DESENVOLVIMENTO DE INTERFACE DE VOZ PARA PROGRAMAÇÃO BACK-END: UM ESTUDO SOBRE ERGONOMIA E ACESSIBILIDADE

ISABELLY NOGAROTO RIBEIRO, MARIA EDUARDA ALVES SBARZI RODRIGUES , PALOMA DA SILVA FREITAS, WAGNER DOS SANTOS CLEMENTINO JESUS .

Universidade do Vale do Paraíba / Instituto de Pesquisa e Desenvolvimento, Av. Shishima Hifumi, 2911, Urbanova, 12244-000, São José dos Campos-SP, Brasil, isabellynribeiro@gmail.com, alvesrodrigues534@gmail.com, palomafreitas600@gmail.com, wagner@univap.br.

Resumo

O crescimento do setor de tecnologia impulsionou a oferta de empregos, mas evidenciou lacunas na acessibilidade para Pessoas com Deficiência (PcD) e riscos de Lesões por Esforços Repetitivos (LER/DORT) devido ao uso contínuo de teclados e mouses. Diante disso, este projeto teve como objetivo geral desenvolver uma ferramenta de construção de código-fonte baseada em reconhecimento de voz para o desenvolvimento back-end. A metodologia fundamentou-se no Design Sprint e no Design Thinking para a validação rápida da solução, cuja arquitetura técnica foi implementada em C# (Windows Forms) com integração ao Microsoft Speech Services para captura fonética em tempo real e saída de scripts estruturados em Python. Os resultados demonstraram que o dicionário de voz customizado processa e formata comandos lógicos e indentação de maneira semanticamente correta, superando limitações de softwares de ditado convencionais. Conclui-se que a ferramenta viabiliza um ambiente de programação inclusivo, ergonômico e de baixo custo técnico, promovendo acessibilidade e saúde ocupacional no ecossistema de software.

Palavras-chave: Acessibilidade. Programação por voz. Python. LER/DORT. Design Sprint.

Área do Conhecimento: Ciências Exatas e da Terra / Ciência da Computação.

Introdução

O setor de desenvolvimento de software no Brasil apresenta expansão constante, impulsionado pela aceleração digital recente (Macedo, 2022). Contudo, essa expansão evidencia a carência na inclusão de Pessoas com Deficiência (PcD) e no bem-estar dos programadores, uma vez que o uso prolongado de periféricos tradicionais auxilia os riscos de Lesões por Esforços Repetitivos (Escudero, 2024). Barreiras físicas e tecnológicas limitam a participação do indivíduo (ONU, 2006), sendo acentuadas pela LER/DORT, que transforma o teclado e o mouse em obstáculos para a atuação profissional (Maciel, 2001). Como alternativa, a comunicação por voz surge para diminuir as dificuldades de acessibilidade e saúde (Liu, 2010; Oliveira; Damasceno, 2021)

A justificativa deste trabalho reside na necessidade de eliminar a dependência exclusiva das mãos no processo de programação, promovendo tanto a inclusão socioprofissional de PcDs quanto a prevenção de lesões ocupacionais em programadores. Além do impacto social e de saúde, a pesquisa justifica-se do ponto de vista técnico ao explorar a viabilidade do reconhecimento de voz na sintaxe do desenvolvimento *back-end*, contribuindo para a área de Interação Humano-Computador (IHC).

Diante desse cenário, o projeto busca o desenvolvimento de software sem a dependência exclusiva das mãos. O objetivo geral é desenvolver uma ferramenta de construção de código-fonte baseada em reconhecimento de voz para o desenvolvimento *back-end*. Para isso, os objetivos específicos

consistem em: a) identificar as palavras-chave do desenvolvimento em *Python* (Van Rossum; Drake, 2009) para criar um dicionário de voz; b) desenvolver uma interface que converta fonemas em sintaxe de programação semântica; e c) estabelecer um protocolo de tratamento de erros para comandos de fala mal interpretados.

Metodologia

A condução do projeto fundamenta-se na metodologia *Design Sprint*, um processo de validação rápida baseado em *Design Thinking* e práticas ágeis. Enquanto o *Design Thinking* foca na resolução de problemas de forma centrada no usuário (Silva, 2018), o *Design Sprint* organiza essa mentalidade em um ciclo intensivo de cinco dias para prototipação e testes com riscos reduzidos (Knapp, 2016).

O sistema foi consolidado como uma aplicação desktop em C# com .NET Framework e Windows Forms (Corrêa, 2014; Lima; Reis, 2002). A entrada de dados ocorre por comandos de voz via atalho Windows + H, utilizando os *Microsoft Speech Services* para conversão de áudio em texto em tempo real por meio de processamento em nuvem baseado em redes neurais profundas (Yu; Deng, 2016).

A Figura 1 detalha o processamento do reconhecimento de voz em cinco etapas, fundamentadas nos modelos de processamento de linguagem e redes neurais (Jurafsky; Martin, 2024; Yu; Deng, 2016): na Captação e Pré-Processamento (1), o áudio analógico é digitalizado e filtrado; na Extração de Características (2), o sinal passa por segmentação (*framing*), janelamento (*windowing*), *Fast Fourier Transform* ou Transformada Rápida de Fourier (FFT) e *Mel-Frequency Cepstrum Coefficients* (MFCCs) para gerar o vetor acústico; o Modelo Acústico (3) aplica Redes Neurais Profundas para mapear os vetores em unidades fonéticas (Soares; Silva; Souza, 2023); a Decodificação (4) cruza essas probabilidades com o Modelo de Linguagem e Léxico via algoritmos como *Viterbi* ou *Beam Search* (Tan; Zhang, 2023; Gulati et al., 2023); por fim, no Reconhecimento e Saída (5), a sequência de palavras é exibida na interface.

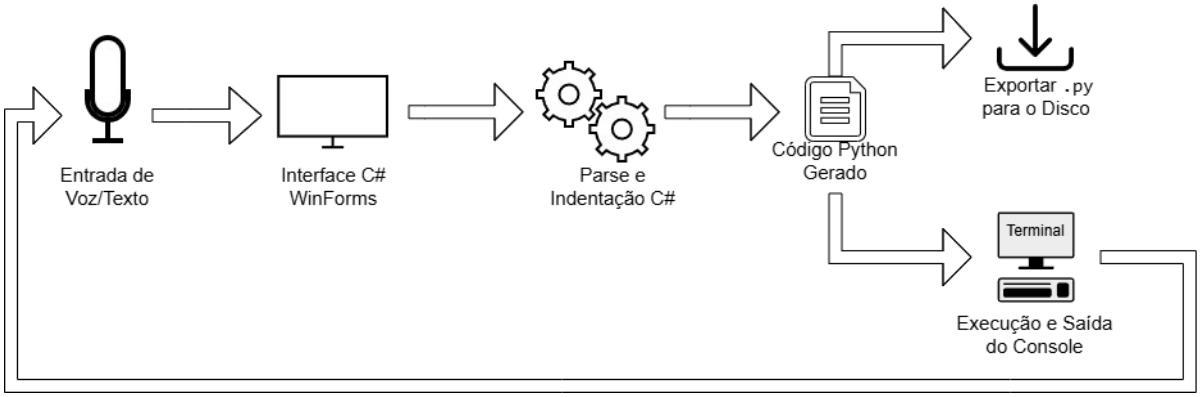
Figura 1 - Diagrama de funcionamento do sistema de reconhecimento de voz.



Fonte: Os Autores, gerada por IA (Google Gemini).

A integração dessas tecnologias no fluxo global da aplicação é sintetizada na Figura 2. O ciclo inicia-se com a Entrada de Voz/Texto, cujos dados transcritos são recebidos pela Interface C# WinForms. Em seguida, o sistema realiza o Parse e Indentação C#, que converte o texto capturado nas estruturas sintáticas correspondentes. Essa etapa resulta no Código Python Gerado, o qual segue por dois caminhos em paralelo: a persistência do arquivo via Exportar .py para o Disco e a Execução e Saída do Console no terminal. O feedback do terminal encerra o fluxo, mantendo o sistema pronto para novas entradas de comandos.

Figura 2 - Diagrama de funcionamento geral do Sistema.



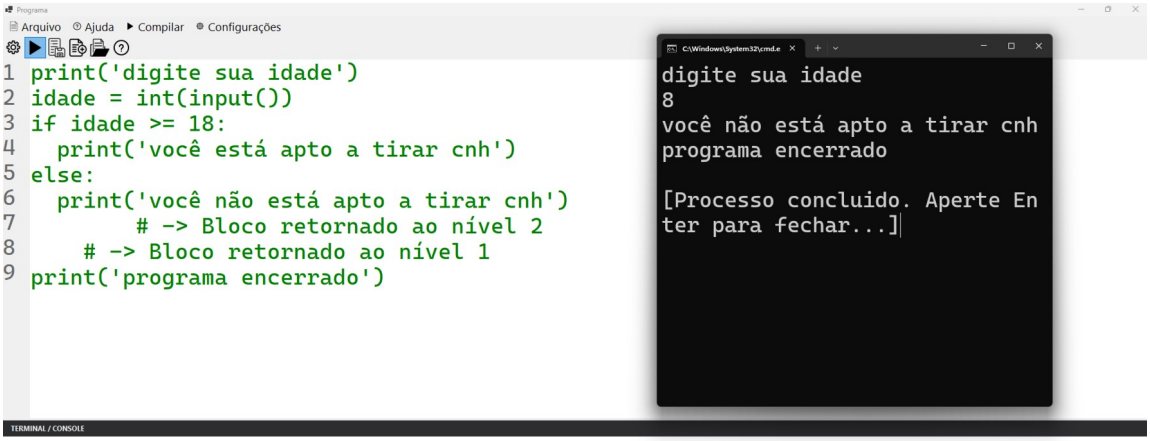
Fonte: Autores, 2026.

Resultados

Os objetivos voltados à identificação de palavras-chave e estruturas recorrentes no desenvolvimento back-end em *Python* foram alcançados, fundamentando a criação de um dicionário de fala integrado a uma interface que converte fonemas em sintaxe de programação semântica. A interface principal do sistema, projetada em *Windows Forms* com foco em usabilidade e simplicidade cognitiva, é estruturada para fornecer feedback visual imediato ao desenvolvedor.

Essa interface principal, conforme indica a figura 2, apresenta um layout desenvolvido para otimizar o conforto visual durante longos períodos de uso. A organização dos elementos é vertical, concentrando no menu lateral esquerdo as opções operacionais de acesso rápido. Na parte central superior, localiza-se o editor de código, onde o script Python gerado a partir do reconhecimento de voz é exibido em tempo real com a formatação correta da sintaxe. Logo abaixo, o terminal inferior de execução funciona como o console, indicando o status do programa e exibindo as saídas processadas, o que comprova o funcionamento prático do compilador interno sem dependências externas.

Figura 2- Interface principal do ambiente de desenvolvimento por voz.



Fonte: Autores, 2026

Discussão

Para validar a relevância da ferramenta, realizou-se uma análise comparativa com softwares de reconhecimento de voz comerciais, como o *Dragon Professional* e ferramentas nativas de ditado operacional. Embora os sistemas de mercado possuam algoritmos complexos, estes não compreendem a semântica de linguagens de programação, gerando erros severos de sintaxe e indentação.

Em contrapartida, a solução desenvolvida destaca-se por focar exclusivamente no mapeamento de estruturas *back-end* em *Python*, garantindo um código semanticamente correto. Sob a perspectiva econômica, enquanto softwares comerciais demandam licenças de alto custo, o sistema proposto demonstrou viabilidade financeira ao operar de forma otimizada em ambientes *Windows* convencionais por meio da plataforma .NET.

Conclusão

O desenvolvimento deste projeto permitiu viabilizar uma alternativa tecnológica eficaz para a programação de software sem a dependência exclusiva de periféricos físicos tradicionais. Ao integrar o reconhecimento de voz dos Serviços de Fala da Microsoft com uma interface *desktop* em C#, o sistema provou ser capaz de interpretar comandos fonéticos e convertê-los em código *Python* semanticamente correto e devidamente indentado. Dessa forma, o objetivo geral da pesquisa foi plenamente atendido, gerando uma ferramenta que atua tanto na inclusão de programadores com deficiências motoras quanto na prevenção de lesões ocupacionais, como a LER/DORT.

A análise comparativa validou o diferencial de mercado do software proposto, que alia especificidade técnica para codificação *back-end* a um baixo custo de investimento técnico por operar na plataforma .NET em ambientes Windows convencionais. Como limitações do ciclo atual, aponta-se a não conclusão de um protocolo robusto de tratamento de erros para fonemas mal interpretados. Para

trabalhos futuros, recomenda-se o refinamento desse módulo de correção, a expansão do dicionário fonético para outras linguagens de programação e a realização de testes de usabilidade com grupos de controle de desenvolvedores.

Referências

CORRÊA, A. M. **Programação C#**. São Paulo: Érica, 2014.

ESCUDERO, R. Como o Home Office pode prejudicar a saúde das suas mãos? **Blog Dr. Ricardo Escudero**, 19 dez. 2024. Disponível em: https://drricardoescudero.com.br/2024/12/19/como-o-home-office-pode-prejudicar-a-saude-das-suas-maos/. Acesso em: 27 mai. 2026.

GULATI, A. et al. Conformer: Convolution-augmented Transformer for Speech Recognition. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Piscataway, v. 45, n. 5, p. 6210-6224, maio 2023.

JURAFSKY, D.; MARTIN, J. H. **Speech and Language Processing**: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. 3. ed. draft. Stanford: Stanford University, 2024.

KNAPP, J. **Sprint**: como resolver grandes problemas e testar novas ideias em apenas cinco dias. Rio de Janeiro: Intrínseca, 2016.

LIMA, A. S.; REIS, M. A. **Desenvolvimento de aplicações com Windows Forms e .NET**. Rio de Janeiro: Ciência Moderna, 2002.

LIU, W. Natural user interface — next mainstream product user interface. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER-AIDED INDUSTRIAL DESIGN & CONCEPTUAL DESIGN, 11., 2010, Yiwu. **Proceedings...** Yiwu: IEEE, 2010. p. 203-205. DOI: 10.1109/CAIDCD.2010.5681374.

MACEDO, A. P. **Aceleração digital e o mercado de software no Brasil**. São Paulo: Novatec, 2022.

MACIEL, V. **LER**: lesões por esforços repetitivos. São Paulo: LTr, 2001.

OLIVEIRA, R. S.; DAMASCENO, L. F. Acessibilidade e interfaces de voz na tecnologia assistiva. **Revista Brasileira de Tecnologia Assistiva**, v. 4, n. 1, p. 45-58, 2021.

ORGANIZAÇÃO DAS NAÇÕES UNIDAS. **Convenção sobre os Direitos das Pessoas com Deficiência**. Nova York: ONU, 2006.

SILVA, E. J. O Design SPRINT como ferramenta para engajamento da equipe: um estudo de caso. **Human Factors in Design**, Florianópolis, v. 7, n. 13, p. 191-202, fev./jul. 2018. DOI: 10.5965/2316796307132018191.

SOARES, G. J. R.; SILVA, J. E. C.; SOUZA, J. F. de. Avaliação de modelos para reconhecimento automático de fala aplicados para identificação da qualidade de leituras em voz alta. **Revista Brasileira de Informática na Educação**, Porto Alegre, v. 31, p. 112-128, 2023.

SWAROOP, C. H. **A Byte of Python**. [S. l.: s. n.], 2004. Disponível em: https://python.swaroopch.com/about_python.html. Acesso em: 10 abr. 2026.

TAN, T.; ZHANG, C. **Recent Advances in End-to-End Speech Recognition**. Singapore: Springer, 2023.

VAN ROSSUM, G.; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley: CreateSpace, 2009.

YU, D.; DENG, L. **Automatic Speech Recognition**: A Deep Learning Approach. London: Springer, 2016. (Signals and Communication Technology).