UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
PRÓ-REITORIA DE GRADUAÇÃO
DEPARTAMENTO DE ENGENHARIAS E TECNOLOGIA
CURSO INTERDISCIPLINAR EM TECNOLOGIA DA INFORMAÇÃO

Pedro Henrique Souza Pessoa

Desenvolvimento de uma API para apoio ao processo de aprendizagem de lógica de programação

PAU DOS FERROS
2025

Pedro Henrique Souza Pessoa

Desenvolvimento de uma API para apoio ao processo de aprendizagem de lógica de programação

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Bacharel Interdisciplinar em Tecnologia da Informação.

Orientador: Prof. Dr. Alysson Filgueira Milanez - UFERSA

PAU DOS FERROS

2025

P475d Pessoa, Pedro Henrique Souza.
Desenvolvimento de uma API para apoio ao processo de aprendizagem de lógica de programação / Pedro Henrique Souza Pessoa. - 2025.
82 f. : il.

Orientador: Alysson Filgueira Milanez.
Monografia (graduação) - Universidade Federal Rural do Semi-árido, Curso de Tecnologia da Informação, 2025.

1. Ensino de Programação. 2. Lógica de Programação. 3. Gamificação. 4. API. I. Milanez, Alysson Filgueira, orient. II. Título.

Ficha catalográfica elaborada por sistema gerador automáto em conformidade com AACR2 e os dados fornecidos pelo) autor(a).
Biblioteca Campus Pau dos Ferros
Bibliotecária: Erlanda Maria Lopes da Silva
CRB: 15/966

Pedro Henrique Souza Pessoa

Desenvolvimento de uma API para apoio ao processo de aprendizagem de lógica de programação

Monografia apresentada à Universidade Federal Rural do Semi-Árido como requisito parcial para obtenção do título de Bacharel Interdisciplinar em Tecnologia da Informação.

Defendida em: 18 / 07 / 2025

BANCA EXAMINADORA

Alysson Filgueira Milanez, Prof. Dr. (UFERSA)
Presidente

Bruno Borges da Silva, Prof. Esp. (UFERSA)
Membro Examinador

Douglas Afonso Tenório de Menezes, Prof. Dr. (IFAL)
Membro Examinador

Dedico esse trabalho a minha mãe, Claudia Maria, que correu sua vida inteira, para que eu pudesse chegar.

AGRADECIMENTOS

Carregar os sonhos e vontades daqueles que você ama não é um fardo, e sim, ânimo, força e coragem, para mim, uma obstinação, o motivo pelo qual me levantei todos os dias e fiquei acordado por muitas madrugadas, o motivo pelo qual cheguei até aqui, orgulhar aqueles me deram todo o apoio possível, e sacrificaram muito em suas vidas para que eu chegasse nesse momento, o mínimo que poderia fazer era dar tudo de mim, e nesse momento agradecer a cada um de deles. Sem antes esquecer de, primeiramente, agradecer a Deus que a todo momento foi meu abrigo, me escutou nos momentos mais difíceis, e que apesar das minhas falhas nunca deixou de me amar. E se iniciei esse agradecimento falando sobre carregar sonhos, era para falar da mulher na qual herdei o sonho de me formar, minha mãe, Claudia Maria, que lutou sua vida inteira e deu tudo de si para que eu conseguisse chegar até aqui. Não existem palavras suficientes para agradecer tudo que foi feito por você, mas saiba que é minha maior inspiração e ânimo, nas noites em claro, me dava forças pensar em como ela ficaria feliz ao fim disso tudo, até hoje não entendo muito bem quando ela me olhava e falava "meu filho, estude para não ficar que nem sua mãe" quando, na verdade, gostaria de um dia ser metade do ser humano que ela é, afinal de contas eu sou a continuação do seu sonho.

Carrego comigo alegria de ter nascido "rico", da riqueza que dinheiro algum pode comprar, a der ter uma família que faria de tudo por mim. Agradeço às minhas tias, Carla, Carleide e Kainara, pelo carinho que sempre tiveram comigo, com todo cuidado do mundo e uma ligação quase que materna. Às minhas primas, Camila, Kaline, Carol, Ana Laura, Évila e ao meu primo Caio Victor, pelas diversas alegrias vividas, os sorrisos arrancados e as emoções compartilhadas. As minhas avós, "Chiquita" e Maria Conceição, pelo amor derramado em forma de mimos e conselhos. Aos meus Padrinhos, Alvaro e Alessandra, pelo apoio e constante ajuda, e juntamente ao meu primo, Alessandro e minha Tia Francisca, serem verdadeiras inspirações e grandes conselheiros nessa minha vida acadêmica que está se iniciando. Ao meu pai, Sebastião, que no início de minha vida deu tudo de si para que eu tivesse o melhor. E por fim, à irmã que a vida me presenteou, Naruna, que apesar de tão diferente, tanto me completa, obrigado por cada conversa, sorriso e principalmente pela parceria.

Se não bastasse uma família de sangue, tive o prazer de formar uma grande família de pessoas que me acompanharam durante toda essa jornada. Gradeço Aldo, por ter me apresentado a cidade, ter me mostrado um novo norte, me acolher em sua casa e por cada momento vivido. A gradeço especialmente a Cicero, meu fiel escudeiro, pela companhia e pelo acolhimento diários,

por trazer essa sensação de casa para a residência. Agradeço ao irmão que a UFERSA me deu, Gerliano, meu companheiro de casa, um verdadeiro irmão, às vezes pai, nada que eu diga aqui vai conseguir expressar minha gratidão a tudo que foi feito, obrigado por cada momento. Agradeço a todos que fizeram parte da minha casa, a ala 2, especialmente a Vitor, Emson, Jean, Habadara, Andrey, Eriky e "Lulu". Não poderia esquecer de agradecer ao grupo que alegrou tantos dias de minha vida, "Coquinha", e aos que o compõem, Thiago, Lucas, Alex, Pedro Damião, Jeferson e Tiago, obrigado pelas risadas, jogatinas e barbaridades faladas. Agradeço também aos meus amigos que apesar da distância nunca deixaram de me apoiar, "Duda", Lázaro, "Duda" Alves e "Manu". Sem esquecer da flor de lírio que nasceu em meu jardim, Hillary, obrigado por cada momento, cada risada e abraço, pela calma e alegria que me traz, e por me fazer acreditar que as coisas podem ser diferentes e principalmente, mais alegres.

Não poderia esquecer do meu pai acadêmico, ao homem que tenho prazer de chamar de meu orientador, não poderia ter escolhido alguém melhor para me guiar nessa jornada. Obrigado pelos ensinamentos e pela paciência.

E por último, gostaria de agradecer ao meu avô, Raimundo, que já não se encontra aqui, mas que carrego seus ensinamentos em meu coração, que infelizmente, me deixou no meio do caminho, mas tenho certeza, que no momento em que eu levantar o canudo, vai me olhar com aquele olhar cheio de orgulho e o sorriso que me encantava, foi um prazer enorme dividir cada momento com o senhor, obrigado.

EPÍGRAFE

“O sonho das pessoas não tem fim”

(Eiichiro Oda)

RESUMO

Este trabalho apresenta o desenvolvimento de uma Application Programming Interface (API) Educacional para apoio ao ensino de lógica de programação, com foco em proporcionar uma experiência interativa e personalizada para alunos e educadores. O contexto atual do ensino de programação é marcado por desafios significativos, como a dificuldade dos alunos em aplicar conceitos teóricos em práticas reais, a falta de engajamento e a alta taxa de evasão em cursos introdutórios. Esses problemas motivaram a criação de uma solução que combinasse gamificação, personalização e interatividade para facilitar o aprendizado. A API foi projetada para gerenciar perfis de usuários, armazenar dados de desempenho, criar atividades gamificadas e fornecer suporte ao acompanhamento do progresso dos alunos. Além disso, incorpora elementos de gamificação, como nível baseado em Experience Points (XP), e permite a integração com ferramentas educacionais, como jogos sérios e plataformas de *e-learning*. Este estudo contribui para o campo do ensino de programação ao propor uma solução que atende às necessidades identificadas na literatura sobre dificuldades no aprendizado de lógica de programação. A API tem potencial para ser uma ferramenta valiosa no desenvolvimento de ambientes educacionais mais dinâmicos e engajadores, promovendo um aprendizado ativo e adaptado às necessidades dos alunos.

Palavras-chave: Ensino de programação; Lógica de programação; Gamificação; API.

ABSTRACT

This work presents the development of an Educational Application Programming Interface (API) to support the teaching of programming logic, focusing on providing an interactive and personalized learning experience for students and educators. The current context of teaching programming is marked by significant challenges, such as the difficulty students have in applying theoretical concepts to real-world practices, lack of engagement, and high dropout rates in introductory courses. These problems motivated the creation of a solution that combined gamification, personalization, and interactivity to facilitate learning. The API was designed to manage user profiles, store performance data, create gamified activities, and support student progress tracking. Additionally, it incorporates gamification elements, such as XP-based levels, and allows integration with educational tools like serious games and e-learning platforms. This study contributes to the field of programming education by proposing a solution that combines gamification, personalization, and interactivity, addressing the needs identified in the literature regarding challenges in learning programming logic. The API has the potential to be a valuable tool in developing more dynamic and engaging educational environments, promoting active learning tailored to students' needs.

Keywords: Programming education; Programming logic; Gamification; API.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CRUD	Create, Read, Update e Delete
DRF	Django Rest Framework
HTTP	Hypertext Transfer Protocol
IA	Inteligencia artificial
ID	Identity
IDEs	Integrated Development Environments
INEP	Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira
JSON	JavaScript Object Notation
JWT	JSON Web Token
LGPD	Lei Geral de Proteção de Dados Pessoais
MVT	Model-View-Template
PBL	Problem-Based Learning
PEP8	Style Guide for Python Code
REST	Representational State Transfer
SQL	Structured Query Language
TI	Tecnologia da Informação
UML	Unified Modeling Language
URL	Uniform Resource Locator
XP	Experience Points
XSS	Cross-Site Scripting

SUMÁRIO

1 INTRODUÇÃO

A Tecnologia da Informação (TI) desempenha um papel central no desenvolvimento econômico e social, impulsionando inovações em diversas áreas, desde saúde até entretenimento (World Bank, 2022). No entanto, a formação de profissionais capacitados para atuar nesse setor enfrenta desafios significativos, especialmente no ensino de lógica de programação, o qual é a base para o desenvolvimento de habilidades essenciais, como resolução de problemas e pensamento crítico (Jenkins, 2002). Apesar de sua importância, o aprendizado de programação frequentemente esbarra em barreiras como a dificuldade de abstração, a falta de motivação e a desconexão entre teoria e prática (Robins; Rountree; Rountree, 2003).

No entanto, o ensino de lógica de programação apresenta diversos desafios tanto para alunos quanto para professores (Jenkins, 2002). Estudantes frequentemente enfrentam dificuldades com a abstração e a aplicação prática dos conceitos teóricos (Jenkins, 2002). Além disso, a falta de motivação e o desinteresse são barreiras comuns, o que pode levar à evasão dos cursos (Gomes; Mendes, 2007). Esses desafios são agravados pela necessidade de adaptar métodos pedagógicos para atender às variadas necessidades e ritmos de aprendizagem dos alunos, o que pode dificultar a manutenção do engajamento e da participação ativa nas aulas (Bennedsen; Caspersen, 2007).

Os métodos tradicionais de ensino, como aulas expositivas e exercícios práticos, muitas vezes não são suficientes para engajar os estudantes e facilitar a compreensão dos conceitos (Milne; Rowe, 2002). Isso levanta a questão de como melhorar o ensino de lógica de programação para torná-lo mais eficaz e acessível.

A introdução de ferramentas digitais e abordagens pedagógicas inovadoras tem o potencial de transformar o aprendizado e ajudar a superar essas dificuldades. Ferramentas como ambientes de desenvolvimento integrados (Integrated Development Environments (IDEs)) e plataformas de aprendizagem on-line oferecem recursos interativos que podem facilitar a compreensão e aplicação dos conceitos de programação (Marcelino *et al.*, 2018).

Este trabalho visa explorar e abordar essas questões por meio da criação de uma API[1] que auxiliará o processo de ensino-aprendizagem de lógica de programação. A metodologia envolvida inclui a identificação das necessidades e dificuldades de professores e alunos mediante questionários e entrevistas, o desenvolvimento de uma API para apoiar o ensino e a realização de

[1] Repositório contendo a implementação da API disponível em: https://github.com/Pedro-Pesssoa/Apoio-aprendizagem

testes para validar a eficácia da solução proposta. A pesquisa foi realizada de forma exploratória e descritiva, com abordagens qualitativas e quantitativas, a fim de fornecer uma compreensão abrangente das questões envolvidas e das soluções possíveis.

Este estudo visa não somente melhorar a prática pedagógica no ensino de lógica de programação, mas também contribuir para o desenvolvimento de ferramentas digitais que apoiem a formação de futuros profissionais na área de TI. Acredita-se que, ao identificar e atender às necessidades específicas dos professores e alunos, é possível promover um ambiente de aprendizado mais eficaz e motivador, que prepare melhor os estudantes para os desafios do mercado de trabalho.

1.1 Problematização

A lógica de programação é um dos pilares no ensino de ciência da computação e áreas correlatas. No entanto, muitos estudantes enfrentam dificuldades significativas ao aprender esses conceitos, o que pode levar à desmotivação e até mesmo à evasão dos cursos (Bennedsen; Caspersen, 2007; Gomes; Mendes, 2007).

A complexidade da abstração e a aplicação prática dos conceitos teóricos são alguns dos principais obstáculos relatados por estudantes e professores. Jenkins (2002) e Gomes e Mendes (2007) destacam que a falta de compreensão dos conceitos básicos de lógica de programação é um problema recorrente, que afeta negativamente o desempenho acadêmico e a continuidade dos estudos.

Além disso, os métodos tradicionais de ensino, como aulas expositivas e exercícios práticos, muitas vezes não são suficientes para engajar os estudantes e facilitar a compreensão dos conceitos (Jenkins, 2002). A diversidade de perfis e ritmos de aprendizado entre os alunos torna o desafio ainda maior para os professores, que precisam adaptar suas estratégias pedagógicas para atender a essas variabilidades (Milne; Rowe, 2002). Bennedsen e Caspersen (2007) e Milne e Rowe (2002) apontam que a alta taxa de falha em disciplinas introdutórias de programação é um indicador claro da necessidade de melhorias nas abordagens de ensino.

Com o avanço da tecnologia, diversas ferramentas digitais se desenvolveram para apoiar o ensino de lógica de programação, como ambientes de desenvolvimento integrados (IDEs), plataformas de aprendizagem on-line e softwares educativos. Ferramentas como *Scratch* (MIT Media Lab, 2025) e *Alice* (Alice Project, 2025) têm se mostrado eficazes para introduzir conceitos básicos de programação devido à sua interface visual e interativa (Marcelino *et al.*, 2018).

É necessário desenvolver uma plataforma que combine métodos pedagógicos eficazes com ferramentas digitais inovadoras, especificamente voltada para o ensino de lógica de programação. Esta plataforma deve conseguir engajar os estudantes, facilitar a compreensão dos conceitos e proporcionar um ambiente de aprendizado dinâmico e acessível, atendendo às necessidades tanto de alunos quanto de professores.

Dessa forma, o presente trabalho visa desenvolver uma API que possa ser utilizada em plataformas e jogos sérios que possam suprir essa lacuna, contribuindo para a melhoria do ensino de lógica de programação e, consequentemente, para a formação de profissionais mais preparados e motivados na área de tecnologia da informação.

1.2 Objetivos

Esta seção, apresenta os objetivos da presente pesquisa, tanto o objetivo geral quanto os específicos.

1.2.1 Objetivo Geral

Desenvolver uma API para um jogo educacional, focada no ensino de lógica de programação.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral, os seguintes objetivos específicos foram elencados:

- Identificar as principais dificuldades enfrentadas por alunos e professores no ensino e aprendizado de lógica de programação, mediante a aplicação de questionários e entrevistas em instituições de ensino;
- Analisar abordagens pedagógicas e ferramentas digitais já existentes para o ensino de lógica de programação, visando identificar práticas eficazes que podem ser incorporadas na plataforma;
- Desenvolver uma API que suporte funcionalidades essenciais da plataforma, como gerenciamento de usuários, armazenamento de dados de desempenho dos alunos e integração com o *front-end*;
- Implementar *endpoints* que permitam a comunicação entre o *back-end* e o *front-end*;

1.3 Justificativa

O ensino de lógica de programação é um componente essencial na formação de profissionais da área de tecnologia, sendo fundamental para o desenvolvimento de habilidades como resolução de problemas, pensamento crítico e criatividade (Robins; Rountree; Rountree, 2003). No Brasil, dados do Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP) revelam que cursos de computação apresentam altas taxas de evasão, com índices superiores a 40% nos primeiros semestres (INEP, 2023). Essa realidade reflete os desafios enfrentados tanto por alunos quanto por professores no processo de ensino-aprendizagem.

Estudantes frequentemente relatam dificuldades relacionadas à abstração de conceitos e à aplicação prática dos mesmos (Alrubaye *et al.*, 2013). Além disso, a falta de motivação e o desinteresse são barreiras que podem ser mitigadas com abordagens pedagógicas inovadoras, como a gamificação e o uso de ferramentas digitais interativas (Deterding *et al.*, 2011). Professores, por outro lado, enfrentam o desafio de adaptar conteúdos e metodologias para atender às necessidades diversificadas dos alunos, muitas vezes sem recursos adequados para manter o engajamento (Mannila *et al.*, 2006).

Nesse contexto, ferramentas como *Scratch* (MIT Media Lab, 2025), *Alice* (Alice Project, 2025), *Codecademy* (Codecademy, 2025) e *Khan Academy* (Khan Academy, 2025) têm se mostrado eficazes para introduzir conceitos básicos de programação de forma acessível e envolvente (Marcelino *et al.*, 2018). Entretanto, ainda há uma lacuna significativa na oferta de plataformas que combinem funcionalidades avançadas de gerenciamento de aprendizado com elementos de gamificação e personalização.

A integração de métodos pedagógicos eficazes com ferramentas digitais tem o potencial de transformar o ensino de lógica de programação. Ferramentas como *Scratch* e *Alice* destacam-se por suas interfaces visuais e interativas, que facilitam a compreensão de conceitos abstratos (Marcelino *et al.*, 2018). Plataformas como *Codecademy* e *Khan Academy* oferecem cursos estruturados que combinam teoria e prática, promovendo um aprendizado mais dinâmico e acessível (Fessakis; Gouli; Mavroudi, 2013). No entanto, essas soluções ainda não atendem plenamente às demandas específicas de professores e alunos no contexto brasileiro, especialmente no que diz respeito à personalização e ao acompanhamento do progresso individual.

Portanto, este trabalho justifica-se pela necessidade de desenvolver soluções educacionais que atendam às demandas específicas de alunos e professores no ensino de lógica de programação. A criação de uma API robusta, utilizando tecnologias como *Python* e *Django*, tem o potencial

de oferecer um recurso valioso para aprimorar o ensino e aprendizado nesta área. Ao integrar funcionalidades como gerenciamento de conteúdo, acompanhamento de desempenho e suporte a atividades gamificadas, a API proposta busca proporcionar uma experiência de aprendizado mais interativa, personalizada e motivadora (Christie, 2013).

1.4 Contribuição do trabalho

Este estudo contribui para a área de ensino de lógica de programação ao propor uma solução para alguns dos desafios enfrentados tanto por alunos quanto por educadores. A principal contribuição é desenvolver uma API gamificada que oferece suporte ao processo de aprendizagem de programação, integrando funcionalidades como gerenciamento de conteúdos educacionais, acompanhamento de progresso e suporte a atividades interativas. Além disso, a API foi projetada para ser flexível e escalável, permitindo sua integração com outras ferramentas educacionais, como plataformas de *e-learning* e jogos sérios.

A abordagem adotada neste trabalho, destaca-se pela combinação de elementos de gamificação e personalização, que visam aumentar o engajamento dos alunos e facilitar a aplicação prática dos conceitos teóricos. Ao centralizar a lógica de *back-end* em uma interface programática, a API também possibilita que desenvolvedores criem novas aplicações educacionais, ampliando o alcance e as possibilidades de uso para diferentes públicos e contextos educacionais.

1.5 Organização do trabalho

Este trabalho está organizado da seguinte forma. O Capítulo 2 apresenta os principais conceitos teóricos relacionados ao ensino de lógica de programação, softwares educacionais e tecnologias de desenvolvimento. No Capítulo 3, são analisados trabalhos relacionados que abordam dificuldades no ensino de programação, abordagens pedagógicas inovadoras e o uso de ferramentas digitais, destacando lacunas que este trabalho busca preencher. O Capítulo 4 descreve a metodologia adotada.

No Capítulo 5, são apresentados os resultados obtidos. Por fim, o Capítulo 6 oferece uma reflexão final sobre o trabalho realizado, discutindo as principais contribuições, limitações do estudo e sugestões para futuros aprimoramentos da ferramenta proposta.

2 FUNDAMENTAÇÃO TEÓRICA

Compreender a lógica de programação é necessário para o desenvolvimento de habilidades de resolução de problemas e pensamento crítico, competências essenciais no campo da computação (Jenkins, 2002; Robins; Rountree; Rountree, 2003).

Este capítulo explora a literatura existente sobre o ensino de lógica de programação (seção 2.1), softwares educacionais (seção 2.2) e apresenta as tecnologias para o desenvolvimento do sistema do presente trabalho (seção 2.3).

2.1 Ensino e Aprendizado de Lógica de Programação

O ensino de lógica de programação é uma área de estudo rica e complexa que envolve a aplicação de diversas metodologias e ferramentas para facilitar a compreensão dos alunos (Jenkins, 2002). Segundo Robins, Rountree e Rountree (2003), a aprendizagem de programação é um processo difícil para muitos estudantes, devido à necessidade de dominar tanto os conceitos teóricos quanto as habilidades práticas necessárias para resolver problemas.

Estudos também indicam que a eficácia do ensino de lógica de programação pode ser aumentada por meio da integração de abordagens pedagógicas inovadoras e ferramentas tecnológicas (Gomes; Mendes, 2007; Milne; Rowe, 2002).

2.1.1 Abordagens Pedagógicas no Ensino de Lógica de Programação

Diversas abordagens pedagógicas têm sido exploradas para ensinar lógica de programação (Robins; Rountree; Rountree, 2003). Entre as mais tradicionais, destacam-se as aulas expositivas, onde os conceitos são apresentados diretamente pelo professor, e os exercícios práticos, que permitem aos alunos aplicar os conhecimentos em problemas reais. No entanto, essas estratégias são frequentemente complementadas por métodos inovadores, como a aprendizagem baseada em projetos (Problem-Based Learning (PBL)) (Bennedsen; Caspersen, 2007), que incentiva os alunos a resolverem problemas complexos em equipe, e a gamificação, que utiliza elementos de jogos para aumentar o engajamento e a motivação dos estudantes (Deterding *et al.*, 2011).

De acordo com Bennedsen e Caspersen (2007), a abordagem baseada em problemas (PBL) tem mostrado eficácia significativa, ao promover o aprendizado ativo e a aplicação prática do conhecimento teórico. Além disso, estudos de Milne e Rowe (2002) sugerem que o uso de jogos e simulações pode aumentar o engajamento dos alunos e facilitar a compreensão de

conceitos complexos.

2.1.2 Dificuldades no Ensino e Aprendizado de Lógica de Programação

O ensino de lógica de programação apresenta diversos desafios (Robins; Rountree; Rountree, 2003). Estudantes frequentemente enfrentam dificuldades com a abstração e a aplicação prática de conceitos teóricos (Jenkins, 2002). Além disso, a falta de motivação e o desinteresse são barreiras comuns, como observado por Gomes e Mendes (2007). Professores, por sua vez, enfrentam o desafio de adaptar o conteúdo e as metodologias de ensino para atender às necessidades diversificadas dos alunos, muitas vezes encontrando dificuldades em manter o engajamento e a participação ativa nas aulas (Bennedsen; Caspersen, 2007).

2.2 Softwares Educacionais

O uso de softwares educacionais no ensino de lógica de programação tem se mostrado uma estratégia eficaz para superar algumas das dificuldades encontradas tanto por alunos quanto por professores. Esses softwares oferecem uma abordagem interativa e prática para o aprendizado, muitas vezes gamificando o processo e tornando-o mais envolvente para os alunos (Fessakis; Gouli; Mavroudi, 2013). Além disso, ferramentas como ambientes de desenvolvimento integrados (IDEs) e plataformas de aprendizagem on-line proporcionam aos alunos a oportunidade de aplicar os conceitos teóricos em cenários práticos, facilitando a compreensão e retenção do conhecimento.

Com o avanço da tecnologia, diversas ferramentas digitais foram desenvolvidas para apoiar o ensino de lógica de programação. Ambientes de desenvolvimento integrados (IDEs), plataformas de aprendizagem on-line e softwares educativos são algumas das ferramentas mencionadas na literatura (Luxton-Reilly *et al.*, 2016).

Ferramentas digitais têm se mostrado eficazes para introduzir conceitos de lógica de programação a iniciantes, especialmente aquelas que utilizam interfaces visuais e interativas (Marcelino *et al.*, 2018). Ademais, cursos estruturados que combinam teoria e prática promovem um aprendizado mais dinâmico e acessível, facilitando a compreensão e a aplicação dos conceitos ensinados (Bennedsen; Caspersen, 2007).

2.3 Tecnologias de Desenvolvimento

No desenvolvimento de sistemas e plataformas educacionais, a escolha das tecnologias adequadas é fundamental para garantir a eficácia e eficiência do produto final. Linguagens de programação, *frameworks* e ferramentas de desenvolvimento desempenham um papel crucial na implementação de funcionalidades que atendam às necessidades dos usuários (Sommerville, 2011).

Neste contexto, tecnologias como *Python*, *Django* e API *RESTful* são amplamente utilizadas devido à sua flexibilidade, facilidade de uso e robustez (Rossum, 1991; Holovaty; Kaplan-Moss, 2009; Fielding, 2000).

2.3.1 Python

Python é uma linguagem de programação interpretada, de alto nível e de propósito geral. Ela foi criada por Guido van Rossum e lançada em 1991 (Rossum, 1991). *Python* é conhecida por sua sintaxe clara e legível, que facilita a aprendizagem e a manutenção do código.

Outrossim, a linguagem possui uma vasta biblioteca padrão e uma comunidade ativa, tornando-a uma escolha popular para desenvolvimento Web, automação, análise de dados, inteligência artificial, entre outras áreas (Rossum, 1991).

2.3.2 API

APIs são conjuntos de definições e protocolos que permitem que diferentes softwares se comuniquem entre si (Fielding, 2000). No contexto do desenvolvimento Web, APIs são frequentemente usadas para permitir que uma aplicação se conecte a serviços externos ou para permitir que outras aplicações interajam com a própria aplicação de maneira estruturada. APIs Representational State Transfers (RESTs) são particularmente populares por serem baseadas em Hypertext Transfer Protocol (HTTP) e serem simples de implementar e consumir (Fielding, 2000).

2.3.3 Django

Django é um *framework* de desenvolvimento Web de alto nível para a linguagem *Python*, que permite o desenvolvimento rápido e com design limpo e pragmático (Holovaty; Kaplan-Moss, 2009). Criado para facilitar a criação de aplicações Web complexas, *Django* oferece várias

funcionalidades prontas, como um sistema de administração, autenticação, roteamento, Uniform Resource Locator (URL), *templates* e muito mais. Sua arquitetura segue o padrão Model-View-Template (MVT), que promove a separação de responsabilidades no desenvolvimento de aplicações Web (Holovaty; Kaplan-Moss, 2009).

2.3.4 Django Rest Framework

Django Rest Framework (DRF) é uma poderosa e flexível biblioteca para construir API Web em *Django*. Ela fornece um conjunto de ferramentas e convenções que permitem criar APIs *RESTful* de maneira rápida e eficiente (Christie, 2013). A biblioteca DRF facilita a serialização de dados, a criação de *views* baseadas em classes e a implementação de autenticação e permissões. Além disso, ela suporta operações Create, Read, Update e Delete (CRUD) e permite a integração com uma ampla gama de bibliotecas e ferramentas do ecossistema *Django* (Christie, 2013).

2.3.5 Banco de Dados e SQL

Bancos de dados são sistemas que permitem o armazenamento, organização e recuperação de dados de maneira eficiente (Date, 2003). Structured Query Language (SQL) é a linguagem padrão usada para gerenciar e manipular bancos de dados relacionais (Date, 2003). SQL permite a realização de operações como inserção, consulta, atualização e exclusão de dados.

Bancos de dados relacionais, como MySQL, PostgreSQL e SQLite, utilizam SQL para gerenciar suas informações. A compreensão de bancos de dados e SQL é crucial para o desenvolvimento *back-end*, pois muitas aplicações dependem de armazenamento e manipulação eficiente de dados (Date, 2003).

2.3.6 Diagramas de Classe

Os diagramas de classe são uma das principais ferramentas utilizadas na modelagem de sistemas orientados a objetos. Eles fazem parte da linguagem de modelagem unificada (Unified Modeling Language (UML)), amplamente adotada na engenharia de software para representar estruturas e comportamentos de sistemas de maneira visual e padronizada (Booch; Rumbaugh; Jacobson, 2005). Um diagrama de classe descreve as classes de um sistema, seus atributos, métodos e as relações entre elas, como associações, heranças e dependências.

No contexto do desenvolvimento de APIs, o diagrama de classe é essencial para planejar

a arquitetura do sistema, identificar entidades-chave (como usuários, conteúdos e perguntas) e definir como essas entidades interagem entre si. Por exemplo, no presente trabalho, o diagrama de classe foi utilizado para mapear as relações entre usuários, progressos, conteúdos e perguntas, garantindo que a API fosse projetada de forma coesa e alinhada aos requisitos funcionais e regras de negócio (Fowler, 2003).

Além disso, o uso de diagramas de classe facilita a comunicação entre desenvolvedores e *stakeholders*, ao fornecer uma visão clara e abstrata da estrutura do sistema antes da implementação. Isso reduz a probabilidade de erros de design e garante que todas as funcionalidades estejam adequadamente representadas (Booch; Rumbaugh; Jacobson, 2005).

2.3.7 Diagramas de Casos de Uso

Os diagramas de casos de uso, outra ferramenta importante da linguagem UML, são utilizados para modelar as interações entre os atores (usuários ou sistemas externos) e o sistema em desenvolvimento (Cockburn, 2000). Esses diagramas descrevem as funcionalidades do sistema sob a perspectiva do usuário, destacando as ações que os atores podem realizar e os cenários correspondentes.

Neste trabalho, os diagramas de casos de uso foram empregados para identificar e documentar as principais interações dos usuários com a API, como autenticação, cadastro de usuários, visualização de conteúdos, respostas a perguntas e acompanhamento de progresso. Cada caso de uso representa uma funcionalidade específica, permitindo uma análise detalhada dos requisitos do sistema e das restrições de acesso, como as permissões de administradores e usuários comuns (Cockburn, 2000).

Os diagramas de casos de uso ajudam a validar o escopo do sistema, permitindo avaliar se todas as necessidades dos usuários estão sendo atendidas e se as funcionalidades estão alinhadas aos objetivos do projeto. Ademais, servem como base para a criação de testes funcionais e para a documentação da API, facilitando a integração com outras aplicações (Fowler, 2003).

3 TRABALHOS RELACIONADOS

Neste capítulo, serão apresentados os principais trabalhos relacionados ao ensino de lógica de programação e ao uso de ferramentas digitais para apoiar esse processo. A análise desses trabalhos forneceu uma base para a compreensão do estado da arte na área e identificou lacunas que o presente estudo planejou preencher.

3.1 Dificuldades no Ensino de Programação

Estudos como o de Jenkins (2002) destacam as dificuldades que os alunos enfrentam ao aprender programação, particularmente na abstração e aplicação prática de conceitos teóricos. Jenkins (2002) argumenta que a programação exige uma combinação de habilidades cognitivas que não são intuitivas para muitos estudantes, como a capacidade de decompor problemas complexos em etapas menores e a necessidade de pensar de forma lógica e sistemática. Ele aponta que os alunos frequentemente enfrentam barreiras na compreensão de estruturas de controle, manipulação de variáveis e no desenvolvimento de algoritmos, levando à frustração e, muitas vezes, à desistência dos cursos.

Além disso, Gomes e Mendes (2007) discutem os fatores que contribuem para as dificuldades no aprendizado de programação, com destaque para a falta de motivação e o desinteresse dos alunos. Gomes e Mendes (2007) ressaltam que os métodos tradicionais de ensino, como aulas expositivas e exercícios repetitivos, muitas vezes não conseguem envolver os estudantes de maneira significativa. Eles identificam que, para muitos alunos, a programação é vista como uma atividade isolada e abstrata, sem conexão clara com aplicações práticas ou problemas reais. Essa desconexão contribui para uma alta taxa de evasão, especialmente em cursos introdutórios. Os autores também sugerem que o uso de abordagens pedagógicas mais dinâmicas e interativas poderia ajudar a reduzir essas dificuldades.

Esses desafios são recorrentes e têm sido documentados na literatura (Bennedsen; Caspersen, 2007), indicando uma necessidade constante de melhorias nas abordagens pedagógicas e nas ferramentas de ensino. Nesse contexto, a utilização de metodologias ativas de aprendizagem, como a gamificação e o uso de jogos sérios, pode ser uma forma eficaz de engajar os alunos e facilitar a compreensão dos conceitos complexos de programação (Alrubaye *et al.*, 2013).

3.2 Abordagens Pedagógicas Inovadoras

Diferentes abordagens pedagógicas foram propostas para melhorar o ensino de programação. Bennedsen e Caspersen (2007) analisam a eficácia de diferentes estratégias de ensino, com um foco específico nas altas taxas de reprovação em cursos introdutórios de programação. Os autores sugerem que a integração de metodologias ativas de aprendizagem, como a aprendizagem baseada em projetos (PBL) e a gamificação, pode aumentar o engajamento dos alunos e facilitar a compreensão dos conceitos de programação. Eles argumentam que essas abordagens permitem que os estudantes apliquem o conhecimento adquirido em situações práticas e reais, promovendo um aprendizado ativo em vez de passivo. O estudo revela que a aprendizagem ativa estimula os alunos a se envolverem mais com o conteúdo, especialmente quando precisam resolver problemas complexos ou trabalhar em equipe. Além disso, a gamificação, quando aplicada corretamente, pode contribuir para a motivação e a persistência dos estudantes, transformando o processo de aprendizado em uma experiência mais dinâmica e desafiadora.

Robins, Rountree e Rountree (2003) discutem a eficácia de utilizar exemplos práticos e problemas reais para contextualizar o aprendizado da programação. O estudo deles é uma revisão sobre o ensino e a aprendizagem de programação, na qual eles destacam que muitos estudantes enfrentam dificuldades devido à desconexão entre a teoria ensinada e a prática necessária para resolver problemas. Os autores argumentam que o uso de exemplos concretos e contextualizados ajuda a tornar o aprendizado mais relevante e interessante para os alunos, uma vez que permite a visualização de como os conceitos de programação se aplicam a cenários do dia a dia. Robins, Rountree e Rountree (2003) enfatizam a importância de combinar teoria e prática, oferecendo oportunidades para os alunos testarem e ajustarem suas soluções em um contexto que simule situações reais. Isso não somente facilita a compreensão dos conceitos de programação, mas também promove o desenvolvimento de habilidades de resolução de problemas.

Esses estudos sugerem que abordagens pedagógicas inovadoras, que vão além das aulas expositivas tradicionais, podem ajudar a reduzir as dificuldades no ensino de programação. A utilização de metodologias ativas, exemplos práticos e problemas contextualizados são estratégias promissoras para tornar o aprendizado mais dinâmico e eficaz (Guo *et al.*, 2014).

3.3 Uso de Ferramentas Digitais

O uso de ferramentas digitais no ensino de programação tem sido um foco significativo de pesquisa nos últimos anos. Marcelino *et al.* (2018) destacam a eficácia de ferramentas como *Scratch* (MIT Media Lab, 2025) e *Alice* (Alice Project, 2025) para introduzir conceitos de lógica de programação a iniciantes, devido à sua interface visual e interativa. Essas ferramentas permitem que os alunos visualizem e manipulem os elementos da programação de maneira intuitiva, o que pode facilitar a compreensão de conceitos abstratos.

Além disso, plataformas de aprendizagem on-line têm se mostrado úteis para complementar o ensino tradicional. (Fessakis; Gouli; Mavroudi, 2013) discutem como plataformas como *Codecademy* (Codecademy, 2025) e *Khan Academy* (Khan Academy, 2025) oferecem cursos estruturados que combinam teoria e prática, promovendo um aprendizado mais dinâmico e acessível. Essas plataformas permitem que os alunos aprendam no seu próprio ritmo e revisem os conceitos conforme necessário, o que pode ser particularmente útil para aqueles que enfrentam dificuldades no aprendizado de programação (Luxton-Reilly *et al.*, 2016).

3.4 Estudos de Caso e Implementações Práticas

Vários estudos de caso demonstram a eficácia de diferentes abordagens e ferramentas no ensino de programação. Milne e Rowe (2002) conduziram uma investigação sobre as dificuldades enfrentadas por estudantes e professores em cursos de programação. Eles identificaram que muitos alunos têm dificuldades em entender os conceitos fundamentais de programação, especialmente em relação à abstração e à lógica de controle. O estudo revelou que as dificuldades dos estudantes não estão somente relacionadas ao conteúdo, mas também aos métodos de ensino utilizados, que frequentemente falham em adaptar-se às diversas formas de aprendizado dos alunos. Os autores sugerem soluções práticas para esses problemas, como o uso de tutoriais interativos, programas de mentoria e suporte personalizado para estudantes que apresentam dificuldades específicas. Essas abordagens visam proporcionar um aprendizado mais orientado e menos dependente de métodos tradicionais, aumentando a taxa de sucesso dos alunos em cursos introdutórios.

Além disso, Holovaty e Kaplan-Moss (2009) descrevem a implementação de um curso de programação baseado em projetos, utilizando o *framework Django* para ensinar desenvolvimento Web. A abordagem deles foi centrada em projetos práticos que os alunos precisavam desenvolver

ao longo do curso, o que incentivou o aprendizado ativo e a aplicação direta dos conceitos de programação em situações reais. Os autores relatam que essa metodologia resultou em um aumento significativo no engajamento dos alunos, que se mostraram mais motivados a aprender e a se aprofundar nos conceitos devido à relevância prática dos projetos. A implementação prática de um projeto real permitiu que os estudantes compreendessem o ciclo completo de desenvolvimento, desde a modelagem de dados até a criação de interfaces Web interativas. Esse tipo de abordagem ajudou a melhorar a compreensão dos conceitos de programação e a promover uma experiência de aprendizado mais significativa e conectada à prática profissional.

Esses estudos indicam que a aplicação de metodologias práticas e baseadas em projetos pode ter um impacto positivo no ensino de programação, abordando tanto as dificuldades enfrentadas pelos estudantes quanto os desafios pedagógicos dos professores. O uso de exemplos práticos, tutoriais personalizados e projetos reais não somente facilita a compreensão dos conceitos, mas também promove um aprendizado mais ativo e engajador (Mannila *et al.*, 2006).

3.5 Gamificação no Ensino de Programação

A gamificação é o uso de elementos de jogos, como pontuação, níveis e recompensas, em contextos que não são de jogos, visando aumentar o engajamento e a motivação (Deterding *et al.*, 2011). No ensino de programação, a gamificação tem se mostrado uma estratégia promissora para melhorar a experiência de aprendizado dos alunos (Deterding *et al.*, 2011).

O estudo de Carvalho, Brandão e Paiva (2017) explorou a aplicação da gamificação no ensino de testes de software, um subcampo da engenharia de software. Os resultados indicam que o uso de competição e elementos visuais de progresso auxiliou os alunos a se manterem motivados e engajados durante o processo de aprendizado. Contudo, os autores também destacam que, para ser eficaz, a gamificação deve complementar uma sólida abordagem pedagógica, e não a substituir. A gamificação funcionou melhor quando os elementos de jogo estavam alinhados aos objetivos educacionais, permitindo que os alunos aprendessem de maneira interativa (Carvalho; Brandão; Paiva, 2017).

Além disso, a literatura sugere que a personalização dos elementos de gamificação pode melhorar significativamente a sua eficácia. Quando os desafios e recompensas são adaptados ao nível de habilidade e ao ritmo de aprendizado de cada aluno, a gamificação tende a ser mais eficaz em promover um aprendizado significativo (Nah *et al.*, 2014). Por outro lado, implementar gamificação sem considerar as características individuais dos alunos pode resultar em frustração

ou desinteresse, especialmente se os desafios forem muito difíceis ou triviais (Burke, 2014). Portanto, o design das atividades gamificadas deve ser cuidadosamente planejado para garantir que proporcionem um equilíbrio adequado entre dificuldade e recompensa, além de incentivar a participação contínua dos alunos (Aponte; Tempel; Carter, 2014).

A aplicação da gamificação no ensino de programação tem o potencial de aumentar o engajamento dos alunos e de tornar o aprendizado mais dinâmico (Nah *et al.*, 2014). No entanto, os desafios relacionados à integração eficaz desses elementos indicam que a gamificação deve ser cuidadosamente planejada e implementada para garantir que os objetivos de aprendizado sejam atingidos sem desviar o foco do conteúdo programático (Aponte; Tempel; Carter, 2014).

3.6 Jogos Sérios no Ensino de Programação

Os jogos sérios, diferentemente da gamificação, são jogos desenvolvidos com o objetivo específico de ensinar, treinar ou simular habilidades em um contexto educacional, ou profissional (Susi; Johannesson; Backlund, 2007). No ensino de programação, os jogos sérios proporcionam um ambiente de aprendizado prático e interativo, permitindo que os alunos apliquem os conceitos de programação em situações simuladas e controladas (Michael; Chen, 2006).

Fernandes, Sousa e Silva (2012) destacam que os jogos sérios têm sido amplamente utilizados na educação de engenharia de software, proporcionando um ambiente envolvente onde os alunos podem praticar habilidades técnicas e enfrentar desafios reais em um ambiente simulado.

Mediante cenários de jogos, os alunos são incentivados a resolver problemas e tomar decisões críticas, ajudando a solidificar sua compreensão dos conceitos de programação. A chave para o sucesso dos jogos sérios é o equilíbrio entre diversão e aprendizado, garantindo que o jogo seja, ao mesmo tempo, envolvente e instrutivo (Fernandes; Sousa; Silva, 2012).

Plataformas como *CodeCombat*[2] e *LightBot*[3] são exemplos de jogos sérios utilizados para ensinar programação. Essas plataformas utilizam uma abordagem interativa para ensinar conceitos fundamentais de lógica de programação, como estruturas de controle e algoritmos. Guo *et al.* (2014) indicam que essas ferramentas são eficazes para introduzir alunos iniciantes à programação, ao permitirem que eles aprendam enquanto jogam, promovendo um aprendizado ativo e engajado. Os jogos sérios também facilitam a aplicação prática de conceitos de progra-

[2] Disponível em: https://codecombat.com. Acesso em 07 jul. 2025
[3] Disponível em: https://lightbot.com. Acesso em: 07 jul. 2025

mação em cenários controlados, o que pode ser benéfico para alunos que têm dificuldade em aplicar teorias abstratas em situações reais.

Os estudos revisados, incluindo Guo *et al.* (2014), indicam que os jogos sérios são particularmente eficazes para engajar os alunos e ajudá-los a aplicar os conceitos de programação de forma prática. No entanto, assim como na gamificação, o design dos jogos sérios deve ser cuidadosamente alinhado aos objetivos educacionais para garantir que atendam às necessidades dos alunos.

3.7 Comparação entre os trabalhos relacionados

A revisão dos trabalhos relacionados demonstra que, apesar de existirem diversas ferramentas e abordagens inovadoras para o ensino de lógica de programação, ainda há desafios significativos, como a necessidade de maior engajamento e a superação das dificuldades relacionadas à abstração de conceitos teóricos. Este estudo visa contribuir para a área com o desenvolvimento de uma API que não somente atenda às necessidades identificadas nos trabalhos analisados, mas que também traga uma solução mais integrada e interativa, com o uso de gamificação e jogos sérios.

O objetivo é preencher as lacunas identificadas, proporcionando uma ferramenta que combine a interatividade das ferramentas digitais e dos jogos com a profundidade necessária para o ensino de lógica de programação, melhorando a experiência tanto de professores quanto de alunos. O Quadro 1 apresenta um comparativo entre os trabalhos relacionados e o presente trabalho.

Essa comparação evidencia que, embora os trabalhos relacionados tenham contribuído significativamente para o avanço no ensino de programação, ainda existem oportunidades de melhoria. O estudo proposto oferece uma solução mais integrada, focada tanto na motivação quanto na aplicação prática, através da combinação de gamificação e jogos sérios. Além disso, a API permitirá que professores e alunos adaptem as atividades conforme suas necessidades, proporcionando um ambiente de aprendizado mais dinâmico e personalizado.

Quadro 1 – Comparação entre os trabalhos relacionados e o estudo proposto

Estudo	**Abordagens Utilizadas**	**Linguagens Abordadas**	**Uso de Gamificação**	**Forma de Validação**
Jenkins (2002)	Aulas expositivas e exercícios práticos para ensino de programação.	Várias linguagens introdutórias (como Java e C).	Não	Estudo qualitativo com base em entrevistas e análise de dados acadêmicos.
Gomes e Mendes (2007)	Análise das dificuldades de aprendizado, incluindo a falta de motivação e metodologias tradicionais.	Não especificado.	Não	Avaliação baseada em taxas de evasão e entrevistas com alunos.
Bennedsen e Caspersen (2007)	Utilização de metodologias ativas, como aprendizagem baseada em projetos e gamificação.	Java, C++	Sim	Estudo comparativo entre turmas que usaram metodologias ativas e turmas que usaram métodos tradicionais.
Marcelino *et al.* (2018)	Ferramentas visuais como Scratch e Alice para introdução de lógica de programação.	Scratch, Alice	Não	Análise do desempenho de estudantes em testes práticos e observação em sala de aula.
Calderón, Ruiz e Pérez (2018)	Uso de gamificação no ensino de engenharia de software.	Não especificado	Sim	Revisão sistemática e mapeamento de estudos existentes.
Carvalho, Brandão e Paiva (2017)	Gamificação no ensino de testes de software, com foco em competição e progresso visual.	Python	Sim	Experimento empírico com grupos de controle e experimentais.
Fernandes, Sousa e Silva (2012)	Uso de jogos sérios para ensino de engenharia de software.	Java	Sim	Estudos de caso e feedback dos participantes sobre a experiência de aprendizado.
Estudo Proposto	Desenvolvimento de uma API gamificada para auxiliar o ensino de lógica de programação, integrando jogos sérios e ferramentas interativas.	Python	Sim	Análise de aspectos técnicos da API

Fonte: Autoria própria (2025)

4 METODOLOGIA

A pesquisa realizada é do tipo exploratória e descritiva, com uma abordagem qualitativa e quantitativa (Gil, 2008; Creswell, 2014). A pesquisa exploratória visa proporcionar maior familiaridade com o problema, tornando-o mais explícito e estabelecendo bases para hipóteses (Gil, 2008). A pesquisa descritiva, por sua vez, foca no público-alvo para identificar e compreender os principais desafios enfrentados no processo de ensino de lógica de programação (Vergara, 2000).

A combinação dessas abordagens foi essencial para identificar as necessidades e dificuldades de professores e alunos no aprendizado de lógica de programação e para fornecer uma base sólida para o desenvolvimento de uma API que atenda a essas demandas específicas.

4.1 Idealização do Estudo

O ponto de partida da metodologia é a idealização do estudo, que envolve o entendimento aprofundado do problema central e a formulação dos objetivos de pesquisa. Durante essa fase, foram identificados os principais desafios enfrentados no ensino de lógica de programação, incluindo a falta de engajamento e a dificuldade dos alunos em compreender conceitos abstratos como estruturas de controle, manipulação de variáveis e algoritmos (Jenkins, 2002).

Essa compreensão inicial orienta as próximas etapas da pesquisa, garantindo que o estudo aborde aspectos críticos para o público-alvo e forneça uma solução relevante (Sommerville, 2011). Com a idealização, foram também definidos o público-alvo (professores e alunos de disciplinas introdutórias de programação) e os principais objetivos da pesquisa, como a criação de uma API para o apoio ao processo de aprendizagem de lógica de programação.

4.2 Elaboração dos Questionários

A próxima etapa consiste na elaboração de questionários estruturados, que visam coletar dados quantitativos e qualitativos sobre as necessidades, dificuldades e expectativas dos professores e alunos em relação ao ensino de lógica de programação. Os questionários foram planejados com base nos problemas identificados na fase de idealização, que incluem:

- **Dificuldade na compreensão de conceitos abstratos**, como *loops*, condicionais e manipulação de variáveis, conforme destacado por Jenkins (2002);
- **Falta de motivação e desinteresse** dos alunos, causados pela desconexão entre teoria e prática, como discutido por Gomes e Mendes (2007);

- **Desafios relacionados à abstração**, como a incapacidade de aplicar conceitos teóricos em situações práticas, apontada por Bennedsen e Caspersen (2007);
- **Altas taxas de evasão e reprovação** em cursos introdutórios de programação, conforme analisado por Milne e Rowe (2002) e Andrade (2023);
- **Limitações dos métodos tradicionais de ensino**, como aulas expositivas e exercícios repetitivos, que não conseguem engajar os estudantes de forma significativa, conforme observado por Robins, Rountree e Rountree (2003).

Esses problemas foram divididos em duas seções principais nos questionários: perguntas fechadas, que permitem a coleta de dados quantitativos, e perguntas abertas, que possibilitam uma análise qualitativa mais profunda.

- Perguntas fechadas: incluem perguntas de múltipla escolha, escalas de concordância e opções binárias (como sim/não), possibilitando a coleta de dados quantitativos para análise estatística. Exemplo de pergunta: "Acredita que a lógica de programação é um desafio significativo para a maioria dos alunos?" com opções de resposta como Sim, Não, Talvez.
- Perguntas abertas: permitem aos participantes expressarem suas opiniões de forma mais livre e detalhada, capturando dados qualitativos que possibilitam uma compreensão mais profunda do contexto e das dificuldades relatadas. Um exemplo de pergunta aberta é: "Quais conceitos de lógica de programação você considera mais difíceis de entender?".

Essas questões foram desenvolvidas para garantir que os dados coletados proporcionem uma visão abrangente das dificuldades enfrentadas pelos alunos e das necessidades dos professores no ensino de lógica de programação (Apêndice A).

4.3 Aplicação dos Questionários

Após a elaboração, os questionários foram aplicados a professores e alunos de instituições de ensino federais, especificamente aqueles envolvidos em cursos introdutórios de programação. A aplicação foi realizada on-line, garantindo que a amostra incluísse uma diversidade de perspectivas e experiências. A amostra foi selecionada intencionalmente, englobando participantes de cursos de TI e áreas correlatas, visando coletar dados representativos e relevantes para o desenvolvimento da API.

Essa etapa visou obter um volume significativo de respostas, permitindo a análise de tendências e padrões entre diferentes perfis de participantes. Os questionários on-line foram distribuídos por meio de plataformas digitais, aplicados em instituições parceiras, sempre com o

cuidado de garantir a clareza das perguntas e a acessibilidade da pesquisa.

4.4 Realização das Entrevistas

Com os dados iniciais coletados pelos questionários, foi possível identificar temas e questões recorrentes que demandavam uma investigação mais aprofundada. A etapa seguinte, então, foi a realização de entrevistas semiestruturadas com uma amostra selecionada de professores. As entrevistas foram planejadas para expandir a compreensão dos dados obtidos nos questionários, permitindo uma análise qualitativa dos fatores que influenciam o aprendizado e o ensino de lógica de programação.

Durante as entrevistas, os participantes foram incentivados a discutir em maior profundidade as dificuldades que enfrentam, as práticas de ensino que consideram mais eficazes, e as características que gostariam de ver em uma ferramenta de apoio ao ensino de programação. As entrevistas seguiram um roteiro de perguntas baseadas nas respostas obtidas nos questionários, mas também permitiram flexibilidade para os entrevistados abordarem outros pontos que considerassem relevantes (Apêndice A).

As respostas das entrevistas foram gravadas e transcritas para posterior análise de conteúdo, visando identificar temas comuns, padrões e sentimentos expressos pelos participantes. Essa análise forneceu subsídios para a próxima etapa de desenvolvimento, garantindo que o design da API e suas funcionalidades estejam alinhados às necessidades identificadas.

4.5 Elicitação de Requisitos

A elicitação de requisitos, etapa crucial no desenvolvimento de sistemas, permite identificar e documentar as necessidades dos usuários e *stakeholders* para garantir que o sistema proposto atenda às expectativas (Sommerville, 2011). Neste trabalho, a elicitação de requisitos foi conduzida com base em três abordagens principais: (i) revisão bibliográfica sobre os desafios no ensino de lógica de programação, (ii) coleta de dados por meio de questionários e entrevistas com professores e alunos, e (iii) análise das funcionalidades oferecidas por ferramentas digitais existentes.

4.5.1 Base Teórica para a Definição de Requisitos

O levantamento bibliográfico realizado no Capítulo 2 forneceu *insights* valiosos sobre as dificuldades enfrentadas no ensino de lógica de programação. Estudos como os de Jenkins (2002) e Gomes e Mendes (2007) destacam a necessidade de abordagens pedagógicas inovadoras que promovam engajamento e facilitem a compreensão de conceitos abstratos.

Além disso, trabalhos como os de Bennedsen e Caspersen (2007) e Robins, Rountree e Rountree (2003) enfatizam a importância de combinar teoria e prática por meio de atividades interativas e contextualizadas. Esses estudos forneceram embasamento para a definição de requisitos funcionais relacionados ao suporte ao aprendizado ativo e à gamificação.

4.5.2 Coleta de Dados por meio de Questionários e Entrevistas

Os questionários e entrevistas aplicados a professores e alunos permitiram identificar necessidades específicas do público-alvo. Os questionários foram estruturados para capturar tanto dados quantitativos quanto qualitativos.

As entrevistas semiestruturadas, por sua vez, proporcionaram uma compreensão mais profunda das práticas pedagógicas adotadas pelos professores e das características desejadas em uma ferramenta de apoio ao ensino.

4.5.3 Análise de Ferramentas Digitais Existentes

A análise de ferramentas digitais já utilizadas no ensino de programação também contribuiu para a elicitação de requisitos. Foram analisadas as seguintes ferramentas: *Scratch* (MIT Media Lab, 2025), *Alice* (Alice Project, 2025), *Codecademy* (Codecademy, 2025) e o *Khan Academy* (Khan Academy, 2025). Essas plataformas foram escolhidas por sua relevância no ensino de programação e lógica. A análise buscou identificar características que pudessem ser incorporadas ao sistema, como feedback imediato e mecanismos de progresso personalizado. Os resultados dessa análise foram utilizados para definir funcionalidades e diretrizes de design durante a modelagem do sistema.

Esses exemplos inspiraram requisitos funcionais como a criação de triagem inicial para avaliar o nível de conhecimento do usuário e a implementação de alternativas ajustadas automaticamente para perguntas de nível fácil (A documentação detalhada da API está disponível no Apêndice B).

4.5.4 Documentação dos Requisitos

A documentação dos requisitos foi realizada de forma estruturada, com base nas informações coletadas durante a fase de elicitação. Para organizar os requisitos funcionais e não funcionais, foi elaborado um quadro-resumo (veja o Capítulo 5 para detalhes) que categoriza cada requisito de acordo com sua descrição, justificativa e relevância para o sistema. Essa abordagem permitiu garantir que todos os requisitos estivessem alinhados aos objetivos propostos e às necessidades dos usuários. Além disso, os requisitos foram priorizados com base em critérios de essencialidade, importância e desejo. Esse processo foi guiado por práticas recomendadas na literatura, como as descritas por Malhotra (2010) e Robins, Rountree e Rountree (2003).

4.6 Modelagem do Sistema

A modelagem do sistema foi realizada com base nos requisitos funcionais e não funcionais. O objetivo desta etapa é representar visualmente a arquitetura do sistema, as interações entre os atores e os principais casos de uso, garantindo uma visão clara das funcionalidades e do comportamento esperado da API.

4.6.1 Diagrama de Classes

O diagrama de classes descreve a estrutura do sistema, detalhando as entidades principais, seus atributos, métodos e relacionamentos (Sommerville, 2011). Para a criação, foi utilizada a plataforma *Lucidchart*[4], que permite a modelagem visual de sistemas orientados a objetos. A escolha da plataforma foi devido ao seu funcionamento intuitivo, utilizando blocos, que facilita a visualização. As entidades foram derivadas dos requisitos funcionais e não funcionais elicitados.

Foram identificadas quatro entidades principais: Usuário, Conteúdo, Pergunta e Progresso, cada entidade foi detalhada com base em seus atributos, métodos e relacionamentos, garantindo que refletissem as necessidades do sistema. Os relacionamentos entre as entidades foram mapeados com base nas regras de negócio estabelecidas. Por exemplo, a relação entre Usuário e Progresso foi definida como 1:1, enquanto a relação entre Conteúdo e Pergunta foi definida como 1:N.

[4] Disponível em: https://www.lucidchart.com/pages/pt. Acesso em: 23 maio 2025

4.6.2 Regras de Negócio

As regras de negócio foram definidas como parte integrante do processo de elicitação de requisitos, visando guiar o comportamento do sistema e garantir que ele atendesse às necessidades dos usuários. As regras de negócio foram extraídas diretamente dos dados coletados por meio de questionários, entrevistas e análise de ferramentas existentes. As regras foram categorizadas em três tipos:

- Restrições: Regras que limitam o acesso ou ações no sistema (ex.: somente administradores podem gerenciar conteúdos).
- Associações: Regras que definem relacionamentos entre entidades (ex.: uma pergunta está associada a um conteúdo específico).
- Interações: Regras que descrevem como os usuários interagem com o sistema (ex.: o sistema valida se a resposta de um usuário está correta).

4.6.3 Diagramas de Casos de Uso

A modelagem dos casos de uso foi realizada para representar as interações previstas entre os atores (usuários e administradores) e o sistema. Esses diagramas foram projetados para mapear as principais funcionalidades do sistema, divididas em dois cenários principais: Gerenciamento de Usuários e Gerenciamento de Perguntas. Durante essa etapa, foram identificadas as operações específicas para cada perfil de usuário, como cadastro, log in, triagem, visualização de conteúdo e resposta a perguntas.

A criação desses diagramas seguiu boas práticas de modelagem, conforme discutido por Sommerville (2011), e teve como objetivo garantir que todas as interações fossem claramente definidas antes da implementação. Os diagramas também serviram como base para a validação das funcionalidades planejadas com os *stakeholders*.

4.7 Implementação

Esta seção descreve as etapas metodológicas adotadas para a construção da API, incluindo a escolha das tecnologias, a organização do código, os padrões de arquitetura utilizados e o processo de desenvolvimento.

4.7.1 Escolha das Tecnologias

A seleção das tecnologias foi guiada pelas necessidades funcionais e não funcionais identificadas na fase de elicitação de requisitos (Seção 4.5). Para garantir escalabilidade, segurança e facilidade de integração, optou-se pelo uso do *framework Django REST Framework* (DRF) em conjunto com *Python*, uma linguagem amplamente utilizada no desenvolvimento de sistemas educacionais (Holovaty; Kaplan-Moss, 2009).

O DRF foi escolhido devido à sua robustez e flexibilidade, características destacadas por Christie (2013), que ressalta como o DRF facilita a criação rápida de APIs *RESTful* com suporte nativo a autenticação via *token JSON Web Token (JWT)*, documentação em *Swagger* e serialização de dados. Além disso, o uso de *Python* facilita a manutenibilidade do código e a integração com ferramentas de gamificação e jogos sérios, alinhando-se aos objetivos do projeto.

Para estruturar o projeto inicial, foi utilizado o *Cookiecutter*, uma ferramenta de *scaffolding* que permite criar projetos Django pré-configurados com boas práticas de desenvolvimento (Holovaty; Kaplan-Moss, 2009). O uso do *Cookiecutter* foi baseado em estudos que destacam sua eficácia na padronização de projetos e na redução do tempo de configuração inicial (Holovaty; Kaplan-Moss, 2009). Essa abordagem garantiu que o projeto seguisse padrões bem definidos desde o início, facilitando a colaboração e a manutenção ao longo do desenvolvimento.

4.7.2 Arquitetura do Sistema

A arquitetura do sistema foi projetada seguindo o padrão (MVT), o qual é amplamente utilizado no desenvolvimento de aplicações Django (Sommerville, 2011). Este padrão promove a separação de responsabilidades entre as camadas de dados, lógica de negócios e apresentação, facilitando a manutenção e a expansão do sistema (Sommerville, 2011).

Os principais componentes da arquitetura incluem: modelos que representam as entidades principais do sistema, como Usuários, Conteúdos, Perguntas e Progressos. Cada modelo foi mapeado diretamente para tabelas no banco de dados relacional *PostgreSQL*, garantindo consistência e integridade dos dados (Date, 2003). A escolha do *PostgreSQL* foi baseada em sua capacidade de lidar com grandes volumes de dados e transações complexas, conforme discutido por Date (2003).

Views, implementam a lógica de negócios e expõem os *endpoints* da API. Foram criadas *views* baseadas em classes para aproveitar a modularidade e reutilização de código, uma prática

recomendada por Sommerville (2011). Serializadores, responsáveis pela conversão de dados entre o formato JavaScript Object Notation (JSON) e os modelos do sistema, garantindo que as operações CRUD sejam realizadas de forma eficiente e segura. A importância dos serializadores no DRF foi destacada por Christie (2013), que enfatiza sua contribuição para a simplificação do processo de manipulação de dados. Como este sistema é uma API REST pura, a camada de *Template* não foi utilizada, pois os dados são retornados diretamente no formato JSON.

4.7.3 Processo de Desenvolvimento

O desenvolvimento foi realizado de forma incremental e iterativa, seguindo os princípios do Desenvolvimento Ágil (Beck *et al.*, 2001). As funcionalidades foram divididas em *sprints*, cada uma correspondendo a um conjunto de requisitos funcionais e não funcionais previamente definidos (Seção 4.5). A cada iteração, os artefatos gerados foram revisados e ajustados conforme necessário, garantindo a conformidade com as expectativas do projeto.

Para garantir a qualidade do código, foram adotados os seguintes padrões e práticas: padrões de código seguindo as diretrizes do Style Guide for Python Code (PEP8), garantindo legibilidade e consistência (Rossum, 1991). A importância do PEP8 como padrão de estilo foi destacada por Rossum (1991), que enfatiza como ele contribui para a manutenibilidade do código.

O sistema foi versionado utilizando *Git*, com *branches* específicos para funcionalidades, correções e refatorações (Fowler, 2018). Isso permitiu um controle eficaz das alterações. A adoção do *Git* como ferramenta de versionamento foi baseada em estudos que destacam sua eficácia no gerenciamento de projetos colaborativos (Sommerville, 2011).

4.7.4 Segurança e Privacidade

A segurança foi uma preocupação central durante a implementação. Para proteger os dados dos usuários e garantir a integridade do sistema, foram adotadas as medidas discutidas a seguir.

A autenticação, que foi implementada utilizando *tokens* JWT, com *tokens* de curta duração e *refresh tokens* para maior segurança. A autorização foi configurada com base em permissões granulares, garantindo que somente usuários autorizados tenham acesso a determinadas funcionalidades. A eficácia do JWT como mecanismo de autenticação foi discutida por Fielding (2000), que destaca sua capacidade de fornecer segurança em sistemas distribuídos.

Os dados de entrada foram validados para evitar vulnerabilidades como *SQL Injection* e Cross-Site Scripting (XSS) (Sommerville, 2011). A importância da validação de dados como medida de segurança foi enfatizada por Sommerville (2011), que destaca como ela pode mitigar riscos associados a ataques cibernéticos. O sistema foi projetado para atender às normas de privacidade estabelecidas pela Lei Geral de Proteção de Dados Pessoais (LGPD), garantindo que os dados dos usuários sejam coletados, armazenados e processados de forma transparente e segura. A importância da conformidade com regulamentações de privacidade foi discutida por Bennedsen e Caspersen (2007), que enfatizam sua relevância no contexto de sistemas educacionais.

4.8 Verificação e Validação

A etapa de Verificação e Validação foi fundamental para garantir a qualidade e confiabilidade do sistema desenvolvido (Pressman; Maxim, 2016). Essa fase envolveu a realização de testes automatizados utilizando a ferramenta *pytest*, que permitiu validar as funcionalidades principais da API e identificar possíveis falhas ou inconsistências no código.

Os testes automatizados foram implementados para verificar o comportamento das funcionalidades definidas nos requisitos funcionais e não funcionais (Seção 4.5). A escolha do *pytest* como ferramenta de teste foi baseada em sua ampla adoção na comunidade Python e sua capacidade de simplificar a escrita e execução de testes unitários e de integração (Holovaty; Kaplan-Moss, 2009).

Os principais aspectos avaliados durante os testes incluíram: Validação de Entradas, foram testados todos os *endpoints* da API para garantir que os dados de entrada fossem validados corretamente, evitando vulnerabilidades como *SQL Injection* e XSS (Sommerville, 2011). Autenticação e Autorização, os testes verificaram se os *tokens* JWT estavam sendo gerados e validados corretamente, garantindo que somente usuários autorizados tivessem acesso às funcionalidades restritas.

As operações de criação, leitura, atualização e exclusão de entidades (usuários, conteúdos, perguntas, progressos) foram testadas para garantir que funcionassem conforme o esperado. A importância dos testes automatizados no desenvolvimento de software foi destacada por Sommerville (2011), que enfatiza como eles contribuem para a prevenção de regressões e a garantia da qualidade do código. Além disso, os testes automatizados permitem que novas funcionalidades sejam adicionadas ao sistema sem comprometer a estabilidade das implementações existentes.

Além dos testes automatizados, o sistema foi verificado quanto à conformidade com as normas de privacidade estabelecidas pela LGPD, garantindo que os dados dos usuários fossem coletados, armazenados e processados de forma transparente e segura (Bennedsen; Caspersen, 2007). Essa abordagem sistemática de verificação e validação buscou garantir que o sistema fosse robusto, seguro e alinhado às necessidades dos usuários finais.

5 API PARA APRENDIZADO DE LÓGICA DE PROGRAMAÇÃO

Este capítulo apresenta os resultados alcançados no desenvolvimento da API para apoio ao ensino de lógica de programação. Incluindo a implementação dos requisitos funcionais e não funcionais identificados na fase de elicitação (seção 5.1), bem como a modelagem do sistema (seção 5.3) e as funcionalidades desenvolvidas. Além disso, são apresentados os diagramas de classes e casos de uso que ilustram a estrutura e o comportamento do sistema.

5.1 Requisitos Implementados

Com base nas informações coletadas durante a fase de levantamento de requisitos, foi possível documentar e implementar os principais requisitos funcionais e não funcionais do sistema. O Quadro 2 lista os requisitos funcionais implementados.

A API desenvolvida para apoio ao ensino de lógica de programação contempla uma série de requisitos funcionais que visam garantir a sua operacionalidade e a usabilidade. Entre os principais requisitos está a autenticação de usuários, que permite o *login* (RF001) e *logout* (RF002) no sistema. O *login* é realizado mediante a inserção de credenciais válidas, como nome de usuário e senha, enquanto o *logout* encerra a sessão ativa de forma segura.

Além disso, o sistema oferece funcionalidades para o gerenciamento de perfis de usuários, incluindo o cadastro de novos alunos (RF003), atualização de dados cadastrais e exclusão de contas. Essas operações são restritas aos próprios usuários ou administradores, garantindo segurança e privacidade. A visualização de perfis é dividida em duas modalidades: pública (RF005), acessível a qualquer pessoa, e privada (RF004), disponível somente para o próprio usuário ou administradores.

Outro conjunto significativo de requisitos funcionais está relacionado ao gerenciamento de conteúdos educacionais e perguntas interativas. Administradores podem cadastrar, atualizar e excluir conteúdos, bem como criar perguntas associadas a esses conteúdos. As perguntas podem variar quanto ao nível de dificuldade, ajustadas automaticamente para alternativas “Verdadeiro” e “Falso” quando classificadas como fáceis.

Usuários autenticados têm permissão para visualizar e responder às perguntas, recebendo feedback imediato sobre suas respostas. Além disso, o sistema mantém um controle detalhado do progresso dos usuários, criando registros automáticos após o cadastro e permitindo a visualização, edição e exclusão desses registros por administradores. Essa funcionalidade é necessária para

Quadro 2 – Requisitos funcionais implementados

Código	**Descrição**
RF001	Realizar login de usuário
RF002	Realizar logout do sistema
RF003	Cadastrar aluno
RF004	Visualizar perfil privado
RF005	Visualizar perfil público
RF006	Atualizar perfil
RF007	Excluir usuário
RF008	Listar usuários
RF009	Cadastrar conteúdo
RF010	Visualizar conteúdo
RF011	Atualizar conteúdo
RF012	Excluir conteúdo
RF013	Listar conteúdos
RF014	Cadastrar pergunta
RF015	Visualizar pergunta
RF016	Atualizar pergunta
RF017	Excluir pergunta
RF018	Listar perguntas
RF019	Responder pergunta
RF020	Criar progresso
RF021	Visualizar progresso
RF022	Editar progresso
RF023	Excluir progresso
RF024	Listar progresso
RF025	Criar triagem
RF026	Visualizar triagem
RF027	Excluir triagem

Fonte: Autoria própria (2025)

acompanhar o desempenho dos alunos e personalizar a experiência de aprendizado.

Além dos requisitos funcionais, foram definidos requisitos não funcionais para garantir a qualidade e usabilidade do sistema (Quadro 3).

Quadro 3 – Requisitos não funcionais implementados

Código	**Descrição**
RNF001	Autenticação via *tokens* JWT com *tokens* de curta duração e *refresh tokens*
RNF002	Validação de todos os dados de entrada para evitar SQL *Injection* ou XSS
RNF003	Conformidade com regulamentações de privacidade seguindo as normas LGPD
RNF004	Documentação da API conforme padrões como *Swagger* para facilitar a integração
RNF005	Respostas de erro claras e seguindo códigos HTTP padrão
RNF006	Código seguindo padrões de estilo PEP8
RNF007	Controle de versão da API para evitar que mudanças quebrem clientes existentes

Fonte: Autoria própria (2025)

Os requisitos não funcionais também desempenham um papel fundamental na qualidade e confiabilidade da API. Um dos aspectos mais críticos é a segurança, abordada por meio da implementação de autenticação via *tokens* JWT, com tokens de curta duração e *refresh tokens* para garantir maior proteção contra acessos não autorizados (RNF001).

Além disso, todos os dados de entrada são validados rigorosamente para evitar vulnerabilidades como SQL *Injection* e XSS, alinhando-se às melhores práticas de desenvolvimento seguro. Outro requisito relevante é a conformidade com regulamentações de privacidade, como a LGPD, assegurando que os dados dos usuários sejam tratados de maneira ética e transparente. A usabilidade também foi priorizada, com uma documentação clara e padronizada seguindo o formato *Swagger*, facilitando a integração com outras aplicações.

Finalmente, o código foi desenvolvido seguindo o padrão PEP8, promovendo consistência e legibilidade, enquanto a API foi versionada para evitar impactos em clientes existentes durante atualizações futuras. Esses requisitos não funcionais garantem que o sistema seja escalável, seguro e fácil de manter, atendendo tanto às necessidades técnicas quanto às expectativas dos usuários finais.

5.2 Regras de Negócio

As regras de negócio definem as restrições e interações fundamentais que regem o comportamento do sistema, garantindo que as funcionalidades estejam alinhadas aos objetivos educacionais e às necessidades dos usuários. Entre as principais regras implementadas, destacam-se: a associação entre perguntas e conteúdos, que estabelece a relação hierárquica entre essas entidades; a personalização das alternativas para perguntas de nível fácil, ajustando-as automaticamente para “Verdadeiro” ou “Falso”; e o controle de acesso, que restringe operações críticas, como criação, atualização e exclusão de conteúdos e perguntas, somente aos administradores.

Além disso, o sistema valida as respostas dos usuários em tempo real, dando feedback imediato sobre o desempenho, e garantindo que informações sensíveis, como a alternativa correta de uma pergunta, sejam exibidas somente para administradores. Essas regras são essenciais para manter a integridade, segurança e usabilidade do sistema. O Quadro 4 apresenta as principais regras de negócio implementadas, acompanhadas de seus respectivos códigos de identificação.

Quadro 4 – Regras de Negócio

Código	Regra de Negócio
RN001	Uma pergunta possui um conteúdo
RN002	Um conteúdo pode possuir 0 ou N perguntas
RN003	Um usuário autenticado pode responder a uma pergunta
RN004	Uma pergunta de nível fácil deve ter alternativas ajustadas para “Verdadeiro” e “Falso”
RN005	A alternativa correta de uma pergunta não deve ser exibida para usuários comuns
RN006	Somente administradores podem criar, atualizar ou excluir conteúdos
RN007	Somente administradores podem criar, atualizar ou excluir perguntas
RN008	Qualquer usuário autenticado pode visualizar conteúdos e perguntas
RN009	O sistema valida se a resposta de um usuário está correta

Fonte: Autoria própria (2025)

5.3 Modelagem do Sistema

A modelagem do sistema foi realizada utilizando diagramas de classes e casos de uso, que ajudaram a estruturar as funcionalidades e os relacionamentos entre as entidades.

5.3.1 Diagrama de Classes

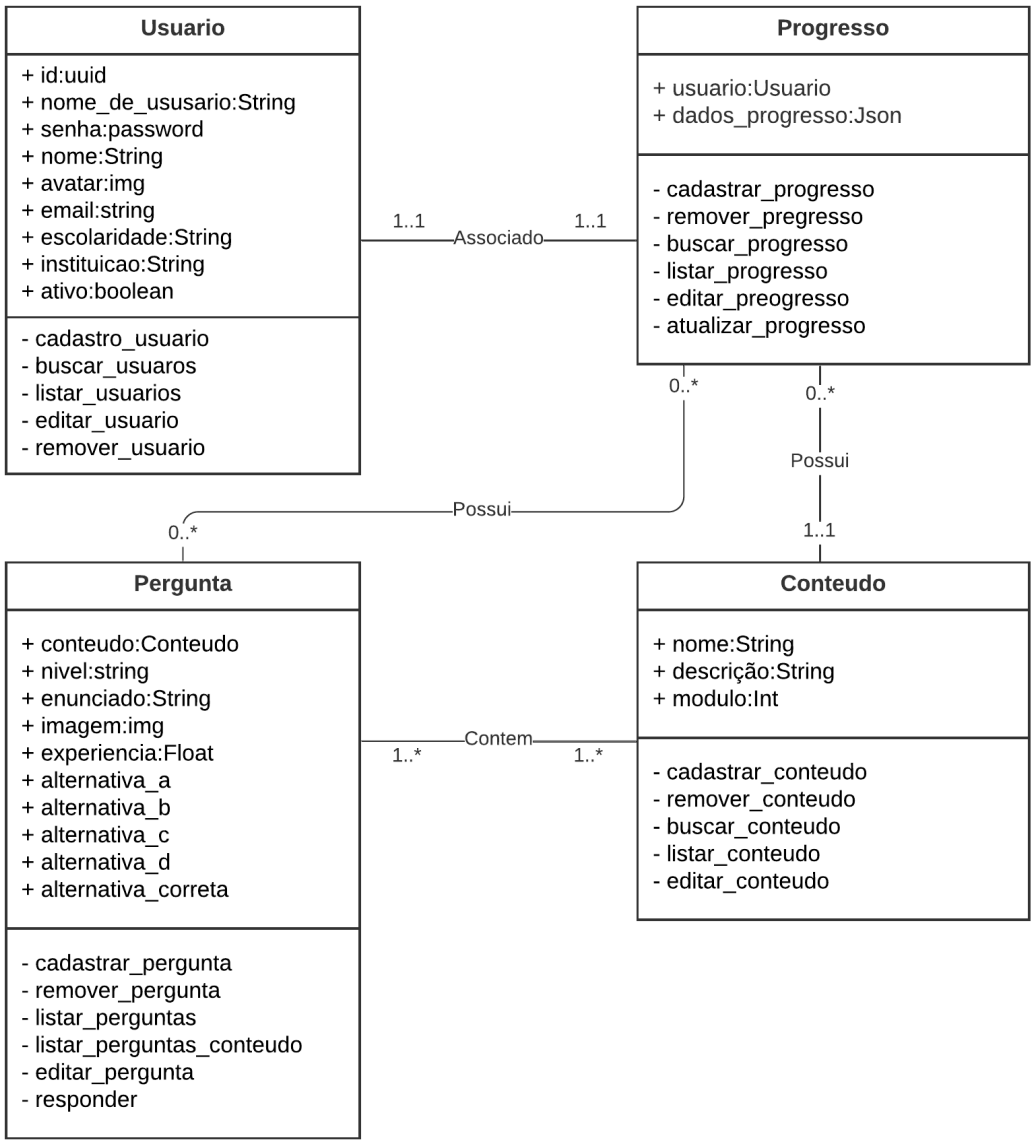
O diagrama de classes (Figura 1) ilustra as principais entidades do sistema e seus relacionamentos. As entidades incluem: Usuário, que representa os perfis de alunos e professores; Conteúdo, que se refere aos módulos educacionais cadastrados no sistema; Pergunta, associada a um conteúdo específico para avaliação dos alunos; Progresso, que armazena o avanço dos usuários em relação aos conteúdos estudados; e Triagem, responsável por registrar informações iniciais sobre o nível de conhecimento do usuário após o cadastro.

A entidade *Usuário* possui atributos como **nome**, **avatar** e **instituição**, além de campos opcionais como **data de nascimento** e **escolaridade**. Essa entidade é fundamental para gerenciar perfis de alunos e professores, permitindo funcionalidades como autenticação, atualização de dados e exclusão de contas. A interação entre usuários e outras entidades, como progresso e triagem, permite um acompanhamento personalizado do aprendizado.

Os conteúdos são representados por atributos como **nome**, **descrição** e **módulo**, organizando o material educacional em categorias claras e acessíveis. Cada conteúdo pode estar associado a múltiplas perguntas, facilitando a criação de atividades avaliativas contextualizadas. Essa estrutura modular permite que os administradores gerenciem dinamicamente os materiais disponíveis no sistema.

As perguntas estão vinculadas a um conteúdo específico e possuem atributos como

Figura 1 – Diagrama de Classes



Fonte: Autoria própria (2025)

enunciado, **nível de dificuldade** e **alternativas**. Para perguntas de nível fácil, as alternativas são automaticamente ajustadas para “Verdadeiro” ou “Falso”. Essa abordagem simplifica a interação dos usuários com questões básicas, enquanto mantém a complexidade para níveis mais avançados. A validação das respostas é realizada pelo sistema, dando feedback imediato aos alunos.

O progresso armazena informações sobre o desempenho dos usuários, como o **nível atual**, a **experiência acumulada** (XP) e as **perguntas respondidas**. Essa entidade permite que os alunos acompanhem seu desenvolvimento ao longo do tempo, enquanto os administradores podem visualizar e gerenciar os progressos de forma centralizada. A exclusão automática do

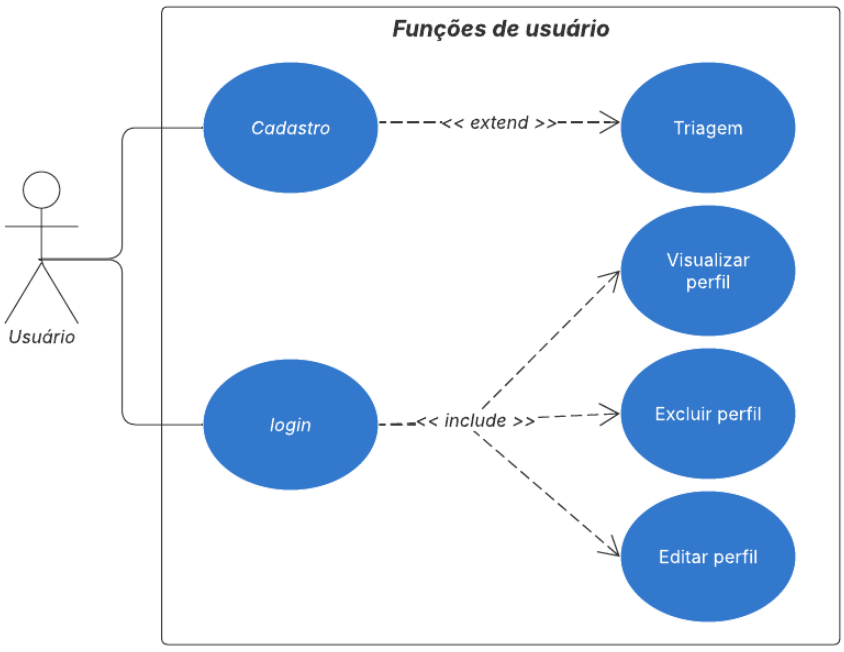
progresso ocorre quando um usuário tem sua conta removida do sistema.

Por fim, a entidade triagem registra informações iniciais sobre o nível de conhecimento do usuário logo após o cadastro. Esses dados são utilizados para personalizar o conteúdo apresentado ao aluno, permitindo uma experiência de aprendizado adaptada às suas necessidades. A triagem é excluída automaticamente quando o usuário é removido do sistema, mantendo a consistência dos dados.

5.3.2 Diagramas de Casos de Uso

Os diagramas de casos de uso descrevem as interações entre os atores (usuários e administradores) e o sistema. Os principais cenários incluem: gerenciamento de usuários e gerenciamento de perguntas. O gerenciamento de usuários inclui operações como cadastro, atualização e exclusão de perfis de usuários (Figura 2). Um ponto importante é que, com exceção de cadastro e triagem, todas as operações requerem que o usuário esteja logado. A triagem realizada após o cadastro permite identificar o nível de conhecimento do usuário, possibilitando a personalização do conteúdo oferecido.

Figura 2 – Diagrama de Casos de Uso - Usuário

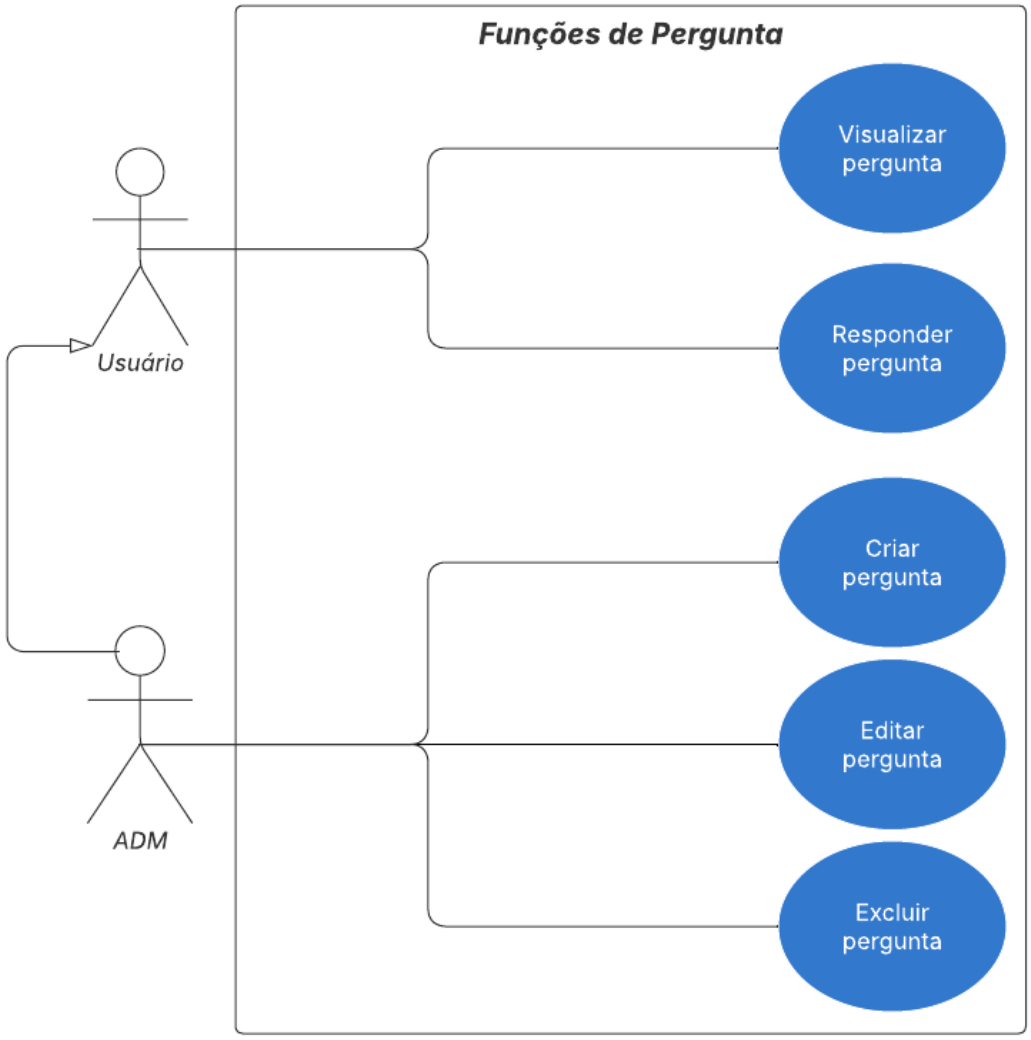


Fonte: Autoria própria (2025)

Já o gerenciamento de perguntas descreve as operações relacionadas ao cadastro, visualização, resposta, atualização e exclusão de perguntas (Figura 3). Esse caso de uso ilustra bem a

principal diferença entre os tipos de usuário, no qual as operações mais gerenciais (cadastro, edição e exclusão) são de responsabilidade somente do administrador, restando ao usuário comum operações mais interativas (visualização e resposta).

Figura 3 – Diagrama de Casos de Uso - Perguntas



Fonte: Autoria própria (2025)

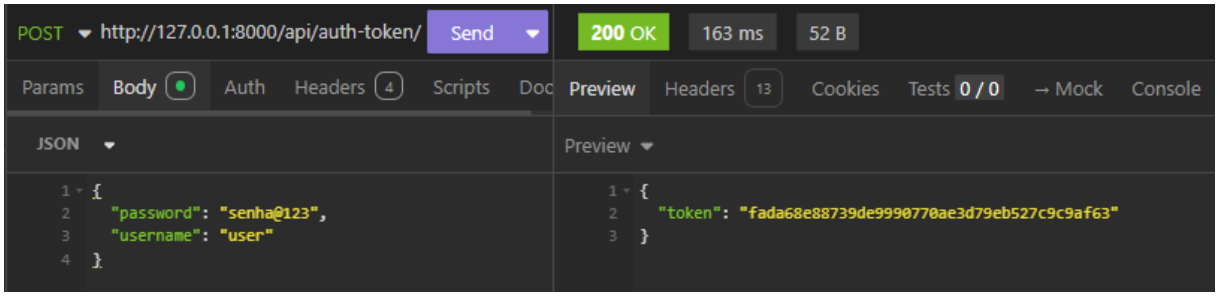
5.4 Funcionalidades Implementadas

As funcionalidades implementadas no sistema são descritas a seguir. A autenticação e autorização desempenham um papel central na segurança do sistema. Foram desenvolvidos *endpoints* específicos para suportar operações de autenticação e autorização, garantindo controle de acesso seguro ao sistema. O processo de log in permite que os usuários validem suas

credenciais, como nome de usuário e senha, e recebam *tokens* JWT para sessões autenticadas.

Além disso, o sistema oferece um *endpoint* para log out, que invalida a sessão ativa do usuário. A API valida todos os *tokens* de acesso recebidos em requisições restritas e suporta a renovação de *tokens* por meio de *refresh tokens*, assegurando uma experiência segura e fluida para os usuários. Na Figura 4 pode-se visualizar a execução do log in e *token* JWT retornado.

Figura 4 – Log in de usuário com geração e token



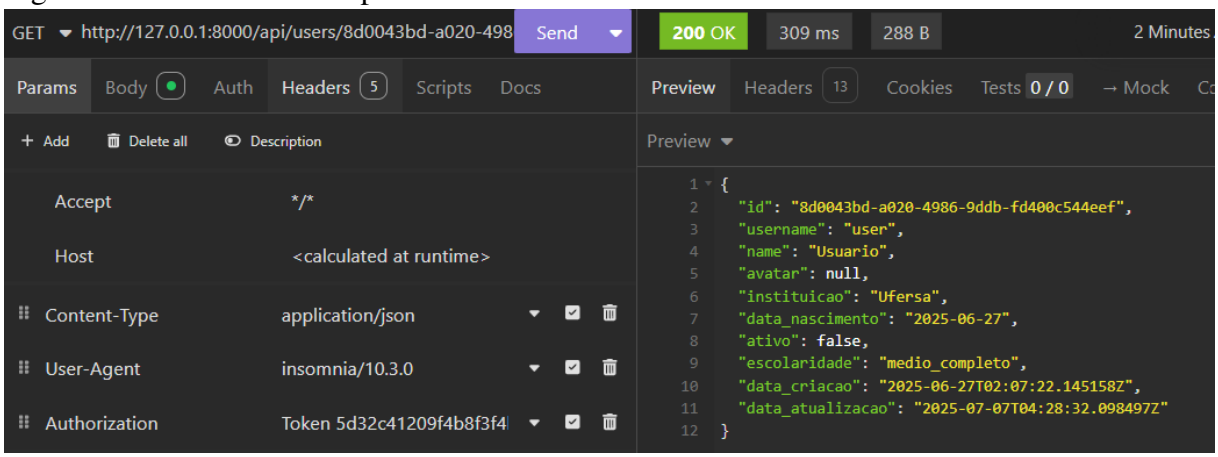
Fonte: Autoria própria (2025)

O gerenciamento de perfis de usuário é outra funcionalidade importante da API. O sistema oferece recursos completos para o cadastro, visualização, atualização e exclusão de perfis de usuários. Durante o cadastro, os usuários podem informar dados obrigatórios, como nome de usuário, senha e nome completo, além de campos opcionais, como avatar, instituição, data de nascimento e escolaridade. Após o cadastro, os usuários podem visualizar seus perfis privados ou públicos, sendo que somente o próprio usuário ou administradores têm acesso aos detalhes do perfil privado.

A listagem de usuários é restrita a administradores, que também têm permissão para realizar operações gerenciais, como edição ou exclusão de perfis. Nas Figuras 5 e 6 pode-se observar a diferença dos dados evidenciados dependendo de qual usuário tenha realizado log in, na Figura 5 tem-se todos os dados que compõem as informações do usuário sendo exibidas, isso acontece, pois o *endpoint* foi executado utilizando as credenciais do perfil que possui essas informações, já na Figura 6 tem-se a demonstração do processo contrário, um usuário buscando o perfil de outro, observe que os dados exibidos foram limitados, visando garantir que informações pessoais não sejam exibidas; outro ponto a se destacar é que em ambos os casos a senha não é exibida, visto que além de não ser uma boa prática também é não permitido pela LGPD.

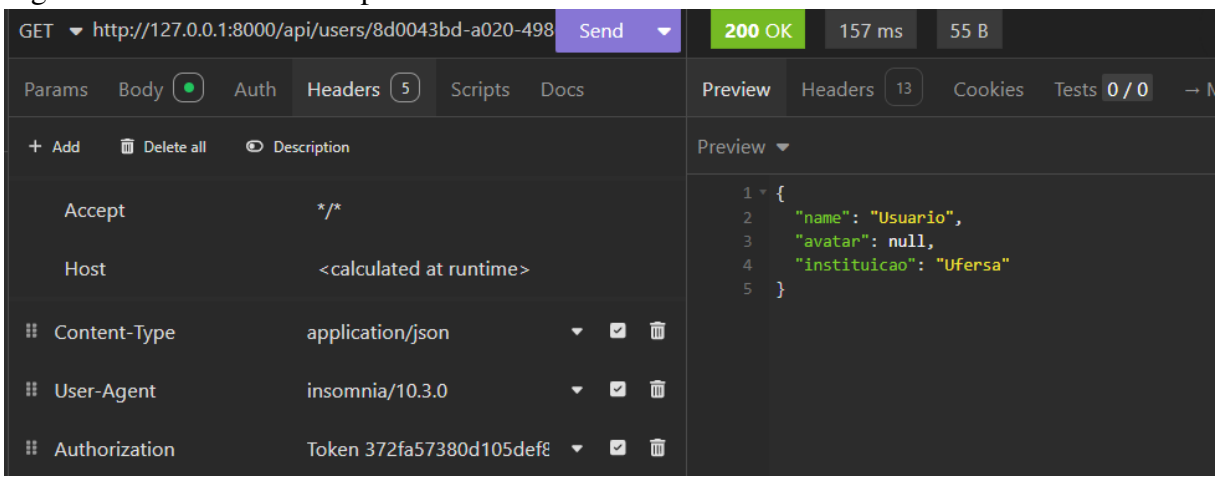
Já o gerenciamento de conteúdo educacional é uma das funcionalidades mais importantes da API. A API disponibiliza *endpoints* para o cadastro, atualização, exclusão e visualização de conteúdos educacionais. Administradores podem cadastrar novos conteúdos, especificando

Figura 5 – Perfil de usuário privado



Fonte: Autoria própria (2025)

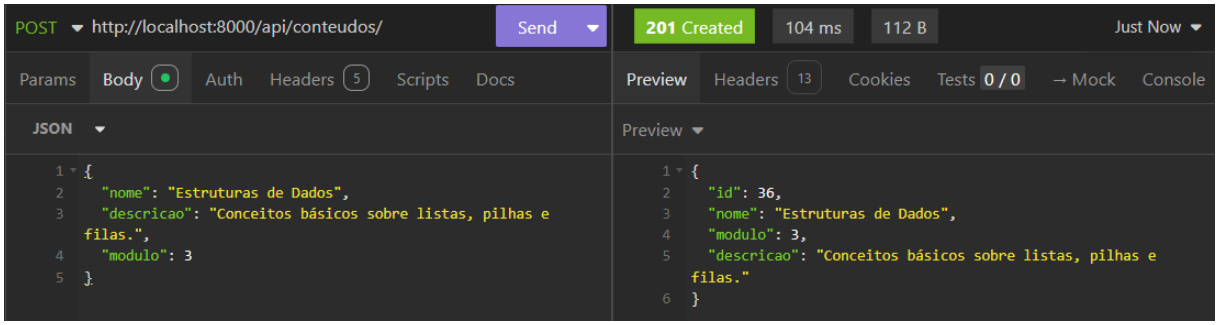
Figura 6 – Perfil de usuário público



Fonte: Autoria própria (2025)

nome, descrição e módulo, enquanto usuários autenticados podem visualizar detalhes dos conteúdos e listar todos os conteúdos disponíveis. Além disso, administradores têm permissão para atualizar ou excluir conteúdos existentes, garantindo flexibilidade no gerenciamento do material educacional. Na Figura 7 há um exemplo da criação de um conteúdo.

Figura 7 – Criação de conteúdo



Fonte: Autoria própria (2025)

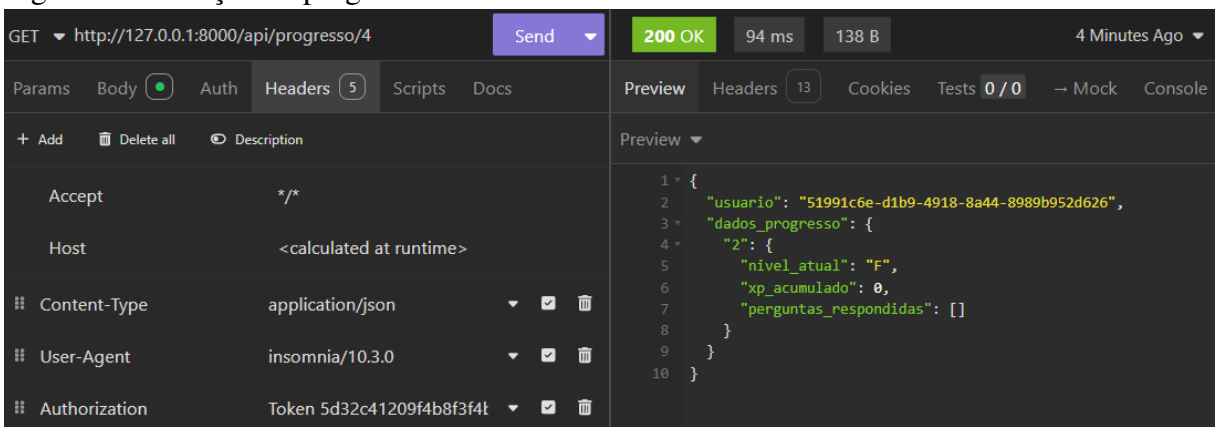
O sistema também inclui funcionalidades robustas para o gerenciamento de perguntas associadas a conteúdos específicos. Administradores podem cadastrar novas perguntas, definindo enunciado, nível, alternativas e alternativa correta. Para perguntas de nível fácil, as alternativas são automaticamente ajustadas para “Verdadeiro” e “Falso”. Usuários autenticados podem visualizar perguntas, respondê-las e receber feedback imediato sobre suas respostas.

A alternativa correta não é incluída no JSON retornado para usuários comuns, evitando possíveis tentativas de descobrir a resposta mediante essa URL. A alternativa correta é enviada somente para administradores, enquanto usuários normais recebem o resultado de sua resposta por meio de outro *endpoint*, que verifica se a resposta está correta e retorna o resultado. Além disso, administradores podem atualizar, excluir ou listar todas as perguntas cadastradas.

Por fim, o controle de progresso permite acompanhar o desempenho dos usuários. Após o cadastro de um usuário, o sistema cria automaticamente registros de progresso para cada conteúdo disponível. Usuários autenticados e administradores podem visualizar o progresso, enquanto somente administradores têm permissão para editar ou excluir registros de progresso.

O sistema também permite listar todos os progressos de um usuário, exibindo detalhes sobre seu desempenho em cada conteúdo. Essa funcionalidade é essencial para fornecer uma visão clara do avanço dos alunos e apoiar o processo de aprendizado. Na Figura 8 há um exemplo de exibição de progresso de usuário.

Figura 8 – Exibição de progresso



Fonte: Autoria própria (2025)

5.5 Validação e Testes

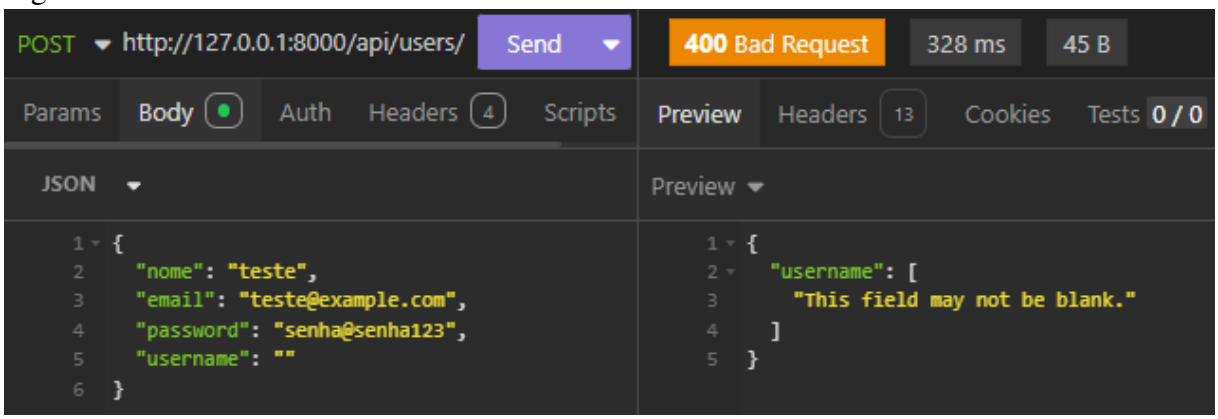
Para garantir a qualidade do sistema, foram realizados testes automatizados utilizando a ferramenta *pytest*, amplamente adotada em projetos Python por sua simplicidade e eficácia

(Team, 2023). Os testes abrangem diferentes níveis de validação, incluindo testes unitários, de integração e funcionais.

5.5.1 Testes Unitários

Os testes unitários foram desenvolvidos para verificar o comportamento isolado de métodos e funções específicas, como validações de modelos, serializadores e lógica de negócios. Entre os exemplos mais relevantes estão, validação de entrada de dados, que garante que os dados fornecidos pelos usuários estejam nos formatos esperados e evitar vulnerabilidades como *SQL Injection* ou *XSS*.

Figura 9 – Cadastro de usuário



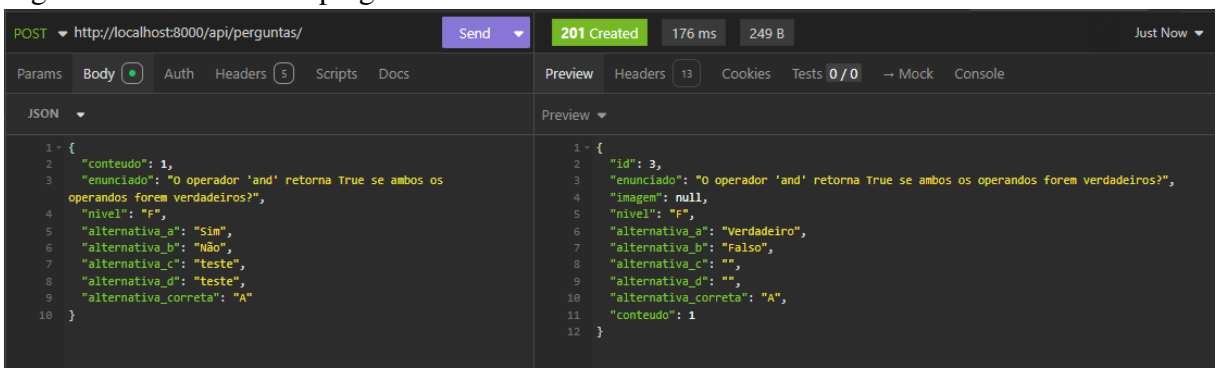
Fonte: Autoria própria (2025)

A Figura 9 ilustra a validação implementada pela API ao lidar com entradas de dados. Ao receber uma requisição com campos obrigatórios ausentes, o sistema responde com um código de status 400 *Bad Request* e uma mensagem de erro orientando o usuário sobre o problema. Essa abordagem não somente garante que os dados fornecidos estejam nos formatos esperados, mas também contribui para a segurança do sistema, mitigando riscos como SQL *Injection* ou *XSS*. Além disso, a estrutura padronizada das mensagens de erro facilita a interpretação por parte dos desenvolvedores e usuários, promovendo uma experiência mais fluida e eficiente durante a interação com a API.

Além disso, foram implementadas verificação de regras de negócio, por exemplo, ajuste automático das alternativas para “Verdadeiro” e “Falso” em perguntas de nível fácil (Figura 10).

A Figura 10 demonstra que ao definir o nível fácil no momento do cadastro da pergunta, as alternativas “c” e “d” são automaticamente definidas como uma *strings* vazias (“”), independente do valor atribuído, respeitando a regra de negócio definida.

Figura 10 – Cadastro de pergunta nível fácil

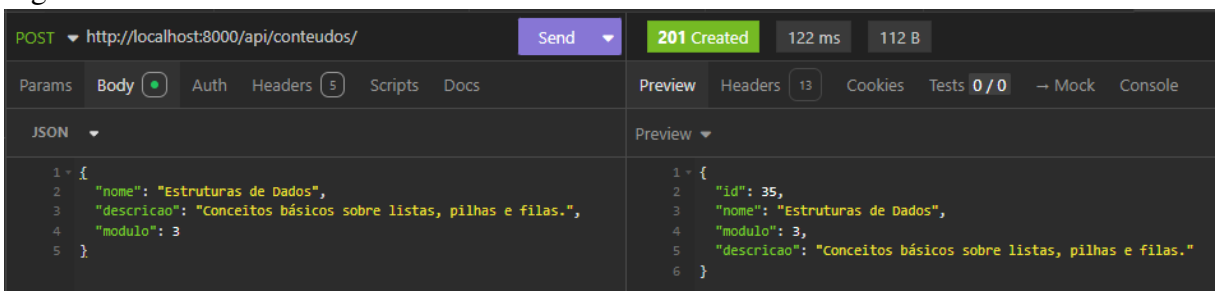


Fonte: Autoria própria (2025)

5.5.2 Testes de Integração

Os testes de integração foram projetados para validar a interação entre diferentes componentes do sistema, como o sistema de autenticação (*allauth*), o banco de dados (PostgreSQL/SQLite) e o framework REST. Um exemplo significativo é a verificação de autenticação e autorização, que valida se somente administradores podem criar, atualizar ou excluir conteúdos e perguntas (RF009, RF011, RF012, RF014, RF016, RF017).

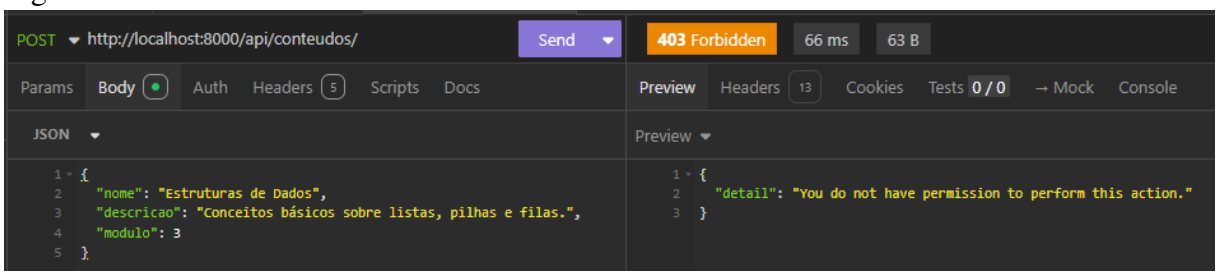
Figura 11 – Cadastro de conteúdo com usuário admin



Fonte: Autoria própria (2025)

A Figura 11 demonstra o cadastro de um conteúdo utilizando a autenticação de um perfil de administrador. Já a Figura 12, também demonstra o cadastro de um conteúdo, porém, com credenciais de autenticação de um perfil de um usuário comum.

Figura 12 – Tentativa de adastro de conteúdo com usuário comum



Fonte: Autoria própria (2025)

Esses exemplos demonstram como a API implementa as regras de negócio relacionadas à autenticação e autorização, garantindo que somente administradores possam realizar operações críticas, como a criação, atualização ou exclusão de conteúdos e perguntas. Além disso, ele ilustra a robustez do sistema ao retornar mensagens de erro claras e códigos HTTP apropriados, conforme especificado nos requisitos não-funcionais (RNF002).

Outro aspecto testado foi o gerenciamento de progresso, incluindo a criação automática de progresso após o cadastro de um usuário (RF020) e a exclusão de progresso ao deletar um usuário (RF023). Além disso, testou-se se as respostas às perguntas atualizam corretamente o progresso do usuário (RF019), demonstrando a integração eficaz entre os módulos.

5.5.3 Testes Funcionais/API

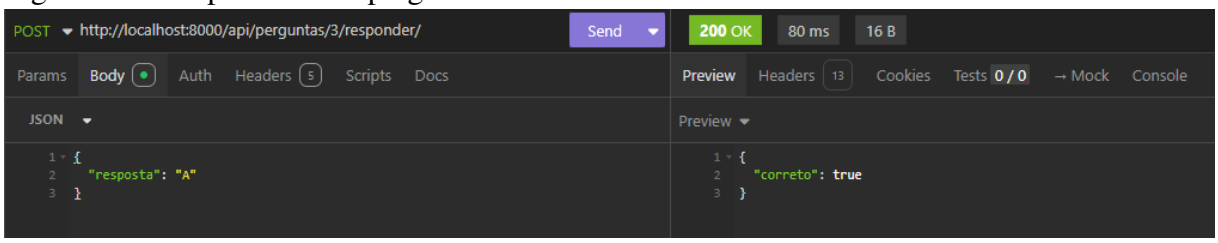
Os testes funcionais simularam requisições HTTP para os *endpoints* da API, validando o comportamento do sistema em cenários reais. Foram utilizadas ferramentas como *pytest-django* e *Insomnia* para realizar esses testes.

Entre os casos testados estão os *endpoints* de autenticação, que validam o log in (RF001), log out (RF002), como a Figura 4, que demonstra o processo de log in, no qual após a submissão do nome de usuário e senha, gera o *token* que dá acesso ao sistema.

Foi testado o gerenciamento de usuários, incluindo o cadastro de novos usuários (RF003), visualização de perfis públicos e privados (RF004, RF005), e exclusão de contas (RF007). Outros testes incluíram a listagem e filtragem de conteúdos (RF013), perguntas (RF018) e progressos (RF024), além da filtragem por dificuldade ou conteúdo específico. Por fim, foi verificado se o sistema valida corretamente as respostas dos usuários e retorna feedback imediato (RF019).

Para garantir que usuários comuns não possam executar o *endpoint* com o intuito de descobrir respostas, o mesmo possui restrição de uso, permitindo que apenas administradores executem, logo, o fluxo para execução funciona da seguinte maneira, o usuário com qualquer credencial envia sua resposta, o *front* recebe e executa o *endpoint* de verificação utilizando credenciais de administrador. A Figura 13 demonstra execução do *endpoint* que retorna se a resposta enviada está correta, a pergunta em questão é passada na URL, através de seu Identity (ID); nesse exemplo, seria a pergunta de ID 3.

Figura 13 – Resposta a uma pergunta



Fonte: Autoria própria (2025)

5.5.4 Resultados dos Testes

Os testes indicaram que o sistema atende aos requisitos funcionais e não funcionais definidos, com alta cobertura de código e ausência de falhas críticas. Alguns destaques incluem:

- **Cobertura de código**: A execução dos testes resultou em uma cobertura superior a 90%, garantindo que a maioria dos cenários foi validada.
- **Ausência de bugs críticos**: não foram identificados bugs críticos ou bloqueadores que comprometessem a funcionalidade do sistema.
- **Segurança**: Os testes de autenticação e validação de entrada demonstraram que o sistema está protegido contra vulnerabilidades comuns, como SQL *Injection* e XSS.
- **Conformidade com as regras de negócio**: os testes validaram o funcionamento de regras como restrições de acesso, estrutura das perguntas e operações de CRUD baseadas em perfis de usuário.
- **Estabilidade funcional**: o sistema respondeu consistentemente durante os testes automatizados, garantindo confiabilidade nos principais fluxos de uso.

5.5.5 Ambiente de Testes

O ambiente de testes foi configurado conforme descrito no Plano de Testes (Apêndice C), com as características apresentadas no Quadro 5.

Quadro 5 – Ambiente de testes do sistema

Sistema Operacional	Windows
Framework Back-End	Django 4.x
Banco de Dados	PostgreSQL (produção) e SQLite (testes)
Ferramentas de Teste	*pytest*, *pytest-django*
Dados de Teste	Banco de dados SQLite temporário (*test_db.sqlite3*) utilizado para simular operações sem impactar o ambiente de produção

Fonte: Autoria própria (2025)

6 CONCLUSÕES E TRABALHOS FUTUROS

O desenvolvimento da API para apoio ao processo de aprendizagem de lógica de programação tem potencial para auxiliar no suporte ao ensino e à prática de conceitos fundamentais de programação. A API foi projetada para oferecer funcionalidades essenciais, como gerenciamento de perfis de usuários, acompanhamento de progresso, criação e consulta de conteúdos educacionais, e interações com perguntas gamificadas. Ao integrar elementos de gamificação e permitir a personalização de experiências de aprendizado, a API busca atender tanto alunos quanto educadores, proporcionando uma base para o desenvolvimento de ferramentas educacionais mais dinâmicas e interativas.

Um dos principais resultados alcançados foi a implementação de um sistema robusto de autenticação e autorização, utilizando *tokens* JWT, que garante segurança e controle de acesso às funcionalidades restritas. Além disso, a API foi desenvolvida seguindo boas práticas de codificação (PEP8) e padrões de documentação, como *Swagger*, facilitando sua integração com outras aplicações. Esses aspectos contribuem para a escalabilidade e manutenibilidade da solução, tornando-a adequada para diferentes contextos educacionais.

Outro ponto relevante foi a incorporação de regras de negócio específicas, como o ajuste automático de alternativas para “Verdadeiro” e “Falso” em perguntas de nível fácil, bem como a validação rigorosa de respostas e restrições de acesso a informações sensíveis, como a alternativa correta das perguntas. Essas características visam promover uma experiência de aprendizado mais fluida e adaptada ao nível de conhecimento do aluno, enquanto garantem a integridade dos dados e a segurança do sistema.

Apesar dos resultados positivos obtidos durante o desenvolvimento e os testes realizados, é importante reconhecer que a efetividade da API no contexto educacional depende de sua aplicação prática em ambientes reais. A API de fato pode ser uma ferramenta útil para engajar estudantes e facilitar o acompanhamento de seu progresso, mas estudos mais abrangentes são necessários para validar seu impacto no aprendizado de lógica de programação. Além disso, a adoção de metodologias ativas, como a gamificação e o uso de jogos sérios, pode ampliar ainda mais o potencial da API como recurso pedagógico.

Mas, ainda há espaço para melhorias e extensões que podem ampliar o impacto e a eficácia da API. Entre as principais propostas para trabalhos futuros estão:

1. Melhoria no Sistema de Progresso com Inteligência Artificial

   Uma das principais áreas de melhoria é a implementação de um sistema de progresso mais

inteligente, utilizando técnicas de Inteligencia artificial (IA). Uma abordagem promissora seria o uso de IA preditiva, que pode analisar os padrões de respostas dos usuários e identificar suas dificuldades específicas. Com base nessa análise, a IA poderia sugerir ajustes no nível de dificuldade ou recomendar a revisão de módulos anteriores para reforçar a compreensão dos conceitos básicos.

Por exemplo, se um aluno apresenta um alto índice de erros em perguntas relacionadas a um determinado conteúdo (e.g. recursão), a IA poderia automaticamente ajustar seu nível atual para um estágio anterior, incentivando uma revisão gradual e estruturada. Esse tipo de abordagem é preferível a métodos tradicionais de avaliação, ao permitir uma análise mais granular e personalizada do desempenho do aluno, considerando nuances que seriam difíceis de capturar manualmente.

2. Melhoria no Sistema de Triagem

   Outro ponto de destaque para trabalhos futuros é a otimização do sistema de triagem. Atualmente, a triagem é responsável por avaliar o nível inicial de conhecimento do usuário em relação aos conteúdos disponíveis. No entanto, essa funcionalidade pode ser aprimorada para fornecer uma análise mais detalhada e precisa do perfil do aluno.

   Isso pode ser alcançado, por exemplo, utilizando algoritmos de IA generativa para criar perguntas adaptativas que se ajustam dinamicamente ao nível de conhecimento do usuário durante o processo de triagem. Essa abordagem não somente melhora a precisão da alocação inicial, mas também garante que o aluno seja direcionado para conteúdos que correspondam adequadamente ao seu nível de habilidade.

3. Adição de uma Classe de Configuração

   Para aumentar a flexibilidade no consumo da API, propõe-se implementar uma classe de configuração que permita personalizar o comportamento da API conforme as necessidades do sistema consumidor. Essa classe poderia definir se a triagem será obrigatória, opcional ou completamente desativada. Outro exemplo, seria a possibilidade de configurar o fluxo de aprendizado, permitindo que o usuário inicie diretamente em um conteúdo intermediário ou avance linearmente, começando pelo primeiro módulo. Essas opções trariam maior adaptabilidade à API, tornando-a mais versátil para diferentes cenários de uso.

4. Inclusão de Novos Elementos de Gamificação

   Embora a API já incorpore elementos básicos de gamificação, como o sistema de progresso baseado em XP, há oportunidades para expandir essas funcionalidades. Por exemplo, pode-

riam ser adicionados sistemas de níveis de dificuldade dinâmicos, desafios temporizados, medalhas de conquista e classificações globais. Esses elementos podem aumentar o engajamento dos usuários e motivá-los a continuar seus estudos, criando uma experiência de aprendizado mais envolvente e divertida.

A API desenvolvida tem o potencial de ser uma solução viável para apoiar o ensino de lógica de programação. Ela atende às necessidades identificadas na literatura, como a falta de engajamento e a dificuldade de aplicar conceitos teóricos, ao mesmo tempo que oferece uma base flexível para integração com outras ferramentas educacionais. Os trabalhos futuros propostos visam superar as limitações atuais e ampliar o escopo da API, incorporando tecnologias emergentes, como IA, e recursos adicionais de personalização e gamificação. Essas melhorias têm o potencial de transformar a API em uma ferramenta poderosa para o suporte ao ensino de programação, beneficiando tanto alunos quanto educadores em sua jornada de aprendizado.

REFERÊNCIAS

Alice Project. **Alice: An easy to use virtual environment creation and animation tool**. 2025. Disponível em: https://www.alice.org. Acesso em: 23 maio 2025.

ALRUBAYE, H. *et al.* Investigating the challenges in learning and teaching programming: Insights from a study. **Journal of Computer Education**, MECS Publisher, v. 21, n. 3, p. 123–135, 2013.

ANDRADE, D. L. A. **Análise da evasão nos cursos da ufersa câmpus pau dos ferros: Um estudo de tendências e causas**. Monografia (Bacharel em Interdisciplinar em Tecnologia da Informação) — Universidade Federal Rural do Semi-Árido, 2023.

APONTE, M. V.; TEMPEL, T.; CARTER, E. J. Gamification, personalization and motivational affordances: Revisiting motivation through the lens of gamification. **Entertainment Computing**, v. 5, p. 59–66, 2014.

BECK, K.; GRENNING, J.; MARTIN, R. C. *et al.* **Manifesto for agile software development**. 2001. Disponível em: https://agilemanifesto.org/. Acesso em: 10 jul. 2025.

BENNEDSEN, J.; CASPERSEN, M. E. Failure rates in introductory programming. In: **ACM SIGCSE Bulletin**. [S.l.: s.n.], 2007. v. 39, n. 2, p. 32–36.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The unified modeling language user guide**. 2. ed. [S.l.]: Addison-Wesley, 2005. ISBN 978-0321267979.

BURKE, B. **Gamify: How gamification motivates people to do extraordinary things**. [S.l.]: Bibliomotion, Inc., 2014.

CALDERóN, A.; RUIZ, D.; PéREZ, M. Initiatives and challenges of using gamification in software engineering: A systematic mapping. **Journal of Systems and Software**, Elsevier, v. 146, p. 124–137, 2018.

CARVALHO, E.; BRANDãO, T.; PAIVA, A. Can gamification help in software testing education? findings from an empirical study. **Journal of Educational Technology & Society**, International Forum of Educational Technology & Society, v. 20, n. 1, p. 26–37, 2017.

CHRISTIE, T. **Django REST framework documentation**. GitHub, 2013. Disponível em: https://www.django-rest-framework.org/. Acesso em: 10 jul. 2025.

COCKBURN, A. **Writing effective use cases**. [S.l.]: Addison-Wesley, 2000. ISBN 978-0201702255.

Codecademy. **Codecademy: Learn to code interactively**. 2025. Disponível em: https://www.codecademy.com. Acesso em: 23 maio 2025.

CRESWELL, J. W. **Research design: Qualitative, quantitative, and mixed methods approaches**. 4. ed. [S.l.]: SAGE Publications, 2014.

DATE, C. J. **An introduction to database systems**. [S.l.]: Addison-Wesley, 2003.

DETERDING, S.; DIXON, D.; KHALED, R.; NACKE, L. From game design elements to gamefulness: Defining "gamification". In: **Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments**. [S.l.]: ACM, 2011. p. 9–15.

FERNANDES, M.; SOUSA, L.; SILVA, F. Gamification in software engineering education. In: IEEE. **Proceedings of the 25th International Conference on Software Engineering Education and Training (CSEET)**. [S.l.], 2012. p. 42–49.

FESSAKIS, G.; GOULI, E.; MAVROUDI, A. Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. **Computers & Education**, v. 63, p. 87–97, 2013.

FIELDING, R. T. Architectural styles and the design of network-based software architectures. **Doctoral dissertation, University of California, Irvine**, 2000. Disponível em: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf.

FOWLER, M. **UML distilled: A brief guide to the standard object modeling language**. 3. ed. [S.l.]: Addison-Wesley, 2003. ISBN 978-0321193681.

FOWLER, M. **Refactoring: Improving the design of existing code**. [S.l.]: Addison-Wesley, 2018.

GIL, A. C. **Métodos e técnicas de pesquisa social**. 6. ed. [S.l.]: Atlas, 2008.

GOMES, A.; MENDES, A. J. Learning to program - difficulties and solutions. In: **International Conference on Engineering Education - ICEE**. [S.l.: s.n.], 2007.

GUO, P. *et al.* Active learning in computing education: Evidence from research. **ACM Transactions on Computing Education**, v. 14, n. 2, p. 10–25, 2014.

HOLOVATY, A.; KAPLAN-MOSS, J. **The definitive guide to django: Web development done right**. [S.l.]: Apress, 2009.

INEP. **Taxas de evasão em cursos de computação no brasil**. 2023. Disponível em: http://www.inep.gov.br. Acesso em: 25 jul. 2025.

JENKINS, T. On the difficulty of learning to program. **Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences**, 2002.

Khan Academy. **Khan academy: Free online courses and lessons**. 2025. Disponível em: https://pt.khanacademy.org. Acesso em: 23 maio 2025.

LUXTON-REILLY, A. *et al.* Blended learning approaches for programming education: A study. **Computer Science Education**, v. 26, n. 1, p. 1–16, 2016.

MALHOTRA, N. K. **Marketing research: An applied orientation**. 6. ed. [S.l.]: Prentice Hall, 2010.

MANNILA, L. *et al.* Challenges in learning programming and how they are overcome: A case study. In: **Proceedings of the 2006 Conference on Information Technology Education**. [S.l.: s.n.], 2006. p. 161–166.

MARCELINO, M. J.; PESSOA, T.; VIEIRA, C.; SALVADOR, T.; MENDES, A. J. Learning computational thinking and scratch at distance. **Computers in uman Behavior**, v. 80, p. 470–477, 2018. ISSN 0747-5632.

MICHAEL, D. R.; CHEN, S. L. Serious games: Games that educate, train, and inform. **Thomson Course Technology**, Cengage Learning, 2006.

MILNE, I.; ROWE, G. Difficulties in learning and teaching programming – views of students and tutors. In: **Education and Information Technologies**. [S.l.: s.n.], 2002. v. 7, p. 55–66.

MIT Media Lab. **Scratch: Programming language for kids**. 2025. Disponível em: https://scratch.mit.edu. Acesso em: 23 maio 2025.

NAH, F. F.-H.; ZENG, Q.; TELAPROLU, V. R.; AYYAPPA, K.; ESCHENBRENNER, B. Gamification of education: A review of literature. **Lecture Notes in Computer Science**, Springer, v. 8527, p. 401–409, 2014.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de software: Uma abordagem profissional**. Porto Alegre, RS: McGraw-Hill, 2016.

ROBINS, A.; ROUNTREE, J.; ROUNTREE, N. Learning and teaching programming: A review and discussion. **Computer Science Education**, v. 13, n. 2, p. 137–172, 2003.

ROSSUM, G. V. **Python tutorial**. [S.l.]: Centrum voor Wiskunde en Informatica (CWI), 1991.

SOMMERVILLE, I. **Software engineering**. 9. ed. [S.l.]: Pearson, 2011. ISBN 978-0137035151.

SUSI, T.; JOHANNESSON, M.; BACKLUND, P. Serious games: An overview. In: **Technical Report HS-IKI-TR-07-001, School of Humanities and Informatics**. University of Skövde, Sweden, 2007. Disponível em: https://www.diva-portal.org/smash/get/diva2:2416/FULLTEXT01.pdf.

TEAM, P. **Pytest documentation**. 2023. Disponível em: https://docs.pytest.org/. Acesso em: 10 jul. 2025.

VERGARA, S. C. **Projetos e relatórios de pesquisa em administração**. 4. ed. [S.l.]: Atlas, 2000.

World Bank. **The role of digital technology in economic development**. 2022. Disponível em: https://www.worldbank.org/en/topic/digitaldevelopment. Acesso em: 25 jul. 2025.

APÊNDICE A – QUESTIONÁRIO

Li, portanto, este termo e manifesto minha concordância com seu conteúdo e tenho acordo com minha participação livre e espontânea como voluntário(a) do estudo "Um protótipo de jogo para o apoio ao ensino introdutório de Programação". Declaro que obtive informações necessárias e que estou orientado(a) quanto ao teor do estudo aqui mencionado e compreendendo sua natureza e objetivo, autorizo o registro de minhas respostas para posterior análise científica, com publicação em periódicos e eventos relacionados.
Desde já, agradecemos a sua colaboração!

* Indica uma pergunta obrigatória

1. Aceita participar desta pesquisa? *

Marcar apenas uma oval.

- ◯ Sim, li o Termo de Consentimento Livre e Esclarecido (TCLE) e aceito participar da pesquisa.
- ◯ Eu não desejo participar do estudo. *Pular para a seção 6 (Agradecimento)*

Identificação

2. Qual é o seu papel na instituição? *

Marcar apenas uma oval.

- ◯ Estudante *Pular para a pergunta 3*
- ◯ Professor(a) *Pular para a pergunta 12*

Visão do estudante

Nesta seção, serão abordadas questões sobre os principais desafios e impactos da introdução da programação na vida dos estudantes, além de explorar o uso de jogos educacionais e seus efeitos.

3. Você já tinha tido algum contato com programação antes de entrar no curso? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não

4. Qual é sua principal dificuldade nas disciplinas de programação? *

Marque todas que se aplicam.

- ☐ Desenvolver a lógica de programação
- ☐ Entender a sintaxe do programa
- ☐ Entender a semântica do programa
- ☐ Compreender as abstrações a respeito do tema
- ☐ Outro: _____

5. Você considera importante desenvolver e/ou aperfeiçoar o raciocínio lógico para o auxílio da programação? *

Marcar apenas uma oval.

	1	2	3	4	5	
Sem	◯	◯	◯	◯	◯	Muito importante

6. Você considera que é importante diversificar o ensino de programação por meio de *metodologias ativas*, como os jogos educacionais? *

Marcar apenas uma oval.

	1	2	3	4	5	
Sem	◯	◯	◯	◯	◯	Muito importante

7. Você já fez uso de *jogos educacionais* durante as aulas de programação ou por conta própria? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

8. Caso já tenha feito uso de algum jogo educacional, quais foram suas principais perspectivas e suas maiores dificuldades?

_____

_____

_____

_____

_____

9. Em sua experiência, os jogos educacionais podem ser adaptados para atender às necessidades de diferentes alunos? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

10. Você acredita que os jogos educacionais podem tornar o processo de aprendizagem de programação mais divertido e envolvente? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

11. Você gostaria de aprender conceitos de programação por meio de jogos educacionais? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

Pular para a pergunta 22

Visão do docente

Nesta seção, serão abordados os principais desafios e impactos da introdução da programação na vida dos estudantes, segundo a visão dos professores. Também serão exploradas as didáticas utilizadas e as principais lacunas nas ferramentas disponíveis.

12. Quais são os principais assuntos abordados na disciplina introdutória de programação? *

13. Quais são as maiores dificuldades que os alunos enfrentam ao aprender lógica de programação nas aulas iniciais? *

14. Acredita que o desenvolvimento lógico cognitivo do algoritmo é realmente um problema frequente entre os alunos? *

Marcar apenas uma oval.

- ◯ Sim
- ◯ Não
- ◯ Talvez

15. Em sua opinião, quais são os conceitos de lógica de programação que os alunos têm mais dificuldade em entender? *

16. Quais são as abordagens didáticas que você utiliza para ensinar lógica de programação e quais têm sido mais eficazes? *

17. Você acredita que uma plataforma dedicada ao ensino de lógica de programação básica poderia ajudar seus alunos? Se sim, de que forma? *

18. Em sua opinião, um aluno que está tendo contato pela primeira vez com programação possui uma dificuldade muito maior do que quem já teve contato prévio com programação? *

19. Quais são as maiores lacunas nas ferramentas de ensino de programação que você utiliza atualmente?

20. Quais métricas ou indicadores você gostaria de acompanhar para avaliar o progresso dos alunos em uma plataforma de ensino de lógica de programação? *

21. Você teria disponibilidade para participar de uma entrevista para entender melhor a proposta do sistema? Caso tenha, por gentileza, forneça um e-mail para contato.

Pular para a pergunta 22

Visão geral do sistema

Nesta seção, buscamos entender a sua visão sobre a ferramenta que pretendemos desenvolver e obter seu feedback sobre a pesquisa.

22. Quais funcionalidades você considera essenciais em uma plataforma voltada para o ensino de lógica de programação? *

23. Quais tipos de exercícios ou atividades você gostaria de ver em uma plataforma de ensino de lógica de programação? *

24. Há mais alguma sugestão ou comentário que você gostaria de adicionar sobre o ensino de lógica de programação e o desenvolvimento desta plataforma?

25. Para você, qual é o grau de relevância dessa pesquisa? *

Marcar apenas uma oval.

	1	2	3	4	5	
Sem	◯	◯	◯	◯	◯	Muito relevante

Agradecimento

Agradecemos sinceramente pelo tempo e esforço dedicados para completar este questionário. Suas respostas são extremamente valiosas e nos ajudarão no desenvolvimento da pesquisa e no planejamento do jogo. Seu feedback é fundamental para nós. Obrigado mais uma vez por sua participação!

APÊNDICE B – DOCUMENTAÇÃO DO SISTEMA

Documentação da API para apoio ao processo de aprendizagem de lógica de programação

IDENTIFICAÇÃO DO PROJETO

Projeto

- API Educacional para apoio ao ensino de lógica de programação

HISTÓRICO DE REGISTROS

Versão	Data	Autor	Descrição
{1.0}	*01/05/2025*	*Pedro Henrique Souza Pessoa*	Elaboração do documento

SUMÁRIO

1. INTRODUÇÃO

Este documento apresenta a primeira etapa do desenvolvimento de uma API dedicada a auxiliar o processo de aprendizagem de ensino de lógica de programação. Para auxiliar educadores e alunos, essa API busca criar uma experiência interativa e personalizada de aprendizado, integrando funcionalidades que permitem o gerenciamento de conteúdo educacional, acompanhamento de desempenho e suporte a atividades gamificadas. No contexto atual, o ensino de programação enfrenta desafios como a falta de engajamento e a dificuldade de aplicar conceitos teóricos em práticas de desenvolvimento. A API proposta visa proporcionar uma plataforma acessível e flexível que favoreça a compreensão e aplicação prática dos conceitos de programação. Além disso, a API será uma base para o desenvolvimento de ferramentas educacionais, como jogos sérios e sistemas de gamificação, que podem ser integrados ao sistema, promovendo uma experiência de aprendizado mais dinâmica e motivadora.

2. DESCRIÇÃO DO SISTEMA

A API de Ensino de Lógica de Programação é um serviço *back-end* projetado para fornecer funcionalidades essenciais ao suporte e monitoramento do processo de aprendizado em programação. A API oferecerá endpoints bem definidos que viabilizarão diferentes funcionalidades, como:

- Gerenciamento de perfis de usuários (alunos e professores):
  - Cadastro de novos usuários.
  - Atualização de dados cadastrais.
  - Exclusão de contas.
  - Visualização de perfis públicos e privados.
- Armazenamento e consulta de dados de desempenho:
  - Registro de progresso dos alunos em relação aos conteúdos estudados.
  - Consulta de histórico de desempenho por parte dos usuários e administradores.
  - Listagem de progressos detalhados para acompanhamento.
- Criação e gerenciamento de atividades e exercícios interativos:
  - Cadastro de perguntas e conteúdos educacionais.
  - Configuração de níveis de dificuldade para as perguntas.
  - Envio e validação de respostas dos alunos.
  - Feedback imediato sobre o desempenho nas atividades.
- Implementação de elementos de gamificação:
  - Configuração de níveis e recompensas para engajamento.
  - Triagem inicial para avaliar o nível de conhecimento do usuário.
  - Personalização de conteúdo com base no progresso do aluno.

Com foco na escalabilidade e segurança, a API será desenvolvida para suportar múltiplos usuários simultâneos, fornecendo uma experiência consistente e eficiente. Além disso, ao centralizar a lógica de *back-end* em uma interface programática, a API permitirá que outras aplicações educacionais, como plataformas de *e-learning* e jogos sérios, integrem-se facilmente, ampliando o alcance e as possibilidades de uso para diferentes públicos e contextos educacionais.

3. REGRAS DE NEGÓCIO

Regra de Negócio	**Entidade I**	**Relação**	**Entidade II**
Uma pergunta possui um conteúdo	Pergunta	Associação	Conteúdo
Um conteúdo pode possuir 0 ou N perguntas	Conteúdo	Associação	Pergunta
Um usuário autenticado pode responder a uma pergunta	Usuário	Interação	Pergunta
Uma pergunta de nível fácil deve ter alternativas ajustadas para "Verdadeiro" e "Falso"	Pergunta	Restrição	-
A alternativa correta de uma pergunta não deve ser exibida para usuários comuns	Pergunta	Restrição	Usuário
Apenas administradores podem criar, atualizar ou excluir conteúdos	Conteúdo	Restrição	Usuário
Apenas administradores podem criar, atualizar ou excluir perguntas	Pergunta	Restrição	Usuário
Qualquer usuário autenticado pode visualizar conteúdos e perguntas	Conteúdo/Pergunta	Restrição	Usuário
O sistema valida se a resposta de um usuário está correta	Pergunta	Interação	Usuário

4. REQUISITOS

4.1. Requisitos Funcionais

4.1.1. Login no Sistema:

- **[RF001] - Login de usuário :**

O sistema deve permitir que o usuário faça log in informando o nome de usuário e a senha cadastrados. Se as informações de login forem incorretas (e-mail e/ou senha inválidos), o sistema deve exibir uma mensagem especificando o erro, orientando o usuário a corrigir as informações fornecidas.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF002] - Logout do Sistema:**

O sistema deve permitir que o usuário se desconecte da aplicação, encerrando a sessão de forma segura. Após o log out, o usuário não terá mais acesso às funcionalidades restritas até que faça um novo log in.

☑ Essencial ☐ Importante ☐ Desejável

4.1.2. Gerenciar Usuário

- **[RF003] - Cadastro de Aluno:**

O sistema deve permitir que um novo usuário seja registrado informando os seguintes dados obrigatórios: nome de usuário, senha e nome completo. Os campos opcionais incluem: avatar, instituição, data de nascimento, escolaridade. O cadastro deve estar disponível para qualquer usuário, sem necessidade de autenticação prévia.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF004] - Visualização de perfil privado:**

O sistema deve permitir que um usuário autenticado visualize seu próprio perfil completo, exceto a senha. Somente o próprio usuário ou um administrador podem acessar essa visualização. Caso o usuário tente acessar o perfil privado de outro usuário, o sistema deve retornar a visualização de perfil publico(RF005).

☐ Essencial ☑ Importante ☐ Desejável

- **[RF005] - Visualização de perfil público:**

O sistema deve permitir que qualquer usuário visualize os dados públicos de outro usuário, incluindo: nome, avatar e instituição. Essa visualização deve estar disponível mesmo para usuários não autenticados.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF006] - Atualização de perfil:**

O sistema deve permitir que um usuário autenticado atualize seus próprios dados cadastrais. Os campos editáveis incluem: nome, avatar, instituição, data de nascimento, escolaridade e senha. Somente o próprio usuário ou um administrador podem realizar essa operação.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF007] - Exclusão de Usuário:**

O sistema deve permitir que um usuário autenticado exclua sua própria conta. Somente o próprio usuário ou um administrador podem realizar essa operação. Após a exclusão, o usuário não poderá mais acessar o sistema com suas credenciais.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF008] - Listagem de Usuário:**

O sistema deve permitir que um administrador visualize uma lista de todos os usuários registrados no sistema. Para cada usuário, devem ser exibidos os seguintes dados: ID, nome de usuário, nome, avatar (avatar), instituição, data de criação. Usuários comuns não devem ter acesso a essa funcionalidade.

☐ Essencial ☑ Importante ☐ Desejável

4.1.3. Gerenciar conteúdo

- **[RF009] - Cadastro de conteúdo:**

O sistema deve permitir que um administrador cadastre um novo conteúdo informando os seguintes dados obrigatórios: nome, descrição e módulo. Somente administradores podem realizar essa operação.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF010] - Visualização de conteúdo:**

O sistema deve permitir que qualquer usuário autenticado visualize os detalhes de um conteúdo, incluindo: nome, descrição e módulo. Essa visualização deve estar disponível para todos os usuários autenticados.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF011] - Atualizar conteúdo:**

O sistema deve permitir que um administrador atualize os dados de um conteúdo existente. Os campos editáveis incluem: nome, descrição e módulo. Somente administradores podem realizar essa operação.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF012] - Exclusão de conteúdo:**

O sistema deve permitir que um administrador exclua um conteúdo existente. Somente administradores podem realizar essa operação.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF013] - Listagem de conteúdos:**

O sistema deve permitir que qualquer usuário autenticado visualize uma lista de todos os conteúdos cadastrados. Para cada conteúdo, devem ser exibidos os seguintes dados: nome, descrição e módulo.

☐ Essencial ☑ Importante ☐ Desejável

4.1.4. Gerenciar Pergunta

- **[RF014] - Cadastro de pergunta:**

O sistema deve permitir que um administrador cadastre uma nova pergunta vinculada a um conteúdo específico. Os dados obrigatórios incluem: enunciado, nível, alternativas (A, B, C, D) e alternativa correta. Se o nível da pergunta for "Fácil", as alternativas serão automaticamente ajustadas para "Verdadeiro" e "Falso". Somente administradores podem realizar essa operação.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF015] - Visualização de pergunta:**

O sistema deve permitir que qualquer usuário autenticado visualize os detalhes de uma pergunta. Para perguntas de nível "Fácil", as alternativas exibidas devem ser somente "Verdadeiro" e "Falso". Para perguntas de nível "Médio" ou "Difícil", todas as alternativas (A, B, C, D) devem ser exibidas. A alternativa correta não deve ser exibida para usuários comuns. Somente administradores podem visualizar a alternativa correta.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF016] - Atualização de pergunta:**

O sistema deve permitir que um administrador atualize os dados de uma pergunta existente. Os campos editáveis incluem: enunciado, nível, alternativas (A, B, C, D), alternativa correta e conteúdo associado. Se o nível da pergunta for alterado para "Fácil", as alternativas devem ser automaticamente ajustadas para "Verdadeiro" e "Falso". Somente administradores podem realizar essa operação.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF017] - Exclusão de pergunta:**

O sistema deve permitir que um administrador exclua uma pergunta existente. Somente administradores podem realizar essa operação.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF018] - Listagem de pergunta:**

O sistema deve permitir que qualquer usuário autenticado visualize uma lista de todas as perguntas cadastradas. Para cada pergunta, devem ser exibidos os seguintes dados: enunciado, nível, alternativas (ajustadas conforme o nível) e conteúdo associado. A alternativa correta não deve ser exibida para usuários comuns. Somente administradores podem visualizar a alternativa correta.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF019] - Responder pergunta:**

O sistema deve permitir que qualquer usuário autenticado responda a uma pergunta enviando sua escolha (A, B, C, D ou Verdadeiro/Falso). O sistema deve validar a resposta do usuário e retornar se a resposta está correta ou incorreta. Essa funcionalidade deve estar disponível para todos os usuários autenticados.

☑ Essencial ☐ Importante ☐ Desejável

4.1.5. Gerenciar Progresso

- **[RF020] - Criação de progresso:**

O sistema deve criar automaticamente o progresso de um usuário após seu cadastro, o sistema deve manter um progresso para cada conteúdo.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF021] - Visualização de progresso:**

O sistema deve permitir que os progressos de cada usuário seja visto por ele mesmo ou um administrador.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF022] - Edição de progresso:**

O sistema deve permitir que um administrador realize edição no progresso de um usuário.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF023] - Exclusão de progresso:**

O sistema deve excluir todo o progresso de um usuário que seja deletado, também deve permitir que um administrador realizar essa exclusão mesmo que o usuário não tenha sido excluído.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF024] - Listagem de progresso:**

O sistema deve permitir que todos os progressos de um usuário sejam listados exibindo campos que permitam visualizar seu progresso em cada conteúdo.

☐ Essencial ☑ Importante ☐ Desejável

4.1.6. Gerenciar Triagem

- **[RF025] - Criação de triagem:**

O sistema permitir que um usuário realize uma triagem após o cadastro do usuário, essa triagem é responsável por receber as informações do usuário sobre o seu nível de conhecimento referente a cada conteúdo.

☑ Essencial ☐ Importante ☐ Desejável

- **[RF026] - Visualização de triagem:**

O sistema deve permitir que um administrador visualize o resultado da triagem de um usuário.

☐ Essencial ☑ Importante ☐ Desejável

- **[RF027] - Exclusão de triagem:**

Após um usuário ter seus dados excluídos, o sistema deve também excluir automaticamente sua triagem.

☐ Essencial ☑ Importante ☐ Desejável

4.2. Requisitos Não-Funcionais

4.2.1. Segurança

- **[RNF001] - Autenticação via token:**

A API deve implementar autenticação JWT com tokens de curta duração e *refresh* tokens.

☑ Essencial ☐ Importante ☐ Desejável

- **[RNF001] - Validação:**

Validar todos os dados de entrada para evitar SQL Injection ou XSS.

☐ Essencial ☑ Importante ☐ Desejável

- **[RNF001] - Regulamentação:**

Garantir conformidade com regulamentações de privacidade seguindo as normas LGPD.

☑ Essencial ☐ Importante ☐ Desejável

4.2.2. Usabilidade

- **[RNF002] - Documentação:**

A documentação da API deve seguir padrões como Swagger para facilitar a integração.

☐ Essencial ☑ Importante ☐ Desejável

- **[RNF002] - Mensagem de erro:**

Respostas de erro devem ser claras e seguir códigos HTTP padrão.

☐ Essencial ☑ Importante ☐ Desejável

4.2.3. Manutenibilidade

- **[RNF002] - Padrão de código:**

O código deve seguir padrões de estilo PEP8 - Python.

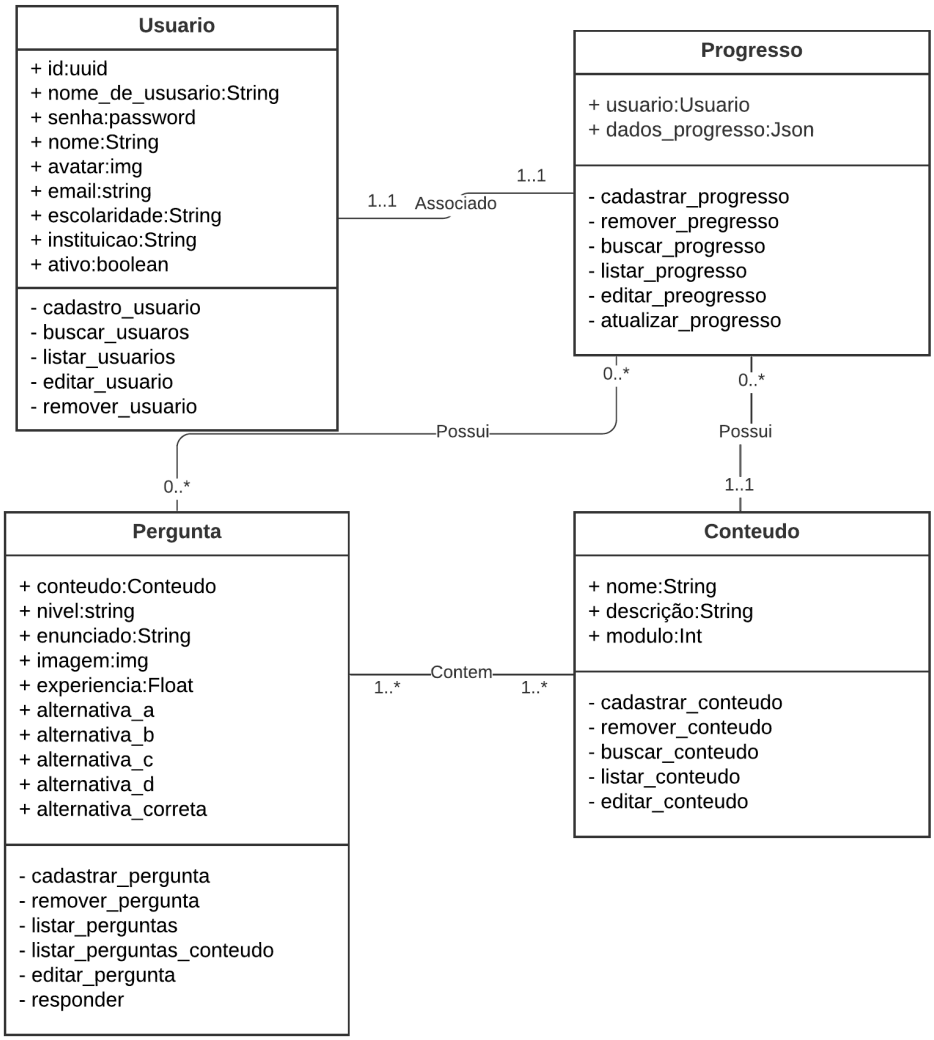
☐ Essencial ☑ Importante ☐ Desejável

- **[RNF002] - Controle de versão:**

A API deve ser versionada para evitar que mudanças quebrem clientes existentes.

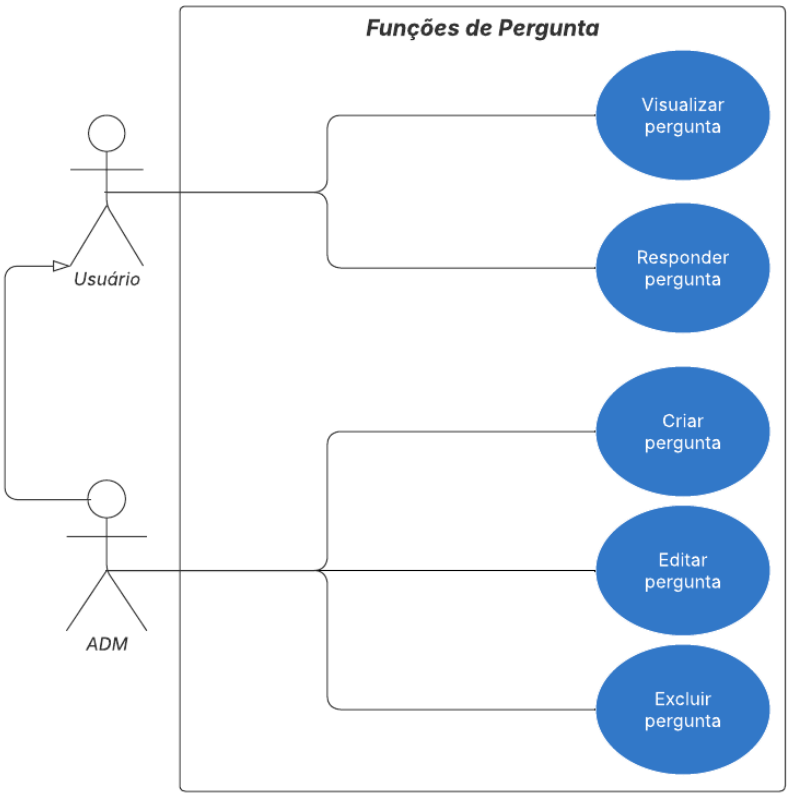
☐ Essencial ☑ Importante ☐ Desejável

4.3. Diagrama de classe

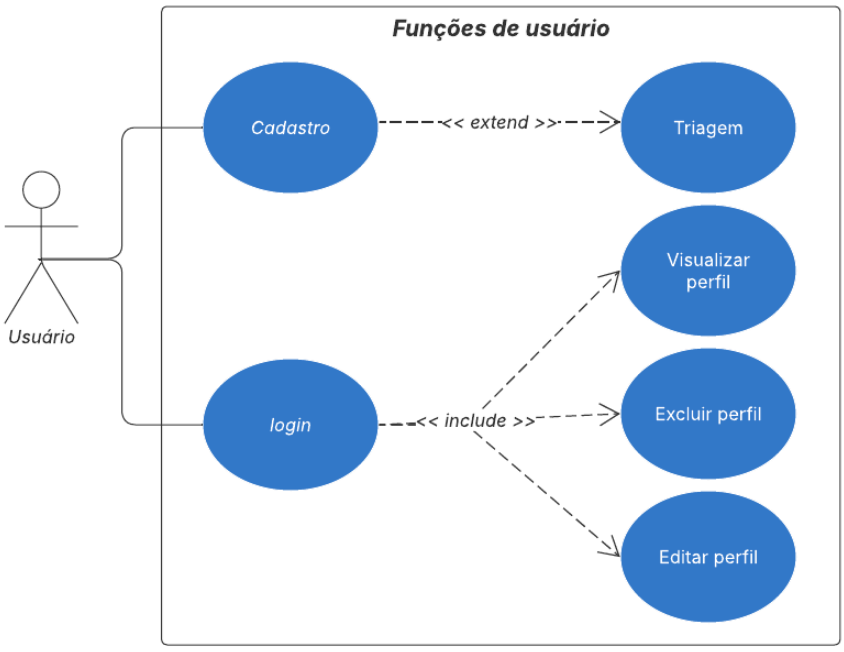


4.4. Diagramas de casos de uso

4.4.1. Interações com pergunta



4.4.2. Interações com funções de usuário



APÊNDICE C – PLANO DE TESTES

Plano de Testes

Software: API para apoio ao processo de aprendizagem de lógica de programação

---

1. Introdução

Este Plano de Teste foi elaborado para definir os objetivos, escopo, estratégia, ambiente e critérios dos testes que serão realizados para garantir a qualidade da API para apoio ao processo de aprendizagem de lógica de programação. A API é responsável por gerenciar usuários, progresso de aprendizado, perguntas e conteúdos educacionais. Este documento também descreve os entregáveis e o cronograma de atividades.

2. Objetivo dos testes

- Verificar se as funcionalidades principais da API estão funcionando conforme o esperado, incluindo autenticação, registro de progresso, visualização de conteúdos e gerenciamento de usuários.
- Validar a integração entre diferentes módulos, como o sistema de autenticação (allauth), o banco de dados (PostgreSQL ou SQLite) e o framework REST (Django REST Framework).
- Identificar e corrigir defeitos críticos antes do lançamento da aplicação em produção.

3. Escopo dos testes

- O que será testado:
    - Autenticação e Registro de Usuários:
        - Criação de novos usuários.
        - Log in e log out.
        - Recuperação de senha.
        - Validação de campos obrigatórios (ex: e-mail, nome, senha).
    - Gerenciamento de Progresso:
        - Criação automática de progresso após o cadastro do usuário.
        - Atualização de dados de progresso (XP, nível, perguntas respondidas).

        - Exclusão de progresso ao deletar um usuário.
    - Gerenciamento de Perguntas e Conteúdos:
        - Listagem de perguntas e conteúdos disponíveis.
        - Filtragem de perguntas por dificuldade ou conteúdo específico.
        - Resposta a perguntas e atualização do progresso.

4. Abordagem dos testes

- Testes Unitários:
    - Verificação de métodos e funções isoladas, como validações de modelos e serializadores.
    - Ferramenta utilizada: pytest
- Testes de Integração:
    - Testes de interação entre componentes, como autenticação (allauth) e banco de dados.
    - Ferramenta utilizada: pytest-django.
- Testes Funcionais/API:
    - Simulação de requisições HTTP para *endpoints* da API.
    - Ferramentas utilizadas: pytest, requests, Insomnia.

5. Ambiente dos testes

- Sistema Operacional: Windows.
- Framework *Back-End*: Django 4.x.
- Banco de Dados: PostgreSQL (produção), SQLite (testes).
- Ferramentas de Teste: pytest, pytest-django.
- URL da Aplicação de Teste: http://localhost:8000/api/.
- Dados de Teste:
    - Arquivo .env configurado com variáveis de ambiente para testes.
    - Banco de dados SQLite temporário (test_db.sqlite3).
- Requisitos do Ambiente:
    - Python 3.12 instalado.

- Dependências instaladas via requirements.txt.
- Variáveis de ambiente configuradas no arquivo .env.

6. Critérios dos testes

6.1. Critérios de entrada

- O ambiente de teste deve estar configurado (banco de dados, servidor local).
- O código-fonte da API deve estar completo e revisado.
- Todos os casos de teste devem estar documentados e prontos para execução.

6.2. Critérios de saída

- Todos os casos de teste devem ser executados.
- Defeitos críticos identificados devem ser corrigidos.

6.3. Critérios de aceitação

- Todas as funcionalidades principais devem passar nos testes unitários, de integração e funcionais.
- Não devem existir bugs críticos ou bloqueadores.