INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* SÃO JOÃO EVANGELISTA
SISTEMAS DE INFORMAÇÃO

Gustavo Brendon Gomes Pimenta
Henrique Sabino Sales

APRIMORAMENTO E EXPANSÃO DE UM SISTEMA *WEB* DE APOIO PARA ESTUDANTES DE COMPUTAÇÃO

São João Evangelista
2026

GUSTAVO BRENDON GOMES PIMENTA
HENRIQUE SABINO SALES

APRIMORAMENTO E EXPANSÃO DE UM SISTEMA *WEB* DE APOIO PARA ESTUDANTES DE COMPUTAÇÃO

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - *Campus* São João Evangelista para a obtenção do título de bacharel em Sistemas de Informação.

Orientador: Prof. Me. Rosinei Soares de Figueiredo

São João Evangelista
2026

P644a Pimenta, Gustavo Brendon Gomes.

Aprimoramento e expansão de um sistema *web* de apoio para estudantes de computação. / Gustavo Brendon Gomes Pimenta, Henrique Sabino Sales – 2026.

53f.;il.

Orientador: Me. Rosinei Soares de Figueiredo.

Trabalho de Conclusão de Curso (bacharelado em Sistemas de Informação) – Instituto Federal Minas Gerais. *Campus* São João Evangelista, 2026.

1. Tecnologia. 2. Acessibilidade. 3. Informação. 4. Educação. I. Pimenta, Gustavo Brendon Gomes. II. Sales, Henrique Sabino. III. Instituto Federal de Minas Gerais *Campus* SJE. IV. Título.

CDD 005.1

Catalogação: Esther Soares Cunha - CRB-6/4333

Gustavo Brendon Gomes Pimenta

Henrique Sabino Sales

APRIMORAMENTO E EXPANSÃO DE UM SISTEMA *WEB* DE APOIO PARA ESTUDANTES DE COMPUTAÇÃO

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais - *Campus* São João Evangelista para a obtenção do título de bacharel em Sistemas de Informação.

Aprovado em: 10/ 02/ 2026 pela banca examinadora:

Documento assinado digitalmente
gov.br
ROSINEI SOARES DE FIGUEIREDO
Data: 02/03/2026 13:45:58-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Rosinei Soares de Figueiredo - IFMG (Orientador)

Documento assinado digitalmente
gov.br
DAYLER VINICIUS MIRANDA ALVES
Data: 02/03/2026 13:53:39-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Dayler Vinicius Miranda Alves - IFMG

Documento assinado digitalmente
gov.br
FERNANDO HENRIQUES MAFRA
Data: 02/03/2026 13:03:56-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Fernando Henriques Mafra - IFMG

AGRADECIMENTOS

Agradecemos às nossas famílias, amigos e professores por acreditarem em nós e pelo constante incentivo na realização do trabalho. Agradecemos também a Deus por nos dar força todos os dias.

Agradecemos ao nosso orientador e todos que contribuíram ao nosso trabalho.

“Education is not preparation for life;
education is life itself."

John Dewey

RESUMO

Nas últimas décadas, a área de Tecnologia da Informação tem passado por transformações significativas, impulsionadas pelo surgimento de novas linguagens de programação, ferramentas e tecnologias. Paralelamente, o domínio da língua inglesa continua essencial para a compreensão de documentações técnicas, comandos e ambientes de desenvolvimento, dificultando o processo de aprendizado para iniciantes. Considerando esse cenário no Brasil, este trabalho apresenta o aprimoramento e a expansão do DicioCode, um sistema *web* de apoio ao ensino de programação desenvolvido originalmente por Milagres, Pimenta e Sales (2025). O processo de evolução do artefato foi estruturado em quatro fases sequenciais: fundamentação teórica e levantamento de requisitos, pesquisa de mercado para seleção de novos conteúdos, projeto de interface baseado nas Heurísticas de Nielsen e a implementação técnica utilizando Node.js, TypeScript e PostgreSQL. O sistema tem como foco auxiliar novos ingressantes da área tecnológica na compreensão de comandos e conceitos básicos da Computação e palavras-chave da programação, com suporte visual e linguístico, especialmente no que se refere à tradução e explicação de termos em inglês. A validação da eficácia do sistema deu-se por meio de uma pesquisa pública, envolvendo testes de usabilidade e questionários aplicados a discentes de diversos cursos e profissionais da área tecnológica. Os testes realizados confirmaram a eficácia do aprimoramento proposto, evidenciando uma interface intuitiva e conteúdos que atendem às necessidades dos usuários, conforme atestado pela pesquisa quantitativa e qualitativa realizada. Espera-se que esta nova versão do sistema contribua de forma mais eficaz para a autonomia dos estudantes e para a melhoria do processo de ensino-aprendizagem no contexto da Informática Educacional.

Palavras-chave: Tecnologia. Acessibilidade. Informação. Educação.

ABSTRACT

In recent decades, the Information Technology sector has undergone significant transformations, driven by the emergence of new programming languages, tools, and technologies. Concurrently, proficiency in the English language remains essential for understanding technical documentation, commands, and development environments, which can hinder the learning process for beginners. Considering this scenario in Brazil, this paper presents the enhancement and expansion of DicioCode, a web-based system to support programming education originally developed by Milagres, Pimenta and Sales (2025). The artifact's evolution process was structured into four sequential phases: theoretical foundation and requirements gathering, market research for new content selection, interface design based on Nielsen's Heuristics, and technical implementation using Node.js, TypeScript, and PostgreSQL. The system focuses on assisting newcomers in the Information Technology field to understand commands, basic Computing concepts, and programming keywords through visual and linguistic support, specifically regarding the translation and explanation of English terms. The effectiveness of the system was validated through public research, involving usability tests and questionnaires applied to students from various courses and IT professionals. The tests confirmed the effectiveness of the proposed enhancement, demonstrating an intuitive interface and content that meet the needs of students and professionals, as attested by the quantitative and qualitative research conducted. It is expected that this new version of the system will contribute more effectively to student autonomy and the improvement of the teaching-learning process within the context of educational informatics.

Keywords: Technology. Accessibility. Information. Education.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ABNT - Associação Brasileira de Normas Técnicas

API – *Application Programming Interface* (Interface de Programação de Aplicações)

CSS – *Cascading Style Sheets* (Folhas de Estilo em Cascata)

DER - Diagrama de Entidade-Relacionamento

DOM – *Document Object Model* (Modelo de Objeto de Documento)

FATECE - Faculdade de Tecnologia, Ciências e Educação

HTML – *HyperText Markup Language* (Linguagem de Marcação de Hipertexto)

HTTP – *HyperText Transfer Protocol* (Protocolo de Transferência de Hipertexto)

IFMG - Instituto Federal de Minas Gerais

IT - *Information Technology* (Tecnologia da Informação)

JSON - *JavaScript Object Notation* (Notação de Objeto de JavaScript)

MDN - *Mozilla Developer Network* (Rede de Desenvolvedor Mozilla)

REST – *Representational State Transfer* (Transferência de Estado Representacional)

RF - Requisitos Funcionais

RNF - Requisitos Não Funcionais

SGBD - Sistema de Gerenciamento de Banco de Dados

SP - São Paulo

SQL - *Structured Query Language* (Linguagem de Consulta Estruturada)

URL - *Uniform Resource Locator* (Localizador Uniforme de Recursos)

UX - *User Experience* (Experiência do Usuário)

WCAG - *Web Content Accessibility Guidelines* (Diretrizes de Acessibilidade em Conteúdo na *Web*)

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, o avanço tecnológico tem promovido transformações significativas e aceleradas em diversas áreas do conhecimento, especialmente na área da Informática. A constante evolução das linguagens de programação, das ferramentas de desenvolvimento de *software*, dos dispositivos inteligentes e dos ambientes computacionais, tem gerado uma demanda crescente por profissionais qualificados para atender às exigências do mercado de trabalho. Diferentemente de cenários anteriores, o mercado atual apresenta um nível de exigência ainda mais elevado. Conforme aponta o relatório da Geek Hunter (2026), as empresas tornaram-se mais criteriosas, priorizando profissionais com alta proficiência técnica e visão de negócio. Esse cenário de dependência tecnológica é corroborado pela Brasscom (2025), que prevê um déficit de 530 mil talentos até 2026 no Brasil, evidenciando que a qualificação profissional tornou-se o principal gargalo para a transformação digital no país.

Impulsionado por um mercado de trabalho aquecido, salários atrativos e novas formas de trabalho flexível, o interesse dos brasileiros por cursos voltados ao setor de Tecnologia da Informação (TI) e Computação apresentou uma ampliação significativa, tanto em nível técnico quanto superior (APAT, 2024). Nesse sentido, o Instituto Federal de Minas Gerais – *Campus* São João Evangelista (IFMG-SJE) tem atendido essa demanda com a oferta dos cursos: Técnico em Informática Integrado ao Ensino Médio e Bacharelado em Sistemas de Informação, ambos com propostas pedagógicas voltadas à formação de profissionais capazes de atuar nas diversas áreas da Computação (INSTITUTO FEDERAL DE MINAS GERAIS, 2025). Contudo, observou-se nesses cursos através do desenvolvimento do projeto de pesquisa realizado por Milagres, Pimenta e Sales (2025), a dificuldade dos estudantes na compreensão e entendimento das linguagens de programação e termos técnicos mais centrados na área da Informação e Tecnologia, em virtude da utilização da língua inglesa como idioma nativo.

A predominância do inglês como idioma padrão na documentação de linguagens, comandos e plataformas de desenvolvimento de *software*, define uma barreira para estudantes com pouca familiaridade com o idioma. Essa limitação impacta diretamente na compreensão teórica e na execução prática de atividades de programação. Conforme aponta pesquisa da Faculdade de Tecnologia, Ciência e Educação (FATECE), a falta de domínio da língua inglesa compromete o acesso a materiais técnicos essenciais, como artigos, fóruns de discussão e repositórios de código (PRISCILA, 2023).

Considerando esse obstáculo, os alunos Fábio Milagres, Gustavo Pimenta e Henrique Sales deram início ao desenvolvimento do DicioCode, um sistema *web* de apoio aos novos programadores e ingressantes na área da Tecnologia da Informação. Conforme Milagres, Pimenta e Sales (2025), o sistema foi construído para servir de âncora para desenvolvedores de *software* que estão ingressando nesta área de conhecimento. Apesar de já possuir diversos recursos para auxiliar os usuários, após análises, foi possível notar que o sistema possui várias necessidades de melhora.

Neste contexto, este trabalho propõe o aprimoramento e expansão do DicioCode. A proposta atual inclui a ampliação das linguagens e campos abordados, o aprimoramento da interface já existente e a integração de exemplos práticos dos termos utilizados nas linguagens de programação.

O aprimoramento do sistema demonstrou potencial para fortalecer a base de conhecimento técnico dos usuários, atuando como um facilitador no processo de aprendizagem de termos fundamentais para a atuação no setor tecnológico.

1.1 Objetivos

A definição dos objetivos é uma etapa fundamental que estabelece as metas a serem alcançadas em um trabalho científico (SEVERINO, 2007). A seguir, são apresentados o objetivo geral e os específicos que norteiam esta pesquisa.

1.1.1 *Objetivo geral*

Dando continuidade ao sistema desenvolvido no projeto de pesquisa anterior, realizado por Milagres, Pimenta e Sales (2025), o DicioCode, o objetivo geral deste trabalho é evoluir o sistema, por meio de melhorias na interface, a ampliação das linguagens suportadas e integração de exemplos práticos dos termos utilizados pelas linguagens de programação.

1.1.2 *Objetivos específicos*

Para a realização deste trabalho, os objetivos específicos são:

a) Reestruturar a interface do sistema;
b) Ampliar o conteúdo da ferramenta;
c) Desenvolver e integrar um módulo de exemplos práticos dos termos utilizados nas

linguagens de programação;

d) Validar a eficácia da nova versão da ferramenta.

1.2 Justificativa

Com o crescimento constante da área de Tecnologia da Informação, a aprendizagem de linguagens de programação e a proficiência em demais áreas da Tecnologia da Informação, tornaram-se habilidades essenciais para quem deseja ingressar no mercado atual. No entanto, muitos estudantes iniciantes enfrentam dificuldades ao lidar com comandos, termos técnicos e documentações produzidos em língua inglesa. Essa barreira linguística compromete a compreensão dos conteúdos, o desempenho e o interesse desses alunos nas disciplinas de Informática.

Essa dificuldade é particularmente observada no IFMG-SJE, a qual foi o ponto principal do projeto de pesquisa anterior, que desenvolveu a primeira versão de um sistema de referência para termos focados no campo da Informação e programação construídos em inglês. A validação desse projeto inicial confirmou seu potencial, mas também revelou a oportunidade para aprimoramentos que pudessem tornar a experiência de aprendizado ainda mais completa.

Diante disso, o aprimoramento deste sistema, justifica-se pela necessidade de uma ferramenta de referência para um ambiente de aprendizagem mais rico e integrado. Enquanto a versão anterior focava na barreira linguística, a proposta atual é incorporar uma interface mais intuitiva, novas linguagens e áreas, e integração de exemplos práticos dos termos utilizados pelas linguagens de programação, visando fortalecer a autonomia do estudante. O objetivo é não apenas apoiar tecnicamente, mas contribuir como um instrumento de inclusão de aprendizado ainda mais eficaz para o ramo de Tecnologia da Informação.

Além do impacto direto na aprendizagem, a versão aprimorada do sistema poderá ser utilizada pelos docentes como um recurso complementar mais robusto em sala de aula. Assim, este trabalho dá continuidade a uma linha de pesquisa de relevância acadêmica, social e educacional, contribuindo para a formação de futuros profissionais da área de TI, com mais segurança e independência no processo de aprendizagem.

2 O PROJETO DICIOCODE

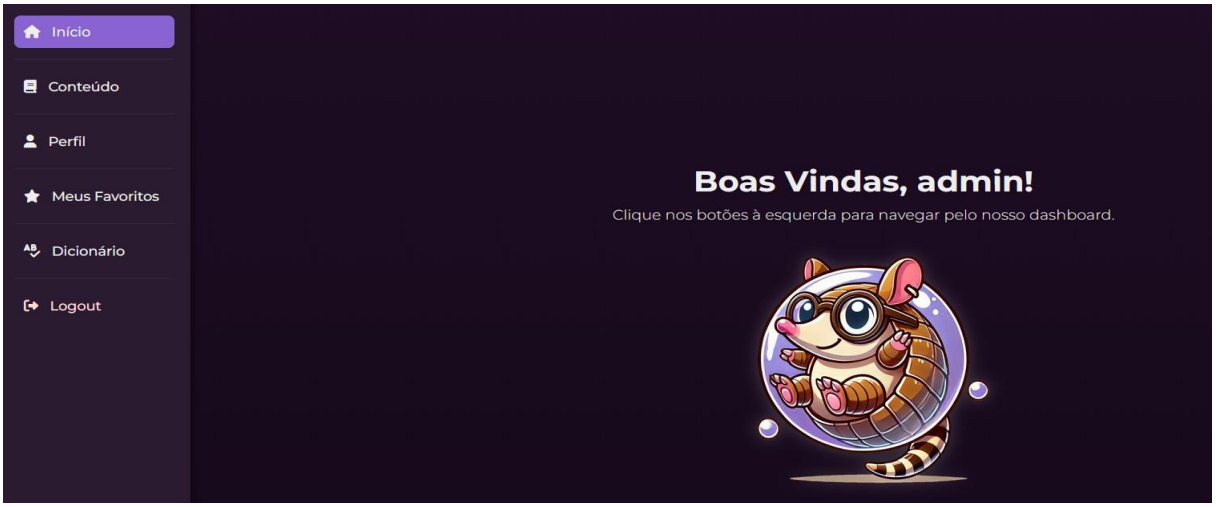
A língua inglesa consolidou-se como o principal meio de comunicação global,

permeando diversos setores profissionais, acadêmicos e de entretenimento. Contudo, apesar de sua ampla difusão, o idioma ainda representa uma barreira significativa para indivíduos que carecem de acesso formal ao aprendizado, o que restringe a absorção de conteúdos didáticos técnicos. Segundo Ortiz (2007), durante o processo de globalização, o inglês tornou-se uma língua que caminha junto à modernidade do mundo, atuando também como idioma base para a sintaxe de linguagens de programação e para a vasta maioria de materiais de apoio. Nesse cenário, o desenvolvimento de ferramentas que auxiliem na construção de uma base de conhecimento torna-se fundamental para promover a inclusão e expandir oportunidades de capacitação.

Nesse contexto, o projeto DicioCode foi estruturado como um repositório terminológico direcionado a estudantes da área de Tecnologia da Informação e campos correlatos. O objetivo consiste em fornecer suporte técnico durante o processo de aprendizagem, independentemente do nível de fluência do usuário no idioma estrangeiro. O sistema disponibiliza uma base de termos técnicos acompanhados de recursos em áudio, permitindo que o discente compreenda a pronúncia correta e a respectiva tradução, estabelecendo, assim, uma correlação direta com a língua portuguesa.

Na interface de usuário padrão, o sistema conta com páginas que contêm os conteúdos presentes para estudo, edição e configuração do perfil, assim como página de favoritos e dicionário, com todos os termos presentes na plataforma, assim como é possível observar na Figura 1.

Figura 1 - Interface do sistema DicioCode base: página inicial e barra lateral.

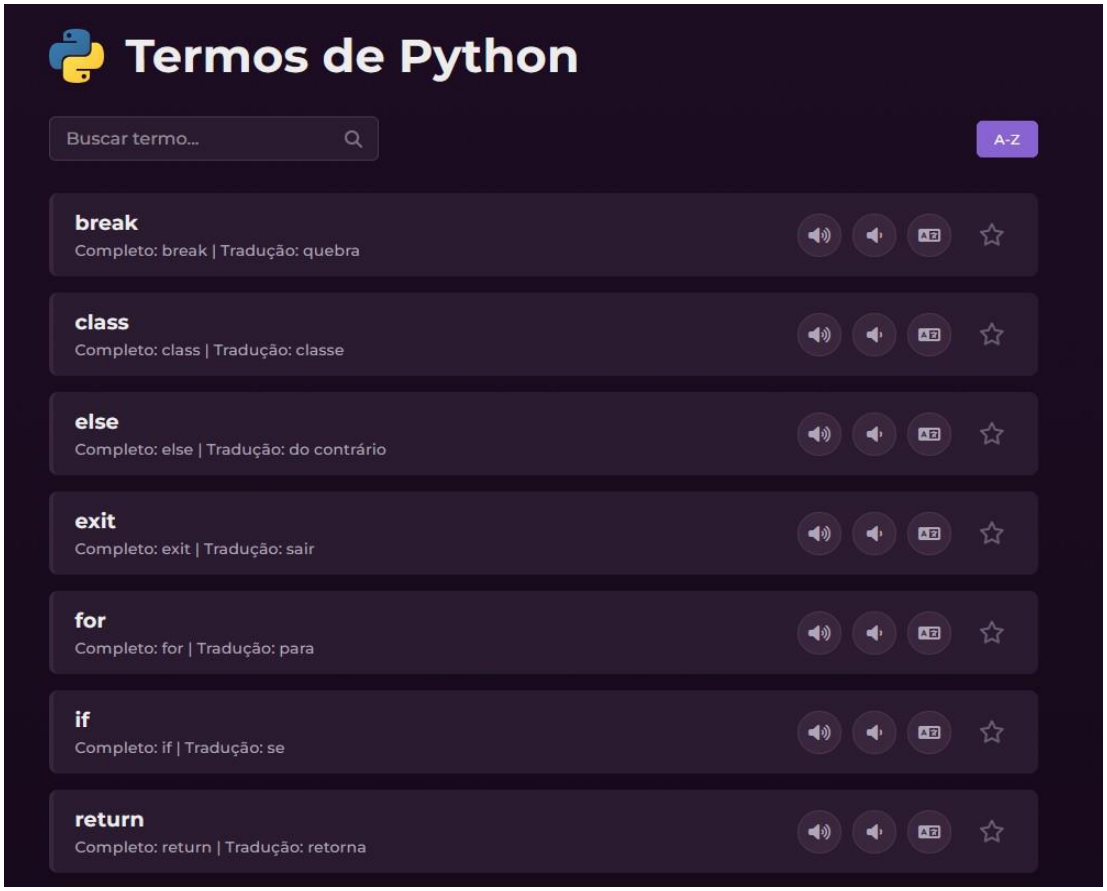


Fonte: Elaborado pelos autores, 2026.

A tela de termos possui a lista com as palavras-chave do conteúdo selecionado,

organizadas em ordem alfabética, e também possui os botões para reprodução dos áudios, nas duas velocidades disponíveis e traduzido, conforme mostra a Figura 2.

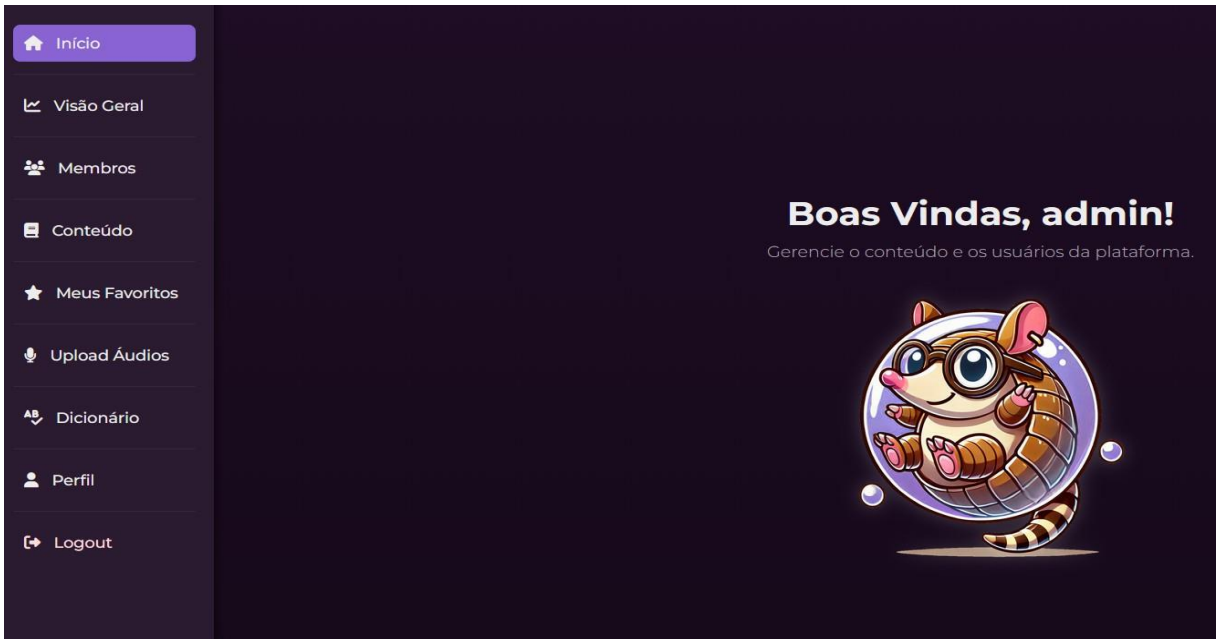
Figura 2 - Interface do sistema DicioCode base: página de conteúdo com lista de termos.



Fonte: Elaborado pelos autores, 2026.

Já para o administrador, além das páginas citadas anteriormente, ele também possui locais para administração de áudios e termos, porém sem a opção dos exemplos práticos, gráficos para controle de frequência de registro, e listas para gerenciar os membros registrados, como é possível observar na Figura 3.

Figura 3 - Interface do sistema DicioCode base: página inicial e barra lateral para administrador.



Fonte: Elaborado pelos autores, 2026.

3 REFERENCIAL TEÓRICO

Neste capítulo, foi apresentado o referencial teórico para este trabalho, que visa oferecer o embasamento necessário para sua realização. O referencial apresenta conceitos básicos sobre as tecnologias utilizadas durante a fase de desenvolvimento do *backend*, *frontend* e banco de dados do sistema, além dos conhecimentos e conceitos teóricos sobre o processo de aprendizagem, utilização e a interação do usuário com o computador, utilizados para sua estruturação.

3.1 Contexto da aprendizagem em Computação

Durante o processo de aprendizagem de um iniciante no mundo do desenvolvimento de *software*, é natural que a maior parte do grupo de discentes deseje focar numa dada linguagem de programação que os favoreça no mercado de trabalho, uma vez que as vagas de emprego no segmento de desenvolvimento de *software* estão em constante crescimento. Segundo pesquisa realizada pela Agência Brasil (2024), na última década, a busca pelas vagas de emprego na área da tecnologia cresceu em aproximadamente 95%, o que motivou ainda mais os novos ingressantes.

No entanto, um obstáculo que interfere no aprendizado de alunos no segmento de desenvolvimento de *software* é a falta de conhecimento da língua inglesa, na qual a maior parte das linguagens de programação é escrita e documentada. Seja na descrição do uso de um comando na documentação da linguagem ou até mesmo na sintaxe de um programa mais complexo e com um corpo mais longo, a interpretação do código é predominantemente feita em inglês.

Adicionalmente, cumpre citar os fóruns de programação existentes, como Reddit, Stack Overflow, W3Schools e outros, que embora existam artigos em português e outros idiomas, a maioria dos tópicos criados ou resolvidos é escrita em inglês, o que também acaba sendo um fator de complicação para estudantes da programação.

A dependência do idioma estrangeiro não apenas dificulta o acesso à informação, mas também impõe uma carga cognitiva adicional ao estudante. A necessidade de compreender o inglês técnico, somada à complexidade da própria linguagem de programação, gera um esforço mental que eleva a carga cognitiva extrínseca. De acordo com a Teoria da Carga Cognitiva, esse esforço sobrecarrega a memória de trabalho do aluno, reduzindo sua capacidade de processar novas informações e construir conhecimento abstrato sobre programação (SWELLER, 1994). Como consequência, essa sobrecarga pode levar à frustração, ao aumento da percepção de dificuldade e, em última análise, contribuir para o aumento das taxas de evasão nos cursos da área.

Para mitigar tais desafios, a literatura em Tecnologia Educacional e Interação Humano-Computador aponta para a eficácia de ferramentas de apoio à aprendizagem (do inglês, *scaffolding* tools). Autores como Wood, Bruner e Ross (1976) defendem que sistemas projetados para oferecer suporte "*just-in-time*" (como glossários interativos, tradutores contextuais e bases de conhecimento de fácil consulta) são cruciais. Tais ferramentas atuam como "andaimes", permitindo que o aluno se concentre na tarefa principal, que seria aprender a programar, ao remover temporariamente o obstáculo secundário - a barreira de idioma.

3.2 Interação humano-computador e experiência do usuário

A qualidade da experiência do usuário em um sistema de aprendizagem é diretamente influenciada pela fluidez de sua interface. Para um estudante, uma interação simples e intuitiva é crucial, pois permite que o foco seja mantido no conteúdo, sem a necessidade de despender tempo considerável apenas para navegar na plataforma em busca de uma consulta. Caso o aprendiz localize dificuldades e cometa erros, o *design* pouco intuitivo da

interface pode ser um empecilho para o estudante (NORMAN, 2006).

Nesse contexto, os princípios de usabilidade são fundamentais. Nesse sentido, o projeto DicioCode pautou-se nas Heurísticas de Jakob Nielsen, descritas em sua obra Usability Engineering, para otimizar a interface para programadores iniciantes. Foram priorizados conceitos como a "correspondência entre o sistema e o mundo real" (Heurística 2), ao traduzir jargões técnicos para uma linguagem acessível, e o "reconhecimento em vez de memorização" (Heurística 6), uma vez que a consulta de termos sob demanda evita a sobrecarga cognitiva do aluno.

Ainda seguindo a linha de pensamento de Nielsen, a abordagem se alinha diretamente ao princípio da prevenção de erros, que o autor considera um dos pilares de um bom *design*. Conforme o autor, "ainda melhor do que boas mensagens de erro é um *design* cuidadoso que previne a ocorrência do problema em primeiro lugar" (Nielsen, 1993, p. 153). Nesse sentido, o DicioCode atua de forma a prevenir que o usuário aplique de forma incorreta um termo ou comando em seu próprio código, evitando a frustração de erros de sintaxe ou de lógica, tornando o ciclo de aprendizado mais eficiente e motivador.

Adicionalmente, para promover o engajamento e reter o interesse do estudante, foram aplicados conceitos de *design* de comunidades *online*. A implementação de perfis de usuário, com a possibilidade de personalização e salvamento do progresso de estudos, funciona como um elemento motivador. Tais funcionalidades buscam transformar a experiência de uso em uma jornada contínua e gratificante, incentivando o retorno e a permanência do usuário na plataforma.

3.3 Arquitetura e tecnologias de sistema *web* modernos

A arquitetura de sistemas *web* mais modernos é principalmente baseada no modelo cliente-servidor, um padrão fundamentalmente distribuído que particiona responsabilidades de um sistema em duas partes, sendo elas o servidor, que gerencia dados e lógica, e o cliente, que consome esses dados e foca na camada de apresentação e na interação com o usuário (TANEMBAUM; WETHERALL, 2011). A principal vantagem desse modelo é a possibilidade de desacoplar módulos, o que permite a interface do usuário e o armazenamento de dados evoluirem de forma independente. Essa separação é, inclusive, a primeira restrição definidora do REST, criada por Roy Thomas Fielding (2000), que rege a comunicação na *web*, como será mencionado posteriormente.

3.3.1 *Arquitetura e desenvolvimento de sistemas web*

No modelo cliente-servidor, o sistema *web* destaca-se por sua natureza centrada em rede, sendo projetado para lidar com um volume expressivo de usuários de maneira concorrente. Por essa razão, tais aplicações são desenvolvidas de forma robusta, priorizando a disponibilidade e a escalabilidade dos serviços (PRESSMAN; MAXIM, 2016). No escopo deste trabalho, o sistema operou estritamente como um cliente *web* dentro dessa arquitetura, consumindo serviços e dados provenientes de um *backend* remoto.

O ponto de partida do desenvolvimento consistiu na implementação do *backend*. Nesta fase inicial, estabeleceram-se os protocolos de comunicação entre o servidor e o banco de dados, responsáveis por processar as requisições dos administradores e gerenciar as rotas dos arquivos de áudio armazenados. Durante esse processo, identificaram-se desafios relativos à definição de formatos para o armazenamento e recebimento de dados, os quais foram solucionados mediante a padronização das interfaces de comunicação.

A segunda fase concentrou-se no desenvolvimento do *frontend*, correspondente à camada de apresentação do sistema. Nesta etapa, dedicou-se atenção especial à UX, visando facilitar a interação do estudante com o sistema. O processo de análise e implementação da interface ocorreu de forma fluida, resultando em um ambiente funcional e aderente aos requisitos de usabilidade estabelecidos.

3.3.2 *HTML - Estruturação semântica*

O HTML foi empregado como a tecnologia fundamental para a estruturação semântica de todo o conteúdo do sistema. Sua utilização correta garante não apenas a organização lógica das informações, como títulos, parágrafos e listas, mas também a acessibilidade da plataforma e a correta interpretação do conteúdo por parte dos navegadores e de tecnologias assistivas, como leitores de tela (MDN WEB DOCS, 2025b).

3.3.3 *CSS - Apresentação e estilização do sistema*

Enquanto o HTML define a estrutura, o CSS foi responsável pela composição da identidade visual e pela estilização da plataforma. Por meio de regras de estilo, foi possível criar uma experiência de usuário bem estruturada, fluida e agradável, definindo cores, fontes, espaçamentos e o *layout* responsivo do sistema, que se adapta a diferentes tamanhos de tela

(MDN WEB DOCS, 2025a).

3.3.4 *JavaScript - Interatividade para o usuário*

Para adicionar fluidez, dinamismo e interatividade à interface, utilizou-se o JavaScript puro, ou *vanilla*. Essa tecnologia, executada diretamente no navegador, é responsável por manipular o DOM para responder às ações do usuário em tempo real, como cliques em botões, validação de formulários e, crucialmente, orquestrar a comunicação assíncrona com o servidor para buscar ou enviar dados sem a necessidade de recarregar a página (MDN WEB DOCS, 2025c).

3.3.5 *Node.js e TypeScript - Rigidez no servidor*

Toda a lógica central do sistema reside no servidor, que foi desenvolvido sobre o ambiente de execução Node.js, utilizado para construir aplicações escaláveis (OPENJS FOUNDATION, 2025). Para a implementação desta camada crítica, optou-se pelo uso do TypeScript em detrimento do JavaScript puro. Sendo um conjunto aprimorado de funções do JavaScript que adiciona tipagem estática, o TypeScript permite a detecção de erros em tempo de desenvolvimento, e não apenas durante a execução (MICROSOFT, 2025). Essa característica confere uma camada extra de robustez e segurança ao código do *backend*, o que é essencial para garantir a integridade dos dados e a previsibilidade do comportamento do sistema em operações como cadastro, autenticação e gerenciamento de conteúdo.

3.3.6 *PostgreSQL - Gerenciamento e maleabilidade de dados*

Para o armazenamento e a persistência dos dados da aplicação, foi escolhido o PostgreSQL, um SGBD relacional de código aberto, amplamente reconhecido por sua estabilidade, segurança e performance. No contexto do sistema apresentado no trabalho, o PostgreSQL é responsável por armazenar informações cruciais, como os dados das contas dos usuários, entre elas as credenciais de acesso e informações do perfil do usuário, e as referências para os arquivos de áudio utilizados na funcionalidade de pronúncia dos termos. A natureza relacional do PostgreSQL garante a integridade e a consistência na associação entre os diferentes dados do sistema (The PostgreSQL Global Development Group, 2026).

3.3.7 *API REST - Comunicação entre cliente e servidor*

A comunicação entre a camada de cliente e a de servidor é orquestrada por meio de uma API que segue os princípios da arquitetura REST. Essa API define um conjunto de rotas e utiliza os verbos do protocolo HTTP como uma interface de comunicação padronizada (FIELDING, 2000). Dessa forma, o *frontend* pode solicitar, enviar, atualizar ou deletar dados no servidor de maneira clara e previsível, tratando o *backend* como uma fonte de dados e serviços com a qual ele pode interagir sem precisar conhecer os detalhes de sua implementação interna.

Na prática, essa arquitetura se materializa em operações concretas do DicioCode: uma busca por um termo no *frontend* dispara uma requisição GET à rota /api/termos/:nome; o envio de uma nova definição por um usuário resulta em uma requisição POST, responsável por criar um novo recurso no sistema; já a atualização de uma informação existente é realizada por meio de uma requisição PUT que atualiza o recurso específico. Essa arquitetura não apenas soluciona os requisitos atuais da aplicação *web*, mas também garante sua escalabilidade e prepara o sistema para futuras expansões, como o desenvolvimento de novos clientes.

3.4 Análise do projeto de base para modificações

Como marco inicial para o desenvolvimento do trabalho, conduziu-se uma análise da arquitetura do sistema, segmentada em seus três componentes principais. Para uma analisar o projeto de maneira eficiente e estruturada, a primeira parte foi analisar o *backend* do trabalho, responsável pela execução da lógica de negócio e pelo processamento central das requisições. Ele contém regras de funcionamento, que disponibiliza um conjunto amplo de serviços, o que permite que a camada de apresentação se interligue com eficiência com a parte lógica, de forma independente, mas que se comuniquem de forma padronizada.

Juntamente à análise do *backend*, também foi analisada a estrutura do banco de dados do sistema, que funciona como a memória da aplicação. Com o uso do PostgreSQL, é possível guardar informações de forma segura e organizada, como dados de usuários e rotas de mídia necessárias para o funcionamento do sistema.

Por fim, a análise focou na camada de apresentação, (*frontend*). Essa camada é construída com três tecnologias que trabalham em conjunto: o HTML, que atua como o esqueleto da página, organizando toda a estrutura e o conteúdo; o CSS, que funciona como a camada de *design*, definindo as cores, fontes e o *layout* para criar uma aparência agradável e fácil de usar; e o JavaScript, que dá vida à página, permitindo a interatividade para responder

aos cliques e ações do usuário. Uma análise cuidadosa do *frontend*, portanto, busca garantir que a experiência de quem usa o sistema seja clara, intuitiva e eficiente.

3.5 Trabalhos correlatos

3.5.1 *Plataformas de ensino de programação*

Nesta seção, são abordados trabalhos que se correlacionam a esta pesquisa, incluindo sistemas comerciais e bases acadêmicas que auxiliam estudantes iniciantes na área da Computação. O objetivo é identificar abordagens, vantagens e desvantagens, posicionando o DicioCode no cenário atual de ferramentas de apoio à aprendizagem. Esta análise foi dividida em dois segmentos principais:

a) Plataformas de ensino de programação;
b) Dicionários técnicos e glossários.

3.5.2 *Plataformas de ensino de programação*

Uma das principais categorias de ferramentas de apoio são as plataformas de ensino interativo. Destaca-se a plataforma W3Schools (W3SCHOOLS, 2026), amplamente utilizada por desenvolvedores com foco em tecnologias *web*. Seu ponto forte reside na vasta documentação sobre HTML, CSS e JavaScript, além de oferecer um editor de código interativo para testes em tempo real.

Complementarmente, os guias da *MDN Web Docs* (MDN WEB DOCS, 2025b; MDN WEB DOCS, 2025a; MDN WEB DOCS, 2025c) são referências globais para a padronização dessas tecnologias. Contudo, tanto o W3Schools quanto a MDN adotam uma abordagem de enciclopédia técnica, sendo majoritariamente estruturados em língua inglesa, o que pode representar uma barreira para iniciantes. Em contrapartida, o aprimoramento proposto para o DicioCode não visa a documentação exaustiva, mas atua como uma ferramenta de "primeiros socorros" para tradução e contextualização, servindo como uma camada de apoio anterior à consulta aprofundada.

3.5.3 *Dicionários técnicos e glossários*

Outra categoria relevante compreende dicionários e glossários, como o Techopedia

(TECHOPEDIA, 2026). Este sistema foca em definições claras e concisas de diversos termos tecnológicos, tratando de forma objetiva um vasto vocabulário. Entretanto, sistemas desse tipo frequentemente carecem de contextualização prática. Embora apresentem definições teóricas corretas, raramente demonstram como um comando é implementado em projetos reais. A proposta deste trabalho busca preencher essa lacuna ao integrar tradução, definição e um módulo de exemplos práticos, conectando o conceito à sua aplicação.

3.5.4 *Síntese da análise*

A análise dos trabalhos correlatos permitiu constatar que, embora existam plataformas robustas para o ensino de programação, há uma carência de ferramentas focadas em apoiar estudantes que enfrentam a barreira inicial do vocabulário em inglês. Conforme discutido por Wood (1976) em teorias de suporte ao aprendizado (*scaffolding*), ferramentas de apoio são essenciais para que o aprendiz supere desafios iniciais e ganhe autonomia.

No contexto brasileiro, a relevância de tais ferramentas é acentuada pela expansão do interesse por cursos de tecnologia (Agência Brasil, 2024). O projeto insere-se como uma solução de referência rápida que visa complementar o processo de aprendizagem ao unir tradução, explicação e exemplos práticos, justificando assim a necessidade de seu aprimoramento.

4 METODOLOGIA

A metodologia científica, compreendida como o estudo dos caminhos e instrumentos para a construção do conhecimento, trata dos procedimentos lógicos e das técnicas que orientam o processo de investigação. Segundo Severino (2007, p. 25), a metodologia “trata, pois, das formas de se fazer ciência; das normas do trabalho científico”. Nesta seção, foram detalhados os procedimentos metodológicos adotados para o desenvolvimento do presente trabalho.

4.1 Classificação da pesquisa

Para fins de estudo, a pesquisa classificou-se da seguinte forma:

a) Quanto à natureza: tratou-se de uma pesquisa aplicada por utilizar o conhecimento científico para desenvolver uma solução direcionada a um problema específico

(GIL, 2008), no caso deste trabalho, a barreira linguística enfrentada pelos ingressantes da área da Tecnologia da Informação.

b) Quanto aos seus objetivos, a presente pesquisa classifica-se como exploratória e descritiva. Caracteriza-se como exploratória por buscar maior familiaridade com o problema da barreira linguística no ensino de programação, culminando na idealização da evolução do sistema DicioCode. Paralelamente, assume um caráter descritivo ao detalhar as propriedades da interface desenvolvida e, sobretudo, ao descrever o perfil socioeducacional e as percepções de usabilidade da amostra de discentes que participou da fase de validação. (GIL, 2008).

c) Quanto aos procedimentos técnicos: para a construção do referencial teórico, utilizou-se a metodologia de pesquisa bibliográfica, desenvolvida a partir de material já elaborado, como livros e artigos científicos (SEVERINO, 2007). Já para o procedimento técnico e para o desenvolvimento do artefato tecnológico proposto pelo trabalho, adotou-se a metodologia de prototipagem de *software* (PRESSMAN; MAXIM, 2016).

4.1.2 *Fases de desenvolvimento do trabalho*

O trabalho foi organizado em um fluxo de quatro fases sequenciais, que cobriram desde a concepção teórica até a implementação final do protótipo.

4.1.3 *Fundamentação e levantamento de requisitos*

A etapa inicial consistiu em uma revisão aprofundada da literatura, cujos resultados foram consolidados no capítulo de Referencial Teórico. Com base nesse estudo, foram levantados os requisitos funcionais e não funcionais para o aprimoramento do sistema DicioCode. Os requisitos funcionais definiram as ações que o sistema deveria executar, como a ampliação das linguagens suportadas, campos abordados e a implementação de um módulo de exemplos em forma textual. Os requisitos não funcionais, por sua vez, definiram as características de qualidade do sistema, como a necessidade de uma interface intuitiva, responsiva e com bom desempenho, baseando-se nas Heurísticas de Usabilidade de Nielsen (1993).

4.1.4 *Pesquisa de mercado*

Para a segunda etapa da fase de fundamentação do sistema, realizou-se uma pesquisa no mercado empregatício para filtrar linguagens e interesses em ascensão para adicionar novos conteúdos à plataforma. Segundo um relatório da JetBrains (2025), a linguagem Go dobrou o número de usuários profissionais nos últimos 5 anos, tornando-se uma peça fundamental para a infraestrutura de nuvem e microsserviços. Seguindo o mesmo plano, a inclusão da linguagem Lua, de origem brasileira, é justificada por ser presente em *scripts* para motores de jogos e automação industrial. A linguagem Lua se mantém influente por ter alta portabilidade e por conta de sua leveza (BAIRESDEV, 2026).

Adicionalmente, a seção dedicada para *hardware* foi incorporada para suprir a lacuna na formação técnica inicial. Conforme apontado por pesquisa realizada pela (BRASSCOM, 2025), o mercado brasileiro encontra um déficit de profissionais que compreendem a integração entre *software* e infraestrutura física. Assim, o DicioCode passou a oferecer suporte para entendimento de funcionamento dos componentes de um computador.

4.1.5 *Projeto da arquitetura e da interface*

Para a fase de aprimoramento do sistema, os requisitos levantados foram traduzidos em um plano técnico e visual. Consolidou-se a arquitetura de *software* no modelo cliente-servidor, onde o contrato de comunicação entre as faces do sistema foi estabelecido com o uso da API REST. Seguindo o mesmo prisma, realizou-se o projeto da interface e da experiência do usuário, sendo elaborada a estrutura visual e o fluxo de navegação para atender aos requisitos de usabilidade e promover uma interação clara e eficiente, conforme os princípios de Interação Humano-Computador.

4.1.6 *Implementação e verificação do protótipo*

A etapa final consistiu na codificação e na verificação do sistema *web* completo, após a revisão de todos os pontos que necessitavam de aprimoramentos. O projeto da fase anterior foi implementado utilizando as tecnologias selecionadas: o *frontend* foi construído com HTML, CSS e JavaScript; o *backend* foi desenvolvido em TypeScript sobre o ambiente Node.js; e para o armazenamento de dados de usuário e rotas de mídia, empregou-se o uso do banco de dados PostgreSQL.

A verificação do sistema foi realizada por meio de testes funcionais internos, nos quais foram checados o cumprimento dos requisitos, a estabilidade das funcionalidades e a ausência de erros, garantindo que o protótipo final estivesse operacional.

Posteriormente, para validar a eficácia da ferramenta no cenário real, realizou-se uma etapa de testes públicos com estudantes do IFMG-SJE e profissionais da área. Para a coleta de percepções, utilizou-se um questionário estruturado (Apêndice A), cujas perguntas foram elaboradas com base nos princípios de usabilidade das Heurísticas de Nielsen (1994). Essa escolha metodológica visou garantir que a avaliação não fosse meramente opinativa, mas fundamentada em critérios de interface como a visibilidade do *status* do sistema, prevenção de erros e *design* minimalista.

Devido ao caráter exploratório desta pesquisa e ao cronograma acadêmico da graduação em Sistemas de Informação, a amostra contou com 15 participantes, o que permitiu identificar tendências de comportamento e oportunidades de melhoria técnica no *software*. Este processo uniu a validação técnica à visão do usuário final, culminando na entrega da versão aprimorada do sistema, representando o principal resultado prático deste trabalho.

5 RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados os aprimoramentos realizados no sistema DicioCode, que teve como objetivo auxiliar estudantes de áreas da tecnologia com a língua inglesa.

5.1 *Software* aprimorado

Para evidenciar a evolução do sistema, os requisitos atendidos foram segmentados entre funcionais, que descrevem as funcionalidades diretas e não funcionais, que tratam das qualidades técnicas e de uso. A Tabela 1 apresenta as funcionalidades implementadas na nova versão.

Tabela 1 - Requisitos Funcionais implementados no DicioCode

ID	**Descrição do Requisito**
RF01	Reestruturação da interface do aplicativo.
RF02	Adição de novos conteúdos (Go, Lua e *Hardware*).
RF03	Seção com exemplos práticos para os termos.
RF04	Reprodução de áudio com duas velocidades.
RF05	Botão de navegação para retorno à seção anterior.
RF06	Botão de navegação para voltar ao início da página
RF07	Gerenciamento de membros e conteúdos (Admin).

Fonte: Elaborado pelos autores, 2026.

A Tabela 2 evidencia o cumprimento dos requisitos que garantem a qualidade técnica e de uso da plataforma.

Tabela 2 - Demonstração de cumprimento dos Requisitos Não Funcionais

ID	**Requisito**	**Evidência de Cumprimento**
RNF01	Usabilidade	Aplicação das Heurísticas de Nielsen, validada pela percepção dos usuários.
RNF02	Desempenho	Uso de processamento assíncrono com JavaScript e Node.js para fluidez.
RNF03	Portabilidade	*Layout* responsivo construído com CSS para adaptação a diversos dispositivos.
RNF04	Confiabilidade	Persistência de dados com PostgreSQL, garantindo integridade de áudios e perfis.

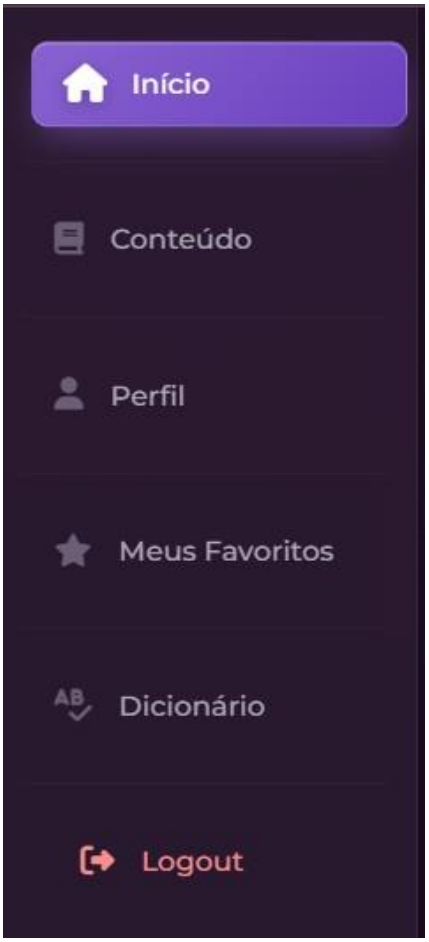
Fonte: Elaborado pelos autores, 2026.

Após a coleta dos requisitos, foram realizadas as mudanças no sistema, juntamente às adições planejadas. O processo de prototipagem de cada alteração foi fundamental para facilitar a visualização e a implementação de cada item do sistema.

5.2 Reestruturação da interface do aplicativo

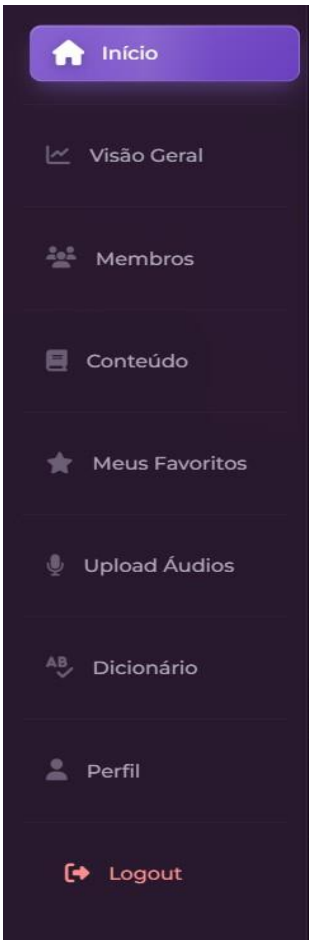
Para tornar a navegação pelo sistema mais responsiva e agradável para o usuário, a barra lateral foi reformulada para ser exibida com um visual mais limpo. A nova interface seguiu as Heurísticas de Nielsen, respeitando a visibilidade do *status* do sistema para que o estudante identifique sua localização atual. É possível observar a barra lateral configurada para o administrador e para o estudante nas Figuras 4 e 5.

Figura 4 - Interface do sistema DicioCode: menu lateral de navegação para o estudante.



Fonte: Elaborado pelos autores, 2026.

Figura 5 - Interface do sistema DicioCode: menu lateral de navegação para o administrador.



Fonte: Elaborado pelos autores, 2026.

Na tela de termos, o container de informações foi ampliado para comportar mais dados de forma clara. Anteriormente, cada segmento continha apenas a tradução e os botões para reprodução dos áudios, conforme ilustrado na Figura 6.

Figura 6 - Interface do sistema DicioCode: tela de termos antes da mudança.

Fonte: Elaborado pelos autores, 2026.

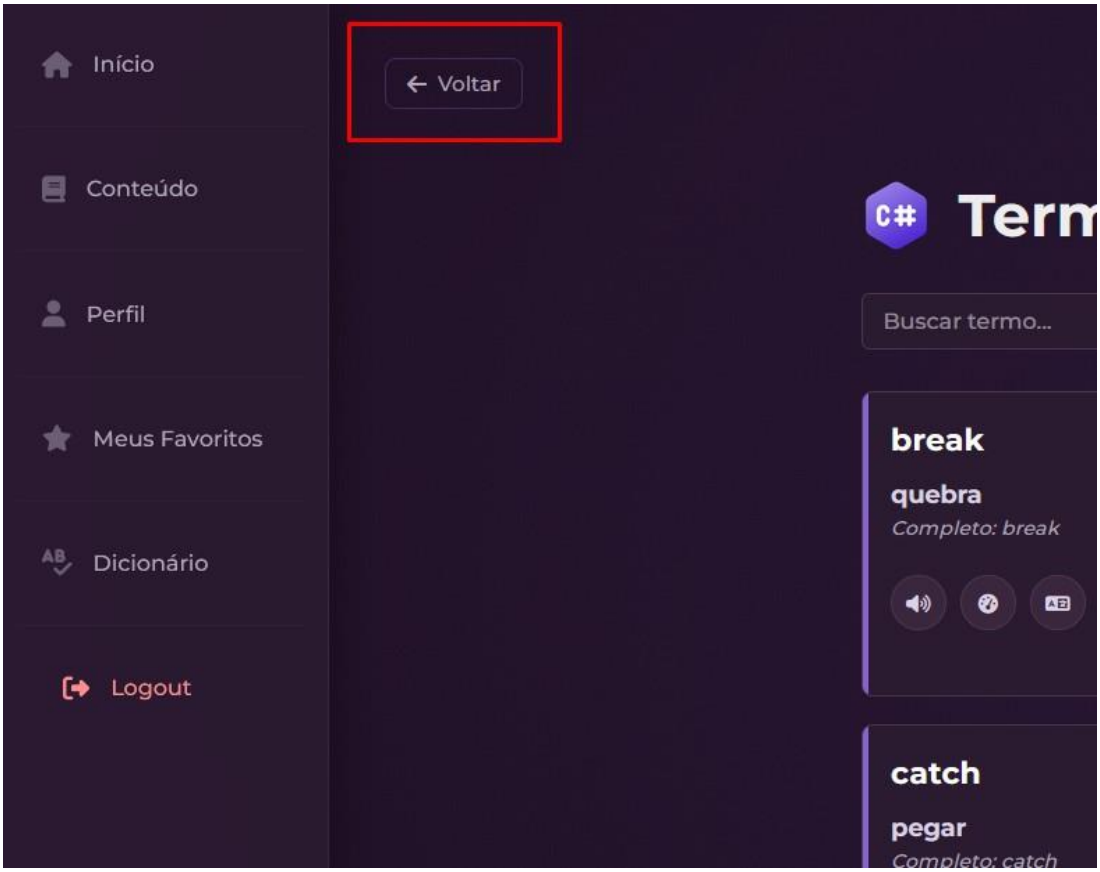
Após as mudanças, cada item passou a ocupar um espaço maior na tela, e o botão para reproduzir o áudio em velocidade reduzida foi alterado para um ícone de velocímetro, tornando a ação mais intuitiva. Além disso, foi implementado o botão “Ver Exemplo”, que permite ao usuário expandir a área de conteúdo. É possível observar a mudança na Figura 7.

Figura 7 - Interface do sistema DicioCode: tela de termos após a mudança.

Fonte: Elaborado pelos autores, 2026.

Para fechar a parte de reestruturação de interface, foi adicionado também um botão “Voltar” no canto superior esquerdo da página, para que o usuário possa voltar ao bloco que estava anteriormente, como mostra a Figura 8.

Figura 8 - Interface do sistema DicioCode: botão “Voltar” no canto superior esquerdo.



Fonte: Elaborado pelos autores, 2026.

5.3 *Adição* de novos conteúdos e base dados

Antes da atualização, o sistema contava com uma quantidade limitada de termos, que abrangiam as linguagens mais utilizadas durante o curso, como demonstrado na Figura 9.

Figura 9 - Interface do sistema DicioCode: conteúdo do sistema antes da mudança.

Fonte: Elaborado pelos autores, 2026.

Para a expansão do conteúdo, foi realizada uma pesquisa de mercado para identificar as linguagens e necessidades mais latentes. Como resultado, foram adicionadas as linguagens Go e Lua, além de uma seção específica sobre *hardware*, abordando componentes físicos do computador, assim como mostrado na Figura 10.

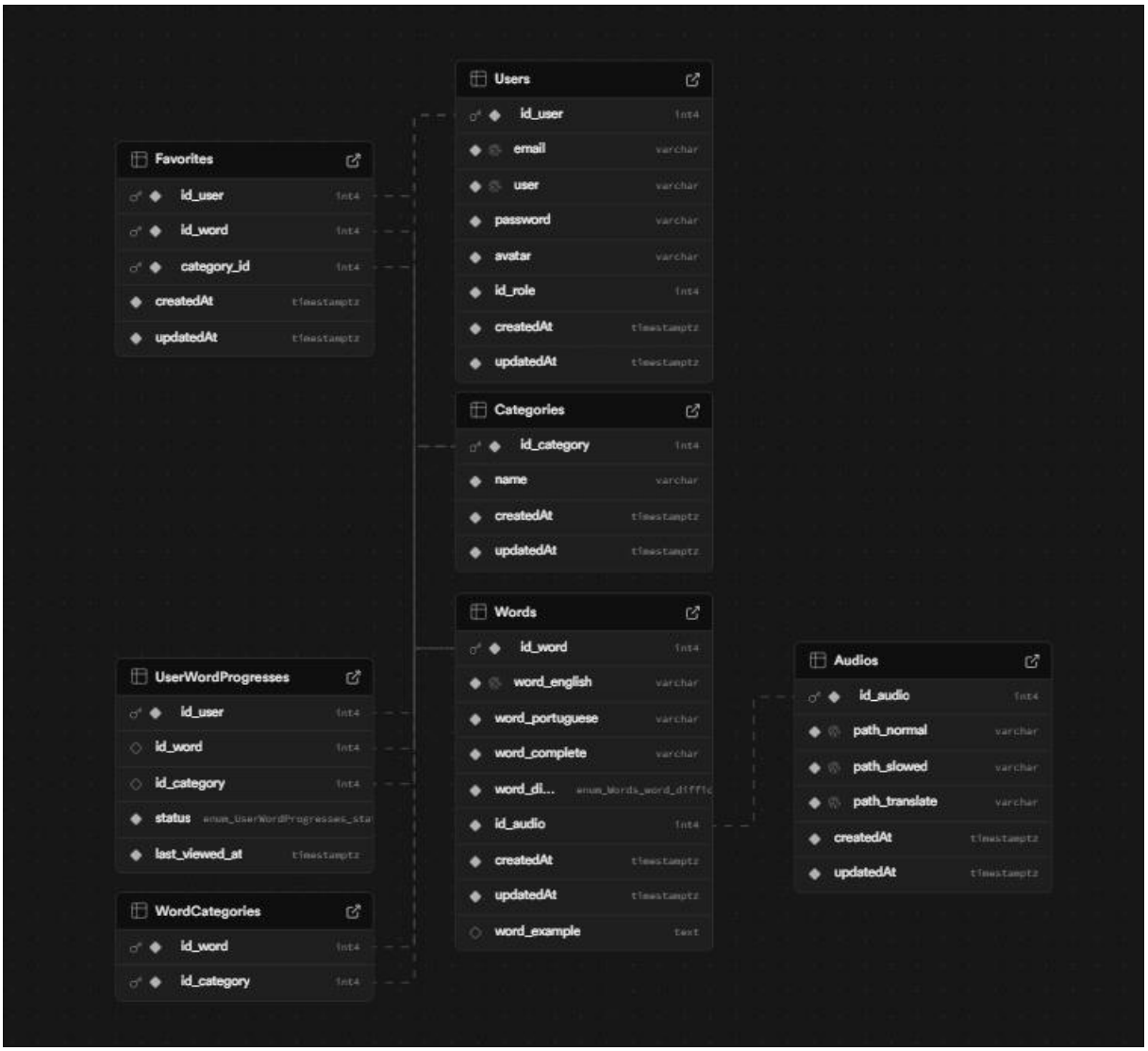
Figura 10 - Interface do sistema DicioCode: novos conteúdos inseridos.

Fonte: Elaborado pelos autores, 2026.

Para processar as informações enviadas pelos formulários e assegurar a fluidez de navegação, os dados são transmitidos em formato `JSON`, permitindo um tráfego de informações

leve e eficiente entre as camadas do *software*. Esses dados são estruturados nas tabelas relacionais da base de dados, conforme ilustrado no Diagrama de Entidade-Relacionamento (DER) na Figura 11. A carga de conteúdo ocorre de forma assíncrona no momento em que a página é acessada, o que minimiza a latência e evita recarregamentos totais da interface, otimizando o desempenho percebido pelo usuário.

Figura 11 - DER do sistema DicioCode.



Fonte: Elaborado pelos autores, 2026.

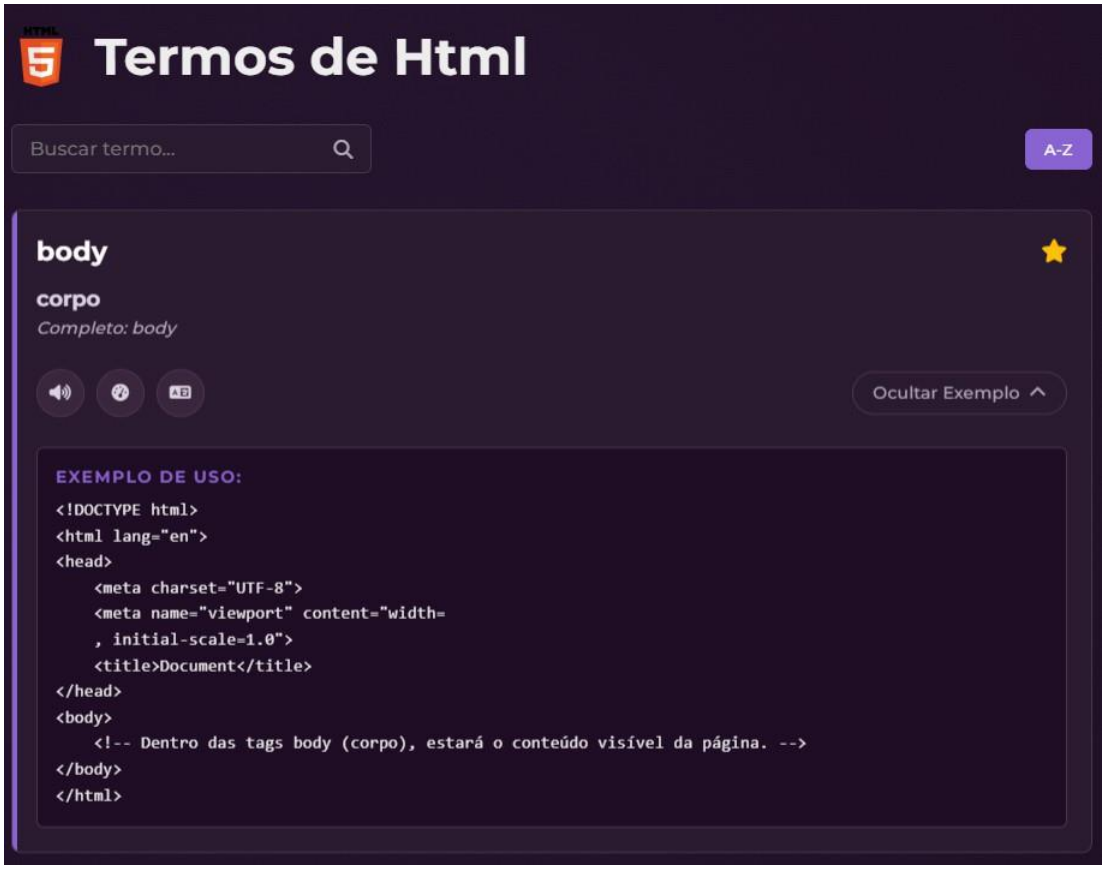
Na tabela Words (Palavras), adicionou-se o campo de texto word_example, onde armazenam-se os exemplos práticos descritos na seção seguinte. Nota-se, também, que a tabela Words utiliza a chave estrangeira id_audio para estabelecer o relacionamento com a tabela Audios (Áudios), permitindo que cada termo seja vinculado aos seus respectivos arquivos de som através dos atributos path_normal, path_slowed e path_translate.

Diferente de sistemas que dependem de conteúdo definido de forma estática, o DicioCode implementa uma arquitetura dinâmica na qual toda a base terminológica e de áudios é gerenciada via interface administrativa. Essa estrutura permite a inclusão de novos conteúdos diretamente pelo módulo administrativo ou via manipulação do banco de dados PostgreSQL, assegurando a escalabilidade do projeto e dispensando qualquer necessidade de alteração no código-fonte para a atualização do sistema.

5.4 *Adição* de uma seção com exemplos práticos para os termos

Como nova funcionalidade, foi incorporado um trecho de código com comentários, acessível por meio do botão “Ver Exemplo”. Ao expandir o campo, o termo passa a ser melhor contextualizado, conforme observado na Figura 12.

Figura 12 - Interface do sistema DicioCode: termo com explicação expandida.



Fonte: Elaborado pelos autores, 2026.

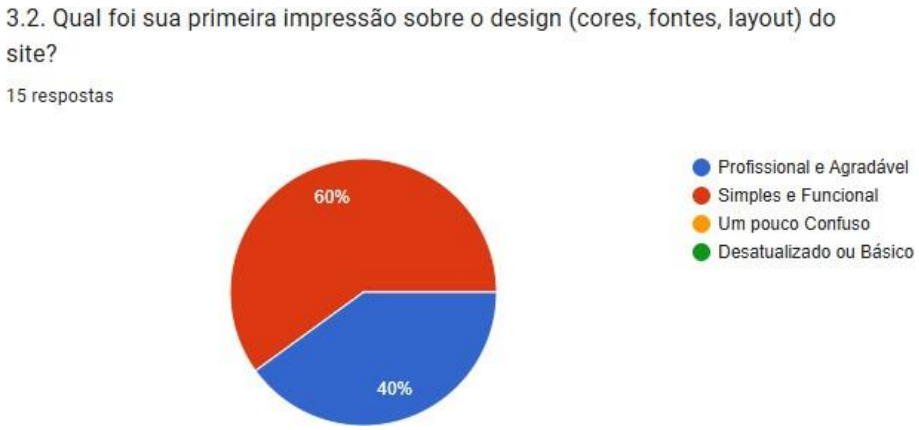
Com a exemplificação do termo em código, a consulta tornou-se mais eficaz para o estudante, permitindo visualizar a aplicação prática de cada palavra.

5.5 *Validação* da eficácia do sistema

Para validar a eficácia do sistema DicioCode, realizou-se uma pesquisa quantitativa e qualitativa via formulário estruturado. Embora a amostra de quinze respostas seja considerada reduzida para generalizações estatísticas amplas, ela cumpre o propósito de uma pesquisa exploratória e qualitativa de usabilidade, permitindo identificar padrões de comportamento e gargalos técnicos em um cenário real de uso acadêmico.

A análise dos dados demonstrou que o *design* do sistema obteve uma recepção positiva, conforme indicado no gráfico da Figura 13.

Figura 13 - Pesquisa de Satisfação: avaliação do *design* geral da interface.

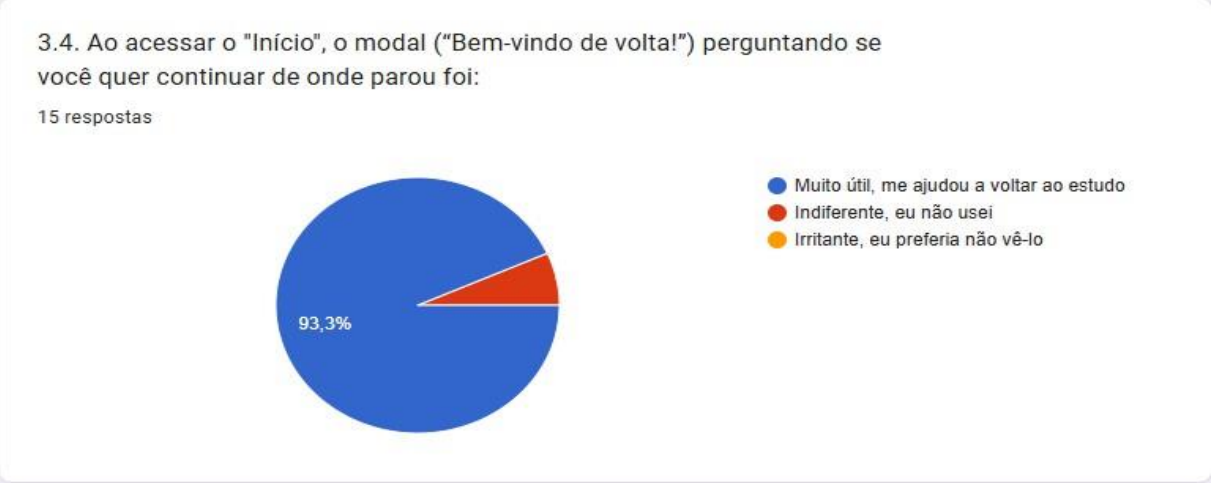


Fonte: Elaborado pelos autores, 2026.

A análise da Figura 13 revela uma recepção positiva da interface. O fato de 60% dos usuários classificarem o *design* como "Simples e Funcional" valida a aplicação da heurística de Estética e *Design* Minimalista, indicando que a remoção de elementos visuais irrelevantes no *frontend* permitiu que os estudantes mantivessem o foco no conteúdo técnico.

Quanto à funcionalidade “Bem-vindo de volta!” (Figura 15), a aprovação majoritária demonstra a eficácia da heurística de Reconhecimento em vez de recordação. Ao utilizar o banco de dados PostgreSQL para persistir o último acesso do usuário, o sistema reduz o esforço cognitivo necessário para retomar os estudos, otimizando o fluxo de aprendizagem.

Figura 14 - Pesquisa de Satisfação: avaliação do recurso de boas vindas.



Fonte: Elaborado pelos autores, 2026.

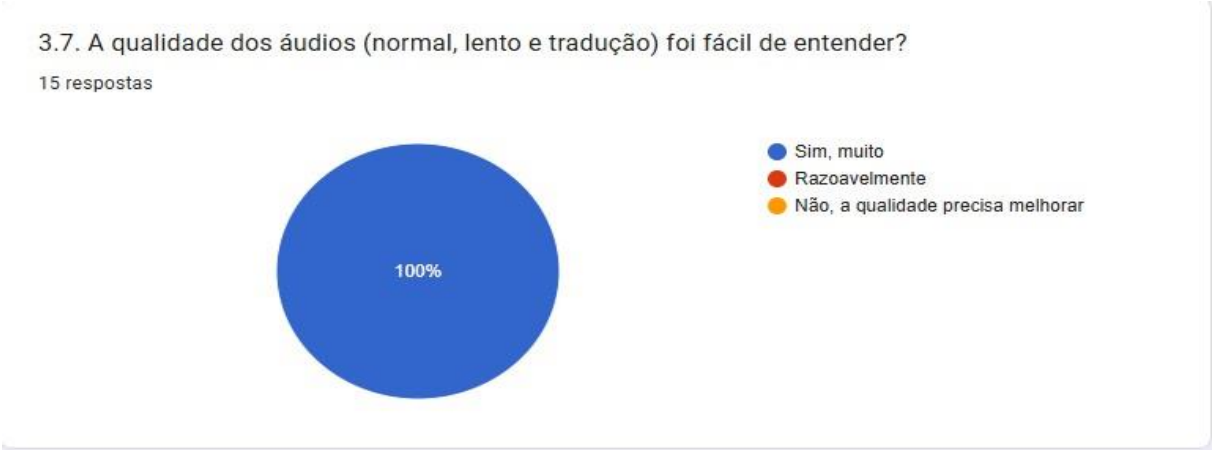
Um ponto de atenção identificado na Figura 15 refere-se à divisão de opiniões sobre a clareza dos ícones de áudio. Essa divergência foi interpretada como uma falha na heurística de Compatibilidade entre o sistema e o mundo real, o que gerou um ciclo de melhoria imediata no projeto, resultando na substituição de ícones genéricos por representações mais intuitivas de velocidade (ex: tartaruga para o áudio lento).

Figura 15 - Pesquisa de Satisfação: avaliação dos botões.

Fonte: Elaborado pelos autores, 2026.

A qualidade dos áudios presentes na plataforma também recebeu avaliações positivas, indicando que a ferramenta cumpre seu papel no auxílio à pronúncia dos termos técnicos, e também auxilia no entendimento da tradução das palavras-chave, como ilustrado na Figura 16.

Figura 16 - Pesquisa de Satisfação: avaliação da qualidade dos áudios.



Fonte: Elaborado pelos autores, 2026.

Após as revisões e testes práticos realizados com os usuários, foi possível constatar que o sistema possui melhorias visuais e estruturais, como visões claras dos exemplos na tela de cada palavra, assim como no dicionário, maior responsividade e fluidez do sistema em termos de navegabilidade, exemplos práticos bem descritos para cada um dos itens nas unidades, e também a adição de novos conteúdos ao sistema.

6 CONSIDERAÇÕES FINAIS

Durante a etapa de desenvolvimento deste trabalho, foram unidas informações conhecidas por ambos autores, e também conhecimentos adquiridos ao longo do tempo de projeto, o que proporcionou desafios técnicos e acadêmicos durante sua execução.

Os resultados obtidos nos testes e a avaliação da interface final demonstram que o aprimoramento do sistema foi realizado com sucesso, garantindo sua plena operabilidade. Dessa forma, os objetivos deste trabalho são considerados atingidos, uma vez que o *software* atende com precisão aos requisitos exigidos para a evolução da plataforma.

Uma sugestão de melhoria para o sistema desenvolvido e reestruturado, seria a possibilidade de adicionar vídeos para a descrição ainda mais detalhada de cada um dos termos apresentados, e também imagens para que fiquem melhor representados segmentos que não falem apenas de código, deixando assim o sistema ainda mais claro para o usuário.

Por fim, foi possível observar que os autores puderam aumentar seu conhecimento sobre a língua estrangeira e também aumentar a base de informação com foco em desenvolvimento para sistemas *web* por meio do desenvolvimento deste trabalho, aplicando conhecimentos e informações do mundo real e obtidos durante o tempo no curso. Para terminar, espera-se que este trabalho possa contribuir para o público a qual é destinado, com a finalidade de contribuir ainda mais para o estudo dos discentes de áreas da Tecnologia da Informação.

6.1 *Trabalhos* futuros

Embora o DicioCode tenha cumprido seus objetivos iniciais ao validar a funcionalidade técnica e a sua usabilidade, o projeto apresenta oportunidades de expansão para consolidar a ferramenta como um recurso educacional ainda mais robusto. Uma das sugestões de evolução consiste na transição de uma base de dados predominantemente textual e sonora para um modelo de conteúdo multimodal. Nesse contexto, uma possibilidade seria a integração com a API do YouTube para a exibição de exemplos práticos em vídeo, o que permitiria ao usuário visualizar o emprego de termos técnicos em contextos reais, como palestras de tecnologia e tutoriais especializados. Essa abordagem abordaria diretamente as observações sobre a dependência de conteúdo estático levantadas durante a fase de avaliação.

Além da expansão de conteúdo, o aprimoramento do sistema poderia priorizar a democratização do acesso por meio de melhorias estruturais focadas em acessibilidade para

pessoas com deficiência. Recomenda-se a implementação de diretrizes em conformidade com o WCAG (*Web Content Accessibility Guidelines*), visando tornar a plataforma plenamente utilizável através de leitores de tela, navegação facilitada via teclado e oferta de modos de alto contraste para usuários com baixa visão. Tais evoluções sugeridas não apenas elevariam o rigor técnico e a maturidade do *software* desenvolvido em Node.js e PostgreSQL, mas também reforçariam o papel social do DicioCode na promoção da inclusão dentro do ensino de tecnologia.

REFERÊNCIAS

Agência Brasil. **Empregos ligados à tecnologia cresceram 95% em 10 anos, diz pesquisa**. 2024. <https://agenciabrasil.ebc.com.br/geral/noticia/2024-11/empregos-ligados-tecnologia-cresceram-95-em-10-anos-diz-pesquisa>. Acesso em: 14 de julho de 2025. Citado na página 26.

APAT. Procura por cursos em áreas de tecnologia aumenta no brasil. **MundoGEO**, aug 2024. Disponível em: <https://mundogeo.com/2024/08/27/procura-por-cursos-em-areas-de-tecnologia-aumenta-no-brasil/>. Acesso em: 16 jul. 2025. Citado na página 1.

BAIRESDEV. The 100 top programming languages in 2026. **BairesDev Blog**, 2026. Disponível em: <https://www.bairesdev.com/blog/top-programming-languages/>. Acesso em: 28 jan. 2026. Citado na página 28.

BRASSCOM. **Relatório de inteligência e demanda por profissionais de TIC**. [S.l.], 2025. Disponível em: <https://brasscom.org.br/>. Acesso em: 28 jan. 2026. Citado na página 28.

FIELDING, R. T. Architectural Styles and the Design of Network-based Software Architectures. 2000. Tese (Doutorado em Ciência da Computação) – University of California, Irvine, CA, USA, 2000. Disponível em: <https://roy.gbiv.com/pubs/dissertation/fielding_dissertation.pdf>. Acesso em: 14 de julho de 2025.

GEEK HUNTER. Guia salarial e tendências de contratação para o setor de tecnologia. **Blog Geek Hunter**, 2026. Disponível em: <https://blog.geekhunter.com.br/>. Acesso em: 28 jan. 2026.

GIL, A. C. **Como Elaborar Projetos de Pesquisa**. 4ª. ed. São Paulo: Atlas, 2008. Citado na página 27.

Instituto Federal de Minas Gerais. **Portal do Campus São João Evangelista**. 2025. Website institucional que apresenta os cursos de nível técnico e superior ofertados pela instituição. Disponível em: <https://www.sje.ifmg.edu.br/portal/index.php>. Acesso em: 28 jan. 2026.

JETBRAINS. **The State of Developer Ecosystem 2025**. [S.l.], 2025. Disponível em: <https://www.jetbrains.com/lp/devecosystem-2025/>. Acesso em: 28 jan. 2026.

MDN WEB DOCS. **CSS**: Cascading Style Sheets. 2025. Website. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS>. Acesso em: 21 jul. 2025. Citado 2 vezes nas páginas 23 e 25.

_____. **HTML**: HyperText Markup Language. 2025. Website. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTML>. Acesso em: 21 jul. 2025. Citado 2 vezes nas páginas 22 e 25.

_____. **JavaScript**. 2025. Website. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 21 jul. 2025. Citado 2 vezes nas páginas 23 e 25.

MICROSOFT. **TypeScript for JavaScript Programmers**. 2025. Website. Disponível em: <https://www.typescriptlang.org/docs/handbook/typescript-for-javascript-programmers.html>. Acesso em: 21 jul. 2025. Citado na página 23.

MILAGRES, F.; PIMENTA, G. B. G.; SALES, H. S. **Relatório de projeto de pesquisa.** Curso de Sistemas de Informação - Instituto Federal de Minas Gerais - Campus São João Evangelista. 2026.

NIELSEN, J. **Usability Engineering**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. Citado na página 21.

NORMAN, D. A. **O Design do Dia a Dia**. Rio de Janeiro: Rocco, 2006. Citado na página 21.

OPENJS FOUNDATION. **About Node.js**. 2025. Website. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 21 jul. 2025. Citado na página 23.

ORTIZ, R. **A diversidade dos sotaques**: o inglês e as ciências sociais. São Paulo: Brasiliense, 2007.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de *Software*:** uma Abordagem Profissional. 8ª. ed. Porto Alegre: AMGH, 2016. Citado 2 vezes nas páginas 22 e 27.

PRISCILA, T. H. P. P. D. J. **A ausência do inglês como fator limitante para o profissional de Tecnologia da Informação**. 2023. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Faculdade de Tecnologia, Ciências e Educação (FATECE), Pirassununga, 2023. Disponível em: < https://fatece.edu.br/sumario/arquivos/A%20aus%C3%AAncia%20do%20ingl%C3%AAs%20como%20fator%20limitante_2023.pdf >. Acesso em: 1 mar. 2026.

SEVERINO, A. J. **Metodologia do trabalho científico**. 23. ed. São Paulo: Cortez, 2007. Revista e atualizada. Citado 2 vezes nas páginas 15 e 27.

TANEMBAUM, A. S.; WETHERALL, D. J. **Redes de Computadores**. 5ª. ed. São Paulo: Pearson Prentice Hall, 2011. Citado na página 22.

TECHOPEDIA. **Techopedia**: Educating IT Professionals. 2026. Disponível em: <https://www.techopedia.com/>. Acesso em: 28 jan. 2026. Citado na página 25.

The PostgreSQL Global Development Group. **PostgreSQL**. [S.l.], 2026. Versão 17. Disponível em: <https://www.postgresql.org/>. Acesso em: 28 jan. 2026. Citado na página 24.

SWELLER, J. Cognitive load theory, learning difficulty, and instructional design. **Learning and Instruction**, [S. l.], v. 4, n. 4, p. 295-312, 1994. Disponível em: <https://doi.org/10.1016/0959-4752(94)90003-5>. Acesso em: 1 mar. 2026.

W3SCHOOLS. **W3Schools Online Web Tutorials**. 2026. Disponível em: <https://www.w3schools.com/>. Acesso em: 28 jan. 2026. Citado na página 25.

WOOD, D.; BRUNER, J. S.; ROSS, G. The role of tutoring in problem solving. **Journal of Child Psychology and Psychiatry**, [S.l.], v. 17, n. 2, p. 89–100, 1976. Disponível em: <https://acamh.onlinelibrary.wiley.com/doi/10.1111/j.1469-7610.1976.tb00381.x>. Acesso em: 14 julho de 2026.

APÊNDICE A – QUESTIONÁRIO DE PESQUISA DE SATISFAÇÃO

Este apêndice apresenta o formulário criado pelo Google utilizado para a coleta de dados e validação do sistema DicioCode. O objetivo da pesquisa foi avaliar a percepção dos discentes quanto à usabilidade, clareza das informações e eficácia das novas funcionalidades implementadas.

1. Perfil do Usuário

1.1 Qual o seu curso atual?

◯ Técnico em Informática
◯ Engenharia da Computação
◯ Sistemas de Informação
◯ Outro: _____

1.2 Em qual período/série você está atualmente?

◯ 1º
◯ 2º
◯ 3º
◯ 4º ou superior

1.3 Como você avalia o seu nível de proficiência em inglês?

◯ Iniciante
◯ Básico
◯ Intermediário
◯ Avançado
◯ Fluente

1.4 Como você avalia o seu nível atual de conhecimento em programação?

◯ Nunca programei
◯ Iniciante (conheço o básico de lógica ou alguma linguagem)
◯ Intermediário (já desenvolvi pequenos projetos)
◯ Avançado (já programo com frequência)

2. Experiência com o Inglês Técnico e Programação

2.1 Você tem dificuldade para compreender textos técnicos em inglês (ex.: documentações, tutoriais, códigos, manuais)?

◯ Sim, frequentemente
◯ Às vezes
◯ Raramente
◯ Nunca

2.2 Em quais situações você sente dificuldade com o inglês técnico? (Marque todas as opções aplicáveis)

□ Leitura de documentações
□ Interpretação de códigos e comandos
□ Entendimento de vídeos/tutoriais
□ Comunicação oral (apresentações, conversas técnicas)
□ Escrita (redação de relatórios, e-mails, etc.)

2.3 Quando você começa a programar, como se sente ao ver comandos e mensagens de erro em inglês?

◯ Fico muito confuso(a), não entendo quase nada
◯ Consigo entender algumas palavras, mas não o sentido completo
◯ Entendo o suficiente para resolver os problemas
◯ Entendo tudo com tranquilidade

2.4 Você costuma traduzir os termos de programação (como *if, while, print, function*) para o português ao estudar?

◯ Sempre
◯ Às vezes
◯ Raramente
◯ Nunca

3. Experiência de Uso da Plataforma

3.1 Em uma escala de 1 (Muito Difícil) a 5 (Muito Fácil), quão fácil foi se cadastrar e fazer o primeiro *login* na plataforma?

◯ 1
◯ 2
◯ 3
◯ 4
◯ 5

3.2 Qual foi sua primeira impressão sobre o *design* (cores, fontes, *layout*) do *site*?

◯ Profissional e agradável
◯ Simples e funcional
◯ Um pouco confuso
◯ Desatualizado ou básico

3.3 Você acha fácil encontrar o que procura usando a barra de navegação (Início, Conteúdo, Dicionário, etc.)?

◯ Sim, é muito intuitiva
◯ Na maior parte do tempo
◯ Achei um pouco confusa

3.4 Ao acessar o "Início", o *modal* ("Bem-vindo de volta!") perguntando se você quer continuar de onde parou foi:

◯ Muito útil, me ajudou a voltar ao estudo
◯ Indiferente, eu não usei
◯ Irritante, eu preferia não vê-lo

3.5 Na página de uma linguagem (ex.: "Termos de JavaScript"), como você classifica a organização das palavras na lista?

◯ Ótima, fácil de ler e entender
◯ Boa, mas poderia ser melhor
◯ Confusa, achei difícil encontrar a informação que eu queria

3.6 Os três botões de áudio (pronúncia normal, lenta e tradução) foram fáceis de entender e usar?

◯ Sim, os ícones são claros e a função é excelente
◯ Sim, mas eu não sabia o que cada um fazia até clicar
◯ Não, achei confuso ou não funcionaram como esperado

3.7 A qualidade dos áudios (normal, lento e tradução) foi fácil de entender?

◯ Sim, muito
◯ Razoavelmente
◯ Não, a qualidade precisa melhorar

3.8 Qual foi sua experiência ao usar a barra de "Buscar termo..." (dentro de uma linguagem ou no Dicionário)?

◯ Foi ótima, encontrei o que precisava rapidamente.
◯ Foi razoável, mas tive alguma dificuldade ou lentidão.
◯ Foi ruim, não encontrou o que eu procurei.

◯ Não utilizei essa funcionalidade.

APÊNDICE B – GUIA DE INSTALAÇÃO E EXECUÇÃO DO SISTEMA

Este apêndice detalha os procedimentos necessários para a configuração do ambiente de desenvolvimento e a execução local do sistema DicioCode.

B.1 Pré-requisitos Técnicos

Para o correto funcionamento da aplicação, é necessário que o ambiente possua as seguintes ferramentas instaladas:

- Node.js(Versão 18 ou superior);
- Gerenciador de pacotes pnpm;
- Banco de dados relacional PostgreSQL.

B.2 Procedimentos de Configuração

O sistema está dividido em duas partes principais: *backend* (servidor) e *frontend* (interface).

B.3 Configuração do Servidor (*Backend*)

- Realize o clone do repositório através do comando *git clone* <https://github.com/DicioCode/back-end.git>;
- Para acessar o diretório, acesse a pasta do servidor através do comando cd *back-end*;
- Para instalar as dependências, execute o comando *pnpm install* para instalar os pacotes necessários;
- Crie um arquivo .env na raiz da pasta seguindo o modelo .env.example e configure as credenciais do seu banco PostgreSQL;
- Execute *npx sequelize-cli db:migrate* para criar a estrutura de tabelas no banco de dados via Sequelize;
- Inicie o servidor em modo de desenvolvimento com o comando *pnpm dev*.

B.4 Configuração da Interface (*Frontend*)

- Realize o clone do repositório através do comando *git clone* <https://github.com/DicioCode/front-end.git>;
- Acesse a pasta da aplicação através do comando *cd front-end*;
- Localize o arquivo index.html na raiz do projeto e abra-o diretamente em um navegador de preferência.

APÊNDICE C – GUIA DE ACESSO À DOCUMENTAÇÃO PELO SWAGGER

Este guia instrui o usuário sobre como acessar a documentação interativa da API REST do sistema DicioCode através da ferramenta Swagger.

C.1 Procedimentos de Acesso

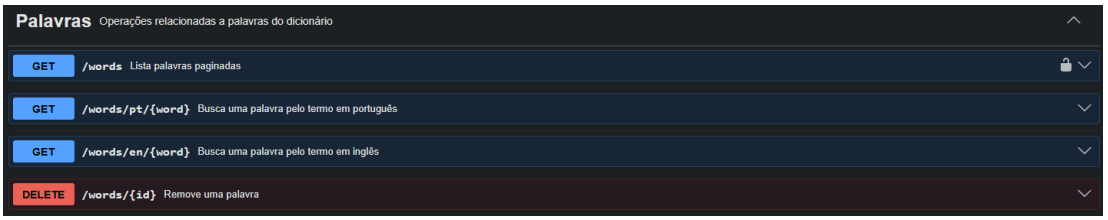
A documentação é gerada dinamicamente pelo servidor e, portanto, requer que o *backend* esteja em execução conforme as instruções de instalação detalhadas anteriormente:

a) Certifique-se de que o servidor foi iniciado através do comando *pnpm run dev* no diretório *back-end*;
b) Com o servidor ativo, abra o navegador e acesse a URL <http://localhost:8080/api-docs/>. Caso tenha alterado a porta padrão no arquivo .env, substitua o valor 8080 pela porta correspondente;
c) Na interface carregada, é possível visualizar todos os *endpoints* disponíveis, organizados por categorias como usuários, termos e áudios;
d) O Swagger permite realizar requisições de teste diretamente pela interface, facilitando a verificação do tráfego de dados em formato JSON entre o cliente e o banco de dados PostgreSQL.

ANEXO A – DOCUMENTAÇÃO DA API (SWAGGER)

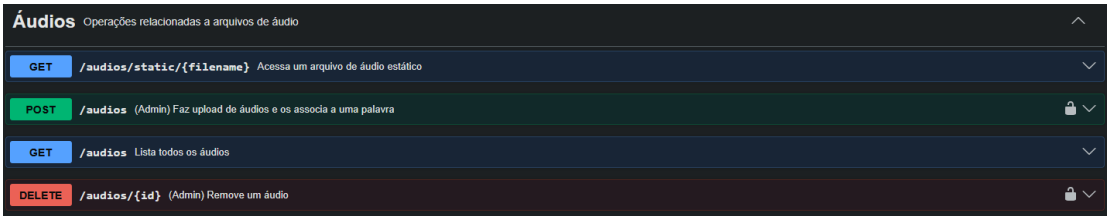
Este anexo apresenta a especificação técnica das rotas da API REST do sistema DicioCode, gerada automaticamente pela ferramenta Swagger/OpenAPI. Esta documentação serve como base para a integração entre as camadas do sistema e para a eventual expansão das funcionalidades por outros desenvolvedores.

Figura 17 - Interface do *Swagger UI*: documentação das rotas de palavras do *backend*.



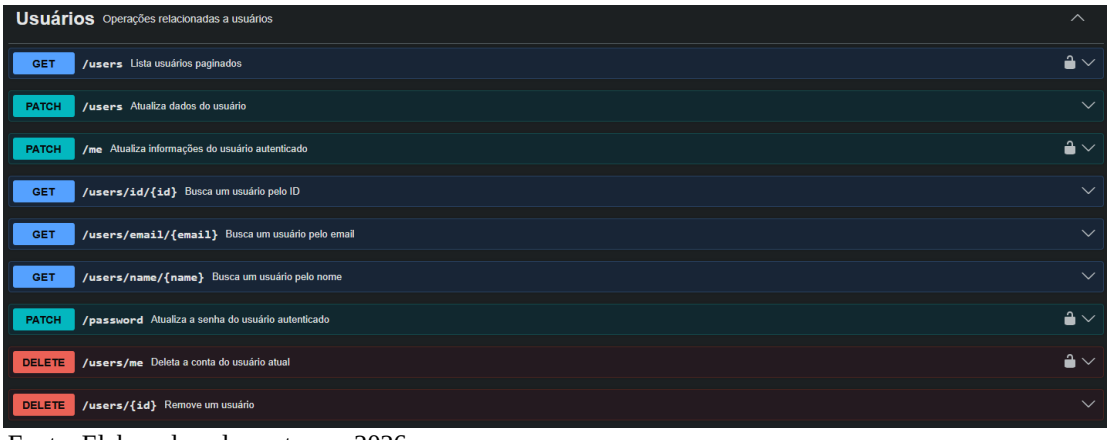
Fonte: Elaborado pelos autores, 2026.

Figura 18 - Interface do *Swagger UI*: documentação das rotas de áudio do *backend*.



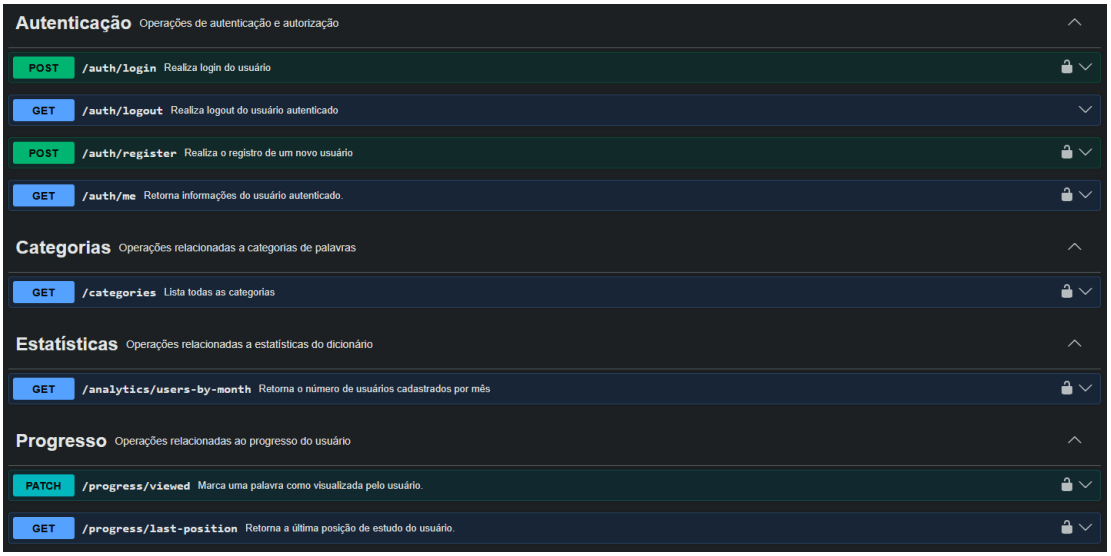
Fonte: Elaborado pelos autores, 2026.

Figura 19 - Interface do *Swagger UI*: documentação das rotas de usuário do *backend*.



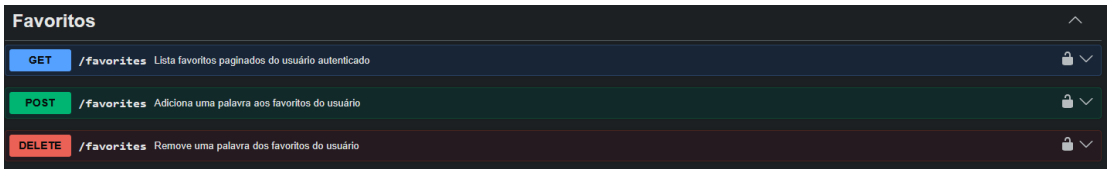
Fonte: Elaborado pelos autores , 2026.

Figura 20 - Interface do *Swagger UI*: documentação das rotas de autenticação, categorias, estatísticas e progresso do *backend*.



Fonte: Elaborado pelos autores, 2026.

Figura 21 - Interface do *Swagger UI*: documentação das rotas de favoritos do *backend*.



Fonte: Elaborado pelos autores, 2026.

A documentação detalha os métodos HTTP (*GET*, *POST*, *PUT*, *DELETE*), os esquemas de dados JSON aceitos e os códigos de resposta configurados para garantir a integridade da comunicação.