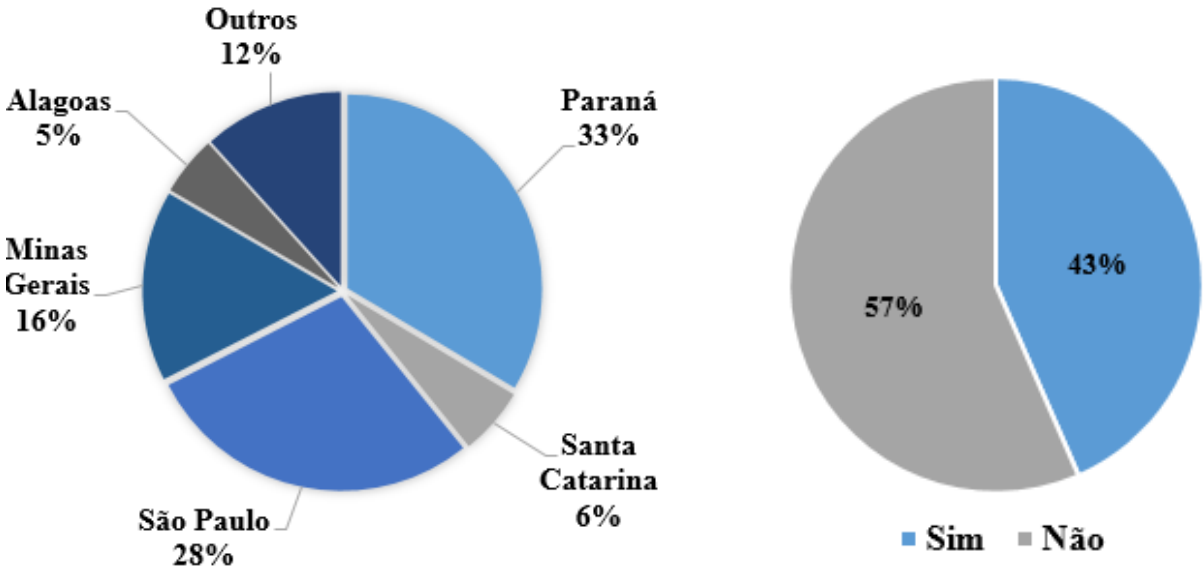
3.2 Execução

Para a aplicação do survey, foi disponibilizado um link[4] com o questionário em listas de e-mails, páginas e grupos de discussão nas redes sociais (como o *Facebook*) relacionados à área de pesquisa. Além disso, foram enviados e-mails para professores de instituições de ensino técnico/superior com informações da pesquisa, solicitando a participação de seus alunos. O questionário foi aplicado no período de Novembro/2017 a Março/2018. Ao final da pesquisa, foram obtidas 284 respostas.

Conforme citado anteriormente, a amostra deste survey foi composta por estudantes da área de TI de diferentes regiões do Brasil que cursam ou já cursaram alguma disciplina relacionada à programação de computadores. Pode-se observar no gráfico da Figura 3 (a) que a maior taxa de respostas vieram dos estados do Paraná (33%), São Paulo (28%) e Minas gerais (16%). Um número consideravelmente pequeno de respondentes de outros estados também foram contabilizados, tais como Rio de Janeiro, Espírito Santo, Amazonas, Brasília, Rio Grande do Sul, Bahia e Maceió (12%).

Buscando conhecer um pouco mais do nível de conhecimento/experiência dos participantes, estes foram questionados se já haviam tido contato com a programação, ou alguma disciplina similar a ela. As respostas encontram-se no gráfico da Figura 3 (b), em que o resultado mostra que mais da metade, ou seja, 57% dos estudantes não tiveram contato com a programação antes dos cursos realizados no momento da pesquisa.

[4] https://goo.gl/forms/XtdiFWMgypsz0nla2



(a): Regiões da pesquisa. (b): Conhecimento prévio em programação.

Figura 3: Regiões da pesquisa versus conhecimento prévio de programação dos participantes.

Fonte: Dados da pesquisa.

O perfil dos respondentes deste questionário são de estudantes graduandos ou recém-graduados nos cursos de TI. Na Tabela 2 apresenta-se um panorama geral do perfil dos respondentes, incluindo a faixa etária dos estudantes, a qual foi utilizada para traçar um perfil com o intuito de observar se a idade poderia influenciar positiva ou negativamente na aprendizagem de programação. A mesma análise foi feita considerando o sexo dos participantes para verificar se a diferença no sexo (masculino ou feminino) poderia exercer alguma influência na aprendizagem.

Neste panorama também apresentam-se: (1) a categoria dos cursos, ou seja, “bacharelado”, “licenciatura”, “tecnólogo” e “cursos técnicos”, (2) o tipo da instituição que os estudantes frequentam, ou no caso dos recém-formados, frequentavam, ou seja, “públicas” ou “privadas”; (3) e por fim, o período do curso, ou seja, “diurno”, “noturno” ou “integral”.

Faixa etária	***Números***	***%***
17 a 20 anos	108	38%
21 a 24 anos	99	35%
25 a 28 anos	34	12%
Acima de 29 anos	43	15%
Sexo		
Masculino	218	77%
Feminino	66	23%
Categoria		
Bacharelado	239	84%
Licenciatura	17	6%
Cursos Técnicos	4	1%
Tecnólogo	24	9%
Instituição		
Pública	239	84%
Privada	45	16%
Período		
Noturno	182	64%
Diurno	20	7%
Integral	82	29%

Tabela 1: Panorama geral do perfil dos respondentes.

Fonte: Dados da pesquisa.

De acordo com o panorama apresentado na Tabela 2, é possível observar que a maior parte dos estudantes frequentam cursos do período noturno. Complementando a Tabela 2, um panorama mostrando se os estudantes somente estudam ou precisam trabalhar durante a realização do curso é apresentado na Figura 4. De acordo com o gráfico da Figura 4, observa-se que 62% dos estudantes estudam e trabalham, enquanto somente 38% só estudam. Isso mostra a razão pela qual muitos estudantes são oriundos dos cursos noturnos.

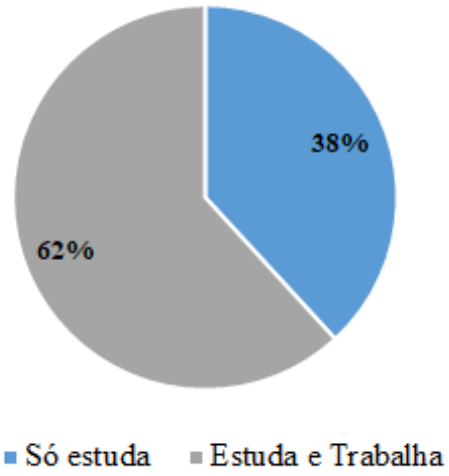


Figura 4: Estudantes que só estudam e os que estudam e trabalham.

Fonte: Dados da pesquisa.

4 ANÁLISE

Neste capítulo, apresenta-se uma análise detalhada dos dados obtidos com o desenvolvimento do survey, além de uma síntese dos principais resultados obtidos.

4.1 Visão Geral

Na Tabela 3 encontram-se os itens relacionados às questões envolvendo conceitos e práticas de ensino abordados no questionário do survey e o grau de dificuldade (%) dos estudantes em relação a estes itens.

Grau de Dificuldade	**1**	**2**	**3**	**4**	**5**
Lógica de programação	20%	29%	29%	16%	6%
Definição e uso de variáveis	41%	28%	22%	6%	3%
Definição e uso de estruturas de decisão	34%	26%	22%	13%	5%
Definição e uso de estruturas de repetição	30%	30%	23%	13%	4%
Definição e uso de funções e métodos	23%	23%	25%	19%	10%
Passagem de parâmetro (valor e referência)	20%	16%	25%	22%	17%
Definição e uso de arrays	22%	20%	24%	18%	16%
Manipulação de Strings	20%	21%	28%	19%	12%
Definição e uso de ponteiros	11%	14%	19%	22%	34%
Definição e uso de algoritmos recursivos	11%	12%	26%	24%	27%
Definição e uso de estruturas de dados	10%	17%	24%	22%	27%

Tabela 2: Panorama geral das dificuldades dos conceitos e práticas de ensino.

Fonte: Dados da pesquisa.

Pode-se verificar na Tabela 3 que as dificuldades enfrentadas pelos estudantes vão desde o entendimento da lógica de programação até a definição e uso de estruturas de dados. Analisando a tabela, nota-se que dos 284 respondentes somente 20%, ou seja, 56 estudantes

disseram não possuir nenhum tipo de dificuldade em relação à lógica de programação, enquanto outros 247 estudantes (80%) possuem alguma dificuldade neste quesito.

As mesmas verificações foram feitas para: (1) definição e uso de variáveis, (2) definição e uso de estruturas de decisão e (3) definição e uso de estruturas de repetição. Nesses quesitos, mais de 180 estudantes, ou seja, pouco mais de 60% disseram que possuem algum grau de dificuldade, ainda que seja considerado "muito baixo".

Nas questões que abordam: (1) definição e uso de funções e métodos, (2) passagem de parâmetro (valor e referência), (3) definição e uso de *arrays* e (4) manipulação de *strings*, nota-se que o número de respondentes que possuem um grau dificuldade considerado "muito alto" varia de 10% a 17%, correspondendo a 28 e 49 estudantes, respectivamente. Porém, ao verificar todas as respostas dos estudantes nessas questões, nota-se que mais de 240 dos estudantes (80%) dizem possuir algum grau de dificuldade, ainda que seja considerado "muito baixo".

Pode-se constatar um índice maior de dificuldade nas questões que abordam: (1) definição e uso de ponteiros, (2) definição e uso de algoritmos recursivos e (3) definição e uso de estruturas de dados. Nesses três itens a porcentagem de respondentes que possuem um grau de dificuldade "muito alto" é de 27% para definição e uso de algoritmos recursivos e também para definição e uso de estruturas de dados. Já para o uso de ponteiros, o grau de dificuldade "muito alto" chega a 34% dos respondentes, o que passa de 90 estudantes. Nesses três itens, menos de 30 dos estudantes (10%) disseram não possuir "nenhuma dificuldade", enquanto mais de 260 estudantes (90%) possuem algum grau de dificuldade, ainda que seja considerado "muito baixo".

4.2 Análise dos Dados

As análises foram realizadas entre o perfil dos participantes apresentado na Tabela 2, junto aos conceitos e práticas de ensino apresentados na Tabela 3. Buscou-se identificar com essa análise as dificuldades encontradas pelos participantes, em que foi verificado se o período do curso poderia influenciar de alguma forma nas dificuldades que os estudantes encontram no aprendizado de programação. O mesmo foi realizado entre o sexo dos participantes, e também em relação aos que trabalham ou só estudam.

Nesta etapa, também buscou-se identificar se algum ambiente ou ferramenta de aprendizagem era utilizado para facilitar a abstração dos conceitos de programação, tornando os exemplos mais fáceis de serem abstraídos.

As análises foram feitas junto às questões na escala de Likert, que analisavam as dificuldades que os estudantes possuíam em determinados conceitos e práticas de programação. A questões na escala de Likert foram estruturadas de forma que as respostas variam de 1 a 5, sendo 1- nenhuma dificuldade, 2- muito baixa, 3- baixa, 4- alta e 5- muito alta.

4.2.1 Análise por Faixa Etária

Para realizar esta análise foi feito um agrupamento das idades dos estudantes respondentes, criando faixa etárias para facilitar a análise desses dados e ajudar na visualização. Agrupou-se as idades dos estudantes em quatro grupos de faixa etária: dezessete a vinte anos (17-20), (2) vinte e um a vinte e quatro anos (21-24), (3) vinte e cinco a vinte e oito anos (25-26) e (4) acima de vinte e nove anos (29-50).

Na Figura 5 apresenta-se uma análise dos níveis de dificuldade dos estudantes em relação à **definição e uso de estrutura de dados**, levando-se em consideração (1) a faixa etária dos estudantes, (2) se durante o curso eles só estudam ou também trabalham e (3) o tipo de instituição (pública ou privada). Na análise, foi utilizado o algoritmo de classificação proposto pelo Weka, no qual os dados são estruturado na forma de árvores de decisão.

Em cada verificação feita na Figura 5 são apresentados três valores. O primeiro valor apresentado é o grau de dificuldade mais assinalado pelos estudantes; em seguida são indicados dois valores entre parênteses, o primeiro valor é o número total de respondentes naquela categoria, enquanto o segundo apresenta o número de respondentes que assinalaram o grau de dificuldade apresentado na ramificação.

Na primeira análise da Figura 5 encontra-se a faixa etária dos participantes, que entre os estudantes de dezessete e vinte anos (17-20), foi identificado o grau de dificuldade “5- muito alto”, seguido dos valores 107 que corresponde ao total de estudantes na categoria e 34 que são os estudantes que disseram possuir um grau de dificuldade “5- muito alto”, (valores entre parênteses).

O grau de dificuldade apresentado é “5”, pois este foi o valor mais assinalado pelos participantes daquela categoria, ou seja 34 dos 108 estudantes possuem um grau de dificuldade

“muito alto” em estrutura de dados. O restante dos participantes assinalaram outros graus de dificuldade.

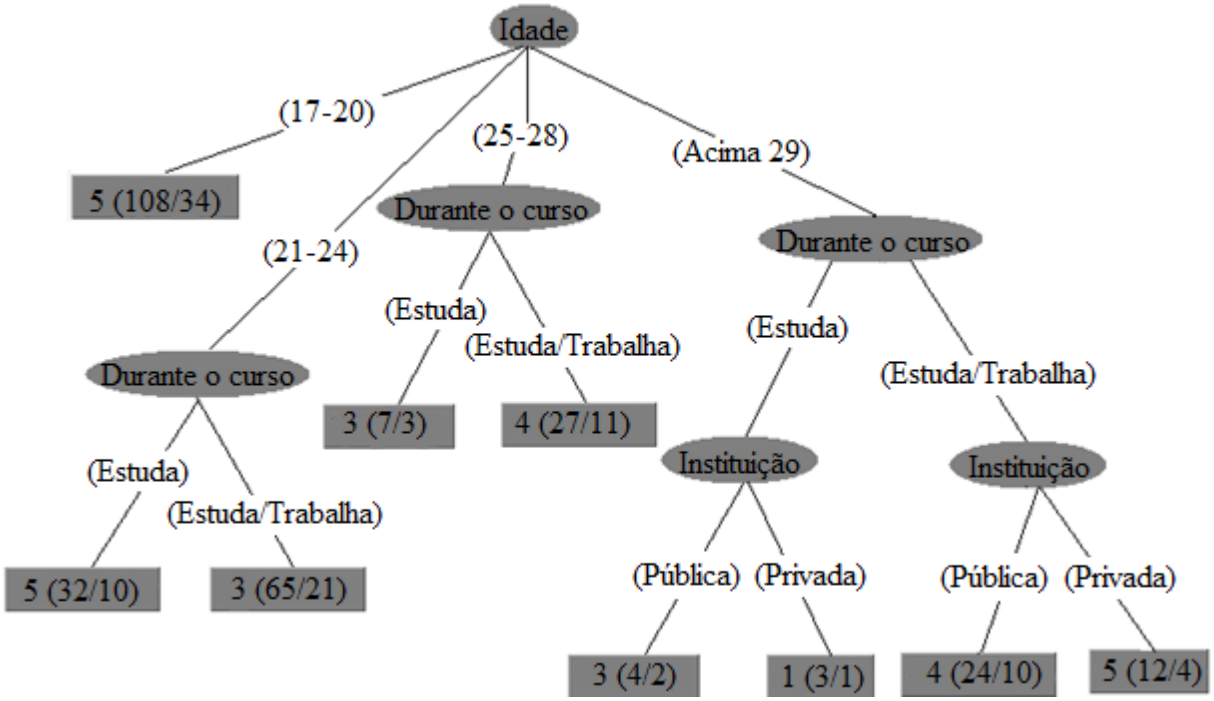


Figura 5: Análise das dificuldades em relação a Estrutura de Dados.

Fonte: Dados da pesquisa.

Ao analisar os estudantes de faixa etária acima de vinte e nove anos (29-50), nota-se que foram realizadas outras análises, considerando os estudantes que só estudam e aqueles que estudam e trabalham, bem como o tipo de instituição. Percebe-se uma grande diferença em relação a dificuldade dos estudantes que só estudam, para aqueles que precisam estudar e trabalhar. Nesse contexto, para os estudantes que só estudam os graus de dificuldade apresentados são “1 - nenhum” e “3 - baixo”, enquanto que para os estudantes que estudam e trabalham, os graus de dificuldade correspondem a “4 - alto” e “5 - muito alto”. Esses resultados “podem sugerir” que o estudantes que trabalham durante o curso tem menos tempo para se dedicar aos estudos e, por consequência, acabam enfrentando mais dificuldades no aprendizado.

Pode-se verificar na Figura 5 que a faixa etária de (17-20) anos não apresenta ramificações das análises para “Durante o curso” e tipo de “Instituição”. Isso ocorre pelo fato, do algoritmo de classificação buscar os valores que aparecem mais vezes, dessa maneira, caso as análises de “Durante o curso” e tipos de “Instituição” fossem realizadas o grau de dificuldade encontrado seria o mesmo já apresentado na faixa etária, “5- muito alto”.

4.2.2 Análise por Regiões da Pesquisa

Ao analisar-se as cidades onde moram os respondentes do questionário, foi possível dividi-las em regiões, e dessa maneira verificar se entre os estados presentes nesta pesquisa haveria grandes divergências em como os estudantes lidam com o aprendizado da programação. O campo “outras” na Figura 6 apresenta estados onde o número de respondentes foi consideravelmente pequeno, o que levou a agrupar essas informações.

Na Figura 6 é apresentado uma análise do grau de dificuldade dos estudantes em relação à **definição e uso de estrutura de dados,** em que foi levado em consideração (1) as regiões da pesquisa e (2) o sexo dos participantes. Nesta análise também foi utilizado o algoritmo de classificação.

Na análise da Figura 6 pode-se observar que o maior grau de dificuldade apresentado (5- muito alto) vem da região Sul, esse fator pode ser levado em conta considerando que a maior parte dos estudantes dessa região estudam e trabalham, e frequentam o período noturno do curso. O mesmo se aplica aos participantes da região Nordeste. Quando verifica-se o sexo dos participantes, pode-se observar que os participantes do sexo feminino possuem um grau de dificuldade maior (3- baixa) comparado aos participantes do sexo masculino, que assinalaram mais vezes um grau de dificuldade considerado menor (2- muito baixa).

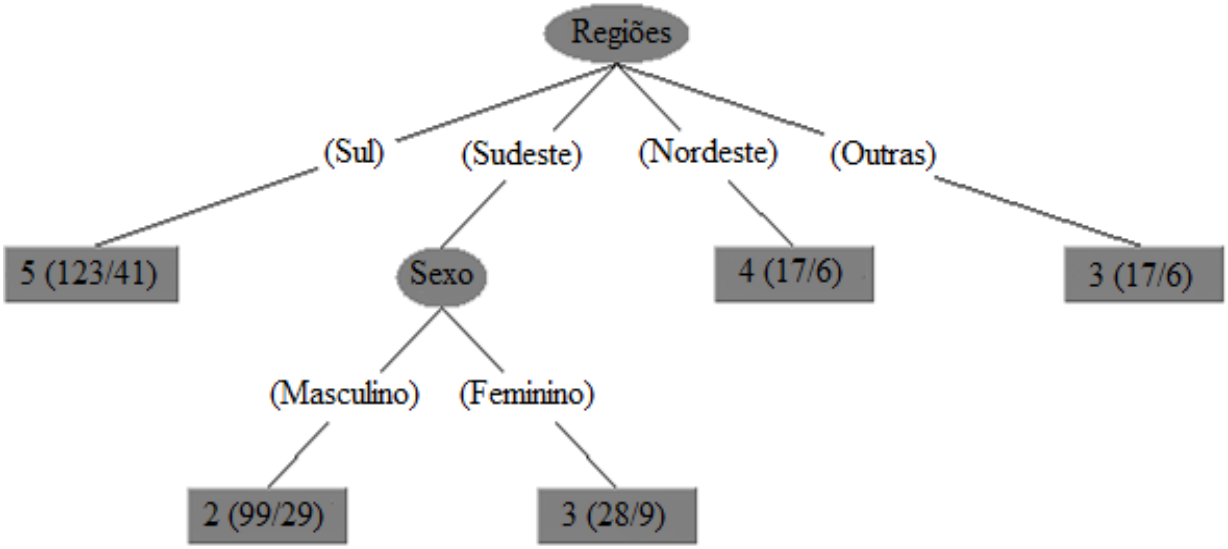


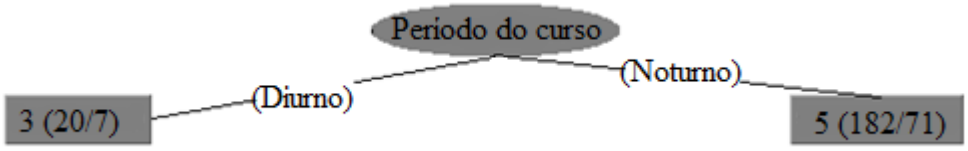
Figura 6: Regiões e sexo dos respondentes e o grau de dificuldade em Estrutura de Dados.

Fonte: Dados da pesquisa.

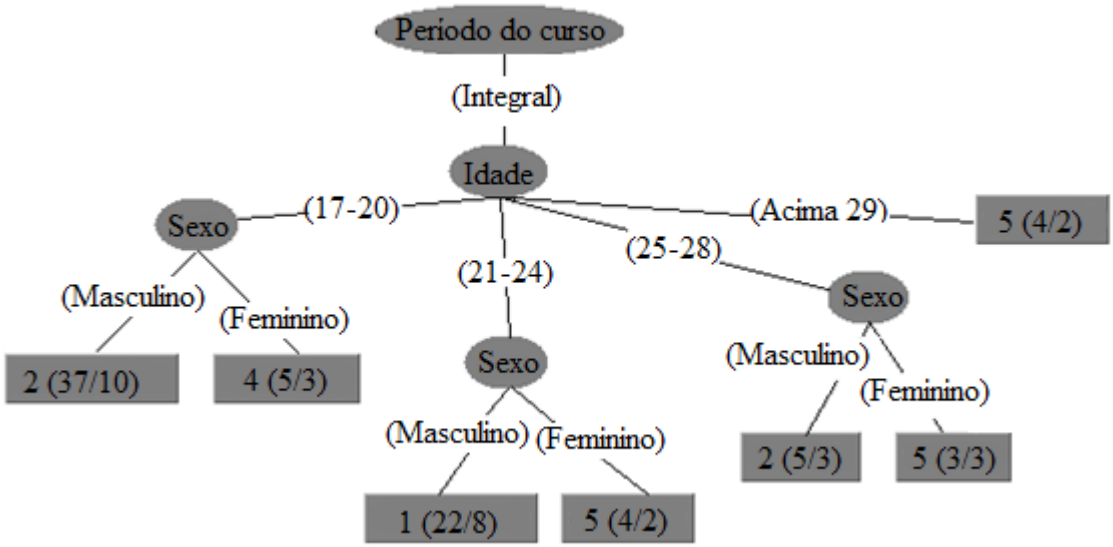
4.2.3 Análise por Período do Curso

Na Figura 7 (a) e (b) apresenta-se uma análise do nível de dificuldade dos estudantes em relação à **definição e uso de ponteiros** usando o algoritmo de classificação, levando-se em consideração (1) o período do curso, (2) a faixa etária e (3) o sexo dos estudantes. Pode-se observar na Figura 7 (a), que os estudantes do período noturno apresentam um grau de dificuldade maior comparado aos estudantes do período diurno. Isso pode ter relação com o tempo dedicado ao estudo, já que a maioria dos estudantes do período noturno trabalham durante o dia.

Considerando o período integral, Figura 7 (b), pode-se observar que os estudantes do sexo feminino possuem maior grau de dificuldade (4- alto) e (5- muito alto), comparados aos participantes do sexo masculino que apresentam menor grau de dificuldade (1- nenhum) e (2- muito baixo). O mesmo se aplica aos estudantes acima de 29 anos do sexo masculino, que apresentam o maior grau de dificuldade (5- muito alto) se comparados aos estudantes mais jovens do mesmo sexo.



(a): Análise da dificuldade entre o período diurno e noturno.



(b): Análise da dificuldade dos participantes do período integral.

Figura 7: Análise da dificuldade em relação a definição e uso de ponteiros.

Fonte: Dados da pesquisa.

4.2.4 Análise por Tempo Dedicado aos Estudos

Nesta análise, foi verificado o tempo que cada estudante consegue dedicar aos estudos, a partir do momento que não está dentro da sala de aula. Analisou-se tantos os estudantes que só estudam, como aqueles que estudam e trabalham. As questões sobre o tempo dedicado aos estudos foram estruturadas na escala de Likert, correspondendo a: 1- Muito Pouco, 2- Pouco, 3- Indiferente, 4- Suficiente e 5- Bastante.

Na Figura 8 pode-se verificar que os estudantes que só estudam conseguem dedicar mais tempo aos estudos quando não estão dentro da sala de aula, sendo que boa parte desses estudantes afirmam que o tempo de estudo é “suficiente” ou até mesmo “bastante” para a aprendizagem.

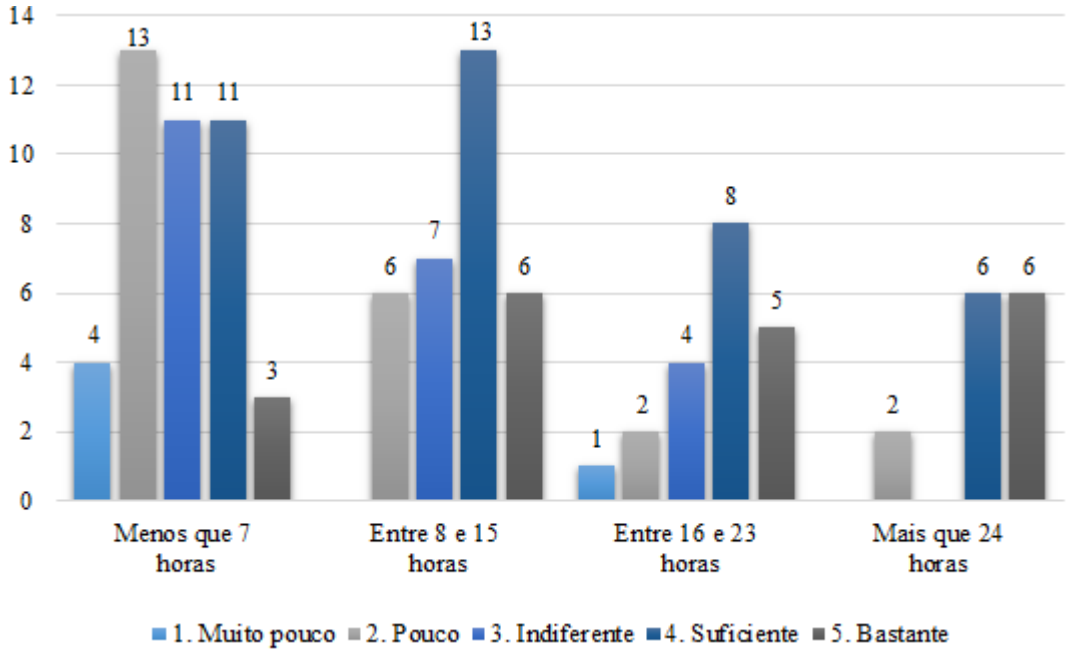


Figura 8: Estudantes que só estudam e o tempo dedicado aos estudos.
Fonte: Dados da pesquisa.

No caso em que os estudantes estudam e trabalham (Figura 9), são poucos aqueles que conseguem dedicar mais do que 7 horas semanais para estudar fora da sala de aula. A maioria desses estudantes afirmam que esse tempo de estudo é “muito pouco” ou “pouco”, ou seja, não sendo suficiente para a aprendizagem dos conceitos apresentados na disciplina de programação.

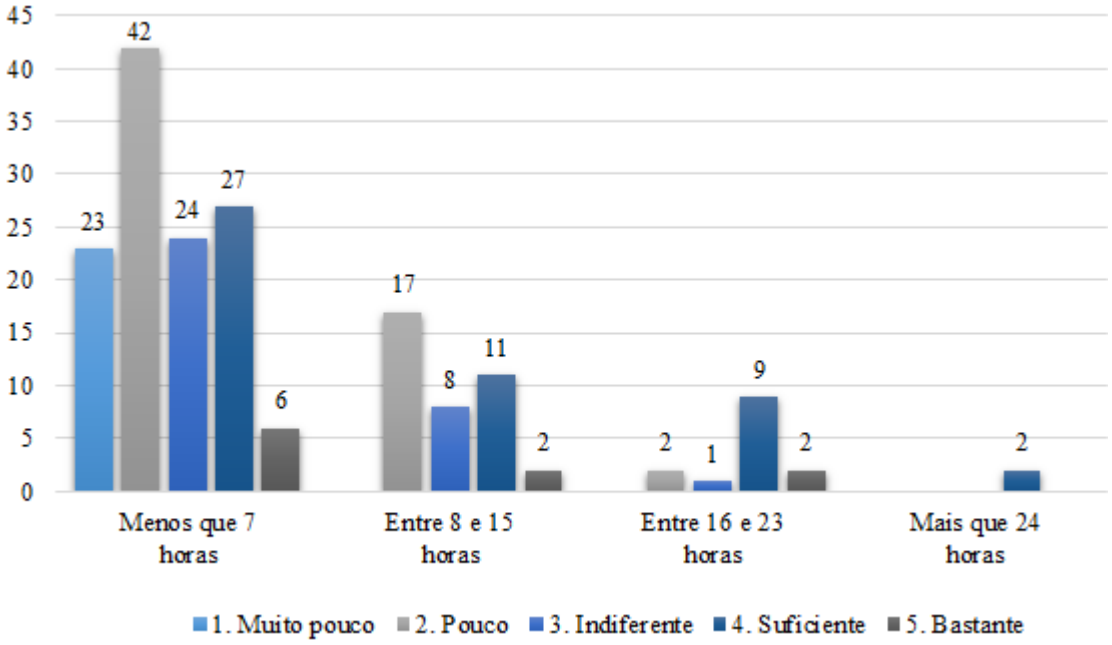


Figura 9: Estudantes que trabalham e estudam e o tempo dedicado aos estudos.
Fonte: Dados da pesquisa.

4.2.5 Ambientes e Ferramentas de Apoio ao Ensino e Aprendizagem

Os estudantes foram questionados se durante os cursos nos quais eles frequentam ou frequentaram recentemente, eram utilizados ambientes ou ferramentas de apoio ao ensino e aprendizagem para facilitar a abstração dos conceitos e práticas de programação discutidos no estudo. Das respostas obtidas, mais da metade responderam que não utilizam quaisquer ferramentas ou ambientes de apoio além da habitual sala de aula (Figura 10).

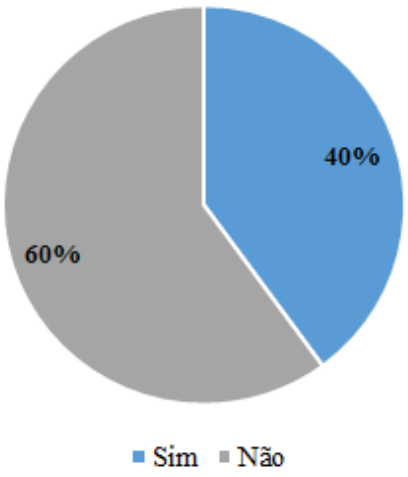


Figura 10: Uso de ambientes e ferramentas de aprendizagem.
Fonte: Dados da pesquisa.

Adicionalmente, no gráfico da Figura 11 considera-se apenas os respondentes que afirmaram utilizar algum ambiente e/ou ferramenta de aprendizagem durante as aulas, tais como laboratórios de informática (32%), ferramentas como o *Scratch* (8%), plataformas de cursos online como a *Udemy* (3%), assim como o *Moodle* (10%) e os sistemas de envio e correção de exercícios como o *Run.codes* (8%).

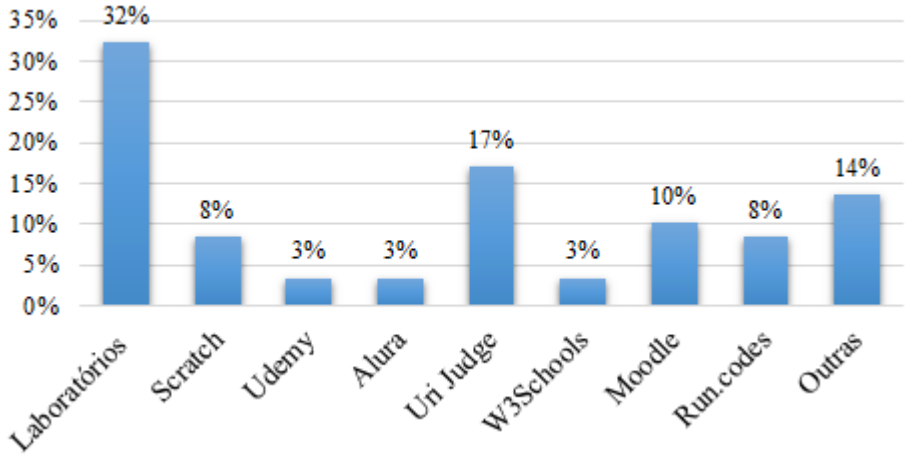


Figura 11: Ambientes e ferramentas de aprendizagem.

Fonte: Dados da pesquisa.

De acordo com Oliveira et al. (2004), um ambiente de aprendizagem pode ser conceituado como os espaços das relações com o saber, o qual é o objeto maior do processo de aprendizagem. Esses espaços são compreendidos pelos autores como ambientes que favorecem a construção do conhecimento que ocorre a partir das interações dos alunos com os conteúdos, com os outros alunos e com os professores (OLIVEIRA *et al*. 2004, p. 118).

Dessa forma, neste estudo foram desconsiderados alguns itens que não se enquadram na definição de ambiente e ferramentas de aprendizagem, tais como minicursos, ferramentas de modelagem e até mesmo linguagens de programação, como por exemplo Java.

4.2. 6. Linguagens de programação

Os estudantes foram questionados sobre quais as linguagens de programação eles possuíam mais dificuldades. Conforme apresentado no gráfico da Figura 12, é possível observar que grande parte das respostas apontam para linguagem Java (31%), seguido da linguagem C (23%) e da linguagem C++ (13%).

A coluna no gráfico da Figura 12 indicada por “Outras”, é um agrupamento de algumas linguagens de programação que tiveram poucos apontamentos pelos respondentes, como por

exemplo *Delphi*, *Assembly* e *Ruby*. Também nota-se que 7% dos estudantes respondentes apontaram possuir dificuldades em todas as linguagens de programação.

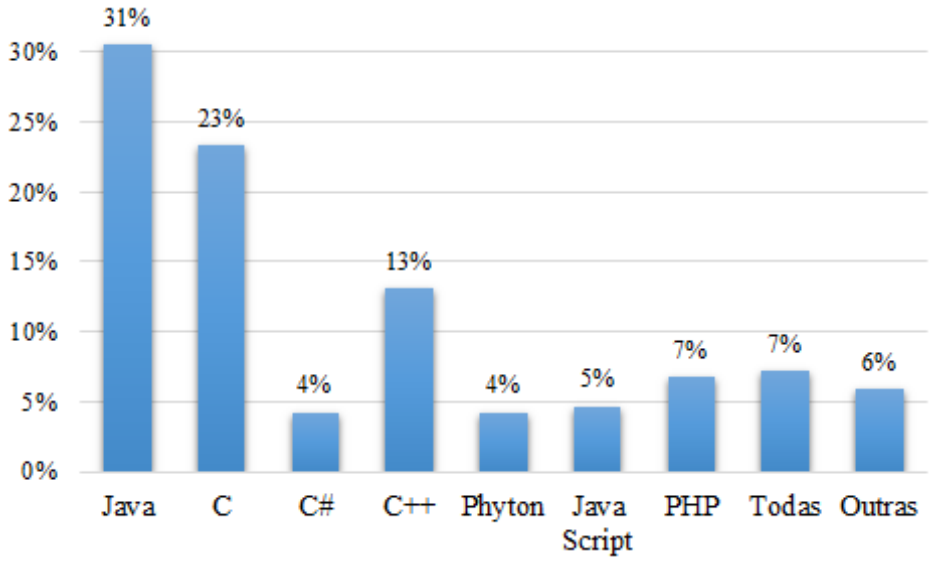


Figura 12: Linguagens de programação que os estudantes possuem dificuldades.

Fontes: dados da pesquisa.

Na análise realizada, foi preciso desconsiderar algumas respostas da questão, pois muitos dos respondentes apontaram dificuldades em SQL, XML, HTML, CSS e outras que não são consideradas linguagens de programação, mas sim linguagens de marcação e linguagem de pesquisa em banco de dados.

Por outro lado, alguns dos estudantes respondentes apontaram que sua maior dificuldade não está em entender ou aprender uma determinada linguagem de programação, mas sim em conseguir entender a lógica de programação.

4.2.7 Sugestões e Melhorias no Ensino e aprendizagem de Programação

Os estudantes foram questionados a dar sugestões para melhorar e facilitar o entendimento dos conceitos abordados na programação. As respostas foram analisadas e agrupadas em categorias que foram criadas de acordo aos termos (sugestões) mais citadas pelos participantes. Veja o gráfico da Figura 13.

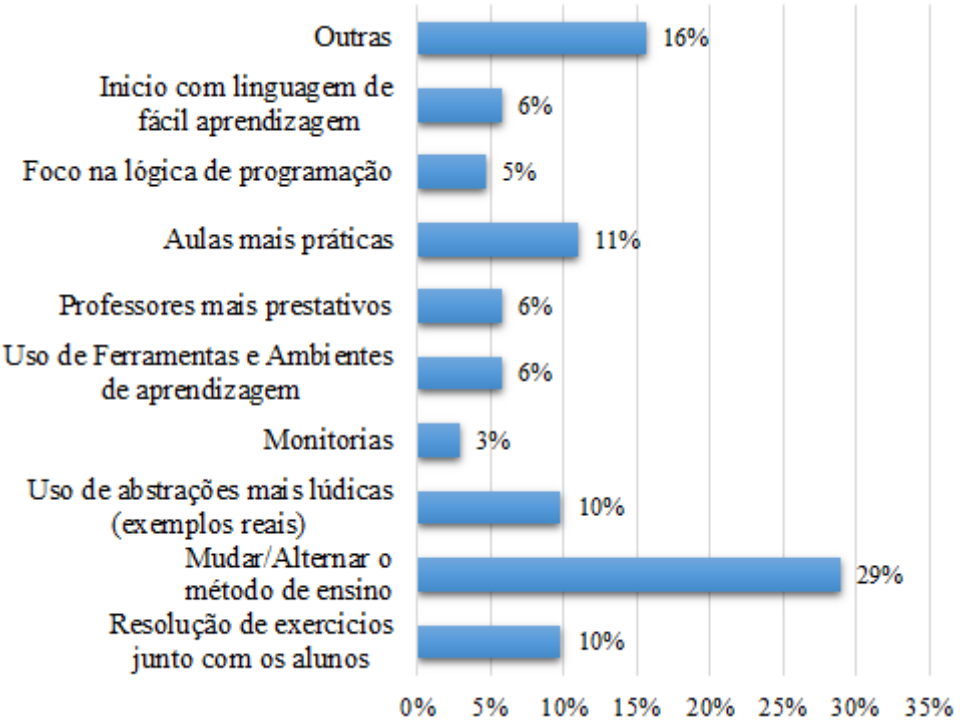


Figura 13: Sugestões para melhorar e facilitar o aprendizado de conceitos de programação.

Fonte: Dados da pesquisa.

Pode-se notar na Figura 13 que a maior parte dos estudantes apontam como uma sugestão a mudança/alternância dos métodos de ensino utilizados no processo de ensino e aprendizagem de programação, totalizando 29% do total. Sugestões como aulas mais práticas e a realização de mais exercícios juntos com os alunos foram outros pontos muito citados pelos participantes, totalizando 11% das respostas. A categoria “Outras” inclui respostas que tiveram um número consideravelmente pequeno, como por exemplo minicursos, uso de linguagens intuitivas, ensinar a debugar, etc.

Sabe-se que é muito difícil para o professor acompanhar individualmente o nível de aprendizado de cada aluno, o que leva muitos a ficarem atrasados e com dificuldades ao longo da disciplina. Nesse sentido, os estudantes também foram questionados a opinar sobre quais seriam as mudanças necessárias para evitar que esse fato continue ocorrendo.

Na Figura 14 apresenta-se uma síntese das mudanças elencadas pelos estudantes no que se refere ao processo de ensino e aprendizagem de programação. O mesmo esquema de agrupamento em categorias foi criado para facilitar a análise. Os resultados vão de encontro às sugestões apresentadas pelos estudantes na questão anterior. Novamente, os estudantes indicam necessidade de mudança nos métodos de ensinos aplicados nas aulas de programação. Destacam-se também a necessidade de monitores nas salas de aulas para auxiliarem o professor,

e também do uso de software que deixem o aprendizado mais lúdico e facilite o acompanhando individual do estudante.

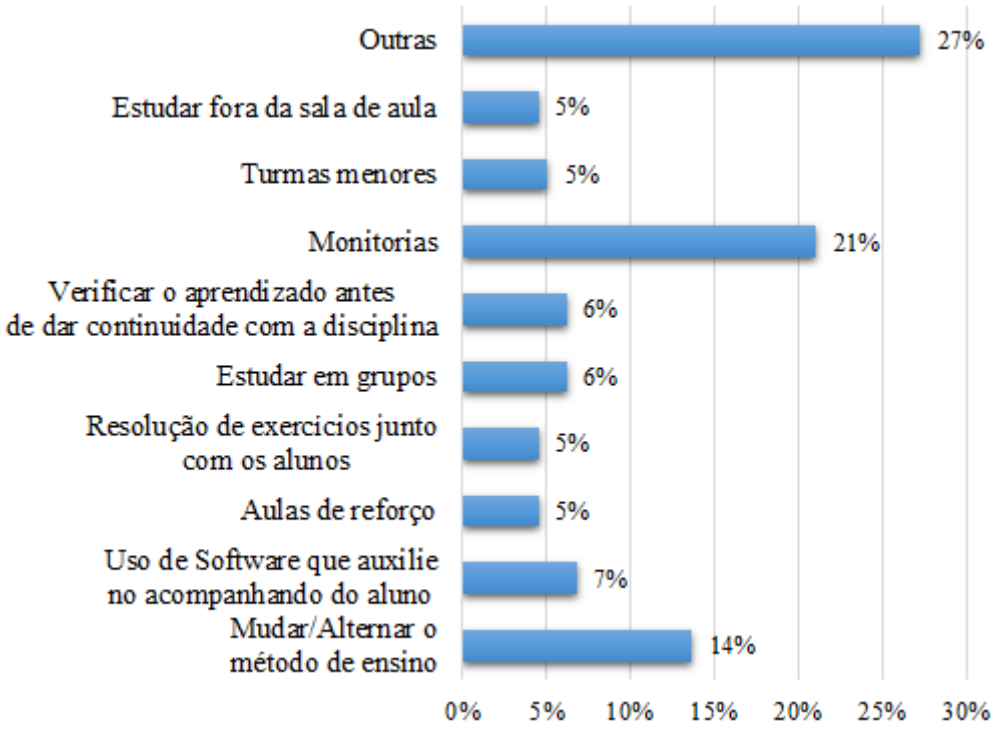


Figura 14: Sugestões de mudanças no processo de ensino e aprendizagem de programação.

Fonte: Dados da pesquisa.

4.3 Síntese dos resultados

Segundo Gomes e Mendes (2007), para muitos estudantes a dificuldade referente ao aprendizado da programação começa na fase inicial de aprendizagem, quando eles precisam entender e aplicar conceitos abstratos de programação.

De acordo com os resultados obtidos neste estudo (Figura 15), verificou-se que aproximadamente 55% dos respondentes concordam que as dificuldades são mais frequentes no início da aprendizagem de programação, enquanto 21% dos respondentes são indiferentes.

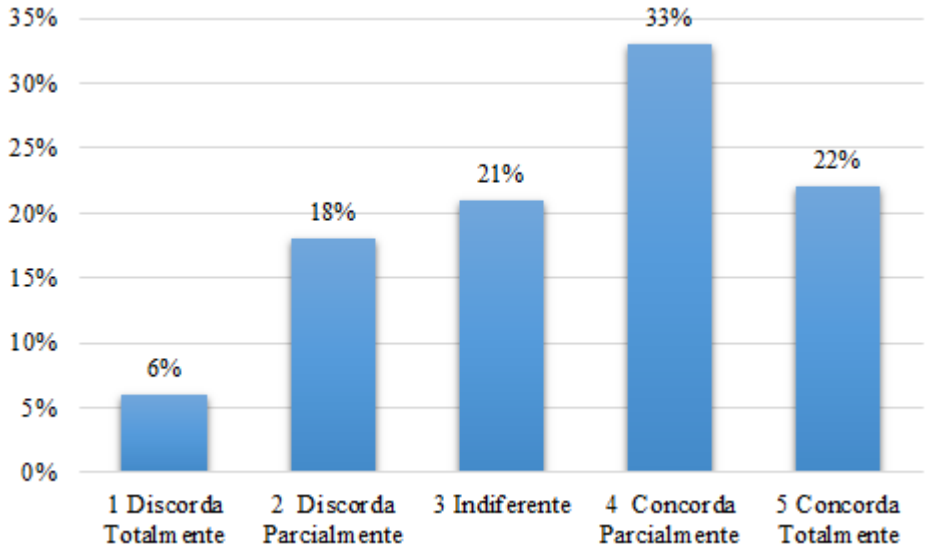


Figura 15: Dificuldades no início da disciplina.

Fonte: Dados da pesquisa.

Kantorski *et al.* (2016) e Pascoal *et al.* (2016) afirmam que os cursos de Computação são considerados um dos que possuem os mais altos índices de reprovação e desistências, tendo relação com as dificuldades dos estudantes no aprendizado das disciplinas de lógica e programação.

Em outros trabalhos encontrados na literatura sobre o aprendizado de programação (Santos e Costas, 2006; Campos, 2010; Júnior, *et al.*, 2014), também foi evidenciado que o nível de evasão em cursos de Computação são muito altos.

Neste estudo, os resultados obtidos estão alinhados com os resultados encontrados na literatura. No gráfico da Figura 16 é possível observar que 45% dos respondentes concordam totalmente que as dificuldades enfrentadas pelos estudantes no ensino e aprendizagem de programação contribuem para os altos índices de evasão nos cursos de TI.

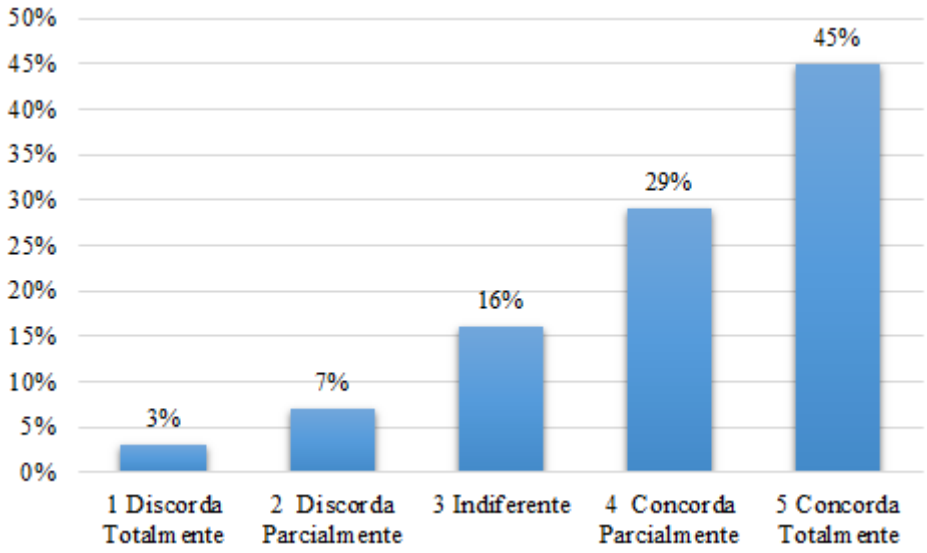


Figura 16: Evasão nos cursos de TI devido às dificuldades no aprendizado.

Fonte: Dados da pesquisa.

Segundo Pimentel, *et al.* (2003) boa parte dos estudantes apresentam grande dificuldade em compreender e aplicar conceitos, como recursão, passagem de parâmetros, e estruturas de seleção e repetição, vistos nas disciplinas introdutórias de programação.

De forma análoga, neste estudo os resultados obtidos os quais foram apresentados anteriormente na Tabela 3, mostram que grande parte dos estudantes possuem alguma dificuldade nesses quesitos.

No estudo conduzido por Ramos *et al*. (2015), cujo objetivo foi verificar artefatos que influenciavam nas taxas de sucesso e reprovação nas disciplinas de programação, verificou-se que a utilização de ambientes e ferramentas aliadas às metodologias de ensino, teve influência significativa na redução da taxa de reprovação nas disciplinas de programação.

Da mesma forma, os resultados obtidos neste estudo mostram que mais de 70% dos participantes concordam que os ambientes e ferramentas de aprendizagem contribuem para uma aprendizagem mais efetiva (Figura 17).

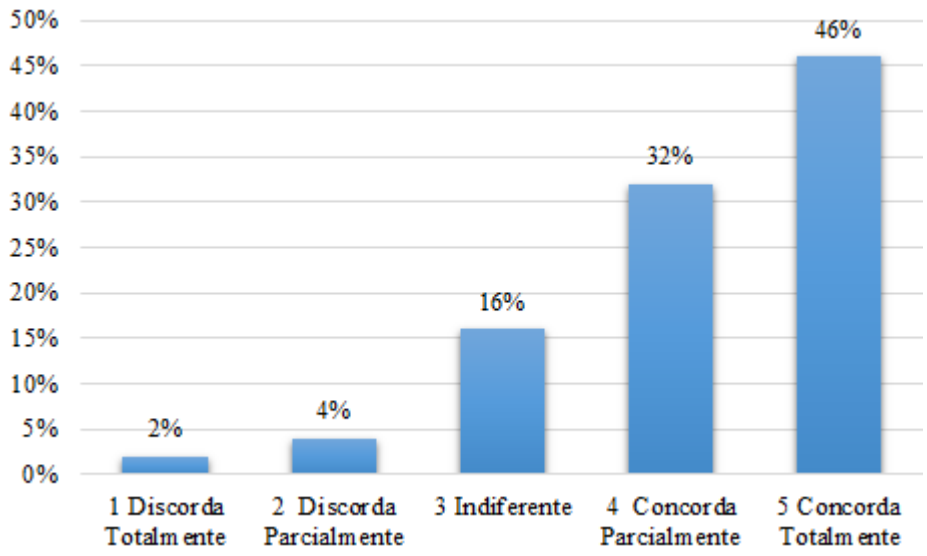


Figura 17: Contribuição dos ambientes e ferramentas de aprendizagem no ensino.

Fonte: Dados da pesquisa.

5 CONSIDERAÇÕES FINAIS

Após apresentar os resultados obtidos com esta pesquisa, este capítulo posiciona a sua relevância, as contribuições, limitações e perspectivas de trabalhos futuros.

5.1 Relevância do Estudo

Pelo fato do ensino e aprendizagem de programação ser complexo e marcado por inúmeros problemas e dificuldades que são alvos de pesquisa no mundo todo, levam as disciplinas de programação e correlatas a possuírem um dos mais altos índices de reprovação. Esses problemas e dificuldades tornam-se muitas vezes desmotivantes aos estudantes, o que ocasiona nos altos índices de evasão nos cursos de Tecnologia da Informação.

Na literatura, existem diversos estudos que tratam da problemática no ensino e aprendizagem de programação (Maltempi e Valente, 2000; Santos e Costa, 2006; Gomes *et. al.*, 2008; Scolari, Bernard e Cordenonsi, 2007; Viegas, *et al.*, 2015). Porém mesmo com esses estudos, ainda não estava claro quais eram os reais problemas e obstáculos que dificultam sua aprendizagem.

Diante do exposto, o principal objetivo deste trabalho foi identificar, compreender e analisar quais são os problemas e dificuldades enfrentados pelos estudantes na aprendizagem de disciplinas relacionadas à programação de computadores. Necessidade de mudanças e melhorias no processo de ensino e aprendizagem foram evidenciadas, a fim de facilitar a introdução de conceitos abstratos de programação.

5.2 Contribuições da Pesquisa

A principal contribuição deste trabalho refere-se ao estudo baseado em survey mostrando um panorama geral das principais dificuldades e problemas enfrentados pelos estudantes na aprendizagem de programação. Dessa forma, este trabalho pode servir como base para outras pesquisas relacionadas, assim como pode ser utilizado por professores e pesquisadores no estudo de propostas e soluções para esses problemas.

Durante a análise dos dados coletados por meio do survey, foi traçado um perfil dos participantes, e dessa forma, foi possível conhecer alguns fatores que levam os estudantes a possuírem dificuldades no aprendizado de programação.

Um das principais dificuldades que os estudantes possuem são em relação a lógica de programação. Esta dificuldade impede que os estudantes consigam aprender e aplicar determinados conceitos abstratos na resolução de um determinado problema. As sugestões propostas pelos próprios participantes da pesquisa refletem a necessidade dos professores darem mais atenção no ensino da lógica de programação, realizando verificações do que foi aprendido pelos estudantes antes de apresentarem uma determinada linguagem de programação.

Um dos problemas do professor é monitorar o aprendizado de todos os estudantes ao longo da disciplina de programação Este fator, faz com que muitos estudantes não consigam acompanhar as aulas devido às dificuldades enfrentadas que não são descobertas a tempo. Como proposta para este problema, foram sugeridos dividir a classe em turmas menores, além de ter mais monitores para tirar dúvidas dos estudantes e auxiliar o professor nas disciplinas.

Outra sugestão foi em separar os estudantes em grupos de estudo, assim, um estudante pode ajudar outro estudante que possui mais dificuldade. Também foram propostas o uso de ferramentas que possibilitem que o professor consiga verificar se os estudantes estão conseguindo realizar as atividades e onde estão encontrando dificuldades.

Identificou-se também a dificuldade em relação às linguagens de programação, em que foi proposto iniciar o aprendizado com linguagens de programação mais fáceis, para facilitar o entendimento dos conceitos abstratos nos anos iniciais dos cursos.

Um dos pontos mais citados pelos estudantes tem relação com as metodologias de ensino utilizadas pelos professores nas disciplinas de programação. Nesse contexto, as sugestões recaem sobre a necessidade de mudança/alteração nas práticas atuais de ensino de programação, incorporando por exemplo, ferramentas e ambientes de aprendizagem que facilitem o entendimento dos conceitos abstratos da programação.

5.3 Dificuldades e Limitações

O questionário foi desenvolvido utilizando a ferramenta do *Google,* o *Google Forms,* o que facilitou o envio deste questionário para estudantes de diversas regiões do Brasil. Porém, não havia como garantir que todos os estudantes que recebiam o questionário iriam respondê-lo. Assim, uma das dificuldades encontradas foi atingir um grupo maior de respondentes. Para isso, o tempo de aplicação do questionário precisou ser estendido.

O tamanho da amostra da população-alvo da pesquisa pode ser considerado uma

limitação. O questionário foi enviado para inúmeras listas de e-mails, páginas e grupos de discussão nas redes sociais, bem como e-mails de professores de várias instituições de ensino técnico/superior. No final, a amostra foi composta por 284 participantes.

5.4 Trabalhos futuros

Como trabalho futuro, propõe-se a condução de estudos que investiguem, além das dificuldades no ensino e aprendizagem de programação, outros fatores que contribuem para os altos índices de evasão e reprovação nos cursos de TI, tais como fatores socioeconômico.

Propõe-se também o estudo e a aplicação de abordagens baseadas em Gamificação, robótica educacional e jogos educacionais (aliada às práticas atuais de ensino) como mecanismos facilitadores e motivadores de aprendizagem de conceitos de programação.

Adicionalmente, propõe-se o desenvolvimento de uma ferramenta ou ambiente de aprendizagem, que facilite o ensino dos conceitos abstratos da programação (tais como definição e uso de ponteiros, algoritmos recursivos e estrutura de dados) e que auxilie no acompanhando dos estudantes, sendo possível verificar os exercícios e trabalhos realizados, assim como as dificuldades encontradas pelos estudantes durante a disciplina.

REFERÊNCIAS

AMARAL, L. R.; SILVA, G. B.; PANTALEÃO, E. (2015). Plataforma Robocode como Ferramenta Lúdica de Ensino de Programação de Computadores - Extensão Universitária em Escolas Públicas de Minas Gerais. *Anais do XXVI Simpósio Brasileiro de Informática na Educação (SBIE 2015).* Maceió, 2015. p. 200-208.

BARBOSA, A. D.; FERREIRA, D. Í.; COSTA, E. B. (2014). Influência da linguagem no ensino introdutório de programação. *III Congresso Brasileiro de Informática na Educação (CBIE 2014). XXV Simpósio Brasileiro de Informática na Educação (SBIE 2014).* Dourados, 2014. p. 612-621.

BINI, E. M.; KOSCIANSKI, A. (2000). O Ensino De Programação De Computadores Em Um Ambiente Criativo E Motivador. *Encontro Nacional de Pesquisa em Educação e Ciência*, (p. 11). Florianópolis.

CAMPOS, R. L. (2009). *ERMC2:* Uma proposta de metodologia para melhoria do ensino-aprendizado de lógica de programação. XI Congreso Chileno de Educación Superior en Computación (CCESC), 2009. Santiago, Chile, Jornadas Chilenas de Computación.

CAVALCANTI, M. (27 de Março de 2015). *Alunos da USP criam sistema simples para aprendizagem de códigos*. Acesso em 25 de Abril de 2018, disponível em Mundo BIT: http://blogs.ne10.uol.com.br/mundobit/2015/03/27/alunos-da-usp-criam-sistema-simples-para-aprendizagem-de-codigos/.

DAMASCENO, M. (2010). INTRODUÇÃO A MINERAÇÃO DE DADOS UTILIZANDO O WEKA. Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, Campus Macau.

FERNANDES, V. D.; JUNIOR, V. F. (2016.). Linguagem de programação: evasão e reprovação no. Instituto Federal Catarinense, Campus Avançado Sombrio.

FILHO, R. L.; MOTEJUNAS, P. R.; HIPÓLITO, O.; LOBO, M. B. (2007). A evasão no ensino superior brasileiro. Cadernos de Pesquisa, 37(132):641–659.

GIL, A. C. (2002). Como Elaborar Projetos de Pesquisa. 4ª ed. São Paulo: Editora Atlas.

GOMES, A.; MENDES, A. J. (2007). Learning to program - difficulties and solutions.

International Conference on Engineering Education – ICEE, (p. 6). Coimbra.

GOMES, A.; AREIAS, C.; HENRIQUES, J.; MENDES, A. J. (2008). Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. *Revista Portuguesa de Pedagogia*, 42(2), 161–179.

GONÇALVES, D. A.; SILVA, G. M.; LUZ, R. S.; SILVA, C. E.(2013). Relato de experiência de alunos do curso de Licenciatura em Computação do IFMG - campus Ouro Branco na utilização de objetos de aprendizagem desplugados e do Scratch como instrumentos no ensino de programação. *II Congresso Brasileiro de Informática na Educação (CBIE).* Minas Gerais, 2013. p. 335-344.

JESUS, A. D.; BRITO, G. S. (2009). Concepção de Ensino-Aprendizagem de Algoritmos e Programação de Computadores: A Prática Docente. *I ENINED - Encontro Nacional de Informática e Educação.* Cascavel, 2009. p. 132-141.

JÚNIOR, J. C.; RAPKIEWICZ, C. E.; DELGADO, C.; XEXEO, J. A. (2014). Ensino de Algoritmos e Programação: Uma Experiência no Nível Médio. *XXV Congresso da Sociedade Brasileira de Computação.* São Leopoldo, 2014. p. 2351-2362.

KANTORSKI, G. Z.; FLORES, E. G.; HOFFMANN, I. L.; SCHMITT, J. A.; BARBOSA, F. P. (2016). Predição da Evasão em Cursos de Graduação em Instituições Públicas. *V Congresso Brasileiro de Informática na Educação (CBIE 2016). Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE 2016).* Santa Maria, 2016. p. 905-915.

LIKERT, R. A technique for the measurement of attitudes. New York: The Science Press, 1932.

LIMA, M. R.; LEAL, M. C. (2010). Uso da linguagem logo no ensino superior de programação de computadores. *Revista Eletrônica Multidisciplinar Pindorama do Instituto Federal de Educação, Ciência e Tecnologia da Bahia – IFBA*, 1(1). p. 1-14.

MALTEMPI, M. V.; VALENTE, J. A. (2000). Melhorando e Diversificando a Aprendizagem via Programação de Computadores. *In: International Conference on Engineering and Computing Education*. São Paulo.

MARTINS, L. A. (2015). A ferramenta scratch como elemento motivador da aprendizagem de algoritmos. *Instituto Federal de Educação, Ciência e Tecnologia do Amazonas.* Campina Grande: Realize.

MATA, E. C.; PINHEIRO, M. F.; JR, A. F.; FRANCÊS, C. R.; SANTANA, Á. L.; COSTA, J. C. (2013). Proposta De Sistema Lúdico Para Ensino De Programação A Alunos Do Ensino Médio. *X Congresso Brasileiro de Ensino Superior a Distância.* Belém, 2013. p. 1-14.

MEDEIROS, T. J.; SILVA, T. R.; ARANHA, E. H. (2013). Ensino de programação utilizando jogos digitais: uma revisão sistemática da literatura. UFRGS, Centro Interdisciplinar de Novas Tecnologias na Educação (CINTED).

OLIVEIRA, C. C; COSTA, J. W.; MOREIRA, M. Ambientes informatizados de aprendizagem. In: COSTA, J. W.; OLIVEIRA, M. A. M. (orgs.) Novas linguagens e novas tecnologias: Educação e sociabilidade. Petrópolis: Vozes, 2004.

PASCOAL, T. A.; BRITO, D. M.; ANDRADE, L. P.; RÊGO, T. G. (2016). Evasão de estudantes universitários: diagnóstico a partir de dados acadêmicos e socioeconômicos. V Congresso Brasileiro de Informática na Educação (CBIE 2016). Anais do XXVII Simpósio Brasileiro de Informática na Educação (SBIE). Paraíba: SBC, p. 926-935.

PAES, R. d.; MALAQUIAS, R.; GUIMARÃES, M.; ALMEIDA, H. (2013). Ferramenta para a Avaliação de Aprendizado de Alunos em Programação de Computadores. Instituto de Computação – Universidade Federal de Alagoas (UFAL), Departamento de Sistemas e Computação – Univ. Federal de Campina Grande (UFCG). *II Congresso Brasileiro de Informática na Educação (CBIE 2013).* Minas Gerais. p. 203-212.

PIMENTEL, E. P.; FRANÇA, V. F.; OMAR, N. (2003). A caminho de um ambiente de avaliação e acompanhamento contínuo da aprendizagem em Programação de Computadores. Instituto Tecnológico da Aeronáutica (ITA), São Paulo.

PIMENTEL, E. P.; FRANÇA, V. F.; NORONHA, R. V.; OMAR, N. (2003). Avaliação Contínua da Aprendizagem, das Competências e Habilidades em Programação de Computadores. *IX Workshop em Informática na Escola - WIE*, 2003. São

Paulo. p. 533-544.

RAABE, A. L.; SILVA, J. M. (2014). Um Ambiente para Atendimento às Dificuldades de Aprendizagem de Algoritmos. *XXV Congresso da Sociedade Brasileira de Computação.* São Leopoldo. p. 2326-2337.

RAMOS, V.; FREITAS, M.; GALIMBERT, M.; MARIANI, A. C.; WAZLAWICK, R. (2015). A Comparação da Realidade Mundial do Ensino de Programação para Iniciantes com a Realidade Nacional: Revisão sistemática da literatura em eventos brasileiros. *Anais do XXVI Simpósio Brasileiro de Informática na Educação.* Maceió, 2015. p. 318-327.

RAPKIEWICZ, C. E.; FALKEMBACH, G.; SEIXAS, L.; ROSA, N. D.; CUNHA, V. V.; KLEMANN, M. (2006). Estratégias Pedagógicas No Ensino De Algoritmos E Programação Associadas Ao Uso De Jogos Educacionais. Revista Novas Tecnologias na Educação (RENOTE), (4)2, p.1-11.

ROSA, M. M.; GIRAFFA, L. M. (2011). O Ensino de Programação De Computadores e EAD: Uma Parceria Possível. PUCRS, Setor Educacional Educação Universitária, Porto Alegre. p. 1-10.

SANTIAGO, A. D.; KRONBAUER, A. H. (2016). Um Modelo Lúdico para o Ensino de Conceitos de Programação de Computadores. In: Simpósio Brasileiro de Informática na Educação (SBIE), Uberlândia, 2016. p. 420-429.

SANTOS, R. P.; COSTA, H. A. (2006). Análise de Metodologias e Ambientes de Ensino para Algoritmos, Estruturas de Dados e Programação aos iniciantes em Computação e Informática. Universidade Federal de Lavras, Minas Gerais.

SBC - Sociedade Brasileira de Computação, (2005). Currículo de Referência da SBC para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação.

SCOLARI, A. T.; BERNARDI, G.; CORDENONSI, A. Z. (2007). O Desenvolvimento do Raciocínio Lógico através de Objetos de Aprendizagem. Centro Universitário Franciscano; Universidade Federal de Santa Maria, Santa Maria, RS.

SILVA, I. F.; SILVA, I. M.; SANTOS, M. S. (2009). Análise de problemas e soluções

aplicadas ao ensino de disciplinas introdutórias de programação.

SOUZA, D. M.; BATISTA, M. H.; BARBOSA, E. F. (2016). Problemas e Dificuldades no Ensino e na Aprendizagem de Programação: Um Mapeamento Sistemático. *Revista Brasileira de Informática na Educação, 24*(1). p. 39-52.

TERENCE, A. C.; FILHO, E. E. (2006). Abordagem quantitativa, qualitativa e a utilização da pesquisa-ação nos estudos organizacionais. *XXVI ENEGEP*. Fortaleza. p. 1-9.

VIEGAS, T. R.; OKUYAMA, F. Y.; PARAVISI, M.; BERTAGNOLLI, S. d. (2015). Uso das TICs no processo de ensino-aprendizagem de programação. *Nuevas Ideas en Informática Educativa TISE,* p. 780-785.

VIEL, F.; RAABE, A.; ZEFERINO, C. (2014). Introdução à Programação e à Implementação de Processadores por Estudantes do Ensino Médio. Universidade do Vale do Itajaí - Univali. *3º Congresso Brasileiro de Informática na Educação (CBIE 2014).* Dourados. p. 248-257.

WAZLAWICK, R. S. (2014). Metodologia de Pesquisa para Ciência da Computação (2ª ed.). Rio de Janeiro. Elsevier Editora Ltda.

APÊNDICE A - Questionário: Dificuldades no aprendizado de Programação

Este questionário é um instrumento de coleta de informações para identificar as principais dificuldades encontradas pelos estudantes nas disciplinas de programação de computadores e correlatas. A pesquisa faz parte de um Trabalho de Conclusão de Curso do aluno Weldrey Toneto de Oliveira, graduando do 9° semestre de Sistemas de Informação na Universidade Estadual do Norte do Paraná (UENP), Campus Luiz Meneghel de Bandeirantes - PR. A ideia da pesquisa é obter uma melhor compreensão acerca da problemática associada ao ensino e aprendizagem dessas disciplinas, a fim de fornecer subsídios para a melhoria da qualidade do ensino e da experiência da aprendizagem dos estudantes, e por consequência, contribuir com a diminuição do alto índice de evasão nos cursos da área.

Espera-se que o questionário não demore mais que 20 minutos para ser preenchido. Os campos marcados com (*) são de preenchimento obrigatório.

Ressalta-se que os dados serão utilizados apenas para fins de pesquisa. Informações pessoais ou confidenciais dos participantes não serão divulgadas.

Desde já, agradecemos a sua participação nesta pesquisa. Ela pode contribuir para um aprofundamento nas discussões e melhorias do processo de ensino e aprendizagem de programação de computadores e disciplinas correlatas.

*Obrigatório

Dados Pessoais

1. Idade: *

2. Sexo: *

Marcar apenas uma oval.

- ◯ Masculino
- ◯ Feminino

3. Cidade: *

Dados sobre a Instituição de Ensino e o Curso

4. Estuda em que tipo de Instituição de Ensino? *

Marcar apenas uma oval.

- ◯ Pública
- ◯ Privada

5. Qual o tipo de curso? *

Marcar apenas uma oval.

- ◯ Bacharelado
- ◯ Licenciatura
- ◯ Curso Técnico
- ◯ Tecnólogo

6. Qual o curso? *

Marcar apenas uma oval.

- ◯ Ciência da Computação
- ◯ Sistemas de Informação
- ◯ Análise e Desenvolvimento de Sistemas (ADS)
- ◯ Engenharia da Computação
- ◯ Gestão de Tecnologia da Informação
- ◯ Design Gráfico
- ◯ Outros

7. Qual o período do curso? *

Marcar apenas uma oval.

- ◯ Diurno
- ◯ Noturno
- ◯ Integral

8. Quais disciplinas relacionadas à programação você cursou ou cursa atualmente? *

Marque todas que se aplicam.

- ☐ Algoritmos;
- ☐ Estrutura de dados;
- ☐ Lógica de programação;
- ☐ Programação I;
- ☐ Programação II;
- ☐ Outras;

9. De acordo com a questão anterior, em que ano as disciplinas foram cursadas?

_____

Dados sobre o tempo dedicado ao Estudo

10. Durante o período do curso você: *

Marcar apenas uma oval.

- ◯ Só estuda
- ◯ Estuda e Trabalha
- ◯ No início do curso só estudava, agora faz ambos
- ◯ No início do curso estudava e trabalhava, agora só estuda

11. De acordo com a questão anterior, caso você trabalhe, qual a carga horária semanal?

Marcar apenas uma oval.

- ◯ Menos de 20 horas semanais;
- ◯ 20 horas semanais;
- ◯ Entre 20 e 30 horas semanais;
- ◯ 40 horas semanais;
- ◯ Mais de 40 horas semanais;

12. A partir do momento em que não está na sala de aula, quantas horas semanais consegue se dedicar aos estudos? *

Marcar apenas uma oval.

- ◯ Menos de 7 horas
- ◯ 8 a 15 horas
- ◯ 16 a 23 horas
- ◯ Mais de 24 horas

13. Em relação ao tempo dedicado aos estudos para aquisição de conhecimento, ele foi: *

1. Muito Pouco, 2. Pouco, 3. Indiferente, 4. Suficiente, 5. Bastante
Marcar apenas uma oval.

	1	2	3	4	5	
Muito Pouco	◯	◯	◯	◯	◯	Bastante

Dados sobre Conhecimento Prévio em Programação

14. Já teve contato com a programação antes de ingressar no ensino superior? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

15. De acordo a questão anterior, se a resposta foi sim, seu nível de conhecimento em programação equivale a:

1. Muito Pouco, 2. Pouco, 3. Indiferente, 4. Suficiente, 5. Bastante
Marcar apenas uma oval.

	1	2	3	4	5	
Muito Pouco	◯	◯	◯	◯	◯	Bastante

16. Linguagem(ns) de programação que você utilizou (conhece):

Dados sobre Conceitos e Práticas de Programação

17. Linguagem(ns) de programação que possui dificuldade:

18. Grau de dificuldade em relação ao entendimento de lógica de programação: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

19. Grau de dificuldade em relação à definição e uso de variáveis: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

20. Grau de dificuldade em relação à definição e uso de estruturas de decisão:

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

21. Grau de dificuldade em relação à definição e uso de estruturas de repetição: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

22. Grau de dificuldade em relação à definição e uso de funções/métodos: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

23. Grau de dificuldade em relação à passagem de parâmetro para função/método (por valor e por referência): *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

24. Grau de dificuldade em relação à definição e uso de arrays (vetores e matrizes):

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

25. Grau de dificuldade em relação à manipulação de strings: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

26. Grau de dificuldade em relação à definição e uso de ponteiros: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

27. Grau de dificuldade em relação à definição e uso de algoritmos recursivos: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

28. Grau de dificuldade em relação à definição e uso de estruturas de dados: *

1. Nenhum, 2. Muito Baixo, 3. Baixo, 4. Alto, 5. Muito Alto
Marcar apenas uma oval.

	1	2	3	4	5	
Nenhum	◯	◯	◯	◯	◯	Muito Alto

Dados sobre o Ensino e Aprendizagem da Disciplina

29. Consegue acompanhar as explicações e o ritmo de andamento da disciplina?

Marcar apenas uma oval.

◯ Sim
◯ Não

30. Possui dificuldade em entender os enunciados dos exercícios? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

31. Quando não consegue resolver algum problema relacionado ao código, o que normalmente você faz: *

Marcar apenas uma oval.

- ◯ Tenta resolver por meio de tentativas
- ◯ Pede ajuda pra alguém
- ◯ Procura o código na internet
- ◯ Tenta entender o que foi pedido, e após entender resolve o problema

32. As dificuldades geralmente surgem no início da disciplina. *

1. Discorda Totalmente, 2. Discorda Parcialmente, 3. Indiferente, 4. Concorda Parcialmente, 5. Concorda Totalmente

Marcar apenas uma oval.

	1	2	3	4	5	
Discorda totalmente	◯	◯	◯	◯	◯	Concorda totalmente

33. As dificuldades no ensino e aprendizagem de programação contribuem para a evasão nos cursos de Tecnologia. *

1. Discorda Totalmente, 2. Discorda Parcialmente, 3. Indiferente, 4. Concorda Parcialmente, 5. Concorda Totalmente

Marcar apenas uma oval.

	1	2	3	4	5	
Discorda totalmente	◯	◯	◯	◯	◯	Concorda totalmente

34. O uso de ferramentas e/ou ambientes de aprendizagem contribui para uma aprendizagem mais efetiva. *

1. Discorda Totalmente, 2. Discorda Parcialmente, 3. Indiferente, 4. Concorda Parcialmente, 5. Concorda Totalmente

Marcar apenas uma oval.

	1	2	3	4	5	
Discorda Totalmente	◯	◯	◯	◯	◯	Concorda totalmente

35. **É utilizado alguma dessas ferramentas ou ambientes de aprendizagem para facilitar o ensino e aprendizagem de programação?** *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

36. **Se sim, qual?**

Sugestões e Melhorias

37. **Se pudesse sugerir algo para melhorar e facilitar o entendimento dos conceitos abordados na programação, qual seria sua sugestão?**

38. **Sabe-se que é muito difícil para o professor acompanhar individualmente o nível de aprendizado de cada aluno, o que leva muitos a ficarem atrasados e com dificuldades ao longo da disciplina. Nesse sentido, o que poderia mudar para evitar que esse fato continue ocorrendo?**