Égua Assist: Uma Ferramenta de Apoio ao Ensino de Lógica de Programação para Estudantes com Deficiência Visual

Daniel S. dos Santos[1], **Paula Christina F. Cardoso**[1], **Victor Hugo S. Costa Pinto**[1]

[1] Universidade Federal do Pará (UFPA) – Belém, PA – Brasil

daniel.santos@icen.ufpa.br, {pcardoso,victor.santiago}@ufpa.br

Abstract. *With the growing application of assistive technologies in education, this paper presents the implementation of an extension to the Égua programming language's integrated development environment, named Égua Assist – Coding by Voice. The tool is designed to support visually impaired students in learning programming logic. It combines a Portuguese-language speech recognition interface, a generative Artificial Intelligence model based on the Transformer architecture responsible for translating spoken prompts into Égua code, and a client-server infrastructure. The solution shows promising results, demonstrating the potential to make programming education more accessible and inclusive.*

Resumo. *Com o avanço de tecnologias assistivas aplicadas à educação, este artigo apresenta a implementação de uma extensão para o ambiente de desenvolvimento integrado da linguagem Égua, intitulada Égua Assist – Coding by Voice. A proposta visa apoiar estudantes com deficiência visual no processo de aprendizagem da lógica de programação. A ferramenta integra uma interface de reconhecimento de voz em português, um modelo de Inteligência Artificial generativa baseado na arquitetura Transformer, responsável por converter a transcrição do prompt falado em código na linguagem Égua, e um ambiente cliente-servidor. A solução apresenta resultados promissores, demonstrando potencial para tornar o ensino de programação mais acessível e inclusivo.*

1. Introdução

No contexto da inclusão educacional, um dos grupos que carecem de abordagens pedagógicas eficientes são as pessoas com deficiência visual (PcDV). O termo "deficiência visual" abrange uma variedade de condições, desde a cegueira total até a baixa visão [Gil 2000]. Com o avanço da acessibilidade no ensino de Ciência da Computação, surgem novos desafios para os educadores, que precisam se adaptar à maior presença de PcDV nesses contextos. Além disso, grande parte dos recursos pedagógicos voltados ao ensino de pensamento computacional ainda apresenta restrições significativas, pois privilegia a visualização por meio de gráficos, ícones, caracteres e números. Como resultado, as PcDV processam o conhecimento em um ambiente essencialmente visual, o que pode colocá-las em desvantagem no aprendizado [de Sá et al. 2007].

No contexto do ensino de lógica de programação para estudantes de graduação, diversos obstáculos se fazem presentes nas salas de aula. Koliver et al. (2004) sugerem que as dificuldades enfrentadas pelos alunos em disciplinas de algoritmos estão associadas

à falta de preparo para lidar com a formulação lógica de soluções. Por sua vez, Wilson (2017) aponta que esses estudantes frequentemente recorrem à mera "memorização" dos conteúdos, em vez de desenvolver uma compreensão aprofundada das estruturas algorítmicas, o que pode intensificar os desafios no processo de aprendizagem. Por fim, Arruda et al. (2024) destacam que o ensino de programação para graduandos frequentemente enfrenta entraves relacionados à dificuldade dos estudantes em assimilar abstrações, raciocínio lógico e conceitos essenciais.

Além disso, a presença de estudantes com deficiência visual em turmas de lógica de programação amplifica os desafios devido a restrições físicas, falta de preparação dos docentes e escassez de materiais acessíveis [Pereira et al. 2018]. Ribeiro Filho et al. (2024) afirmam que o ensino de programação para alunos com deficiência visual exige abordagens adaptativas que superem obstáculos de acessibilidade e incentivem a inclusão. Os autores enfatizam a importância de metodologias interativas, como atividades sensoriais, fluxogramas táteis e o uso de leitores de tela, além do suporte técnico e pedagógico.

Diante do contexto apresentado, este trabalho descreve o desenvolvimento de uma ferramenta denominada *Égua Assist*, cujo objetivo é apoiar pessoas com deficiência visual (PcDV) no processo de aprendizagem da lógica de programação, por meio do ambiente de desenvolvimento integrado (*Integrated Development Environment* (IDE), em inglês) da linguagem Égua [Neves 2025b], acessado via navegador *web*, denominado IDÉgua. A ferramenta combina tecnologias de reconhecimento de voz e geração automática de código, oferecendo um ambiente acessível e promissor para iniciantes no estudo de algoritmos e programação de computadores.

2. Fundamentação Teórica

2.1. Educação Inclusiva e Pensamento Computacional

De acordo com Mantoan (2005, p.96), a inclusão vai além de simplesmente aceitar a presença do outro; ela envolve a interação e o compartilhamento de experiências com aqueles que possuem diferenças em relação a nós. Estar fisicamente próximo de alguém não significa, necessariamente, que há uma verdadeira interação. O verdadeiro sentido da inclusão está na troca de ideias, vivências e aprendizados. Nesse contexto, a educação inclusiva não estabelece barreiras entre as pessoas, abrangendo todos os grupos que enfrentam discriminação e preconceito devido a suas condições físicas ou cognitivas.

Segundo Wing (2006), o pensamento computacional envolve não apenas a resolução de problemas, mas também o desenvolvimento de sistemas e a compreensão do comportamento humano, tendo como base os fundamentos da Ciência da Computação. Esse conceito abarca um conjunto de habilidades cognitivas que refletem a diversidade e a complexidade dessa área do conhecimento. Além disso, trata-se de uma competência essencial para todas as pessoas, não se limitando apenas aos especialistas em computação. O pensamento computacional considera tanto as possibilidades quanto as limitações da computação, seja ela realizada por seres humanos ou por máquinas. A utilização de métodos e modelos computacionais permite lidar com desafios complexos e criar soluções que seriam impraticáveis sem o auxílio dessas abordagens [Wing 2006].

2.2. A Linguagem Égua

A linguagem Égua [Neves 2025b] é uma opção acessível e de fácil compreensão para o ensino de programação em português. Desenvolvida com foco educacional, ela cria um

ambiente intuitivo e acolhedor para iniciantes, tornando o aprendizado mais simples e interativo. Por esse motivo, ela foi escolhida para o desenvolvimento desta proposta, pois a ferramenta *Égua Assist* é direcionada para estudantes que estão começando a aprender sobre os conceitos iniciais de algoritmos e programação.

A linguagem Égua destaca-se por sua sintaxe clara e regras bem estruturadas, o que torna a aprendizagem e a implementação de algoritmos mais acessíveis. Essa característica permite que os alunos concentrem seus esforços nos conceitos fundamentais da programação, sem serem sobrecarregados por detalhes excessivamente complexos [Neves 2025a]. Na Figura 1 descreve-se um bloco de código com a declaração de uma variável na linguagem Égua, incluindo a manipulação de funções de repetição.

```
var contador = 10;

enquanto (contador > 0)
{
  escreva ("Detonação em: " + texto(contador));
  contador = contador - 1;
}

escreva ("Booom!");
```

Figura 1. Declaração de variável e repetição utilizando a linguagem Égua.

2.3. Processamento de Linguagem Natural (PLN)

O Processamento de Linguagem Natural (PLN) tem como objetivo pesquisar e criar sistemas que permitam aos computadores entender, interpretar e produzir a linguagem humana de maneira eficiente. Além disso, esse campo está fortemente vinculado à Inteligência Artificial e à Linguística Computacional, áreas que se complementam dentro da Ciência da Computação [Caseli and Nunes 2024].

Neste estudo, o PLN é empregado na ferramenta *Égua Assist*, considerando dois cenários principais: (1) a conversão de fala em texto e (2) a utilização de um *prompt* de entrada para gerar trechos de código. No primeiro caso, a técnica de Geração de Linguagem Natural (*Natural Language Generation* (NLG), em inglês) é aplicada para transformar a entrada de áudio do usuário em um texto preciso. No segundo, um modelo de Inteligência Artificial interpreta o comando fornecido e produz a linha de código correspondente com base na solicitação do usuário.

2.4. Redes Neurais

Segundo Mehlig (2021), o conceito de redes neurais tem origem na biologia, referindo-se às conexões entre neurônios no cérebro dos mamíferos. Essas redes processam informações de forma dinâmica, ajustando-se a estímulos externos e modificando suas conexões ao longo do tempo para aprimorar funções. Essa capacidade de adaptação inspira os modelos computacionais de redes neurais artificiais, que buscam reproduzir o aprendizado e a evolução observados nas redes biológicas [Mehlig 2021].

Neste trabalho, foi utilizado uma arquitetura de rede neural denominada *Transformer*, cujo objetivo é processar os *tokens* extraídos da transcrição da fala do usuário, e gerar código em linguagem de programação Égua. Esse modelo, desenvolvido por Vaswani et al. (2017), trouxe uma grande transformação para o campo do Processamento de Linguagem Natural (PLN) e serviu de base para arquiteturas como BERT, GPT e T5. O *Transformer* foi projetado para lidar com sequências de dados, como textos, sem a necessidade de redes neurais recorrentes ou convolucionais. Em vez disso, ele utiliza um mecanismo chamado *self-attention*, que permite identificar e estabelecer conexões entre palavras em uma frase, independentemente da distância entre elas [Vaswani et al. 2017].

3. Trabalhos Relacionados

A literatura acadêmica apresenta diversas iniciativas para reduzir as barreiras enfrentadas por PcDV no aprendizado de lógica de programação. Algumas soluções incluem o *Emacspeak* [Raman 1996] e sistemas de programação com *feedback* sonoro, projetados para tornar o ambiente de codificação mais acessível a pessoas cegas [Stefik et al. 2009]. Estudos como os de Junior et al. (2009) e Ferreira et al. (2016) analisaram o uso de *podcasts* no ensino de algoritmos, destacando que recursos alternativos, como o áudio, podem aumentar o engajamento desses alunos. Além disso, Branham e Kane (2015) ressaltam a relevância de tecnologias assistivas, como *feedback* tátil e auditivo, para incentivar a colaboração entre crianças com diferentes níveis de visão.

Do Nascimento et al. (2023) apresentaram a ferramenta *IVProg4All*, um ambiente de programação visual baseado em HTML/CSS para integrar alunos com e sem deficiência visual. Já Pereira et al. (2018) exploraram o uso de fluxogramas físicos e leitores de tela[1], como DOSVOX[2], JAWS[3] e NVDA[4], combinados com a linguagem Pascal, oferecendo uma abordagem tátil e auditiva para o ensino de algoritmos. Essas estratégias promovem a inclusão e incentivam a colaboração entre estudantes com diferentes necessidades, tornando o aprendizado mais acessível e interativo.

Thieme et al. (2017) desenvolveram a linguagem Torino, que usa peças físicas (*beads*) e o método de Consulta Cooperativa para ensinar programação a crianças, incluindo PcDV. Rong et al. (2020) apresentaram o *CodeRhythm*, um sistema baseado em blocos tangíveis que ensina programação por meio de melodias, combinando *feedback* auditivo e tátil. Já Kakehashi et al. (2014) criaram o P-CUBE, um sistema que utiliza blocos com Identificação por Radiofrequência (*Radio Frequency Identification* (RFID), em inglês) para ensinar programação, permitindo que alunos com deficiência visual programem um robô móvel. Todas essas ferramentas adotam uma abordagem tátil, tornando o aprendizado mais acessível e favorecendo a inclusão de PcDV no ensino de programação.

Algumas abordagens citadas anteriormente de ensino de lógica de programação para PcDV ainda são insuficientes para suprir as principais necessidades de alunos com deficiência visual no aprendizado dessa disciplina, pois trabalham majoritariamente com

[1]O leitor de tela é um software utilizado principalmente por deficientes visuais, que fornece informações através de síntese de voz sobre os elementos exibidos na tela do computador [de Biomedicina (CRBM) 2021]

[2]https://intervox.nce.ufrj.br/dosvox/

[3]https://www.tecassistiva.com.br/catalogo/jaws/

[4]https://www.nvaccess.org/download/

recursos físicos e táteis para introduzir conceitos de programação. Embora essas estratégias sejam essenciais para a fase inicial do aprendizado, chega um momento em que o aluno precisa avançar para a escrita direta de código, enfrentando desafios como a sintaxe da linguagem e a organização estrutural dos programas. Nesse contexto, a ferramenta desenvolvida neste trabalho surge como um complemento a essas metodologias, permitindo que PcDV façam a transição do aprendizado tátil para a produção de código de maneira acessível e intuitiva. Ao integrar a conversão de voz para código na linguagem Égua, essa abordagem possibilita que os alunos desenvolvam autonomia na programação, ampliando o alcance das estratégias inclusivas já existentes.

4. Metodologia

Para o desenvolvimento deste trabalho, foi conduzido, inicialmente, um Mapeamento Sistemático da Literatura (MSL), um método apropriado para mapear um tópico específico onde existem evidências limitadas ou o tópico de pesquisa é vasto ou disperso [dos Santos et al. 2025]. Ao concluir o MSL, identificaram-se diversas lacunas no ensino de algoritmos para estudantes com deficiência visual. A primeira constatação é que a maioria das iniciativas existentes concentra-se no público infantojuvenil, havendo uma **escassez de abordagens voltadas especificamente para o contexto do ensino superior**. Em segundo lugar, observou-se que **as ferramentas disponíveis são mais direcionadas ao desenvolvimento inicial do raciocínio lógico e do pensamento computacional**, sem abranger aspectos mais avançados da construção de algoritmos, como a declaração de variáveis, estruturas condicionais, laços de repetição e outros elementos fundamentais para a programação.

Partindo desse cenário, iniciou-se o desenvolvimento de uma ferramenta que gera linhas de código em linguagem de programação Égua a partir de uma entrada de voz do usuário, para auxiliar PcDV de cursos de graduação em Computação no desenvolvimento e compreensão de lógica de programação. A escolha dessa linguagem de programação motivou-se pela sua estrutura baseada na língua portuguesa, e em sua alta flexibilidade e capacidade de execução direta em navegadores *web*, sem a necessidade de um sistema operacional específico.

Posterior à escolha da linguagem de programação, o próximo passo foi determinar como a ferramenta irá auxiliar PcDV na escrita de algoritmos, considerando que esses usuários possuem pouca ou nenhuma interação visual com interfaces gráficas. Em 2024, a Microsoft lançou o *Copilot Voice*[5], uma ferramenta que possibilita a escrita de código por meio da voz, destinada a usuários com dificuldades no uso do teclado. O sistema permite a navegação pelo código, o controle do ambiente de desenvolvimento integrado e o resumo do código, proporcionando explicações sobre trechos de código que possam ser incompreensíveis ao usuário. Inicialmente, a extensão está disponível apenas no *Visual Studio Code*[6].

A extensão *Copilot Voice* não possui o intuito de gerar pseudocódigo, mas sim códigos em linguagens de programação amplamente utilizadas, como *Python*, *Java*, entre outras. Ao considerar o propósito deste estudo, a escolha da linguagem Égua foi imprescindível, devido às características previamente destacadas. Em seguida, foi definida

[5]https://githubnext.com/projects/copilot-voice/

[6]https://code.visualstudio.com/

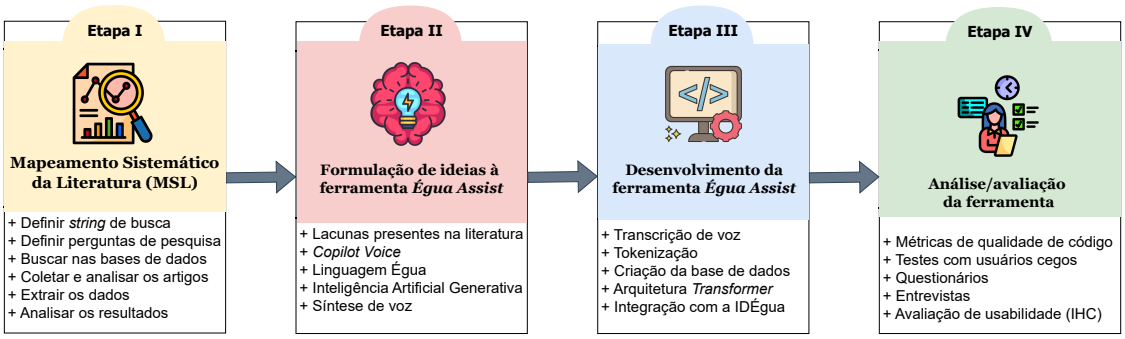

Figura 2. Resumo da metodologia aplicada no trabalho.

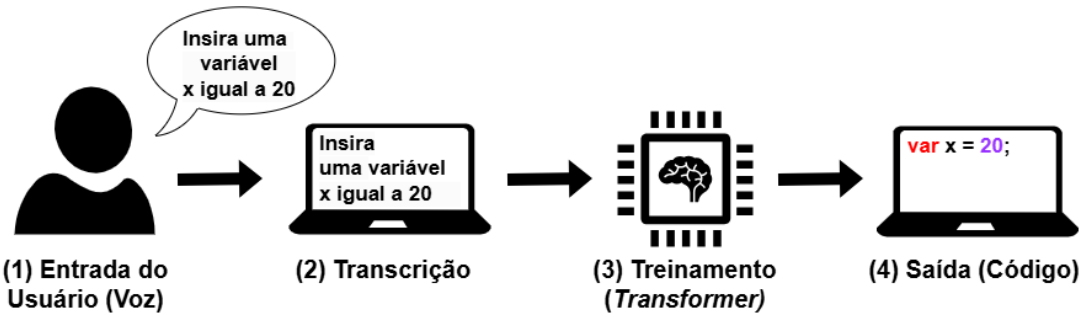

Figura 3. Resumo da metodologia aplicada na ferramenta *Égua Assist.*

a estratégia para integrar a IDE da linguagem selecionada. Inspirado no *Copilot Voice*, foi proposta uma interação por meio de comandos de voz, nos quais as respostas geradas seriam trechos de código na linguagem Égua. A Figura 2 ilustra a metodologia adotada, enquanto as etapas de desenvolvimento da ferramenta são apresentadas na Figura 3. Nas próximas seções descrevem-se as etapas de desenvolvimento da aplicação.

4.1. Conversor Voz-Texto

O primeiro passo para a elaboração da ferramenta proposta é a criação de um conversor de voz para texto em linguagem natural. Foram testadas duas Interfaces de Programação de Aplicação (*Application Programming Interface* (API), em inglês) que realizam essa tarefa no *backend*: a *Web Speech*[7] e a OpenAI *Whisper*[8].

A *Web Speech* permite integrar reconhecimento de voz e conversão de texto em fala em aplicativos *web*, sem depender de tecnologias específicas. Ela possibilita comandos de voz ou captação contínua de áudio e fornece múltiplas interpretações dos dados de reconhecimento para melhorar a interação [Liu et al. 2025]. Já a OpenAI *Whisper* é um serviço pago de conversão de fala em texto, utilizando o modelo *Whisper large-v2*. Ela oferece transcrições e traduções de áudio, retornando resultados em formatos como JSON ou texto simples [OpenAI 2025].

Na página HTML da IDÉgua, foi adicionado um botão para ativar o microfone e duas saídas de texto: uma para a *Web Speech* e outra para a OpenAI *Whisper*. No *JavaScript*, foram implementados métodos da *Web Speech* para iniciar o reconhecimento

[7]https://webaudio.github.io/web-speech-api/
[8]https://openai.com/index/whisper/

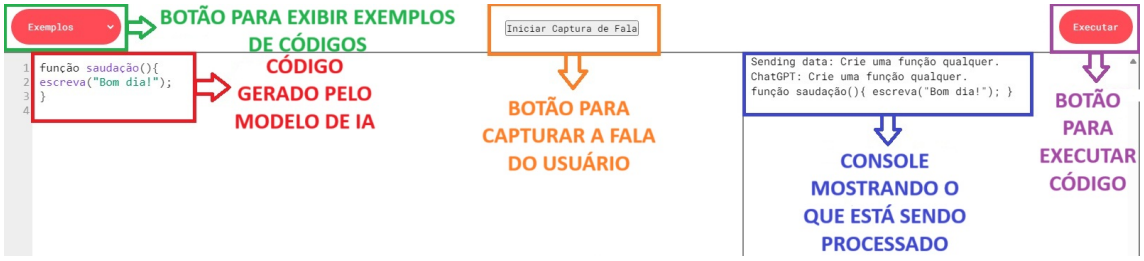

Figura 4. Alterações implementadas na IDÉgua: captura de voz, geração de código e console com o processamento dos dados.

de voz, armazenar transcrições, reiniciar a captura de fala, tratar erros e verificar a entrada de voz. Por outro lado, comunicação com a API *Whisper* é feita via POST, utilizando uma chave secreta para processar o áudio e retornar o texto. A API Whisper é tarifada com base no tempo de áudio processado, adotando um modelo de cobrança por minuto. Atualmente, o custo é de US$ 0,006 por minuto de áudio transcrito[9].

As duas escolhas, *Web Speech* e OpenAI *Whisper*, são complementares e podem ser usadas juntas neste projeto. A *Web Speech* é útil para reconhecimento de voz em tempo real diretamente no navegador, enquanto a API *Whisper* oferece transcrições e traduções mais precisas, com um modelo mais robusto, mas requer um serviço pago e processamento externo. Se o projeto precisar de um sistema de reconhecimento de voz simples e acessível, a *Web Speech* pode ser suficiente. No entanto, para transcrições e traduções mais complexas e precisas, a *Whisper* seria a escolha ideal.

4.2. Tokenização

De acordo com Caseli e Nunes (2024), um *token* é qualquer conjunto de símbolos com significado específico. Em idiomas europeus, a tokenização divide o texto por espaços, mas em muitos idiomas globais, essa divisão não se baseia em espaços. *Tokens* incluem palavras e símbolos gráficos, como vírgulas e pontos finais.

Na linguagem Égua, os *tokens* incluem palavras reservadas, variáveis, valores numéricos, operadores, delimitadores, cadeias de caracteres e espaços, gerados por regras definidas pela sintaxe da linguagem. A tokenização converte fala transcrita e código em elementos discretos, facilitando a manipulação pelo modelo *Transformer*. Esse processo é aplicado tanto na entrada quanto nos dados de treinamento, e após a geração de saídas, os *tokens* são convertidos de volta em texto ou código compreensível.

Na Figura 5 ilustra-se um exemplo de código escrito na linguagem Égua, no qual são atribuídos valores inteiros a duas variáveis, seguidos da impressão de seus respectivos valores. Na linha 7 do bloco, são exibidos os *tokens* extraídos do código-fonte.

4.3. Base de Dados

Para treinar o *Transformer*, foi criada uma base de dados com *prompts* em português e código em Égua, gerados artificialmente, já que não existia uma base pronta. Utilizaram-se o GPT-4o, *Deepseek* V3 e *Gemini* 2.0 Flash para criar 200 exemplos de treino/teste a partir de *prompts* padronizados, sendo 65 ao *Deepseek*, 48 ao GPT e 87 ao *Gemini*. A

[9]https://platform.openai.com/docs/pricing

```
var a = 10;
var b = 4;

escreva("Valor de A: " + texto(a));
escreva("Valor de B: " + texto(b));

[(1, 'var'), (2, 'a'), (54, '='), (3, '10'), (55, ';'), (1, 'var'),
    (2, 'b'), (54, '='), (3, '4'), (55, ';'), (1, 'escreva'), (55,
    '('), (56, '"Valor de A: "'), (54, '+'), (2, 'texto'), (55, '('),
    (2, 'a'), (55, ')'), (55, ')'), (55, ';'), (1, 'escreva'), (55,
    '('), (56, '"Valor de B: "'), (54, '+'), (2, 'texto'), (55, '('),
    (2, 'b'), (55, ')'), (55, ')'), (55, ';'), (4, ''), (0, '')]
```

Figura 5. *Tokens* extraídos do código fonte.

variação nas quantidades aconteceu devido às redundâncias e erros nos códigos gerados. A Tabela 1 ilustra o desempenho dos modelos na geração dos dados, sendo que o Gemini teve o melhor desempenho.

Tabela 1. Desempenho dos Modelos na Geração de Código

Modelo	Qtd. de Código Gerado	Erros e Redundâncias	% de Aproveitamento
Deepseek	100	35	65%
GPT	100	52	48%
Gemini	100	13	87%

A principal razão à criação de uma base de treinamento artificial foi a ausência de bancos de dados com códigos em linguagem Égua. Além disso, a comunidade ativa de programadores em linguagem Égua ainda é bastante restrita, e os exemplos de código disponíveis na *web* são extremamente limitados. A linguagem Égua é bastante nova, com a sua primeira versão disponível no GitHub (1.1.14) datada de fevereiro de 2021. A Figura 6 mostra um resumo do processo de criação da base de dados de treinamento.

4.4. O modelo *Transformer*

Para o desenvolvimento da ferramenta de conversão de voz para código em linguagem Égua, uma das etapas cruciais é a implementação de um modelo de Inteligência Artificial generativa denominada *Transformer*. Neste trabalho, O modelo foi treinado em 500 iterações completas sobre o conjunto de treinamento e teve 6,969,586 de parâmetros treináveis. O modelo foi treinado em um computador com processador AMD Ryzen 5 Mobile 5500U, com 8GB de Memória RAM, gráficos integrados AMD Radeon 2GB, e 256GB de SSD NVMe.

A Tabela 2 descreve os parâmetros do modelo.

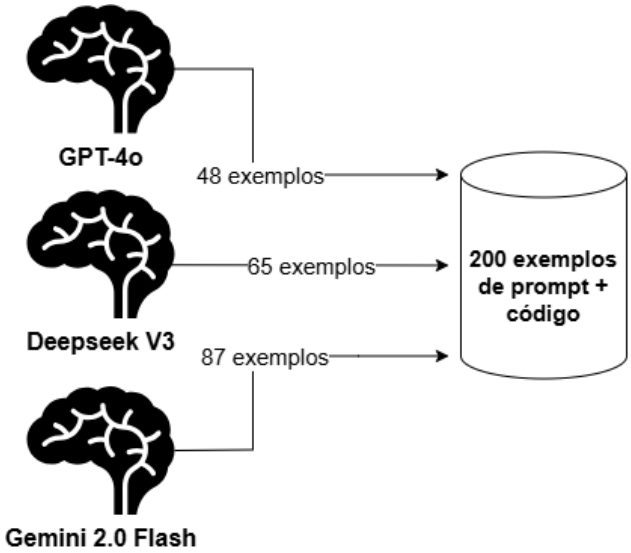

Figura 6. Esquema de geração dos dados artificiais.

Tabela 2. Parâmetros do modelo *Transformer*

Parâmetros	**Descrição**
INPUT_DIM / OUTPUT_DIM	Tamanho do vocabulário da entrada (frases em linguagem natural) e da saída (código na linguagem Égua).
HID_DIM (256)	Controla o tamanho das representações internas usadas pelo *Transformer*.
ENC_LAYERS / DEC_LAYERS (3)	Número de camadas no *Encoder* e *Decoder* do *Transformer*.
ENC_HEADS / DEC_HEADS (16)	Número de cabeças de atenção no mecanismo de *self-attention*.
ENC_PF_DIM / DEC_PF_DIM (512)	Tamanho das camadas *feedforward* dentro de cada bloco *Transformer*.
ENC_DROPOUT / DEC_DROPOUT (0.1)	Taxa de *dropout* para reduzir *overfitting* desligando 10% dos neurônios.
LEARNING_RATE (0.0005)	Taxa que define o tamanho do passo na atualização dos pesos do modelo.
BATCH_SIZE (16)	Número de exemplos processados antes de atualizar os pesos.

4.5. A comunicação entre o *backend* e a IDÉgua

Para que a IDÉgua se comunique com o modelo *Transformer* já treinado, foi necessário implementar um *backend* em *Python* para que a IDE enviasse o *prompt* do usuário transcrito à rede neural e recebesse de volta o código sugerido pelo modelo. Além disso, também foi implementado um *backend* em PHP para facilitar a comunicação da IDÉgua com os servidores da OpenAI, a fim de utilizar a API *Whisper* para transcrever os comandos de voz do usuário. A Figura 7 ilustra esse processo.

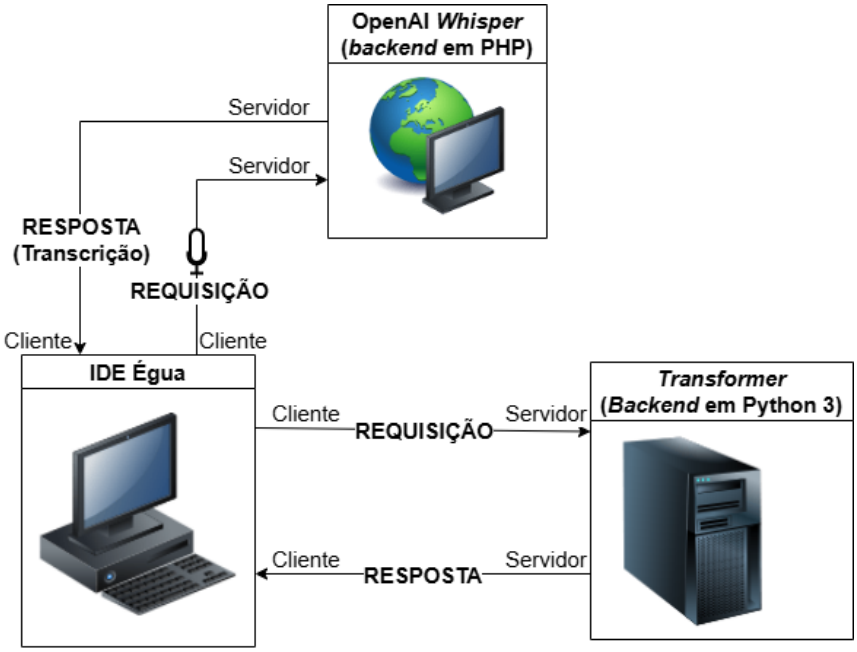

Figura 7. Arquitetura cliente-servidor utilizada na ferramenta *Égua Assist*.

5. Resultados Esperados e Considerações Finais

Este trabalho apresentou uma ferramenta voltada para a inclusão de pessoas com deficiência visual no aprendizado de programação, denominada *Égua Assist - Coding by Voice*[10]. A solução proposta combina reconhecimento de voz e geração automática de código na linguagem Égua, possibilitando que os usuários interajam com o ambiente de programação por meio de comandos falados. A expectativa é que essa abordagem facilite o ensino e a prática da programação, promovendo maior acessibilidade e autonomia para esse público.

Após a aprovação pelo Comitê de Ética em Pesquisa (CEP), a ferramenta será testada em ambientes reais para avaliar sua usabilidade, acessibilidade e efetividade no ensino de lógica de programação para PcDV. O processo de avaliação incluirá coleta estruturada de *feedback* qualitativo, por meio de entrevistas semiestruturadas [Seidman 2006, Nicolaci-da Costa et al. 2004, Blandford 2013] e observação direta das interações dos usuários com a ferramenta, buscando identificar dificuldades, estratégias e sugestões de melhorias. O *feedback* quantitativo será obtido com questionários baseados em escalas Likert, que avaliarão usabilidade, acessibilidade, compreensão dos conceitos lógicos e satisfação geral.

Como trabalhos futuros, planeja-se aprimorar o modelo de processamento de linguagem natural utilizado, buscando aumentar a precisão da conversão de voz para código. Outra possibilidade é a integração com ambientes de ensino de programação já estabelecidos. Em suma, este trabalho representa um passo significativo para a acessibilidade no ensino de programação, e há um grande potencial para futuras melhorias e expansões que possam beneficiar um público ainda mais amplo.

[10]https://github.com/daniel-saavedra-santos/online-coding-for-blinds/

Referências

Arruda, J., Rocha, P. H., Francês, R., and Pinto, V. H. (2024). Explorando a robótica para mitigar desafios comportamentais e de aprendizado em programação na graduação. In *Anais do XXX Workshop de Informática na Escola*, pages 243–253, Porto Alegre, RS, Brasil. SBC.

Blandford, A. E. (2013). Semi-structured qualitative studies. Interaction Design Foundation.

Bottentuit Junior, J. B. and Coutinho, C. P. (2009). Podcast : uma ferramenta tecnológica para auxílio ao ensino de deficientes visuais.

Branham, S. M. and Kane, S. K. (2015). Collaborative accessibility: How blind and sighted companions co-create accessible home spaces. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 2373–2382, New York, NY, USA. Association for Computing Machinery.

Caseli, H. M. and Nunes, M. G. V., editors (2024). *Processamento de Linguagem Natural: Conceitos, Técnicas e Aplicações em Português*. BPLN, 3 edition.

de Biomedicina (CRBM), C. R. (2021). *Tecnologia Assistiva: Sugestões de leitores de tela para deficientes visuais*. Acessado em: 8 de março de 2025.

de Sá, E. D., de Campos, I. M., and Silva, M. B. C. (2007). Formação continuada a distância de professores para o atendimento educacional especializado - deficiência visual.

Do Nascimento, M. D., Brandão, A. A. F., De Oliveira Brandão, L., and Casal, T. M. (2023). Towards ivprog4all: An accessibility test with blind. In *2023 IEEE Frontiers in Education Conference (FIE)*, pages 01–05.

dos Santos, D. S., Shibata, N. N., and Pinto, V. H. S. C. (2025). Teaching programming logic for people with blindness or visual impairments: a systematic mapping study. EasyChair Preprint 15953. "no prelo".

Ferreira, C., Anjos, J., Normando, J., Castro, M., Odakura, V., Sacchi, R., and Barvinski, C. (2016). Uso de podcast para apoio a aprendizagem de algoritmos em curso de graduação em computação. In *Anais dos Workshops do Congresso Brasileiro de Informática na Educação*, volume 5, page 1208.

Filho, J. R., dos Santos, D., Macedo, J. G., and Araújo, F. (2024). Ensino de algoritmos para pessoa com deficiência visual: Um relato de experiência no ensino superior. In *Anais do XXX Workshop de Informática na Escola*, pages 187–198, Porto Alegre, RS, Brasil. SBC.

Gil, M. (2000). *Deficiência Visual*. Number 1 in Cadernos da TV Escola. MEC, Secretaria de Educação a Distância, Brasília. Ilustrado.

Kakehashi, S., Motoyoshi, T., Koyanagi, K., Oshima, T., Masuta, H., and Kawakami, H. (2014). Improvement of p-cube: Algorithm education tool for visually impaired persons. In *2014 IEEE Symposium on Robotic Intelligence in Informationally Structured Space (RiiSS)*, pages 1–6.

Koliver, C., Dorneles, R. V., and Casa, M. E. (2004). Das (muitas) dúvidas e (poucas) certezas do ensino de algoritmos. *Anais do XII Workshop de Educação em Computação*, 24.

Liu, E., Natal, A., Shires, G., Jägenstedt, P., and Wennborg, H. (2025). Web speech api. `https://webaudio.github.io/web-speech-api/`. Acessado em: 10 de fevereiro de 2025.

Mantoan, M. T. E. (2005). Inclusão é o privilégio de conviver com as diferenças. *Nova Escola*.

Mehlig, B. (2021). *Machine learning with neural networks: an introduction for scientists and engineers*. Cambridge University Press.

Neves, L. P. (2025a). *Documentação — Linguagem Égua*. Programação simples e moderna em português. Acessado em: 8 de março de 2025.

Neves, L. P. (2025b). Linguagem Égua. `https://egua.dev/`. Programação simples e moderna em português. Acessado em: 1 de março de 2025.

Nicolaci-da Costa, A. M., Leitão, C. F., and Romão-Dias, D. (2004). Como conhecer usuários através do método de explicitação do discurso subjacente (meds). *VI Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais, IHC*, pages 47–56.

OpenAI (2025). *Speech to text*. Acessado em: 02 de março de 2025.

Pereira, R. M., da Silva, F. F., and Silla, C. N. (2018). Teaching algorithms for visually impaired and blind students using physical flowcharts and screen readers. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–9.

Raman, T. V. (1996). Emacspeak—a speech interface. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '96, page 66–71, New York, NY, USA. Association for Computing Machinery.

Rong, Z., Chan, N. f., Chen, T., and Zhu, K. (2020). Coderhythm: A tangible programming toolkit for visually impaired students. In *Proceedings of the Eighth International Workshop of Chinese CHI*, Chinese CHI '20, page 57–60, New York, NY, USA. Association for Computing Machinery.

Seidman, I. (2006). *Interviewing as qualitative research: A guide for researchers in education and the social sciences*. Teachers college press.

Stefik, A., Haywood, A., Mansoor, S., Dunda, B., and Garcia, D. (2009). Sodbeans. In *2009 IEEE 17th International Conference on Program Comprehension*, pages 293–294.

Thieme, A., Morrison, C., Villar, N., Grayson, M., and Lindley, S. (2017). Enabling collaboration in learning computer programing inclusive of children with vision impairments. In *Proceedings of the 2017 Conference on Designing Interactive Systems*, DIS '17, page 739–752, New York, NY, USA. Association for Computing Machinery.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wilson, G. (2017). *How to Teach Programming (And Other Things)*. Lulu.com.

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, 49(3):33–35.