UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

ERICA YURIE SAITO

OBJETOS DE APRENDIZAGEM ACESSÍVEIS DE DIFERENTES ABORDAGENS PARA O ENSINO DE PROGRAMAÇÃO E AVALIAÇÃO DA MOTIVAÇÃO

CAMPO MOURÃO

2021

ERICA YURIE SAITO

OBJETOS DE APRENDIZAGEM ACESSÍVEIS DE DIFERENTES ABORDAGENS PARA O ENSINO DE PROGRAMAÇÃO E AVALIAÇÃO DA MOTIVAÇÃO

Accessible learning objects from different approaches for teaching programming and motivation assessment

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Dr. André Luiz Satoshi Kawamoto

CAMPO MOURÃO

2021

ERICA YURIE SAITO

OBJETOS DE APRENDIZAGEM ACESSÍVEIS DE DIFERENTES ABORDAGENS PARA O ENSINO DE PROGRAMAÇÃO E AVALIAÇÃO DA MOTIVAÇÃO

Trabalho de Conclusão de Curso de Graduação apresentado como requisito para obtenção do título de Bacharel em Ciência da Computação do Curso de Bacharelado em Ciência da Computação da Universidade Tecnológica Federal do Paraná.

Data de aprovação: 25/junho/2021

André Luiz Satoshi Kawamoto
Doutorado
Universidade Tecnológica Federal do Paraná

Narci Nogueira da Silva
Mestrado
Universidade Tecnológica Federal do Paraná

Rafael Liberato Roberto
Doutorado
Universidade Tecnológica Federal do Paraná

CAMPO MOURÃO

2021

RESUMO

Os cursos da área de Computação apresentam altas taxas de evasão. Para muitos novos alunos que têm o primeiro contato com programação nesses cursos, a forma de ensinar o raciocínio lógico contrasta com a forma como estão acostumados, tornando-se assim um fator desmotivador. A literatura também aponta para a falta de materiais acessíveis específicos para alunos com necessidades especiais. Do ponto de vista dos professores, é preciso repassar o conteúdo para apresentar a programação como uma atividade instigante e buscar uma abordagem de ensino que traga melhores resultados. Este trabalho tem como objetivo Investigar e selecionar abordagens de ensino atualmente utilizadas em programação e ferramentas para avaliar a motivação dos alunos em cursos desta natureza. Depois disso, criar Objetos de Aprendizagem (OA) acessíveis para as abordagens selecionadas e propor um método de avaliação da motivação. Por fim, disponibilize esses recursos em um repositório online. Para tanto, primeiramente, realizamos pesquisa bibliográfica sobre abordagens de ensino de programação, métodos de avaliação de motivação e acessibilidade em repositórios relevantes. Após, selecionamos as práticas de interesse e criamos o OA acessível para cada abordagem de ensino. O questionário CIS, uma ferramenta de avaliação de motivação para o modelo Attention, Relevance, Confiance, Satisfaction (ARCS), foi adaptado e integrado a um repositório online. As fichas de exercícios, o questionário e o repositório foram elaborados considerando os padrões de acessibilidade. Estudos relacionados indicam que o questionário CIS é adequado para avaliar a motivação dos alunos. O material criado, aliado às métricas estatísticas, pode fornecer dados suficientes para um professor utilizar esses materiais e comparar as abordagens por conta própria.

Palavras-chave: ensino de programação; abordagem de ensino; objetos de aprendizagem, acessibilidade, motivação.

ABSTRACT

Courses in the Computing field present high dropout rates. For many new students who have first contact with programming in these courses, the way of teaching logical reasoning contrast with how they are used to, thus becoming a demotivating factor. Literature also points to a lack of accessible materials specific to special-needs students. From the teachers' point of view, it is necessary to pass on the content to introduce programming as an exciting activity while seeking an approach to teaching that brings better results. This work aims to Investigate and select teaching approaches currently used in programming and tools to assess students' motivation in courses of this nature. After this, to create accessible Learning Objects (LO) for the selected approaches and propose a method for assessing motivation. Finally, make these resources available in an online repository. To do so, first, we performed bibliographic research on programming teaching approaches, motivation assessment methods, and accessibility in relevant repositories. After, we selected the practices of interest and created the accessible LO for each teaching approach. The CIS questionnaire, a motivation assessment tool for the ARCS model, was adapted and integrated into an online repository. The exercise sheets, the questionnaire, and the repository were prepared considering accessibility standards. Related studies indicate that the CIS questionnaire is suitable for assessing students' motivation. The material created, combined with statistical metrics, can provide enough data for a teacher to use these materials and compare the approaches on his own.

Keywords: teaching programming; teaching approach; learning objects; accessibility; motivation.

LISTA DE ABREVIATURAS E SIGLAS

Siglas

ARCS	Attention, Relevance, Confiance, Satisfaction
AVA	Ambiente Virtual de Aprendizagem
CIS	Course Interest Survey
EAD	Ensino a Distância
IEEE	Institute of Electrical and Electronics Engineers
IMMS	Instructional Materials Motivation Survey
Inep	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
LO	Learning Objects
MEC	Ministério da Educação
OA	Objetos de Aprendizagem
PBL	Problem Based Learning
PDF	Portable Document Format
RIUT	Repositório Institucional da Universidade Tecnológica Federal do Paraná
SAEB	Sistema de Avaliação da Educação Básica
TI	Tecnologia da Informação
UnB	Universidade de Brasília
USP	Universidade de São Paulo
W3C	World Wide Web Consortium
WCAG	Web Content Acessibility Guidelines

SUMÁRIO

1 INTRODUÇÃO

1.1 Considerações preliminares

Para muitos alunos, o ingresso em cursos superiores da área da computação representa o primeiro contato com a atividade de programação. Esse primeiro contato pode ser desmotivador.

Muitos autores apontam que a sobrevivência nessa área não é fácil. Segundo Kassai *et al.* (2010), na Universidade de São Paulo (USP), a área de Ciências Exatas foi a que mais sofreu evasão no período de 1998 a 2008, com a taxa de 79,88%. Comparado à área de Biológicas, cuja taxa foi de 8,80% e à de Humanas que foi de 31,52%, é possível ver quão alto é esse índice na área de Exatas.

Na área da Computação, Hoed (2016), que utilizou os dados do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep) em suas pesquisas, apresenta resultados desanimadores. A taxa de evasão nos cursos de Computação, tanto nas instituições públicas quanto nas privadas, ficou acima da média nacional no período 2010-2014, sendo uma das maiores entre todas.

Um fator que contribui na sobrevivência é a não reprovação na disciplina de Algoritmos, presente no início do curso. Segundo Hoed (2016), na Universidade de Brasília (UnB), a aprovação nessa matéria resultou em uma menor evasão. Diversos outros autores também afirmam que o insucesso nas disciplinas relacionadas a algoritmos é um forte fator que contribui na evasão do curso. ((GOMES; MENDES, 2000), (HINTERHOLZ Jr., 2009)„ (VALASKI; PARAISO, 2012))

Com a pandemia de COVID-19, no ano de 2020, a taxa de evasão ficou ainda pior. Muitos veículos de imprensa noticiaram o grande aumento de alunos do ensino superior que foram obrigados a interromper seus estudos ((BERMÚDEZ, 2020), (OLIVEIRA, 2020), (MIOLA; GOTTI, 2021)). Oliveira (2020) aponta as áreas mais afetadas pela pandemia no que diz respeito à saída de alunos, e inclui a de Tecnologia da Informação (TI) como uma delas.

1.2 Problema de Pesquisa

Dentre os diversos motivos para a desistência do aluno apresentados por Hoed (2016), a “Dificuldade dos professores em repassar de maneira compreensível o conteúdo” foi considerada como uma das grandes influenciadoras na evasão.

Uma outra pesquisa, realizada por Lahtinen, Ala-Mutka e Järvinen (2005), mostrou as maiores dificuldades dos alunos nas matérias de programação, que foram:

- entender como projetar um programa para resolver uma determinada tarefa;
- dividir funcionalidades em procedimentos;

- encontrar erros nos próprios códigos.

Além dessas, Monclar, Silva e Xexéo (2018) apontam a falta de motivação por parte do aluno, pois muitos veem a programação como uma atividade cansativa e entediante.

Para tentar conter o cenário de evasão que foi agravado pela pandemia da COVID-19 e prosseguir com o ano letivo, muitas instituições adotaram a modalidade Ensino a Distância (EAD), ou híbrido. Por conta disso, muitos professores foram obrigados a adaptar seu material de ensino. Nesse sentido, convém apontar que existe uma carência de materiais didáticos específicos para essas modalidades. Sanchotene *et al.* (2020) apresentam uma pesquisa na qual cerca de 65% dos professores participantes responderam que poucas, ou somente uma parte das aulas foram desenvolvidas durante o isolamento social, mostrando uma falta nos materiais. Além disso, somente 35.5% dos professores conseguiram ministrar as aulas remotamente.

O desenvolvimento de materiais didáticos, sobretudo para alunos à distância deve, além do aspecto motivacional, considerar a acessibilidade. Nesse sentido, Tiziotto e Oliveira Neto (2010) apontam que o uso de materiais didáticos com *design universal* proporciona uma melhora na aprendizagem e na motivação.

Além disso, segundo (RIOS *et al.*, 2016), materiais didáticos sem recursos de acessibilidade favorecem a evasão, uma vez que tornam-se obstáculos para pessoas com necessidades especiais ao tentarem acessar o conteúdo do curso.

Entretanto, esses materiais devem ser cuidadosamente elaborados, uma vez que materiais mal concebidos podem ser, também, fatores desmotivadores para os alunos. Coombs (2010) apresenta uma pesquisa em que aproximadamente 1/3 dos entrevistados que usam alguma tecnologia assistiva para acessar ferramentas educacionais *online* relatam essa experiência como “não confiável”, “inconsistente” ou “sem uso/acesso bem-sucedido”.

1.3 Objetivos

Os objetivos desse trabalho consistem em fazer um levantamento e selecionar abordagens de ensino atualmente utilizadas em programação, além de ferramentas para avaliação da motivação de alunos em cursos dessa natureza.

Ao terminar essa triagem, elaborar OAs atrativos e com a acessibilidade visual corrigida para cada abordagem selecionada, adaptar uma ferramenta de avaliação da motivação e construir um repositório para armazenar esses recursos.

1.4 Contribuições

O conjunto de OAs apresentado nesse trabalho serve como apoio para docentes em diferentes abordagens de ensino de programação. A acessibilidade desses OAs auxilia os alunos com deficiência visual. Além disso, a disponibilização da ferramenta para avaliação da motiva-

ção dos alunos contribui para uma melhor escolha de abordagem de ensino de programação. Essas ferramentas podem ser usadas em cursos de programação nas modalidades EAD, híbrida, ou presencial.

1.5 Organização do Texto

No Capítulo 1, encontrou-se a introdução com a contextualização, problemas de pesquisa, objetivos e contribuições. No Capítulo 2 são apresentados os conceitos utilizados no trabalho. No Capítulo 3, encontram os trabalhos relacionados a essa pesquisa. O Capítulo 4 mostra os passos de como esse trabalho foi feito e, por fim, os resultados e conclusões são apresentados no Capítulo 5.

2 CONCEITOS

Nesse capítulo encontra-se os conceitos utilizados nesse trabalho, como o raciocínio lógico, algoritmos e lógica de programação, OA, algumas abordagens para ensino de programação e um modelo para avaliação de motivação, conhecido como ARCS.

2.1 Raciocínio Lógico

De acordo com o dicionário de português licenciado para Oxford University Press, a lógica, em termos de filosofia, trata das formas do pensamento em geral (dedução, indução, hipótese, inferência etc.) e das operações intelectuais que visam à determinação do que é verdadeiro ou não.

Para Japiassú e Marcondes (2011), o raciocínio lógico examina como relacionamos inferencialmente as proposições para delas extrair conclusões. Raciocinar ou fazer inferências, segundo (MORTARI, 2001), é manipular a informação disponível, ou seja, aquilo que sabemos ou supomos ser verdadeiro, e extrair consequências disso, obtendo uma informação nova.

O raciocínio lógico ajuda a responder questões como:

- Se isso é verdade, o que mais deve ser verdade?
- Se isso é verdade, o que mais pode ser verdade?
- Se isso não é verdade, o que mais não pode ser verdade?

Exemplificando de modo mais compreensível, a partir de proposições como “Todo mamífero é um animal” e “Todo cavalo é um mamífero” pode-se deduzir corretamente que “Todo cavalo é um animal”. (FORBELLONE; EBERSPACHER, 2000)

O raciocínio lógico pode ser trabalhado com certos parâmetros, sendo: abstração, compreensão (interpretação), o número e suas relações, argumentação com base em critérios e em princípios logicamente validados e a expressão de ideias de forma lógica e organizada (OLIVEIRA; ROCHA, 2011).

A falta do desenvolvimento do raciocínio lógico pode ter consequências severas. Segundo (RAUBER *et al.*, 2003), é comum encontrar alunos universitários com dificuldade para interpretar o que estão lendo, por não terem sido alfabetizados para entender o que está “por trás” daquilo que está escrito, ou seja, o real significado e contexto.

O raciocínio lógico pode ser estimulado por meio de atividades simples e dinâmicas, como por exemplo, exercícios de matemática, jogos de quebra-cabeça e jogos de tabuleiros.

Na área da Computação, o raciocínio lógico é uma competência fundamental. Arboleda, Mazuera e Montemiranda (2015) apresentam 8 habilidades que facilitam o aprendizado inicial de programação, em ordem crescente de importância, incluindo o raciocínio lógico, sendo elas:

- Compreensão de leitura;
- Conhecimento de matemática;
- Descrever os passos necessários para a solução do problema;
- Dividir o problema;
- Capacidade de abstração;
- Modelar o problema;
- Pensamento lógico;
- Relatar condições de entrada e saída.

Entretanto, de acordo com o Ministério da Educação (MEC), do total de estudantes do terceiro ano do Ensino Médio que participaram do Sistema de Avaliação da Educação Básica (SAEB) no ano de 2017, somente 4,5% apresentaram aprendizagem adequada (entre os níveis 7 e 10) em Matemática, enquanto apenas 1,6% dos estudantes encontraram-se nessa faixa em Português (BRASIL, 2018). Isso indica que possivelmente, alunos ingressantes no Ensino Superior careçam de habilidades como o “Raciocínio lógico”, além de outras como a “Compreensão de leitura” e os “Conhecimentos de matemática”.

2.2 Algoritmos e Lógica de Programação

Um algoritmo é a descrição de um padrão de comportamento de um repertório finito de ações nomeadas das quais se assume, *a priori*, que são possíveis de serem feitas (DIJKSTRA, 1971). Algoritmos são ações sequenciais e executáveis para resolver um certo problema que pode estar presente no nosso cotidiano. Por exemplo, uma receita de bolo ou um manual de instalação de um aparelho podem ser considerados algoritmos.

A lógica de programação é a aplicação do raciocínio lógico para criar algoritmos que possam ser implementados, utilizando uma linguagem de programação (FORBELLONE; EBERSPACHER, 2000). Segundo (ROCHA *et al.*, 2010), a lógica de programação é um requisito fundamental nos cursos de computação, independente da metodologia e dos paradigmas utilizados, sendo um instrumento importante na estruturação do raciocínio lógico e na formulação de algoritmos corretos.

Alves (2014) afirma que:

> O estudo de algoritmos e de lógica de programação é essencial no contexto do processo de criação de um software. Ele está diretamente relacionado com a etapa de projeto de um software em que, mesmo sem saber qual será a linguagem de programação a ser utilizada, especifica-se completamente o

software a ponto de na implementação ser possível traduzir diretamente essas especificações em linhas de código em alguma linguagem de programação como pascal, C, Java e outras.

2.3 Objetos de Aprendizagem

A Institute of Electrical and Electronics Engineers (IEEE), organização responsável pelas normas que são implementadas nas áreas de engenharia elétrica e informática, define que um OA é qualquer entidade, digital ou não, que pode ser utilizada, reutilizada ou referenciada durante o processo de aprendizagem que utilize tecnologia.

São exemplos de OA: conteúdos multimídia, conteúdos instrucionais, objetivos de ensino, software instrucional e software em geral e pessoas, organizações ou eventos referenciados durante um ensino com suporte tecnológico.

Alguns pesquisadores discutem a interatividade dos OAs e afirmam que, para realizar uma aprendizagem significativa, é necessário que o aluno interaja com o ambiente e que para estabelecer uma verdadeira interatividade, o aluno precisa se sentir participante da ação.

Assim, a atuação do professor é bastante relevante. Segundo (BRAGA, 2014), a seleção dos OAs deve ser feita pelo professor adequadamente, conforme o conteúdo que ele deseja abordar e os objetivos que ele queira alcançar.

2.4 Abordagens para Ensino de Programação

Muitos autores se referem a diversos métodos e abordagens e propõem uma taxonomia para o ensino de programação.

Szlávi e Zsakó (2003) apresentam uma lista com os dez métodos mais disseminados para o ensino de programação:

- Metódica, Orientada a Algoritmos, onde o método engloba todo o processo de programação e a principal importância está na elaboração do algoritmo;
- Orientado a Dados, que possui uma semelhança com o método anterior, porém a principal importância são as estruturas de dados e os refinamentos de tipos;
- Orientado a Problemas, onde a programação é vista como uma atividade global que não pode ser separada em partes, sempre sendo vista como um processo todo, inicia-se definindo um problema para resolver com a programação;
- Orientado a Especificações, que considera a especificação formal como a parte mais importante do processo e o algoritmo é derivado da especificação;

- Orientado a Linguagens, que é um dos métodos mais antigos e está relacionada intimamente com uma linguagem de programação específica;
- Orientado a Instruções, que é semelhante ao método anterior com a exceção que é baseado nas instruções que a maioria das linguagens possuem e não em uma linguagem em específico;
- Orientado a Matemática, que tem como objetivo resolver questões de outros tópicos utilizando a programação;
- Orientado a Hardware, onde assume que para ter conhecimentos de algoritmos há a necessidade de entender aspectos de hardware, como Assembly, linguagem de máquina, funcionamento de processadores etc;
- Orientado a Modelo, onde modelos de algoritmos ou programação de código são introduzidos aos alunos, que adquirem o conhecimento estudando esses modelos.

De acordo com os autores, somente os dois primeiros métodos são voltados para aqueles que concluíram o ensino médio e querem atuar na área de informática. O terceiro método (Orientado a Problemas), por sua vez, é recomendado para qualquer nível de educação.

Os autores afirmam, ainda, que o restante dos métodos não é recomendado para o ensino de programação, pois são desatualizados como, por exemplo, os métodos Orientado a Linguagem, Orientado a Instruções e Orientado a Hardware; ou necessitam de um grande conhecimento prévio, como o Orientado a Matemática e o Orientado a Especificações; ou não é voltado para educar programadores profissionais, como o Orientado a Modelo. Assim, a maior parte dos métodos listados não são adequados para iniciantes.

Em seu estudo, Selby (2011) apresenta quatro abordagens para o ensino de programação:

- Análise de Código, onde os alunos analisam códigos existentes antes de produzir os seus próprios;
- Uso de Blocos, análoga ao Aprendizado de Vocabulário, onde os alunos aprendem os substantivos e verbos antes de construir as sentenças;
- Unidades Simples, nas quais os alunos dominam os problemas simples antes de resolver os problemas mais complexos;
- Sistemas Completos, em que os alunos projetam uma solução para um problema não trivial e os conceitos de programação e construções de linguagem são introduzidos apenas quando a solução para o problema requer sua aplicação.

Selby (2011) afirma, ainda, que a competência em programação pode ser alcançada a partir da combinação do domínio parcial de cada uma das abordagens.

Mohorovičić e Strčić (2011) apresentam, de forma geral, os diversos métodos de ensino de programação utilizados em cursos superiores. A classificação proposta por eles inclui:

- Aprendizagem Baseada em Problemas, em que o foco é o engajamento do aluno na resolução dos problemas;
- Aprendizagem Baseada em *Puzzle*, que visa ensinar raciocínio crítico e resolução de problemas aos estudantes;
- Programação em Pares, que é um estilo de programação em que dois programadores trabalham lado a lado, no mesmo código e na mesma máquina;
- Aulas Pré-gravadas, que são compostas de apresentações feitas em computador narradas e servem para complementar as aulas dadas;
- *Game-Themed Programming*, em que o foco é fazer com que os alunos aprendam sobre conceitos de programação explorando e programando pequenas aplicações de jogos.

Mohorovičić e Strčić (2011) concluem que a diferença entre essas abordagens é o foco. Enquanto algumas se preocupam em como o conceito é apresentado, outras se preocupam em como trabalhar com esses conceitos. Afirmam, ainda, que essas abordagens possuem mais semelhanças do que diferenças. A principal semelhança é a tentativa de aumentar a motivação dos alunos, reduzindo a frustração diante os conceitos e abstrações desafiadoras, e encorajar os alunos a participarem mais nas tarefas. Além disso, todos os métodos visam melhorar a qualidade dos cursos de computação já existentes e têm como objetivo tornar os conceitos de programação mais compreensíveis.

Em vez de classificar abordagens existentes, alguns autores propõem a sua própria, como Jackson e Miller (2009), que desenvolveram uma nova disciplina no curso de programação chamada de Elementos de Construção de Software (*Elements of Software Construction*), na qual os alunos desenvolvem três projetos. Em todos esses projetos, inicialmente é ensinado aos alunos como articular o problema a ser resolvido, em uma notação de modelagem leve e adequada, e após, é ensinado como implementar a solução, usando uma coleção de padrões de design que capturam a experiência típica dos desenvolvedores.

Outro exemplo de abordagem de autoria própria é de Guzdial (2013) que sugere um método onde os alunos manipulam imagens, áudios e vídeos nas primeiras tarefas de programação e conclui que aumentou a retenção dos alunos, tanto na parte de conhecimento dos alunos como na permanência deles no curso.

Nesse trabalho, o termo “abordagens” se refere às diferentes formas de ensinar algum conteúdo específico. Existem diversas maneiras para ensinar programação e cada uma possui suas vantagens e desvantagens.

A seguir são apresentadas as abordagens de ensino de programação que foram identificadas e classificadas, de acordo com sua popularidade, a partir de uma revisão bibliográfica.

2.4.1 Ensino Tradicional

Definimos como Ensino Tradicional o que é visto normalmente em salas de aula: Um professor explica o conteúdo da matéria aos alunos, escrevendo os termos importantes em um quadro negro enquanto os alunos fazem suas anotações.

Esse estilo de ensino existe há muito tempo, desde o século 19, e continua sendo utilizado. Mizukami (1986) afirma que no Ensino Tradicional, as situações de sala de aula recebem destaque, onde os alunos são instruídos e ensinados pelo professor. Os conteúdos e as informações precisam ser adquiridos e os modelos imitados. No Ensino Tradicional, o aluno é basicamente um ser passivo. Nesse relacionamento entre o professor e o aluno, o professor é o agente e o aluno, o ouvinte.

Na área de computação, geralmente ocorre a teorização dos conteúdos de introdução de programação, com a apresentação dos conceitos aos alunos pelo professor, com aplicação de provas e listas de exercícios como método de avaliação da aprendizagem (BERSSANETTE, 2016).

Esses métodos tradicionais podem não ser tão atrativos aos alunos. Freeman *et al.* (2014) apontam que os métodos de ensino em que os alunos participam mais ativamente reduzem as taxas de reprovação e melhoram as notas comparados aos métodos tradicionais.

2.4.2 Programação Baseada em Blocos

A Programação Baseada em Blocos utiliza o conceito de peças de quebra cabeça que fornecem dicas para facilitar a composição do programa (WEINTROP; WILENSKY, 2017).

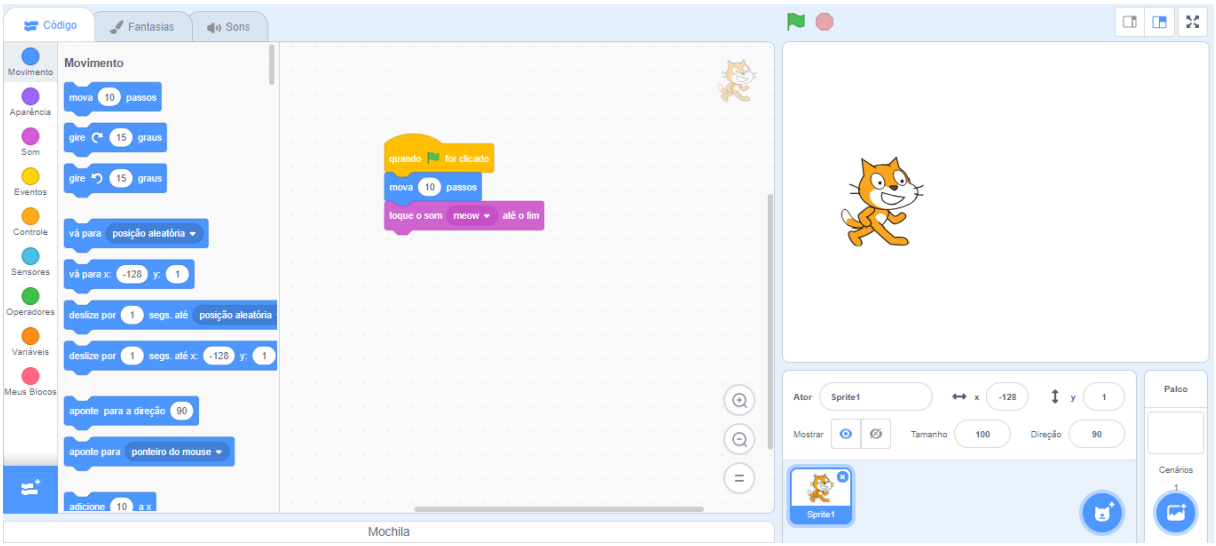
Essa abordagem é muito utilizada para introduzir conceitos de programação, principalmente para crianças, e funciona combinando blocos de instruções já prontas. Alguns exemplos de ferramentas que utilizam essa abordagem são o Scratch[1], Alice[2], Blockly[3], entre muitos outros.

Para ilustrar esse estilo de programação, a interface do Scratch é apresentada na Figura 1. Essa interface é dividida em três partes: na área esquerda são disponibilizados os blocos de programação; na área central esses blocos são combinados; e na área direita, o resultado da execução desse código é mostrado.

Segundo (MEERBAUM-SALANT; ARMONI; BEN-ARI, 2011), os blocos de programação disponíveis são encorajadores e incentivam os alunos a se aventurar na codificação, porém

[1] https://scratch.mit.edu/
[2] https://www.alice.org/
[3] https://developers.google.com/blockly/

Figura 1 – Interface de Programação do Scratch



Fonte: Autoria própria (2021).

podem criar hábitos indesejados, como a programação *bottom-up*, onde ao montar um sistema, o programador inicia com instruções pequenas que são combinadas até formar um subsistema maior e isso é repetido até que o sistema inteiro seja completo.

2.4.3 Aprendizagem Baseada em Jogos Digitais

A aprendizagem baseada em jogos é uma metodologia pedagógica que concentra na concepção, desenvolvimento, uso e aplicação de jogos na educação e na formação (CARVALHO, 2015).

Na área da programação, há diversos jogos disponíveis como o CodeMonkey[4] ou CodeCombat[5], entre outros. Os elementos presentes nos jogos, como a história, desafios, recompensas podem ser mais atrativos para o discente. As taxas de concentração de alguém que está interagindo com um jogo tendem a alcançar níveis mais elevados comparado a outras metodologias que visam manter o foco do estudante a longo prazo (ABREU; PAULA, 2013).

Para exemplificar a abordagem, é apresentado a interface do CodeCombat. No CodeCombat, encontra-se um mapa, presente na Figura 2, com diversos continentes, onde cada um deles aborda algum conceito de programação.

Conforme as fases de um continente são concluídas, o próximo continente vai sendo desbloqueado e assim, mais conceitos de programação são aprendidos pelo jogador.

Para a conclusão das fases, é possível a seleção de personagens, de linguagem de programação e de itens, como mostra a Figura 3. Há diversas opções de personagens e 3

[4] https://www.playcodemonkey.com/
[5] https://br.codecombat.com/play

Figura 2 – Mapa do CodeCombat

Fonte: Autoria Própria (2021).

opções de linguagens, Python, JavaScript, CoffeeScript e as instruções são disponibilizadas conforme a escolha dos itens.

Figura 3 – Tela de Seleção do CodeCombat

(a) **(b)**

Fonte: Autoria Própria (2021).

A interface de codificação do CodeCombat é dividida em 3 partes, como pode ser visto na Figura 4. Na coluna esquerda estão as missões que devem ser concluídas, que nesse caso são “Evite os espinhos” e “Colete a gema”; na coluna central há os comandos disponíveis para completar a fase e, na coluna direita encontra-se o campo de codificação para inserir os comandos necessários.

Uzunca e Jansen (2016) apresentam a aprendizagem com jogos que simulam tarefas significativas, como cirurgias médicas e cuidado de pacientes. Os benefícios dessa abordagem são: a aprendizagem por realidade virtual e a simulação de papéis, como médicos e enfermeiros; a aprendizagem prática, envolvente, motivadora e divertida; a capacidade de descobrir e

Figura 4 – Tela de Codificação do CodeCombat

Fonte: Autoria Própria (2021).

utilizar os conhecimentos e não somente memorizá-los; a simulação de ambientes complexos/ custosos e situações perigosas/críticas.

Os jogos feitos para a aprendizagem precisam considerar fatores como a manutenção do interesse e da motivação dos estudantes e, ao mesmo tempo, garantir o correto entrelaçamento entre os objetivos instrucionais e a jogabilidade (SENA *et al.*, 2016).

2.4.4 Práticas Desplugadas

As Práticas Desplugadas têm como objetivo ensinar os fundamentos da Ciência da Computação, principalmente em crianças, sem a necessidade de utilizar computadores.

Bell, Witten e Fellows (2002) afirmam que as grandes vantagens dessa abordagem são a independência de recursos de hardware ou software, podendo ser aplicadas também em lugares com acesso precário de infraestrutura e podem ser ministradas por não especialistas em computação. As Práticas Desplugadas são uma boa forma de ensinar o pensamento computacional de maneira descontraída.

Um exemplo de atividade de Práticas Desplugadas é apresentado na Figura 5. Nessa atividade os alunos aprendem os métodos de ordenação utilizando recipientes de areia ou água com pesos diferentes.

Um problema dessa abordagem é que apenas o pensamento computacional é ensinado, ou seja, conceitos e linguagens de programação não são levados em consideração.

2.4.5 Aprendizagem Baseada em Problemas

A Aprendizagem Baseada em Problemas - Problem Based Learning (PBL), enfatiza a atividade dos próprios alunos no aprendizado de problemas, na definição de seus próprios ob-

Figura 5 – Exemplo de Atividade Desplugada

Folha de Atividade: Ordenando pesos

Objetivo: Encontrar o melhor método para ordenação de um grupo de pesos desconhecidos

Você precisará de: Areia ou água, 8 recipientes idênticos e um conjunto de balanças

O que fazer:

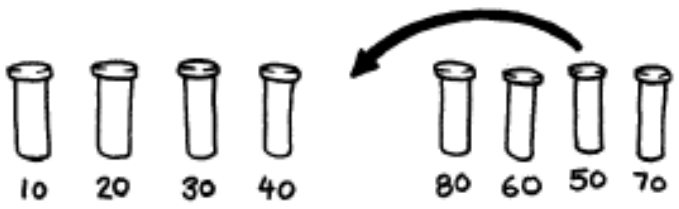
1. Encher cada recipiente com uma quantidade diferente de areia ou água. Sele-o firmemente.
2. Misture-os de modo que você já não saiba a ordem dos pesos.
3. Encontre o menor peso. Qual é a maneira mais fácil de fazer isso ?

   Nota: Você só está autorizado a utilizar a balança para descobrir quanto pesa cada recipiente. Apenas dois pesos podem ser comparados ao mesmo tempo.
4. Escolha 3 pesos aleatoriamente e coloque-os na ordem do mais leve para o mais pesado usando somente a balança. Como você fez isso ? Qual é o número mínimo de comparações que você pode fazer ? Por quê ?
5. Agora coloque todos os objetos na ordem do mais leve ao mais pesado.

Quando você achar que terminou, verifique sua ordenação pesando novamente cada par de objetos.

Ordenação por Seleção

Um método que o computador pode utilizar é chamado de *ordenação por seleção*, que funciona da seguinte forma. Primeiro, encontre o peso mais leve no conjunto e o coloque de lado. Em seguida, encontre o mais leve dos pesos restantes e o retire. Repita esse procedimento até que todos os pesos sejam removidos.



Fonte: Bell, Witten e Fellows (2002).

jetivos de aprendizagem e na busca ativa e análise de informações (NUUTILA; TÖRMÄ; MALMI, 2005).

Segundo (KODJAOGLANIAN *et al.*, 2003), na abordagem PBL, os alunos são ativos e os professores são substituídos por tutores, que são facilitadores de aprendizagem. O papel do

tutor é passivo, ele tira as dúvidas quando for solicitado e guia os alunos quando estiverem indo ao caminho errado.

A PBL inicialmente foi aplicada na área de Medicina, mas escolas de outras áreas, como economia e engenharia, foram aderindo, demonstrando que esse método é aplicável a qualquer ramo do conhecimento (KODJAOGLANIAN *et al.*, 2003).

Uma aplicação na área da Computação feita por Nuutila, Törmä e Malmi (2005) foi um curso introdutório de programação na linguagem Java utilizando a PBL, onde um grupo de 7 a 10 estudantes com 1 tutor realizavam reuniões semanais de 30 minutos a 1 hora para discutir o assunto dado como tarefa. O objetivo era que, ao final do curso, o estudante aprendesse o básico de programação orientada a objetos e conseguisse criar independentemente aplicativos Java com interfaces gráficas voltadas ao usuário.

Outro exemplo é o de Silva (2017) que desenvolveram OAs para o ensino de computação voltados a alunos de engenharias (Engenharia Ambiental, Engenharia de Alimentos e Engenharia Civil), utilizando conceitos como vigas e treliças. Os alunos organizaram-se em grupos de maneira que houvesse pelo menos um aluno da Engenharia Civil em cada grupo e foram atribuídos alguns papéis importantes, o Líder que organiza o grupo e o Relator que realiza as notações. Dois OAs, com duas atividades cada, foram utilizados. Atribuíram a maior parte das aulas para os alunos debaterem sobre o problema dado. Ao final da aula, reservaram um tempo de debate entre professor e alunos e quaisquer dúvidas eram respondidas nesse tempo. Além disso, o relator devia entregar um relatório parcial ao final da aula e no dia seguinte, um e-mail com o diagrama de corpo livre e o algoritmo desenvolvido para a solução dos problemas devia ser enviado ao professor. Foi concluído que os alunos cumpriram os objetivos estabelecidos nas atividades e que o método auxiliou no entendimento do conteúdo.

2.4.6 Robótica Educacional

Segundo (PEREIRA, 2010), a Robótica Educacional, ou Robótica Pedagógica, é aplicada em ambientes educacionais onde o aluno pode montar, desmontar, programar e reprogramar um robô ou sistema robotizado.

A Robótica Educacional faz a utilização de microcontroladores, como Arduino e Raspberry Pi, ou robôs programáveis para o ensino de programação. Um exemplo de ferramentas são os kits robóticos do Lego Mindstorms[6], muito utilizados no aprendizado de programação voltada para crianças.

Essa ferramenta pode ser programada em diversas linguagens utilizando plataformas específicas. Por exemplo, em Java, há uma pequena máquina virtual Java chamada LeJOS[7]

[6] https://www.lego.com/en-us/mindstorms/products/mindstorms-ev3-31313
[7] http://www.lejos.org/ev3.php

que possibilita a programação nos robôs; o RobotC[8], uma linguagem baseada em C; Swift Playgrounds[9], um ambiente para a linguagem Swift, porém somente disponível para iPad.

Uma ferramenta interessante é o ev3dev[10], um sistema operacional que executa nas plataformas Lego Mindstorms e no Raspberry Pi, possibilitando o uso de diversas linguagens, como Python, Java, C/C++ e Go.

O kit do Lego Mindstorms EV3, apresentado na Figura 6, possui um “bloco” programável, sensores de cor, toque e infravermelho, sinalizador infravermelho remoto e motores, além de 14 adesivos decorativos e um manual de instruções.

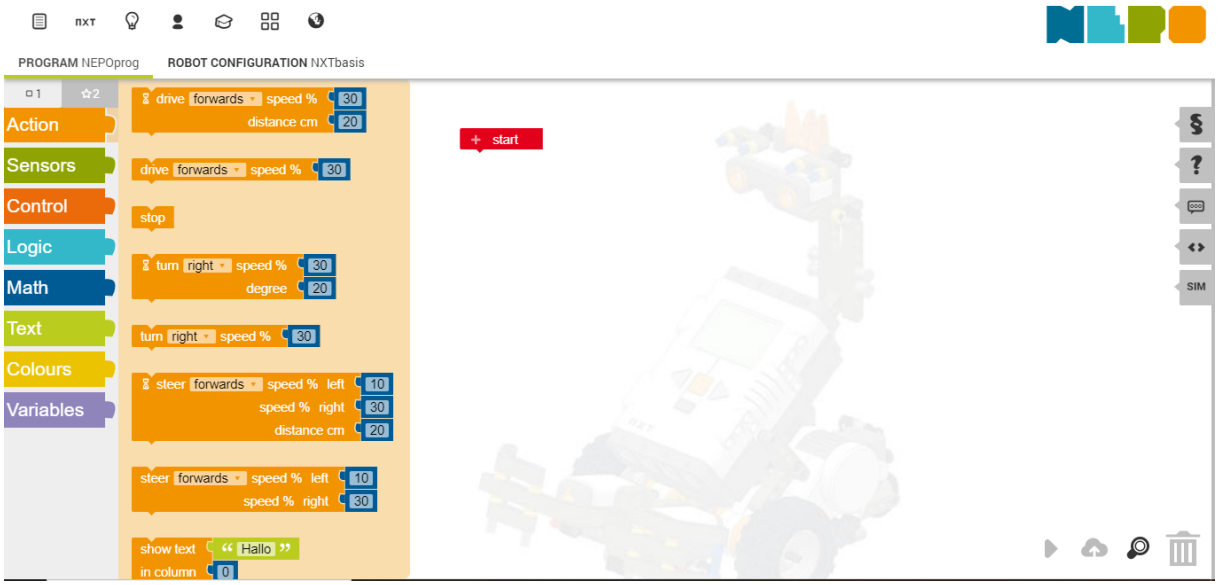
Figura 6 – Kit do Lego EV3

Fonte: LEGO (2020).

Outra ferramenta interessante para a Robótica Educacional é o Open Roberta Lab, uma ferramenta *online* que possui a estrutura de programação em blocos, do projeto da iniciativa educacional alemã “Roberta—Learning with robots”. Um ponto positivo dessa ferramenta é a sua compatibilidade com diversos sistemas, como o mBot, WeDo, micro:bit, Lego EV3, entre outros. A Figura 7 mostra a interface do Open Roberta Lab. No canto esquerdo da tela estão os comandos em blocos divididos por comportamento e esses blocos são combinados na área central.

8 http://www.robotc.net/
9 https://www.apple.com/swift/playgrounds/
10 https://www.ev3dev.org/

Figura 7 – Tela de Codificação do NXT no Open Roberta Lab



Fonte: Autoria Própria (2021).

O Open Roberta Lab também possui um simulador onde é possível testar o comportamento do código feito, diminuindo a necessidade de carregar diversas vezes o código para testá-lo.

Yu (2012) fez o uso dos robôs do Lego Mindstorms em um curso de Tecnologia da Informação, realizado em laboratório durante duas semanas, no qual estudantes trabalhavam na criação de projetos divididos em pequenos grupos. Os problemas resolvidos pelos alunos foram o segue-linha, na qual o robô precisa andar sobre o caminho traçado; o procure e resgate, em que o robô precisa achar um objeto e trazê-lo de volta; e o futebol RoboCup, um futebol de robôs. Para a programação dos robôs, foi utilizado o NXC, uma versão simplificada da linguagem C.

2.4.7 Outras Abordagens

Existem outras abordagens como o *Dojo* de Programação, onde são realizadas reuniões semanais entre um grupo de programadores que se juntam para aprender, praticar e compartilhar experiências. Essas reuniões são organizadas em torno de desafios de programação, incentivando os programadores a participar e mostrar as suas habilidades enquanto resolvem o problema (SATO; CORBUCCI; BRAVO, 2008).

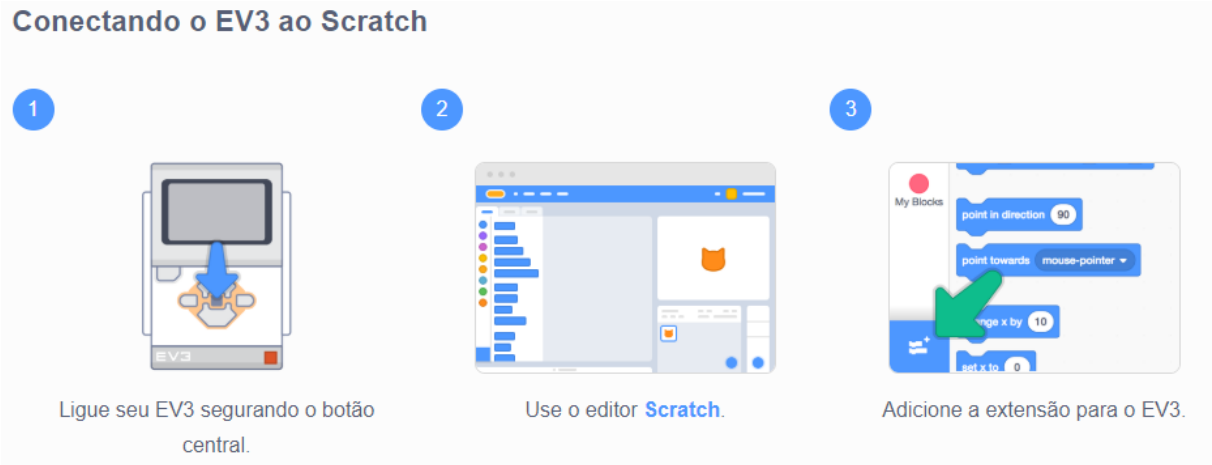
Há práticas que podem ajudar em uma melhor aprendizagem como a Programação em Pares (*Pair Programming*), onde a programação é feita em duplas de alunos. Enquanto um aluno codifica, o outro observa, opina e aponta possíveis erros. Zacharis (2011) define Programação em Pares como um estilo de programação colaborativa onde dois programadores sentam lado a lado no mesmo computador, discutem sobre o que eles estão prestes a programar, analisar,

projetar e depois escrever o código. Após a escrita do código, também participam da fase de teste e da depuração, participando constantemente do processo. Zacharis (2011) comparou os papéis dos programadores aos de “piloto” e de “navegador”, em que o piloto digita o código e sempre explica ao navegador o que ele está fazendo e o que está pensando.

2.4.8 Combinação de Abordagens de Ensino

É possível combinar as abordagens de ensino. Há uma extensão do Scratch[11] que possibilita a combinação da ferramenta de Programação em Blocos com o Lego Mindstorms, exemplo da Robótica Educacional. A conexão do Lego Mindstorms com o Scratch é apresentada na Figura 8.

Figura 8 – Conexão do Lego EV3 com o Scratch



Fonte: LEGO (2020).

Rebouças *et al.* (2010) combinaram o Ensino Tradicional e a Aprendizagem Baseada em Jogos Digitais para propor um curso de programação no ensino médio que, ao final, os alunos constroem jogos de Química, Português e Geometria utilizando a linguagem Python. Foi realizado um planejamento e preparação do curso para ensinar a linguagem de programação aos alunos e implementado os jogos educacionais. Rebouças *et al.* (2010) concluem que a utilização de jogos pode ser um forte fator motivacional, pois tem servido em cursos introdutórios de programação nas universidades. Além disso, os jogos desenvolvidos podem ajudar os alunos a praticar os conteúdos das matérias de aula de forma mais atrativa.

Schumacher, Welch e Raymond (2001) utilizaram os robôs Lego Mindstorms e a PBL para ensinar conceitos básicos de TI e programação para os alunos da Academia Militar dos Estados Unidos na disciplina de Introdução à Computação. Foi utilizado um ambiente criado pelos próprios autores, que possibilita a programação em Java e a simulação com os robôs. Os alunos criaram um algoritmo em que o robô transitava com sucesso um labirinto de testes, que

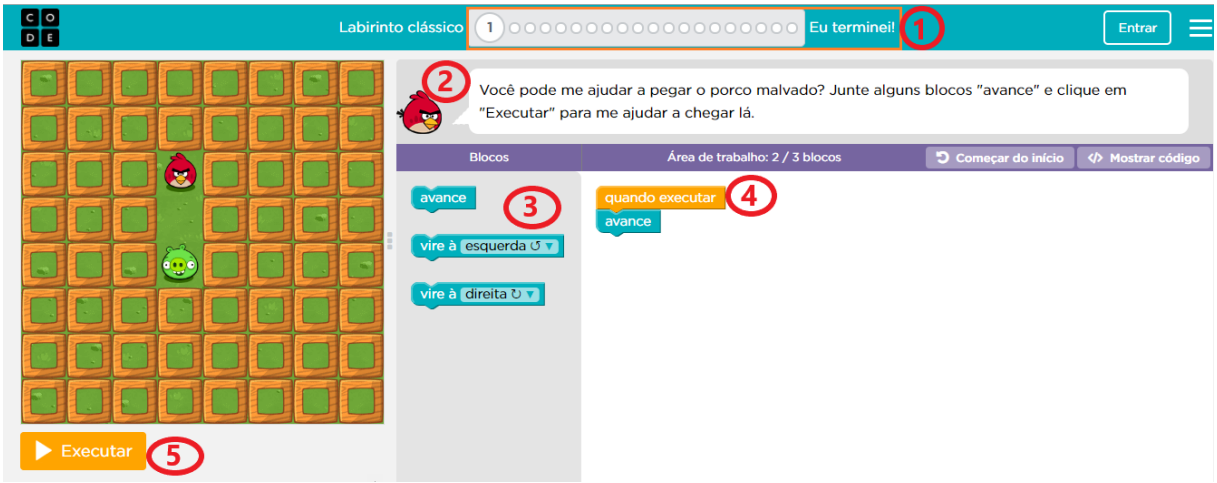
[11] https://scratch.mit.edu/ev3

simula um possível cenário no Iraque, onde o objetivo é encontrar um esconderijo iraquiano no menor tempo possível. A maioria dos alunos conseguiu finalizar o projeto com sucesso.

Um caso interessante de combinação de métodos pode ser visto no contexto do projeto Hora do Código[12]. A Hora do Código é um evento anual para celebrar a Ciência da Computação que ocorre no mundo inteiro organizado pela Code.org[13], uma organização sem fins lucrativos apoiada por empresas e pessoas importantes da área da tecnologia, como a Google, Microsoft, Drew Houston (CEO do Dropbox), Reid Hoffman (Co-fundador do LinkedIn), entre muitas outras, dedicada a expandir o acesso à Ciência da Computação em escolas e aumentar a participação das mulheres e das minorias não representadas.

Os projetos desse evento combinam Programação em Blocos e Aprendizagem Baseada em Jogos. A interface de um desses projetos[14] é apresentada na Figura 9.

Figura 9 – Projeto da Hora do Código



Fonte: Code.org (2020).

No exemplo dessa Figura, na área 1 são apresentados a fase atual e o avanço do jogador ao longo do projeto. Na área 2 há uma breve descrição do que deve ser feito na fase corrente. Os blocos de instruções disponíveis estão na área 3 e são combinados na área 4. Ao selecionar o botão Executar (5), o personagem se comporta de acordo com o que foi programado e isso pode ser visualizado na tela.

É interessante notar a mensagem contida na caixa de texto “Área de Trabalho”, que mostra a quantidade de blocos utilizada até o momento, e a quantidade ideal para concluir a fase.

A combinação de diferentes abordagens de ensino visa garantir as vantagens de todas as abordagens utilizadas, podendo tornar a aprendizagem mais interessante.

[12] https://hourofcode.com/br/pt
[13] https://code.org/international/about
[14] https://studio.code.org/hoc/1

2.5 Modelo ARCS

ARCS é nome de um modelo criado por Keller (1987) com o objetivo de encontrar formas mais eficazes de compreender as principais influências sobre a motivação de aprender e de encontrar métodos sistemáticos de identificar e resolver problemas com motivação de aprendizagem.

O nome ARCS é o acrônimo dos termos "*Attention*", "*Relevance*", "*Confiance*" e "*Satisfaction*" (Atenção, Relevância, Confiança e Satisfação). Segundo Keller (1987), cada termo representa um aspecto da motivação, além de condições que devem ser satisfeitas para que as pessoas se tornem motivadas e permaneçam nesse estado.

A **Atenção** é um pré-requisito para a aprendizagem. Além de despertar a atenção, é importante mantê-la durante todo o processo e direcioná-la para os estímulos apropriados. As estratégias sugeridas pelo próprio modelo incluem a participação ativa do aluno, a aplicação de doses de humor, a utilização de exemplos específicos e a resolução de exercícios por parte dos alunos (KELLER, 1987).

A **Relevância** consiste em mostrar que o conteúdo que está sendo abordado é pertinente e que há um significado em estudá-lo. Para isso, o modelo sugere práticas como relacionar o tópico novo com os conhecimentos prévios, mostrar vantagens de aprender o assunto e apresentar suas utilidades futuras.

A **Confiança** é obtida de forma fácil fornecendo objetivos claros, exemplos de sucesso e estabelecendo expectativas positivas aos alunos. A Confiança é um fator importante para a persistência dos estudantes. As práticas sugeridas são encorajar os alunos, mostrando que eles são capazes de dominar o assunto, usar metas significativas e fornecer *feedback* e apoio.

Para obter a última condição, a **Satisfação**, é necessário que os alunos tenham sentimentos positivos, que podem ser adquiridos por meio de recompensas e reconhecimento. A Satisfação é percebida quando o aluno sente-se realizado com a experiência de aprendizado, isso é, quando o esforço dos estudos foi apropriado e houve consistência entre os objetivos, o conteúdo e os testes. Exemplos de práticas são fazer o aluno ter algum sentimento de conquista, não ser indulgente com os alunos e fornecer oportunidades para usar o novo conhecimento.

Keller (2009) apresenta dois modelos para medir o ARCS: o Course Interest Survey (CIS) e o Instructional Materials Motivation Survey (IMMS). O CIS foi projetado com o objetivo de medir a motivação dos estudantes em relação a um curso ou disciplina ministradas por um instrutor e o IMMS para medir reações a um material didático autodirigido.

As perguntas desses questionários foram escolhidas a partir de um conjunto de itens potenciais elaborados, revisados e testados para aprimorar os conceitos chaves e evitar ambiguidades e falsas respostas.

Uma vez que o IMMS é utilizado para materiais voltados à autoaprendizagem, nesse trabalho será abordado somente o CIS, pois pressupomos a existência de um educador.

2.6 Questionário CIS

O CIS é um questionário de 34 itens desenvolvido por Keller (2009), distribuídos quase igualmente entre os aspectos do modelo ARCS. As respostas desse questionário utilizam o modelo de cinco pontos da escala Likert. As questões do CIS, a distribuição das questões e a pontuação são apresentadas no Apêndice A.

A análise dos resultados do CIS são realizadas utilizando dados estatísticos como média, desvio padrão e variância, comparando os valores obtidos. Essas comparações podem ser feitas em grupos de controle/teste ou entre diferentes abordagens.

As pontuações podem ser calculadas separadamente, por cada aspecto do ARCS, ou pelo escore total do questionário. No modelo original não é especificado uma pontuação mínima desejada, mas muitos pesquisadores consideram a média 3.0 de 5.0 como estimativa, sendo acima de 3.0 uma boa pontuação. ((HUETT, 2006; SAITO *et al.*, 2017; PEREIRA; BITTENCOURT, 2020))

Além desses valores, o alfa de Cronbach é utilizado para estimar a confiabilidade do questionário.

2.6.1 Alfa de Cronbach

Para avaliar a consistência interna de um questionário aplicado em uma pesquisa, muitos pesquisadores utilizam o valor do coeficiente do alfa de Cronbach.

Streiner (2003) define a consistência interna como um índice que aponta se todas as subpartes de um instrumento medem um mesmo atributo.

Segundo (KESZEI; NOVAK; STREINER, 2010), um valor de consistência interna baixa pode significar que os itens estão medindo construtos diferentes ou que as respostas das questões estão inconsistentes. Ou seja, uma consistência interna alta em um questionário significa que as perguntas foram elaboradas de forma coerente, respondendo os objetivos do questionário.

O alfa de Cronbach é calculado utilizando a Equação 1:

$$\alpha = \frac{k}{k-1}\left[1 - \frac{\sum_{i=1}^{k}]S_i^2}{S_{soma}^2}\right] \tag{1}$$

Onde o k representa o número de questões presentes no questionário, o n, o número de respondentes, s^2i é a variância dos n escores das pessoas a i-ésima questão ($i = 1, ..., k$), S^2soma é a variância dos totais $Tj(j = 1, 2,...,n)$ dos escores de cada respondente. A variância é calculada utilizando a Equação 2:

$$S^2 = \frac{\sum_{(} x - \bar{x})^2}{n-1} \tag{2}$$

A Tabela 1 mostra os valores do alfa de Cronbach relacionados à consistência interna segundo Landis e Koch (1977).

Tabela 1 – Consistência Interna Segundo o Valor de Alfa de Cronbach

O alfa de Cronbach	Consistência Interna
Maior do que 0,80	Quase perfeito
De 0,80 a 0,61	Substancial
De 0,60 a 0,41	Moderado
De 0,40 a 0,21	Razoável
Menor do que 0,21	Pequeno

Fonte: Landis e Koch (1977).

Keller (2009) utilizou o alfa de Cronbach para validar o CIS e esses valores são apresentados na Tabela 2.

Tabela 2 – Valores de Referência da Consistência Interna do CIS

Fatores	Estimativa de Confiabilidade (Alfa de Cronbach)
Atenção	0.84
Relevância	0.84
Confiança	0.81
Satisfação	0.88
Total	0.95

Fonte: Keller (2009).

Conforme os valores de referência apresentados na Tabela 1, o questionário CIS de Keller (2009) possui uma consistência interna extremamente alta, quase perfeita, em todos os aspectos do ARCS.

2.7 Acessibilidade

A acessibilidade é definida como:

> A possibilidade e condição de alcance para utilização, com segurança e autonomia, de espaços, mobiliários, equipamentos urbanos, edificações, transportes, informação e comunicação, inclusive seus sistemas e tecnologias, bem como de outros serviços e instalações abertos ao público, de uso público ou privados de uso coletivo, tanto na zona urbana como na rural, por pessoa com deficiência ou com mobilidade reduzida (BRASIL, 2015).

Já o termo acessível é definido como espaços, mobiliários, equipamentos urbanos, edificações, transportes, informação e comunicação, inclusive seus sistemas e tecnologias ou elemento que possa ser alcançado, acionado, utilizado e vivenciado por qualquer pessoa.

No Brasil, o estudo de Oliveira *et al.* (2012) mostrou que quase 1/4 da população brasileira possui algum tipo de deficiência visual, auditiva, motora e mental ou intelectual.

Rios *et al.* (2016) apresenta em seu trabalho a necessidade da garantia, tanto de um espaço físico acessível, como fornecer a acessibilidade em materiais didáticos digitais e em Ambiente Virtual de Aprendizagem (AVA), dado que é preciso empoderar o estudante e consolidar sua autonomia e aprendizagem.

2.7.1 Acessibilidade Digital

A acessibilidade digital é a eliminação de barreiras na Web, pressupondo que sites e portais sejam projetados de forma que possam ser percebidos, entendidos, navegados e interagidos por todos.

O World Wide Web Consortium (W3C)[15] desenvolve padrões para a web, como HTML, CSS, SVG e muitos outros. Além desses, padrões como o Web Content Acessibility Guidelines (WCAG) foram desenvolvidos com o objetivo de tornar as páginas web acessíveis a todos.

Os padrões do WCAG são divididos em camadas de orientação: princípios, diretrizes e critérios de sucesso. A base da acessibilidade web é constituída por 4 princípios: perceptível, operável, compreensível e robusto. Abaixo dos princípios estão as 12 diretrizes que são os objetos básicos que devem ser atingidos para fornecer a acessibilidade. Os princípios e as diretrizes são representados da seguinte forma:

1. Perceptível
    a) Alternativas em texto
    b) Multimídia baseada em tempo
    c) Adaptável
    d) Discernível
2. Operável
    a) Acessível por teclado
    b) Tempo suficiente
    c) Ataques epiléticos
    d) Navegável
    e) Modalidade de Entrada
3. Compreensível
    a) Legível
    b) Previsível

[15] https://www.w3.org/

c) Assistência a entrada

4. Robusto

a) Compatível

Dentro dessas divisões estão as recomendações, chamadas de critérios de sucesso, que são divididas em 3 níveis, A, AA e AAA, de forma crescente pelo grau de conformidade.

O Quadro 10 mostra a relação dos princípios, das diretrizes e dos critérios de sucesso.

Quadro 10 – Relação dos Princípios, Diretrizes e Critérios de Sucesso do WCAG 2.1

Princípios	Diretrizes	A	AA	AAA
1. Perceptível	1.1. Alternativas em texto	1.1.1		
	1.2. Multimídia baseada em tempo	1.2.1 - 1.2.3	1.2.4 - 1.2.5	1.2.6 - 1.2.9
	1.3. Adaptável	1.3.1 - 1.3.3	1.3.4 - 1.3.5	1.3.6
	1.4. Discernível	1.4.1 - 1.4.2	1.4.3 - 1.4.5, 1.4.10 - 1.4.13	1.4.6 - 1.4.9
2. Operável	2.1. Acessível por teclado	2.1.1 - 2.1.2, 2.1.4		2.1.3
	2.2. Tempo suficiente	2.2.1 - 2.2.2		2.2.3 - 2.2.6
	2.3. Ataques epiléticos	2.3.1		2.3.2 - 2.3.3
	2.4. Navegável	2.4.1 - 2.4.4	2.4.5 - 2.4.7	2.4.8 - 2.4.10
	2.5. Modalidade de Entrada	2.5.1 - 2.5.4		2.5.5 - 2.5.6
3. Compreensível	3.1 Legível	3.1.1	3.1.2	3.1.3 - 3.1.6
	3.2. Previsível	3.2.1 - 3.2.2	3.2.3 - 3.2.4	3.2.5
	3.3. Assistência a entrada	3.3.1 - 3.3.2	3.3.3 - 3.3.4	3.3.5 - 3.3.6
4. Robusto	4.1. Compatível	4.1.1 - 4.1.2	4.1.3	

Fonte: Adaptado de World Web Consortium (W3C) (2020).

Os critérios mais simples que representam as barreiras mais significativas de acessibilidade encontram-se no nível A. Somente com os critérios desse nível não é garantido um site altamente acessível.

A seguir são apresentados alguns exemplos de critérios de sucesso do nível A.

- 1.1.1 - Conteúdo não textual: Todo conteúdo “não textual” deve trazer uma alternativa em texto para identificar o conteúdo.
- 1.4.1 - Utilização de cores: Cores não devem ser utilizadas como única maneira de transmitir conteúdo ou distinguir elementos visuais.
- 2.4.2 - Página com título: Páginas ou telas devem possuir um título que descreva claramente a sua finalidade.
- 3.1.1 - Idioma da página: O idioma do conteúdo deve ser definido em cada uma das páginas ou telas da aplicação.
- 4.1.1 - Análise (código): Erros significativos de validação ou análise de semântica de código devem ser evitados.

Estar em conformidade com os critérios do nível AA garante uma acessibilidade à maior parte dos conteúdos para a maioria dos usuários.

Alguns critérios de sucesso de nível AA são:

- 1.2.4 - Legendas (ao vivo): Devem ser fornecidas legendas para todo conteúdo que contenha áudio ao vivo (apenas áudio ou vídeo com áudio).
- 1.4.11 - Contraste Não-Textual - Componentes de interface (exemplo: botões) e imagens essenciais para o entendimento do conteúdo devem ter uma relação de contraste entre primeiro e segundo plano de ao menos 3:1.
- 2.4.5 - Várias formas - O usuário sempre deve ter opções e formas diferentes para acessar ou localizar um determinado conteúdo.
- 3.3.3 - Sugestão de erro - Forneça sugestões simples para que o usuário consiga corrigir facilmente os erros de preenchimento.
- 4.1.3 - Mensagens de *status* - Qualquer tipo de mensagem informacional e relevante ao usuário após executar uma ação deve ser transmitida sem que haja mudança de foco no elemento que originou a informação.

Os critérios do nível AAA representam um nível de acessibilidade mais sofisticado, assegurando um site muito acessível. A maioria dos critérios desse nível são para situações bem específicas, com o objetivo de refinar os critérios dos níveis anteriores. Porém, garantir um nível de acessibilidade AAA é uma tarefa muito custosa, pois a sua implementação é complexa e o nível anterior AA já garante uma acessibilidade satisfatória.

Exemplos de critérios de sucesso de nível AAA são:

- 1.2.6 - Língua de sinais (pré-gravado) - Deve ser fornecida interpretação em língua de sinais (exemplo: Libras) para todo conteúdo que contenha áudio pré-gravado.
- 1.4.9 - Imagens de texto (sem exceção) - Evitar o uso de textos em imagens a não ser que sejam essenciais (exemplo: marcas e logos).
- 2.2.3 - Sem limite de tempo - Nenhuma funcionalidade deve possuir limite de tempo para que uma ação seja executada.
- 3.1.3 - Palavras incomuns - Caso use palavras técnicas ou jargões, forneça um glossário ou explicações que informem ao usuário seu significado.
- 3.1.6 - Pronúncia - Deve-se fornecer um mecanismo que identifique a pronúncia correta de determinadas palavras que possam gerar ambiguidade fora do contexto.

Todos os critérios de sucesso traduzidos para o português estão presentes na página do W3C[16].

Além da acessibilidade da página web em si, os conteúdos presentes, como imagens e documentos, também precisam estar acessíveis.

Conteúdos como imagens de texto e textos convertidos para imagens para obter um determinado efeito visual precisam ter um tamanho e contraste mínimo.

Para arquivos em formato Portable Document Format (PDF), SALTON, AGNOL e TURCATTI (2017) mostram a necessidade de uma estrutura que define a ordem lógica de leitura do documento. Para isso, é feito o uso de *tags*, ou marcações, e com o seu uso é possível fornecer:

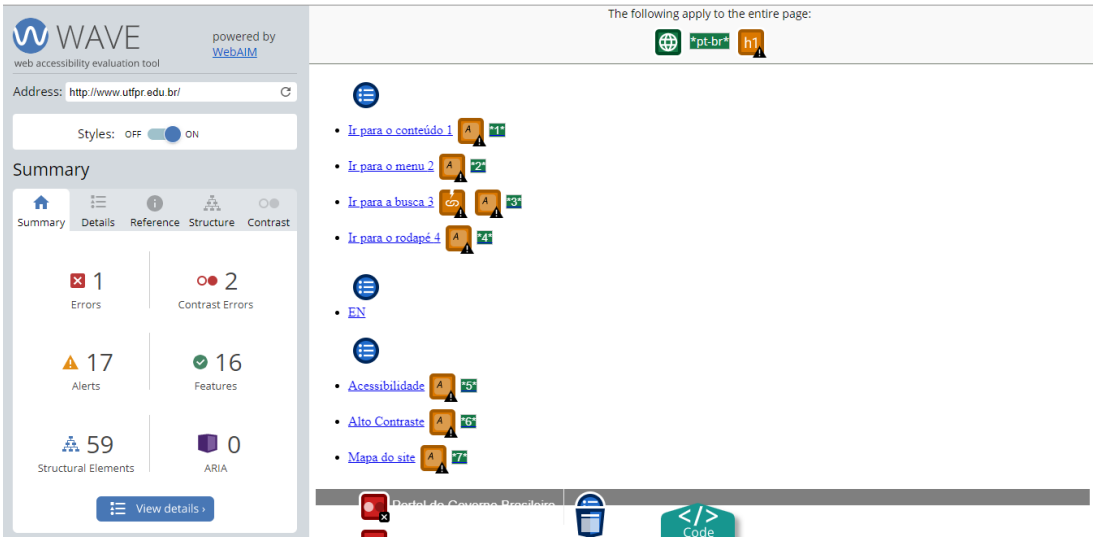
- Ordem lógica de leitura;
- Texto alternativo para as imagens;
- Tabelas com estrutura correta (células de cabeçalho e células de dados);
- Campos de formulário acessíveis.

2.7.2 Ferramentas para Verificação de Acessibilidade em Páginas Web

Existem diversas ferramentas que ajudam a fornecer a acessibilidade digital. Ao pesquisar sobre essas ferramentas, vimos que a maioria é relacionada a deficiências visuais.

O Wave[17] é uma ferramenta *online* de avaliação de acessibilidade da web. Ao inserir o link da página, um relatório detalhado dos erros de acessibilidade é retornado, como é apresentado na Figura 11.

Figura 11 – Exemplo de Validação do Wave



Fonte: Autoria Própria (2021).

[16] https://www.w3c.br/traducoes/wcag/wcag21-pt-BR/
[17] https://wave.webaim.org/

Nessa figura, no canto esquerdo encontra-se o endereço da página avaliada, um botão *switch* para ativar/desativar o estilo da página e um menu com diversos recursos para facilitar o usuário, como o Summary que mostra um resumo da verificação. Uma visão geral da página é apresentada na área direita restante, com ícones representando os erros encontrados. Nesse exemplo, foram encontrados 3 erros, sendo esses, 2 de contraste de cores.

Existem ferramentas específicas para verificar o uso adequado de cores, como sites, extensões de navegadores e softwares. O CorHexa[18] e o WebAim[19] são ferramentas *online* que verificam o contraste de duas cores em formato hexadecimal. Um recurso interessante dessas duas ferramentas é que elas mostram o nível de acessibilidade atingido, AA ou AAA, em fontes de diferentes tamanhos. As Figuras 12 e 13 apresentam, respectivamente, a verificação de contraste de cores do Corhexa e do WebAim.

Figura 12 – Verificação de Contraste de cores utilizando CorHexa

Fonte: Autoria Própria (2021).

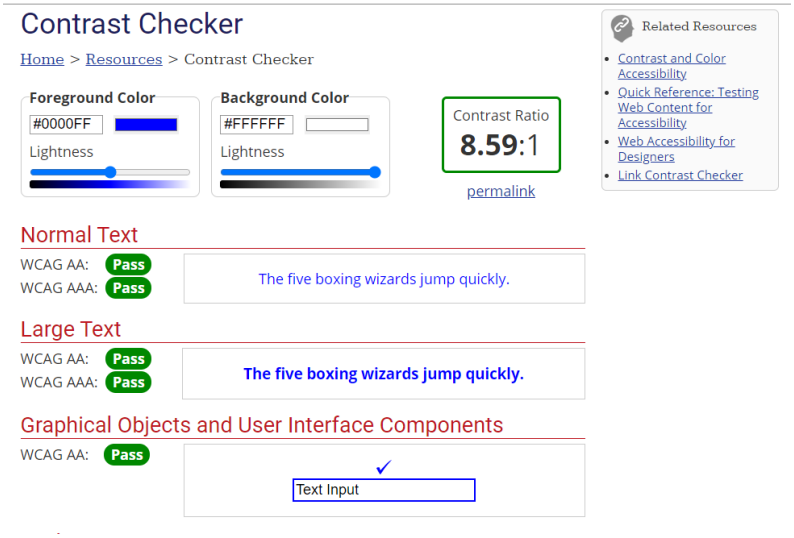
O Lighthouse, apresentado na Figura 14, é uma extensão de código aberto disponível para o Google Chrome, que além de verificar o contraste dos textos, analisa de itens relacionados a acessibilidade, além de outros recursos.

O Accessibility Insights é uma ferramenta desenvolvida pela Microsoft disponível em 3 versões: Android, web (extensão do Google Chrome e Microsoft Edge) e Windows. Com essa ferramenta é possível verificar se os critérios do nível AA do WCAG 2.1 estão em conformidade em sites e aplicativos Android e web.

O aplicativo CSS HTML Validator[20], disponível para Windows e Linux, realiza a edição e validação de códigos HTML e CSS. Além dos arquivos HTML e CSS, ele verifica outros tipos de arquivos, como PHP, JavaScript, SEO, e valida links e acessibilidade.

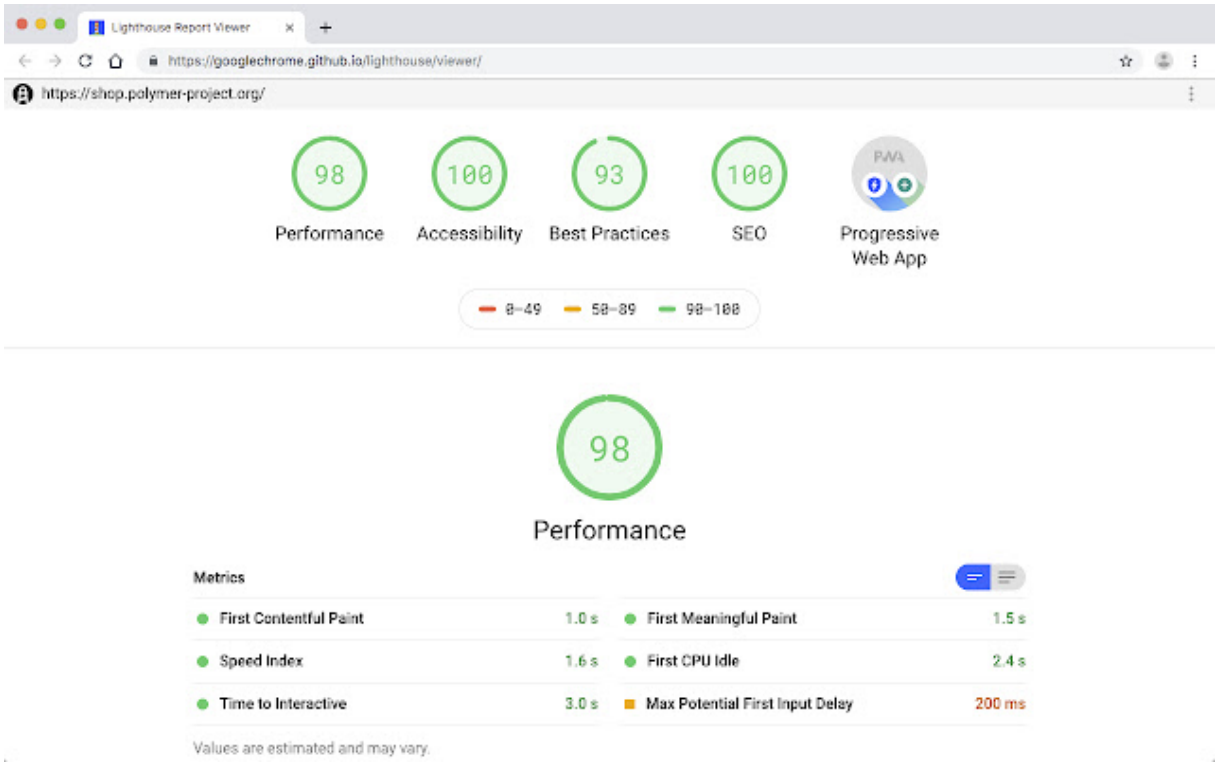
[18] https://corhexa.com/verificador-contraste
[19] https://webaim.org/resources/contrastchecker/
[20] https://www.htmlvalidator.com/

Figura 13 – Verificação de Contraste de cores utilizando WebAim



Fonte: Autoria Própria (2021).

Figura 14 – Avaliação do Lighthouse



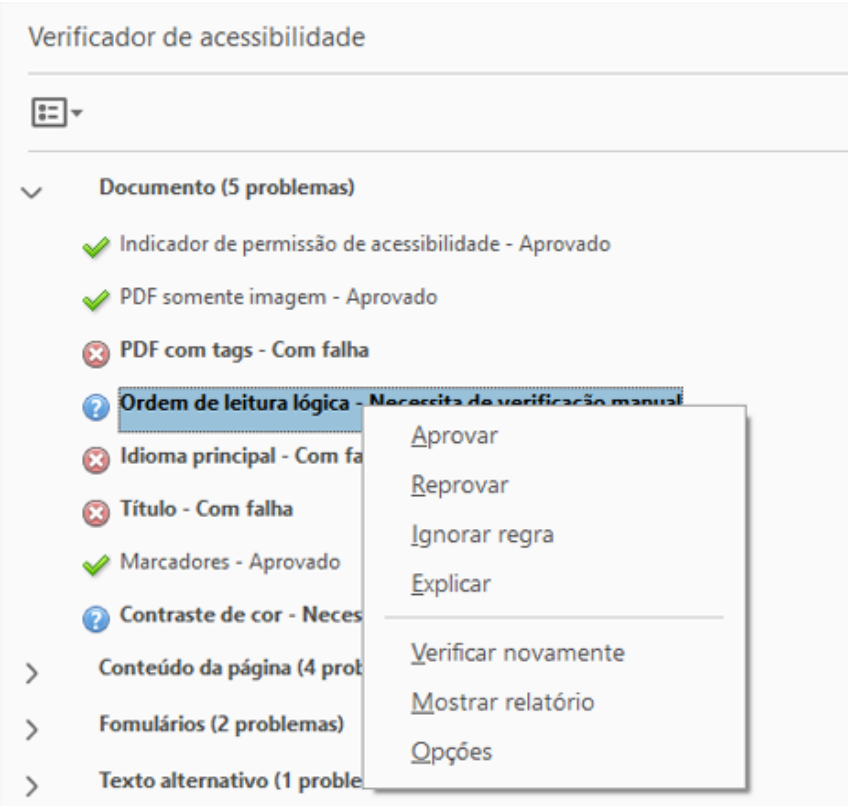
Fonte: Autoria Própria (2021).

2.7.3 Ferramentas para Verificação de Acessibilidade em Arquivos PDF

Um erro recorrente de acessibilidade encontrado em arquivos PDF é a ordem de leitura lógica. Essa ordem pode prejudicar usuários com deficiência visual que utilizam softwares leitores de tela, uma vez que o conteúdo não será percorrido corretamente.

Ferramentas como o Adobe Acrobat Pro possuem recursos que auxiliam a edição e verificação de acessibilidade. Ao realizar a verificação, essa ferramenta gera um relatório que apresenta os erros encontrados, como a Figura 15.

Figura 15 – Status das Regras do Verificador de Acessibilidade.



Fonte: Autoria Própria (2021)).

Também existem ferramentas gratuitas, como o PAVE[21], desenvolvido pelo Laboratório de Acessibilidade da Universidade de Ciências Aplicadas de Zurique em colaboração com a Federação Suíça de Cegos e Deficientes Visuais. O PAVE é uma ferramenta *online* e está disponível de forma gratuita para uso pessoal. A Figura 16 mostra a verificação e a correção da acessibilidade feita pelo o PAVE.

O processo de verificação de erros utilizando o PAVE requer o *upload* do arquivo PDF. A partir disso, a ferramenta faz a verificação e fornece os erros de acessibilidade. Alguns desses erros são corrigidos automaticamente, mas outros requerem intervenção manual, como a especificação da linguagem e o nome do arquivo, por exemplo.

Além das ferramentas apresentadas, existem organizações que trabalham com a correção e solução de conteúdos web, que disponibilizam diversos guias e *checklists* de critérios de acessibilidade disponíveis *online*, como a Cartilha de Acessibilidade na Web do W3C Brasil[22], a

[21] https://pave-pdf.org
[22] https://ceweb.br/cartilhas/cartilha-w3cbr-acessibilidade-web-fasciculo-IV/

Figura 16 – Exemplo de Validação do PAVE

Fonte: Autoria Própria (2021).

Web Content Accessibility Guidelines do WCAG 2.1[23], o Guia WCAG[24]. Esse material pode ser utilizado por pessoas interessadas em adotar esses critérios ao disponibilizar seus arquivos.

[23] https://www.w3.org/TR/WCAG21/
[24] https://guia-wcag.com/

3 TRABALHOS RELACIONADOS

Diversos autores avaliaram a motivação e o rendimento dos alunos utilizando diferentes métodos. Martins (2016) propôs o ensino de lógica de programação utilizando robôs Lego Mindstorms EV3 para alunos do quinto ao sétimo ano do ensino fundamental. Nesse projeto, utilizou a plataforma RachaCuca[1] que, segundo o autor, possui uma interface agradável e diversos problemas de lógica em formato de jogos. Após estimular o raciocínio lógico, foram aplicadas aulas teóricas para ensinar as definições de algoritmos e, na sequência, aulas práticas com a montagem dos robôs LEGO. Para a construção dos programas, foi utilizado o Scratch e por último a plataforma EV3, que possui a mesma lógica do Scratch de arrastar e soltar blocos, para a programação dos robôs.

Na análise de rendimento, foi aplicado uma avaliação para cada turma e também foi levado em consideração a opinião dos docentes em relação ao comportamento da turma e relatos dos alunos sobre a aula. Em geral, Martins (2016) conclui que, apesar das dificuldades, foram obtidos resultados positivos, a maior parte dos alunos alcançou uma nota acima de 8.0 na avaliação, e com bons relatos dos alunos em relação às aulas.

Tsukamoto *et al.* (2016) compararam o uso de linguagens textuais com linguagens visuais no ensino de programação para crianças do terceiro ao sexto ano do ensino Fundamental. Para isso, foram abertos cursos divididos em 6 a 8 tópicos, e avaliaram a motivação aplicando um questionário baseado no modelo ARCS. As questões aplicadas por (TSUKAMOTO *et al.*, 2016) são apresentadas no Quadro 3.

Quadro 3 – Questionário aplicado por Tsukamoto *et al.* (2016)

Atenção	
A1	Perspectiva de estímulo: A educação em programação lhe proporcionou uma surpresa agradável?
A2	Inquérito de estímulo: Você sentiu vontade de aprender mais durante a educação em programação?
A3	Variabilidade: Você poderia estudar sem se entediar pois havia variações no conteúdo de aprendizagem?
Relevância	
R1	Familiaridade: Você sentiu que o conteúdo da aprendizagem era familiar?
R2	Objetivo da orientação: Você entendeu o objetivo e a importância da aprendizagem?
R3	Correspondência de motivos: Você teve chances de selecionar os métodos de aprendizagem mais adequados a você?
Confiança	
C1	Requerimentos de aprendizagem: O objetivo que você deve alcançar está claro?
C2	Oportunidade de sucesso: Você teve a ocasião de sentir que escreveu bem o seu programa?
C3	Controle pessoal: Você sentiu que escreveu bem o seu programa por causa de seus esforços e habilidades?
Relevância	
R1	Consequência natural: Você teve a ocasião de usar seus novos conhecimentos adquiridos?
R2	Consequência positiva: Você ficou feliz quando fez um bom programa?
R3	Justo: Seu desempenho foi avaliado razoavelmente com um padrão consistente?

Fonte: Traduzido de Tsukamoto *et al.* (2016).

[1] https://rachacuca.com.br/

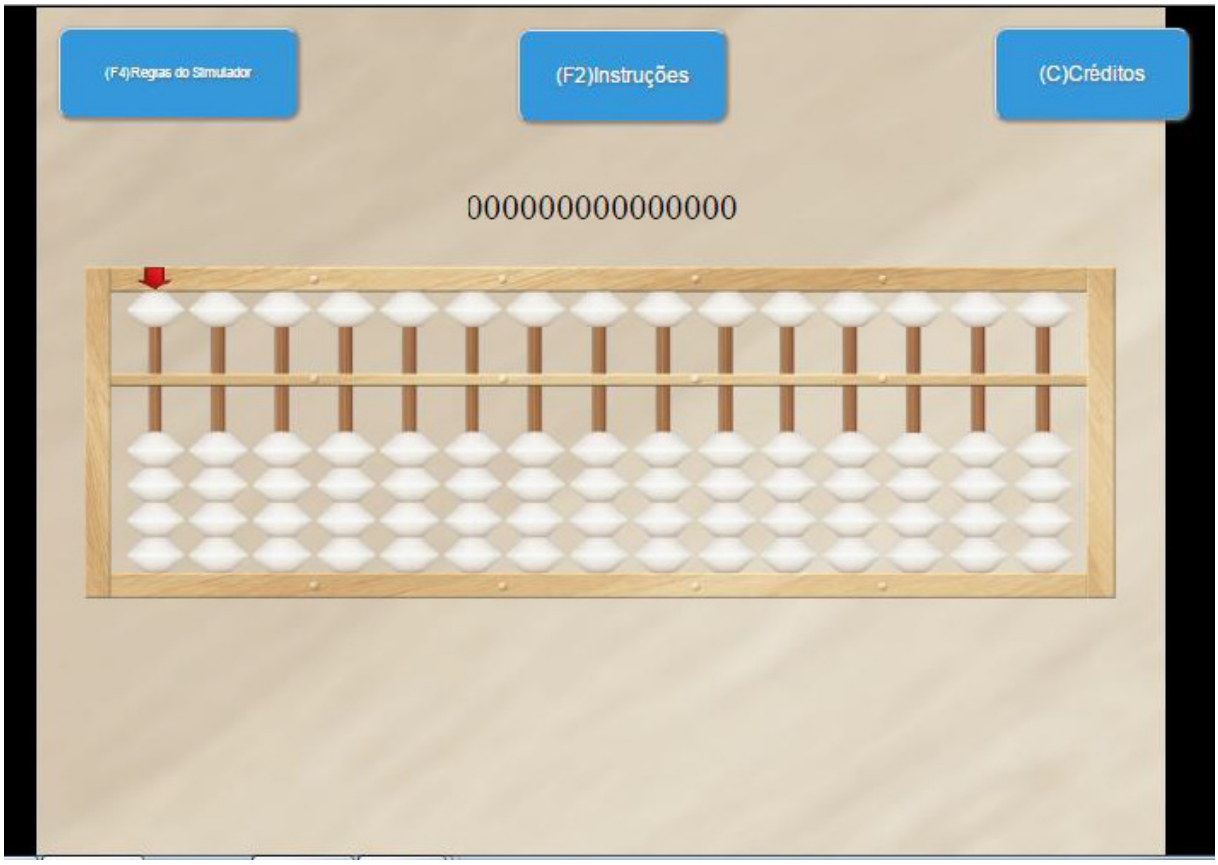
Ao final de cada tópico do curso, esse questionário foi aplicado a três turmas diferentes: uma turma que foi submetida somente à linguagem textual e duas turmas à linguagem visual.

Foram calculadas a média e a variância dos resultados coletados, de cada aspecto do ARCS e o total, e foi percebido um aumento gradativo na motivação dos alunos das turmas com a linguagem visual no decorrer das aulas, o que não ocorreu na outra turma. Com isso, (TSUKAMOTO *et al.*, 2016) concluíram que a linguagem visual é mais motivadora para o ensino de programação para crianças.

O questionário CIS foi utilizado em diversos estudos, e todos eles mostraram uma boa consistência, com um valor de alfa de Cronbach maior que 0,80 (GABRIELLE, 2003; HUETT, 2006; ROBB, 2010). Além do alfa de Cronbach, foram aplicados a média e o desvio padrão comparando os resultados de cada aspecto e o total do ARCS em grupos de controle e experimental. Huett (2006) utilizou outras diferentes fórmulas estatísticas, como o d de Cohen e valor de p.

O trabalho de Bataliotti *et al.* (2016) consiste na construção de objetos educacionais acessíveis. Nesse trabalho um *soroban online* acessível, apresentado na Figura 17, foi elaborado, construído e validado. O processo desse trabalho foi custoso, passando por diversas etapas e testes para garantir o acesso a todos.

Figura 17 – Soroban Acessível



Fonte: Bataliotti *et al.* (2016).

Gkatzidou e Pearson (2009) realizaram um estudo de caso com um *vodcasting* de aprendizagem acessível. Foi realizado uma transformação de um OA do formato web para o *vodcast* com recursos de acessibilidade. O OA foi bem recebido, especialmente por alunos com deficiência auditiva.

Diferente dos trabalhos apresentados, que focam sobretudo alunos dos ensinos fundamental e médio, o presente trabalho propõe a utilização de diferentes abordagens de ensino em alunos das disciplinas introdutórias de programação do ensino superior. Para isso, foi desenvolvido OAs no formato de fichas de exercícios acessíveis, disponibilizadas em um repositório online. Nesse projeto, os critérios de acessibilidade observados focam sobretudo pessoas com deficiência visual, uma vez que as ferramentas de verificação encontradas são voltadas para esse tipo de acessibilidade e que o material produzido é impresso.

Um outro diferencial é que a presente proposta possibilita a aplicação de um método de análise de motivação, visando incluir esse aspecto como um parâmetro adicional na avaliação das abordagens.

4 METODOLOGIA

Nesse capítulo são apresentados o método utilizado para o desenvolvimento do presente trabalho.

4.1 Levantamento e Seleção das Abordagens

Inicialmente, uma pesquisa bibliográfica em bases de dados significativas foi realizada. Para isso, utilizou-se os termos de pesquisa "*Teaching Programming*", "*Methods Teaching Programming*" em bases estrangeiras e "Ensino de Programação", "Métodos de Ensino de Programação" em bases nacionais. Dos diversos trabalhos encontrados, destacaram-se as abordagens em si, a aplicação dessas e os resultados.

As abordagens e práticas de ensino de programação identificadas foram:

- Ensino Tradicional;
- Aprendizagem Baseada em Blocos;
- Aprendizagem Baseada em Jogos;
- Práticas Desplugadas;
- Aprendizagem Baseada em Problemas;
- Robótica Educacional;
- *Pair Programming*;
- *Dojo* de Programação.

Ao pesquisar sobre as abordagens de ensino de programação, diversos métodos de avaliação de interesse dos alunos também foram identificados, como aplicações de questionários, observação do comportamento dos alunos, mudança de desempenho na disciplina, entre outros. Esses métodos de avaliação mostraram-se relevantes e, por isso, foi decidido a aplicação de algum desses métodos para utilizar o interesse dos alunos como um parâmetro de comparação das abordagens.

A princípio, das abordagens de ensino de programação listadas, foram selecionadas o Ensino Tradicional, a Robótica Educacional, as Práticas Desplugadas e a Aprendizagem Baseada em Problemas. Essa seleção foi baseada em critérios como a possibilidade da aplicação em sala de aula e a disponibilidade de materiais.

Porém, ao aprofundar a pesquisa sobre a Aprendizagem Baseada em Problemas, vimos que é um processo demorado, que requer que os alunos estudem previamente o assunto e discutam problemas, o que dificultaria a aplicação dessa abordagem em um curto período de

tempo. Por esse motivo, a Aprendizagem Baseada em Problemas não foi considerada como uma possibilidade viável dentro do escopo desse projeto.

Para analisar a motivação e interesse dos alunos foi escolhido o questionário CIS. Esse questionário foi selecionado por ser capaz de medir a motivação dos estudantes em relação a um curso ou disciplina presencial. Além disso, as questões do CIS pressupõem a existência de um instrutor, o que condiz com o cenário deste trabalho.

4.2 Elaboração das Fichas de Exercícios

Para a elaboração das fichas das abordagens selecionadas, foram utilizadas bibliografias relacionadas à programação e materiais fornecidos por professores das disciplinas escolhidas para a aplicação.

O conteúdo desse material está relacionado ao plano de ensino dessas disciplinas, a saber, Fundamentos da Programação, dos cursos de Engenharias, Fundamentos de Programação I, do curso de Engenharia Eletrônica e Algoritmos, do curso de Bacharelado em Ciência da Computação, e foi adaptado para cada abordagem de ensino selecionada.

Os conteúdos selecionados foram: Estrutura Condicional, Estrutura de Repetição e Vetores (Variáveis Homogêneas Unidimensionais).

Como a melhora no desempenho nas avaliações pode ser um critério de comparação das abordagens, previa-se a aplicação das fichas de exercícios no período entre a primeira e a segunda prova, utilizando o conteúdo da segunda avaliação e os conteúdos anteriores como forma de revisão.

As fichas elaboradas para o Ensino Tradicional contém exercícios que consideram que houve uma explicação teórica do conteúdo seguida por exemplos bem explicados e detalhados. Esses exercícios requerem soluções na linguagem de programação utilizada na sala de aula.

Para a Robótica Educacional, priorizou-se a programação dos robôs em vez da montagem dos mesmos, visando economizar tempo. Por isso, as fichas dessa abordagem consideram a existência de um robô capaz de se movimentar de forma autônoma e o uso da ferramenta de apoio Open Roberta Lab (descrito no Capítulo 2). As instruções para montagem desse robô encontram-se disponibilizadas na Internet[1] e no material.

Para os exercícios dessa abordagem, os alunos devem focar em controlar o robô para que ele faça os movimentos desejados, e fornecer como resposta a sequência de blocos da solução.

Finalmente, para as Práticas Desplugadas, foram elaborados exercícios para ensinar programação sem o uso de computador. Esses exercícios utilizam objetos comuns no dia a dia, como cartas de baralho, dados, caixas de ovo e bolas de isopor. Por exemplo, considerar uma caixa de ovo como um vetor no qual alguns elementos (bolinhas de isopor numeradas) devem

[1] http://nxtprograms.com/NXT2/five_minute_bot/steps.html

ser inseridos de forma ordenada. A resposta para esses exercícios deve ser na forma de um algoritmo que apresenta a lógica da solução do problema.

Os exercícios de cada abordagem são apresentados em ordem crescente de complexidade, a partir do primeiro. Essa ordem de complexidade foi baseada em critérios como a quantidade de linhas necessárias para a solução do exercício e a ordem dos exercícios dos materiais fornecidos pelos professores. Uma vez que a taxa de acerto dos exercícios pode ser um critério de comparação entre abordagens, houve o cuidado para que todos possuíssem níveis de dificuldade semelhantes. Por exemplo, o primeiro exercício de Estrutura de Repetição, em cada abordagem, foi:

- Ensino Tradicional: Faça um código que imprima os números ímpares de 1 a 100.
- Robótica Educacional: Faça com que o carrinho emita uma sequência de sons.
- Práticas Desplugadas: Tire as cartas do monte até sair uma carta ímpar.

A complexidade desses exercícios iniciais é semelhante, e aumenta gradativamente conforme o avanço na lista.

Considerando que é difícil de se aplicar o conceito de Vetores na Robótica Educacional, os conteúdos elaborados nessa abordagem foram limitados a Estrutura Condicional e Estrutura de Repetição.

Esses exercícios constituem um conjunto de OAs que se apresenta no formato de fichas.

4.2.1 Fichas de Exercícios

Dois tipos de fichas foram elaboradas, uma para os alunos e outra para os professores.

As fichas dos alunos visam motiva-los, pela apresentação de exercícios de forma mais atrativa, sendo elaboradas de forma colorida, ilustrativa e informativa. Além disso, considerando o acesso ao material como um instrumento de motivação, essas fichas cumprem os critérios de acessibilidade em arquivos PDFs.

Na Psicologia, trabalhos como o de Farina, Perez e Bastos (2011), mostram que a cor é um fator importante para chamar a atenção do ser humano. As cores básicas, como amarelo, azul, vermelho, são as mais fortes e causam uma grande força emotiva. Já as cores mais suaves têm um efeito oposto. Por esses motivos, priorizou-se o uso de cores primárias.

No que diz respeito à acessibilidade das fichas, questões de fácil percepção, como o contraste das cores e tipo e o tamanho das fontes foram levados em consideração desde o início de sua concepção.

Entretanto, alguns problemas, mais difíceis de serem percebidos, foram corrigidos com o auxílio das ferramentas PAVE e Adobe Acrobat. O uso dessas ferramentas proporcionou que legendas fossem inseridas em todas as imagens, além da inserção de propriedades essenciais, como a identificação do autor, nome e idioma do arquivo. A ordem de leitura foi estabelecida,

reparando as *tags* dos elementos de cada ficha, possibilitando que as ferramentas de leitura de PDF o fizessem na ordem correta.

Essa verificação de acessibilidade e consequente correção de erros foi realizada em todas as 40 fichas dos alunos criadas no contexto desse projeto.

A Figura 18 mostra um exemplo de uma ficha do aluno. Essa ficha apresenta o primeiro exercício de Estrutura Condicional da abordagem Práticas Desplugadas.

Figura 18 – Exemplo de Ficha de Aluno

(a) **(b)**

Fonte: Autoria Própria (2021).

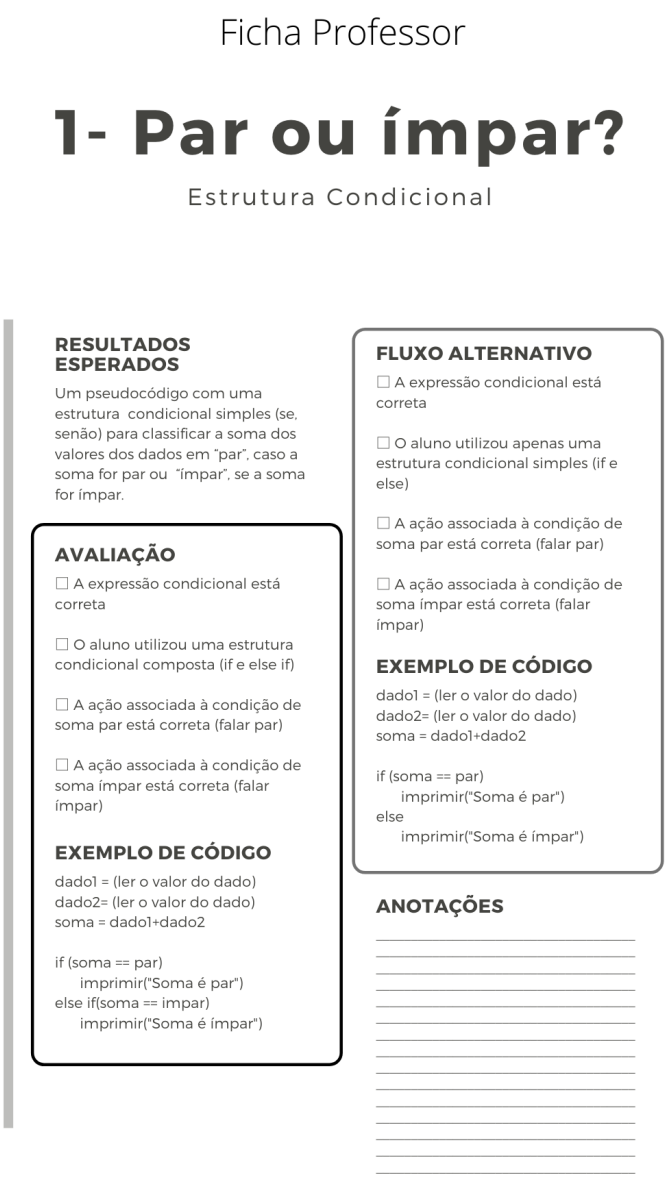
A primeira página dessa ficha contém uma lista dos materiais necessários para a atividade, uma descrição do exercício passo a passo, o objetivo desse exercício, o que deve ser feito e os conhecimentos prévios necessários (Figura 18a). Na segunda página, Figura 18b, há um espaço para o aluno apresentar a solução (algoritmo, código, diagrama de blocos), a data e o nome do aluno.

Todas as fichas obedecem o mesmo modelo, mas há uma diferença entre as abordagens. A ficha de Ensino Tradicional contém uma introdução do tema, uma breve descrição da atividade, os objetivos, os conhecimentos prévios necessários e o que fazer. A ficha de Robótica Educacional é semelhante à ficha de Ensino Tradicional, com a diferença que contém os blocos necessários, substituindo os conhecimentos prévios necessários, e uma introdução do Open Roberta Lab.

As fichas dos professores, por sua vez, não são coloridas e ilustradas como as fichas dos alunos, pois o foco é apresentar a resolução dos exercícios de forma clara e objetiva.

Um exemplo da ficha de professores está apresentado na Figura 19.

Figura 19 – Exemplo de Ficha de Professor



Fonte: Autoria Própria (2021).

Na ficha do professor encontra-se um breve resumo da resolução esperada em forma de texto, uma lista em forma de *checklist* dos elementos necessários no código com um exemplo de resolução.

A possibilidade de existir mais de uma resolução correta foi considerada. Para esses casos, reservou-se uma seção com um *checklist* dos elementos necessários e o exemplo de

código para cada fluxo alternativo encontrado. Uma área com o nome “Anotações” foi prevista para o caso de o educador encontrar um fluxo alternativo ou fazer alguma anotação.

As fichas dos professores não foram verificadas pelas ferramentas de acessibilidade, pois o foco do trabalho é a motivação dos estudantes, atribuindo uma atenção maior nas fichas destinadas a eles. Porém, houve um cuidado na elaboração nos materiais dos professores, utilizando um modelo simples com cores de alto contraste, preto e branco, e fontes de fácil leitura.

Por fim, as fichas criadas abrangem as três abordagens de ensino (Ensino Tradicional, Robótica Educacional e Práticas Desplugadas). Para cada abordagem, três Tópicos (Estrutura Condicional, Estrutura de Repetição e Vetores) são contemplados; e para cada tópico foi elaborado um conjunto de 5 exercícios. Cada exercício possui 2 fichas (1 para o aluno, 1 para o professor), totalizando 80 fichas. A Tabela 4 sintetiza todas as fichas criadas no contexto desse trabalho.

Tabela 4 – Tabela das Fichas de Exercícios

	Ensino Tradicional	Práticas Desplugadas	Robótica Educacional
Estrutura Condicional	1. Positivo, Negativo ou 0? 2. Vogal ou Consoante? 3. Maior e Menor 4. Qual é o Consumo? 5. Faça a Comparação!	1. Par ou impar? 2. Letra, número ou coringa? 3. Qual é o naipe? 4. O maior vence! 5. Acerte o número!	1. Sons 2. Qual o Lado? 3. Quadrado 4. Frente e para 5. Controle o Carrinho
Estrutura de Repetição	1. Números Ímpares 2. Potência 3. Tabuada 4. Letras e Consoantes 5. Poupança	1. Ímpar! 2. Qual o maior valor? 3. Qual é o naipe 2? 4. Somando valores 5. Não ultrapasse!	1. Faça uma música 2. Quadrado 3. Ande em S 4. _|—|_ 5. Aumente o quadrado
Vetores	1. Dobro 2. Inverso 3. Média 4. Par e Ímpar 5. Frequência	1. Alfabeto 2. Maior e menor letra 3. Inverta a ordem 4. Inverta a ordem 2 5. Ordene!	
Total de Fichas	Alunos: 15 Professores: 15	Alunos: 15 Professores: 15	Alunos: 10 Professores: 10

Fonte: Autoria Própria (2021).

Deve ser ressaltado que a ordem dos exercícios das fichas estão conforme a sua complexidade, mas não há uma dependência entre os exercícios.

4.2.2 Questionário CIS

Originalmente, a proposta previa que, para a análise de interesse dos alunos, fosse utilizado o questionário CIS. A tradução para o português foi realizada e os termos padrão das perguntas desse questionário foram modificados para adequar-se ao cenário de pesquisa. Idealmente, esse questionário deveria ser utilizado após a aplicação de cada abordagem. Após todas as aplicações, a pontuação obtida poderia ser calculada.

Esse cálculo é feito conforme a escala Likert-5, em que a opção “Concordo totalmente” vale 5 pontos e o “Discordo totalmente”, 1 ponto, com a exceção das perguntas *reverse*, que são valoradas de forma inversa. Cada questão se refere a um único aspecto do ARCS. O questionário e as pontuações são apresentados no Apêndice A.

Após o cálculo, é possível comparar as abordagens no que diz respeito aos aspectos de motivação propostos pelo ARCS. Sugere-se, a partir de relatos encontrados na literatura ((GABRIELLE, 2003), (HUETT, 2006), (ROBB, 2010)), que o alfa de Cronbach seja calculado para verificar a consistência interna das respostas. Uma vez validado o questionário, outras medidas estatísticas, como média, desvio padrão e variância podem ser obtidas.

Para a representação dos resultados, há diversas opções como gráfico de linhas ou colunas. Pereira e Bittencourt (2020) utilizaram o formato de *box-plot* nas médias de cada aspecto do ARCS. Tsukamoto *et al.* (2016), por sua vez, calcularam a média e a variância de cada aspecto do ARCS e o total e representaram em um gráfico de linhas.

Uma vez que a proposta original foi alterada por conta de restrições impostas pela pandemia de COVID-19 que impossibilitaram a aplicação do método em ambiente escolar no contexto dessa pesquisa, esse questionário foi elaborado em dois formatos: um modelo disponibilizado no Google Forms e um *script* em Python para que instrutores ou professores fossem capazes de estimar o aspecto motivacional dessas abordagens por conta própria.

Questões de segurança impedem que planilhas contendo *scripts* sejam acessadas por usuários externos. Por conta disso, é preciso ter uma cópia da planilha (e do respectivo formulário) no drive pessoal da pessoa interessada em utilizar essa planilha. O questionário criado, bem como as instruções para obter uma cópia dele são fornecidas no Apêndice C.

Uma versão em *script* foi desenvolvida em Python. Ao executar o *script*, as questões exibidas uma a uma e ao responder a última pergunta é apresentado a pontuação.

Esse *script* está disponível para *download* no repositório web.

4.3 Repositório Web

Para armazenar todos os recursos desenvolvidos, um repositório com as fichas e o questionário CIS foi elaborado, observando os critérios de acessibilidade do nível AA do padrão WCAG estabelecido pelo W3C.

A Figura 20 mostra o *layout* da página principal do repositório.

Para assegurar o nível de acessibilidade pretendido dos arquivos HTML e CSS, utilizou-se a ferramenta CSS HTML Validator. O nível de contraste, o tamanho das fontes, a estrutura dos cabeçalhos e a definição da linguagem do documento são exemplos de erros que foram corrigidos após a aplicação dessas ferramentas.

Todas as fichas de exercícios foram divididas por abordagem e tópicos na área “Fichas”. Na parte “Questionário”, uma breve apresentação do questionário CIS e o *script* em Python está disponível para *download*.

Figura 20 – Repositório dos Materiais

Fonte: Autoria Própria (2021).

A execução desse *script* provê, ao terminar de responder o questionário, a pontuação obtida em cada aspecto do ARCS, bem como o valor total. Um exemplo de exibição desses resultados é mostrado na Figura 21.

Figura 21 – Pontuações Obtidas pelo *script* Python para o CIS

```
33. A quantidade de trabalho que tenho que fazer é apropriada.
1 2 3 4 5
3
34. Eu recebo feedback suficiente para saber o quão bem eu estou no curso.
1 2 3 4 5
4
Total de questões: 34
Atenção: 27
Relevância: 27
Confiança: 32
Satisfação: 28
Total do ARCS: 114
```

Fonte: Autoria Própria (2021).

5 CONCLUSÕES

Como diversos estudos mostram, muitos alunos ingressantes na área de computação possuem dificuldade nas disciplinas introdutórias de programação e, consequentemente, isso pode levar à desistência e evasão.

Visando melhorar esse cenário que inclui a dificuldade de aprendizagem, a desmotivação dos alunos, entre outras coisas, o presente trabalho tem como objetivo apresentar abordagens de ensino de programação que podem trazer melhores resultados na motivação e consequente aprendizagem dos alunos.

O processo de levantamento originalmente proposto foi realizado. Diversas abordagens de ensino, além de combinações de abordagem, ferramentas e métodos para avaliar a motivação dos alunos em sala de aula foram identificados. Durante esse processo, vimos que a acessibilidade dos materiais é um fator importante o qual influencia na motivação dos alunos. Por isso, critérios de acessibilidade no material produzido foram incorporados.

Em seguida, selecionamos as abordagens de ensino, levando em consideração a facilidade de aplicação em sala de aula e a disponibilidade dos recursos. Depois, a escolha dos exercícios e a pesquisa por ferramentas para a avaliação de motivação foram concluídas.

Para que o professor consiga aplicar os materiais por conta própria, optou-se pelo formato de fichas para os materiais didáticos. Dois formatos de ficha foram elaborados, uma voltada ao aluno e outra voltada aos professores. Nas fichas dos alunos foi feita uma verificação de acessibilidade visual com apoio de ferramentas específicas.

O questionário CIS de Keller (2009) foi escolhido para a ferramenta de avaliação de motivação. Esse questionário de 34 questões baseado no modelo ARCS é utilizado para medir a motivação dos estudantes em relação a um curso ou disciplina presencial. As questões foram traduzidas e adaptadas para o cenário de pesquisa e dois formatos foram disponibilizado: um questionário no Google Forms e um *script* em Python.

Para disponibilizar tanto as fichas de exercícios quanto o questionário CIS foi elaborado um repositório web acessível, levando em consideração o nível AA de acessibilidade do WCAG. Esse repositório foi validado com o apoio de ferramentas para a verificação da acessibilidade.

As principais contribuições desse trabalho são: o conjunto de fichas de exercícios específicos para diferentes abordagens de ensino, concebidas para professores e alunos; o questionário CIS adaptado e desenvolvido em formato de planilha e *script* executável; orientações para a aplicação desse questionário por parte de educadores, a fim de avaliar o nível de motivação dos alunos; e um repositório idealizado com critérios de acessibilidade. Todo esse material permanecerá disponível no Repositório Institucional da Universidade Tecnológica Federal do Paraná (RIUT)[1].

Como trabalhos futuros, sugere-se incentivar a aplicação dos materiais e do questionário em sala de aula e a análise dos resultados das aplicações pelos professores. Também há a

[1] https://repositorio.utfpr.edu.br/

possibilidade de adaptar este trabalho em outras disciplinas da computação ou em áreas diferentes e adaptar o material desenvolvido, de forma a satisfazer a taxonomia de Bloom (BLOOM *et al.*, 1956). Existe ainda, a possibilidade de estender a quantidade de fichas de exercícios, a fim de incluir novos assuntos, além de disponibilizá-los em repositórios oficiais. No que diz respeito ao questionário CIS, uma possibilidade é desenvolver uma interface gráfica, tanto para a versão online quanto para e versão desktop, obedecendo aos critérios de acessibilidade.

REFERÊNCIAS

ABREU, R. S. d.; PAULA, B. C. d. Contextualizando o desenvolvimento de jogos digitais como estratégia de ensino e aprendizagem. *In*: **IX Seminário Jogos Eletrônicos, Educação e Comunicação – construindo novas trilhas**. [*S.l.*: *s.n.*], 2013.

ALVES, W. P. **Linguagem e Lógica de Programação**. [*S.l.*]: Editora Érica Ltda., 2014. v. 1.

ARBOLEDA, A. V. P.; MAZUERA, L. R. R.; MONTEMIRANDA, G. S. Competences that facilitate the achievement of the objectives of an introductory programming course. *In*: **IEEE 2015 International Conference on Interactive Collaborative Learning (ICL)**. [*S.l.*: *s.n.*], 2015. p. 1007–1012.

BATALIOTTI, S. E. *et al.* A construção de objetos educacionais acessíveis. **Journal Of Research In Special Educational Needs**, Wiley Online Library, v. 16, p. 41–45, 2016.

BELL, T.; WITTEN, I.; FELLOWS, M. **Computer science unplugged**. 2002.

BERMÚDEZ, A. C. **Na pandemia, 608 mil alunos interrompem curso no ensino superior privado**. 2020. **Disponível em:** https://educacao.uol.com.br/noticias/2020/10/19/na-pandemia-inadimplencia-e-evasao-crescem-no-ensino-superior-privado.htm. **Acesso em:** 19 out. 2020.

BERSSANETTE, J. H. **Ensino de programação de computadores: uma proposta de abordagem prática baseada em Ausubel**. 2016. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2016.

BLOOM, B. *et al.* **Taxonomy of Educational Objectives: The Classification of Educational Goals. Cognitive domain**. [*S.l.*]: Longman, 1956. ISBN 9780582280106.

BRAGA, J. C. **Objetos de Aprendizagem: Introdução e Fundamentos**. 1. ed. [*S.l.*]: Editora UFABC, 2014. ISBN 978-85-68576-03-8.

BRASIL. **Lei Brasileira de Inclusão da Pessoa com Deficiência**. **Diário Oficial da República Federativa do Brasil**, 2015.

BRASIL. **Sistema de Avaliação da Educação Básica (Saeb)**. 2018. **Disponível em:** https://portal.inep.gov.br/documents/186968/484421/Relat%C3%B3rio+Saeb+2017/e683ba93-d9ac-4c2c-8f36-10493e99f9b7?version=1.0. **Acesso em:** 15 mai 2020.

CARVALHO, C. V. de. Aprendizagem baseada em jogos-game-based learning. *In*: **II World Congress on Systems Engineering and Information Technology**. [*S.l.*: *s.n.*], 2015. p. 176–181.

Code.org. **Code.org**. 2020. **Disponível em:** https://code.org/. **Acesso em:** 18 ago. 2020.

COOMBS, N. **Making online teaching accessible: Inclusive course design for students with disabilities**. [*S.l.*]: John Wiley & Sons, 2010. v. 17.

DIJKSTRA, E. W. **A short introduction to the art of programming**. [*S.l.*]: Technische Hogeschool Eindhoven Eindhoven, 1971. v. 4.

FARINA, M.; PEREZ, C.; BASTOS, D. **Psicodinâmica das cores em comunicação**. [*S.l.*]: Editora Blucher, 2011.

FORBELLONE, A. L. V.; EBERSPACHER, H. F. **Lógica de programação**. [*S.l.*]: Makron Books, 2000.

FREEMAN, S. *et al.* Active learning increases student performance in science, engineering, and mathematics. **Proceedings of the National Academy of Sciences**, National Acaddemy of Science, v. 111, n. 23, p. 8410–8415, 2014.

GABRIELLE, D. **The effects of technology-mediated instructional strategies on motivation, performance, and self-directed learning**. *In*: ASSOCIATION FOR THE ADVANCEMENT OF COMPUTING IN EDUCATION (AACE). **EdMedia+ Innovate Learning**. [*S.l.*], 2003. p. 2568–2575.

GKATZIDOU, S.; PEARSON, E. The potential for adaptable accessible learning objects: A case study in accessible vodcasting. **Australasian Journal of Educational Technology**, v. 25, n. 2, 2009.

GOMES, A. de J.; MENDES, A. J. **Ambiente de suporte à aprendizagem de conceitos básicos de programação**. 2000. Tese (Doutorado) — Universidade de Coimbra, 2000.

GUZDIAL, M. Exploring hypotheses about media computation. *In*: **Proceedings of the ninth annual international ACM conference on International computing education research**. [*S.l.*: *s.n.*], 2013. p. 19–26.

HINTERHOLZ Jr., O. Tepequém: uma nova ferramenta para o ensino de algoritmos nos cursos superiores em computação. *In*: **Anais do XVII Workshop sobre Educação em Informática**. [*S.l.*: *s.n.*], 2009. v. 20, p. 21.

HOED, R. M. **Análise da evasão em cursos superiores: o caso da evasão em cursos superiores da área de Computação**. 2016. Dissertação (Mestrado) — Universidade de Brasília, 2016.

HUETT, J. B. **The effects of ARCS-based confidence strategies on learner confidence and performance in distance education**. [*S.l.*]: Citeseer, 2006.

JACKSON, D.; MILLER, R. **A new approach to teaching programming**. **Cambridge**, v. 77, 2009.

JAPIASSÚ, H.; MARCONDES, D. **Dicionário básico de filosofia**. [*S.l.*]: Jorge Zahar Editor Ltda., 2011. v. 5.

KASSAI, J. R. *et al.* Reflexões sobre o nível de evasão e o custo anual per capita das unidades de ensino da USP com base no método inquired balance sheet. *In*: **XXXIV Encontro da ANPAD**. [*S.l.*: *s.n.*], 2010.

KELLER, J. M. Development and use of the arcs model of instructional design. **Journal of instructional development**, Springer, v. 10, n. 3, p. 2, 1987.

KELLER, J. M. **Development of two measures of learner motivation**. 2006. **Disponível em:** https://studylib.net/doc/7446614/development-of-two-measures-of-learner-motivation. **Acesso em:** 25 mai 2020.

KELLER, J. M. **Motivational design for learning and performance: The ARCS model approach**. [*S.l.*]: Springer Science & Business Media, 2009.

KESZEI, A. P.; NOVAK, M.; STREINER, D. L. **Introduction to health measurement scales**. **Journal of psychosomatic research**, Elsevier, v. 68, n. 4, p. 319–323, 2010.

KODJAOGLANIAN, V. L. *et al.* Inovando métodos de ensino-aprendizagem na formação do psicólogo. **Psicologia: ciência e profissão**, SciELO Brasil, v. 23, n. 1, p. 2–11, 2003.

LAHTINEN, E.; ALA-MUTKA, K.; JÄRVINEN, H.-M. A study of the difficulties of novice programmers. **ACM Sigcse Bulletin**, v. 37, n. 3, p. 14–18, 2005.

LANDIS, J. R.; KOCH, G. G. The measurement of observer agreement for categorical data. **Biometrics**, [Wiley, International Biometric Society], v. 33, n. 1, p. 159–174, 1977. ISSN 0006341X, 15410420.

LEGO. **Lego Mindstorms**. 2020. **Disponível em:** https://www.lego.com. **Accesso em:** 15 ago. 2020.

MARTINS, L. Ensinando lógica de programação aplicada a robótica para alunos do ensino fundamental. *In*: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [*S.l.*: *s.n.*], 2016. v. 27, n. 1, p. 31.

MEERBAUM-SALANT, O.; ARMONI, M.; BEN-ARI, M. Habits of programming in scratch. *In*: ACM. **Proceedings of the 16th annual joint conference on Innovation and technology in computer science education**. [*S.l.*], 2011. p. 168–172.

MIOLA, C.; GOTTI, A. **Quatro milhões a mais de cadeiras vazias**. 2021. **Disponível em:** https://politica.estadao.com.br/blogs/fausto-macedo/quatro-milhoes-a-mais-de-cadeiras-vazias. **Acesso em:** 01/02/2021.

MIZUKAMI, M. d. G. N. **Ensino: as abordagens do processo**. [*S.l.*]: Editora Pedagógica e Universitária São Paulo, 1986.

MOHOROVIČIĆ, S.; STRČIĆ, V. An overview of computer programming teaching methods. *In*: **XXII Central European Conference on Information and Intelligent Systems**. [*S.l.*: *s.n.*], 2011. p. 1–6.

MONCLAR, R. S.; SILVA, M. A.; XEXÉO, G. Jogos com propósito para o ensino de programação. *In*: **Anais do XVII Simpósio Brasileiro de Jogos e Entretenimento Digital – SBGames**. [*S.l.*: *s.n.*], 2018. p. 1132–1140.

MORTARI, C. A. **Introdução à lógica**. [*S.l.*]: SciELO-Editora UNESP, 2001.

NUUTILA, E.; TÖRMÄ, S.; MALMI, L. Pbl and computer programming—the seven steps method with adaptations. **Computer Science Education**, Taylor & Francis, v. 15, n. 2, p. 123–142, 2005.

OLIVEIRA, E. **Nº de alunos que abandonam faculdade deve subir após a pandemia, e setores poderão enfrentar falta de mão de obra**. 2020. **Disponível em:** https://tinyurl.com/mpj8epsn. **Acesso em:** 13 out. 2020.

OLIVEIRA, L. M. B. *et al.* **Cartilha do Censo 2010: pessoas com deficiência**. Secretaria de Direitos Humanos da Presidência da República (SDH-PR), 2012.

OLIVEIRA, P. A. d.; ROCHA, A. J. d. O. Raciocínio lógico, conceitos e estabelecimentos de parâmetros para a aprendizagem matemática. **Revista do Acadêmico de Matemática**, 2011.

PEREIRA, F. T. S. S.; BITTENCOURT, R. A. Avaliação da motivação da aprendizagem de programação avançada através de pbl. *In*: SBC. **Anais do V Congresso sobre Tecnologias na Educação**. [*S.l.*], 2020. p. 138–147.

PEREIRA, G. Q. **O uso da robótica educacional no ensino fundamental: relatos de um experimento**. 2010. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Federal de Goiás, Goiânia.

RAUBER, J. *et al.* **Que tal um pouco de Lógica**. **Porto Alegre: Clio**, 2003.

REBOUÇAS, A. D. D. S. *et al.* Aprendendo a ensinar programação combinando jogos e python. *In*: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [*S.l.*: *s.n.*], 2010. v. 1, n. 1.

RIOS, G. A. *et al.* Cultura inclusiva na educação a distância: Concepção de cursos acessíveis. **Journal Of Research In Special Educational Needs**, Wiley Online Library, v. 16, p. 332–335, 2016.

ROBB, C. **The impact of motivational messages on student performance in community college online courses**. 2010. Tese (Doutorado) — University of Illinois at Urbana-Champaign, 2010.

ROCHA, P. S. *et al.* Ensino e aprendizagem de programação: análise da aplicação de proposta metodológica baseada no sistema personalizado de ensino. **Revista Novas Tecnologias na Educação (RENOTE)**, v. 8, n. 3, 2010.

SAITO, D. *et al.* Program learning for beginners: Survey and taxonomy of programming learning tools. *In*: **2017 IEEE 9th International Conference on Engineering Education (ICEED)**. [*S.l.*: *s.n.*], 2017. p. 137–142.

SALTON, B. P.; AGNOL, A. D.; TURCATTI, A. **Manual de acessibilidade em documentos digitais**. **Bento Gonçalves: Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul**, 2017.

SANCHOTENE, I. J. *et al.* Competências digitais docentes e o processo de ensino remoto durante a pandemia da covid-19. **EaD em Foco**, v. 10, n. 3, 2020.

SATO, D. T.; CORBUCCI, H.; BRAVO, M. V. Coding dojo: An environment for learning and sharing agile practices. *In*: IEEE. **Agile 2008 Conference**. [*S.l.*], 2008. p. 459–464.

SCHUMACHER, J.; WELCH, D.; RAYMOND, D. Teaching introductory programming, problem solving and information technology with robots at west point. *In*: IEEE. **31st Annual Frontiers in Education Conference. Impact on Engineering and Science Education**. [*S.l.*], 2001. v. 2.

SELBY, C. Four approaches to teaching programming. *In*: **Learning, Media and Technology: a doctoral research conference**. [*S.l.*: *s.n.*], 2011.

SENA, S. de *et al.* Aprendizagem baseada em jogos digitais: a contribuição dos jogos epistêmicos na geração de novos conhecimentos. **Revista Novas Tecnologias na Educação (RENOTE)**, v. 14, n. 1, 2016.

SILVA, N. N. da. **Desenvolvimento de objetos de aprendizagem para a aprendizagem baseada em problemas no ensino de computação para engenharias**. 2017. Dissertação (Mestrado) — Universidade Tecnológica Federal do Paraná, 2017.

STREINER, D. L. Starting at the beginning: an introduction to coefficient alpha and internal consistency. **Journal of personality assessment**, Taylor & Francis, v. 80, n. 1, p. 99–103, 2003.

SZLÁVI, P.; ZSAKÓ, L. Methods of teaching programming. **Teaching mathematics and Computer Science**, v. 1, p. 247–258, 01 2003.

TIZIOTTO, S. A.; OLIVEIRA NETO, J. D. d. Design universal: solução para a acessibilidade no ensino superior a distância. *In*: **Congresso Internacional de Educação a Distância**. [*S.l.*: *s.n.*], 2010.

TSUKAMOTO, H. *et al.* Textual vs. visual programming languages in programming education for primary schoolchildren. *In*: IEEE. **2016 IEEE Frontiers in Education Conference (FIE)**. [*S.l.*], 2016. p. 1–7.

UZUNCA, B.; JANSEN, S. How do ecosystem dynamics work in serious gaming ecosystems? challenges and opportunities. **Strategic Management Society**, 2016.

VALASKI, J.; PARAISO, E. C. Limitações da utilização do alice no ensino de programação para alunos de graduaçã. *In*: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [*S.l.*: *s.n.*], 2012. v. 23, n. 1.

WEINTROP, D.; WILENSKY, U. Comparing block-based and text-based programming in high school computer science classrooms. **ACM Transactions on Computing Education (TOCE)**, ACM, v. 18, n. 1, p. 3, 2017.

World Web Consortium (W3C). **The W3C Web Accessibility Initiative (WAI)**. 2020. **Disponível em:** https://www.w3.org/WAI/standards-guidelines/wcag/. **Accesso em:** 15 ago. 2020.

YU, X. Using lego mindstorms in the undergraduate curriculum of it. *In*: IEEE. **2012 International Symposium on Information Technologies in Medicine and Education**. [*S.l.*], 2012. v. 1, p. 270–273.

ZACHARIS, N. Z. Measuring the effects of virtual pair programming in an introductory programming java course. **IEEE Transactions on Education**, IEEE, v. 54, n. 1, p. 168–170, 2011.

APÊNDICE A – Questionário CIS Adaptado

Nesse apêndice é apresentado uma adaptação das 34 questões presentes no questionário CIS propostas por Keller (2006).

As questões devem ser respondidas na escala Likert, utilizando números de 1 a 5, ou letras de 'a' a 'e', sendo o menor valor significando "Discordo totalmente" ou "Totalmente falso" e o maior valor, "Concordo totalmente" ou "Totalmente verdadeiro".

No questionário CIS, a pontuação mínima possível é 34 pontos, a máxima é 170 pontos e a média é 102 pontos.

As questões do CIS são divididas conforme o aspecto do ARCS, como é apresentado na Tabela 5. O "(reverse)" na frente de algumas questões significa que a pontuação é contada de forma inversa. Por exemplo, se em uma questão "(reverse)" for respondido com 1, a pontuação deve ser contada como 5.

Tabela 5 – Divisão das Questões do Questionário CIS
Fonte: Keller (2006)

Aspecto	Números das questões
Atenção	1, 4 (reverse), 10, 15, 21, 24, 26 (reverse), 29
Relevância	2, 5, 8 (reverse), 13, 20, 22, 23 25 (reverse), 28
Confiança	3, 6 (reverse), 9, 11 (reverse), 17 (reverse), 27, 30, 34
Satisfação	7 (reverse), 12, 14, 16, 18, 19, 31 (reverse), 32, 33

A pontuação mínima, máxima e o valor médio de cada aspecto variam pois o número de questões é diferente.

As questões presentes no questionário CIS são apresentadas na Tabela 6.

Tabela 6 – Questionário CIS Adaptado ao Português

1	O professor sabe nos entusiasmar com o assunto desse curso.
2	O que estou aprendendo nesse curso será útil para mim.
3	Eu me sinto confiante que irei bem nesse curso.
4	Essa aula tem pouca coisa que prende a minha atenção.
5	O professor faz o assunto do curso parecer importante.
6	É preciso ter sorte para conseguir boas notas neste curso.
7	Eu preciso me esforçar muito para ir bem neste curso.
8	NÃO vejo como o conteúdo deste curso se relaciona com algo que eu já conheço.
9	O sucesso ou não neste curso, depende de mim.
10	O professor cria um suspense quando constrói um ponto.
11	O assunto deste curso é muito difícil para mim.
12	Eu sinto que este curso me dá muita satisfação.
13	Neste curso, eu tento definir e alcançar altos padrões de excelência.
14	Eu sinto que as notas que recebo são justas em comparação com outros alunos.
15	Como aluno desta aula, estou curioso sobre o assunto.
16	Eu gosto de trabalhar neste curso.
17	É difícil prever qual nota o professor dará às minhas tarefas.
18	Estou satisfeito com as avaliações do meu trabalho do professor comparado com o quão bem eu acho que fui.
19	Eu me sinto satisfeito com o que estou recebendo deste curso.
20	O conteúdo deste curso está conforme às minhas expectativas e objetivos.
21	O professor faz coisas incomuns ou surpreendentes que são interessantes.
22	Os estudantes participam ativamente nessa aula.
23	Para atingir meus objetivos, é importante que eu me saia bem neste curso.
24	O professor utiliza uma variedade de técnicas de ensino que são interessantes.
25	Eu NÃO acho que vou me beneficiar muito deste curso.
26	Eu frequentemente sonho acordado quando estou nesta aula.
27	Enquanto estou tendo esta aula, acredito que posso ter sucesso se eu me esforçar.
28	Os benefícios pessoais deste curso são claros para mim.
29	A minha curiosidade é estimulada muitas vezes pelas perguntas feitas ou pelos problemas dados sobre o assunto desta aula.
30	Eu acho que o nível de desafio neste curso é quase ideal: nem muito fácil, nem muito difícil.
31	Eu me sinto muito desapontado com este curso.
32	Sinto que obtenho reconhecimento suficiente do meu trabalho neste curso por meio de notas, comentários ou outros feedback.
33	A quantidade de trabalho que tenho que fazer é apropriada.
34	Eu recebo feedback suficiente para saber o quão bem eu estou no curso.

Fonte: Adaptação de Keller (2006).

APÊNDICE B – Lista de Exercícios

Nesse apêndice está as listas de exercícios que foram elaboradas para as turmas das disciplinas introdutórias de programação.

As listas de exercícios foram feitas para os conteúdos de condicional, laços de repetições e vetores utilizando as abordagens de Ensino tradicional,Robótica educacional e Praticas desplugadas.

Os exercícios de cada lista ficaram da seguinte forma:

Lista de Exercícios de Condicional

- Ensino tradicional

    1. Faça um programa que peça um valor e mostre na tela se o valor é positivo ou negativo ou 0 (zero).
    2. Faça um programa que verifique se uma letra informada é vogal ou não.
    3. Faça um programa que leia 3 números inteiros e mostre o maior e o menor deles.
    4. Faça uma programa que receba um percurso em quilômetros e o tipo de combustível do carro, e retorne o consumo estimado de combustível. Os tipos são (A)álcool, (G)gasolina e(D)diesel. O carro a álcool faz 8 km por litro, à gasolina faz 12 Km por litro e à diesel faz 5 Km por litro.
    5. Elabore um código que receba 2 valores numéricos e um símbolo. Caso o símbolo seja um dos relacionados abaixo efetue a operação correspondente com os valores. Símbolos: “+” operação de soma, “-” operação de subtração, “*” operação de multiplicação, “/” operação de divisão.

- Robótica educacional (Utilizar o bloco repetir indefinidamente)

    1. Faça o carrinho tocar sons diferentes para cada botão pressionado. São 3 botões, direito, esquerdo e enter.
    2. Faça o carrinho andar para a direita se o botão direito for pressionado e para a esquerda se o botão esquerdo for pressionado. Caso contrário, o carrinho deve andar para frente.
    3. Faça o carrinho fazer um quadrado pela direita se o botão direito for pressionado e fazer um quadrado pela esquerda se o botão esquerdo for pressionado.

4. Faça o carrinho andar para frente quando o botão direito ou esquerdo for pressionado e faça ele parar quando o botão enter for pressionado.
5. Faça as suas próprias instruções para o carrinho. Selecione as ações que quiser conforme os botões forem pressionados.

- Práticas Desplugadas

1. Jogue os dados e diga se a soma dos números foi par ou ímpar.
2. Identifique se a carta é um número (2 a 10), letra (A, J, Q, K) ou coringa.
3. Identifique o naipe de uma carta. Os naipes são: Espadas; Paus; Copas; Ouros).
4. Cada jogador deve tirar 3 cartas. O jogador que tiver a soma maior dos números vence.
5. Cada jogador deve jogar os dados. Aquele que tirar a carta com o valor da soma dos valores dos dados vence.

Lista de Exercícios de Repetição

- Ensino tradicional

1. Faça um código que imprima os números ímpares de 1 a 100.
2. Faça um código que calcule a potência de um número. O usuário deve informar a base e o expoente.
3. Faça um código que imprima as tabuadas de 1 a 10.
4. Faça um código que peça uma certa quantidade N. Após isso, informar N letras e mostrar a quantidade de vogais e de consoantes.
5. Dada uma poupança com um saldo de R$ 1000,00 no início de janeiro de 2017. Determinar o saldo da poupança em janeiro de 2018. Sabe-se que o juros é de 0,5% ao mês e o aniversário de rendimento é todo dia 1.

- Robótica educacional

1. Faça com que o carrinho toque uma sequência de sons 3 vezes.
2. Faça com que o carrinho ande em “□”

3. Faça com que o carrinho ande em "S".
4. Faça com que o carrinho ande em “_|—|_”
5. Faça com que o carrinho faça um “□” e vá aumentando o tamanho do “□” a cada volta.

- Práticas Desplugadas

1. Tire as cartas até sair um número ímpar.
2. Informe o maior valor entre 5 cartas.
3. Pegue 10 cartas e divida por naipes. Os naipes são: Espadas; Paus; Copas; Ouros.
4. Some os valores das cartas até sair uma carta diferente de um número (A, J, Q, K e coringa).
5. Definam um limite e cada jogador deve tirar uma carta de cada vez. O jogador que ultrapassar o limite perde e o jogo termina.

Lista de Exercícios de Vetores

- Ensino tradicional (Considere o vetor (lista) tenha o tamanho 10)

1. Escreva um código que imprima o dobro dos elementos da lista a serem informados pelo usuário.
2. Faça um programa que leia 10 números e os imprima em ordem inversa a qual foram lidos.
3. Escreva um código que imprima a média dos elementos da lista a serem informados pelo usuário.
4. Faça um código que, considerando a lista **numeros**, crie duas novas lista com os números pares e ímpares da lista:
   numeros = [11,14,29,33,42,55,68]
5. Dado uma lista de inteiros, faça um código que exibe a frequência de cada elemento.

- Práticas Desplugadas

1. Insira 12 bolinhas em ordem alfabética na caixa.

2. Insira 12 bolinhas na caixa e fale a menor e maior letra presente. (Ordem: a, b, c....)
3. Insira 6 bolinhas na fileira de cima da caixa. Após isso, faça uma nova fileira abaixo invertendo a ordem das bolinhas de cima.
4. Insira 6 bolinhas na fileira de cima da caixa e inverta a ordem depois. (sem usar a fileira de baixo)
5. Insira 11 bolinhas na caixa, Após isso, ordene em ordem alfabética. O último lugar da caixa é o auxiliar.

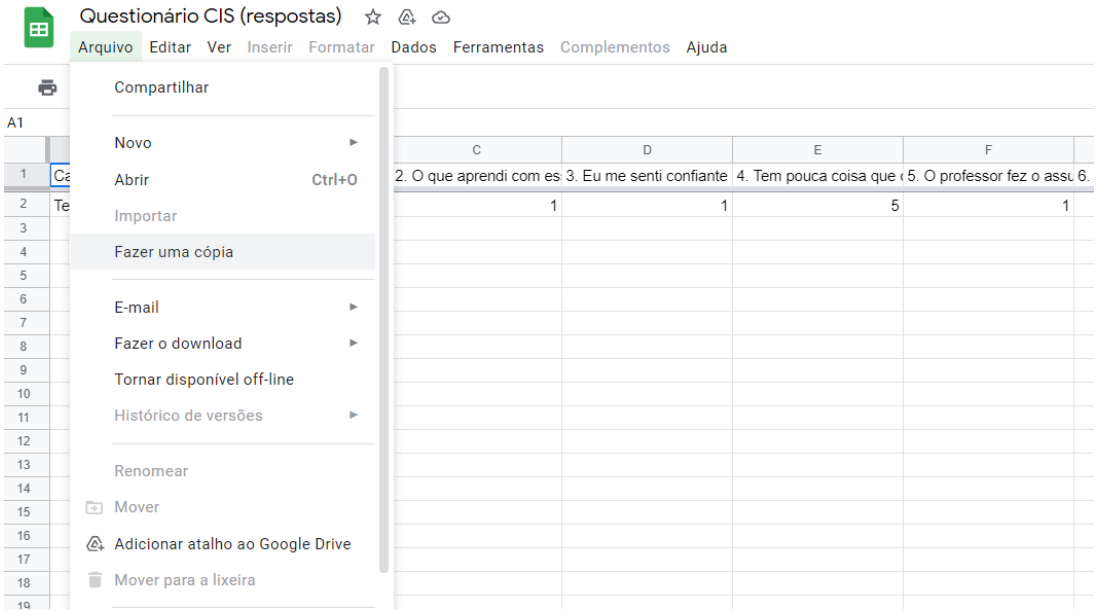
APÊNDICE C – Cópia do Modelo do Google Forms

Para obter a cópia do modelo do Google Forms do questionário CIS, é preciso seguir alguns passos:

1. Acesse a planilha do questionário[1].

2. Clique em “Fazer uma cópia”(Figura 22).

Figura 22 – Clique no “Fazer uma cópia”



Fonte: Autoria própria (2022).

3. Uma cópia da planilha do questionário é obtida.

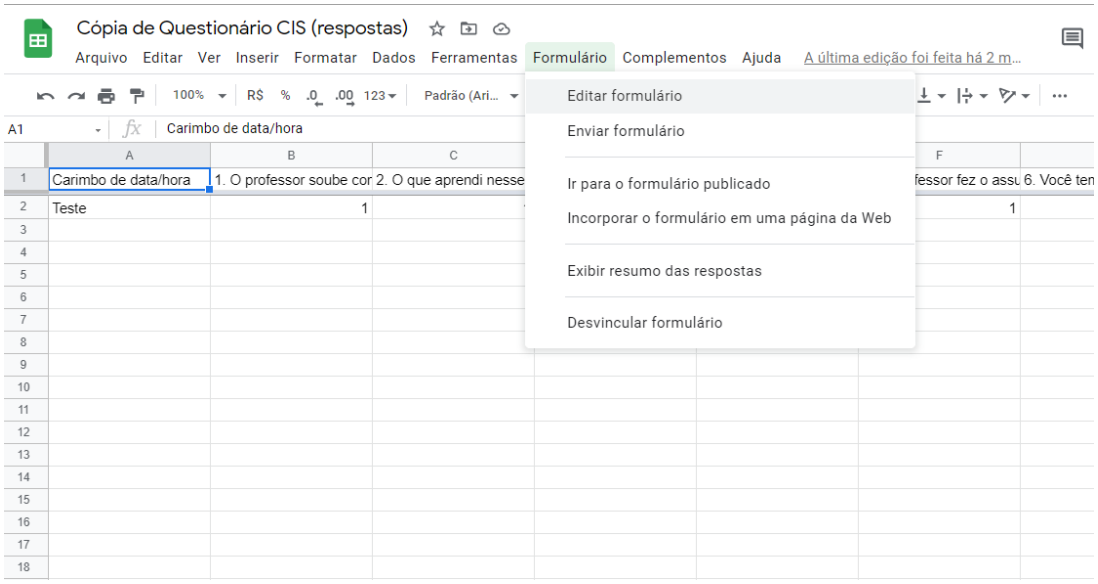
4. Na cópia da planilha, clique em “Editar o formulário” (Figura 23).

5. Uma cópia do questionário é obtida (Figura 24).

A planilha do questionário calcula a soma e a média dos valores de cada aspecto do ARCS e o escore total de cada resposta enviada e o questionário possui as 34 questões do CIS traduzidas e modificadas para o cenário da pesquisa.

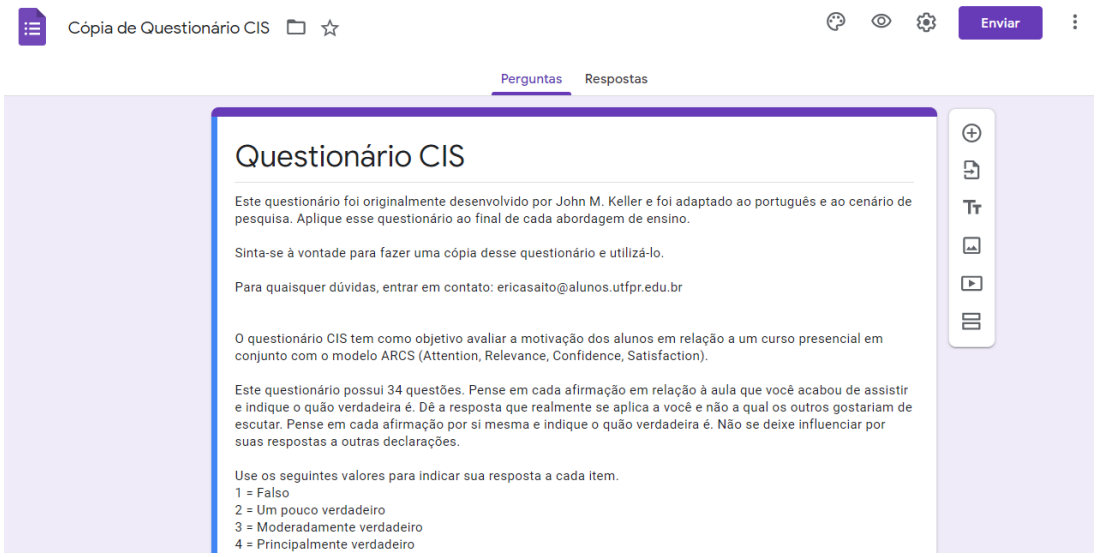
[1] https://docs.google.com/spreadsheets/d/1vm8X_aSezT0r3dlUndj43zdp8mwU7A8CW9kZcVgwQhk/edit?usp=sharing

Figura 23 – Clique no “Editar o formulário”



Fonte: Autoria própria (2022).

Figura 24 – Cópia do Questionário CIS



Fonte: Autoria própria (2022).