Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Scriptonauta - Ensino de Javascript através de uma plataforma lúdica

Nilton Rêgo Teixeira

JUIZ DE FORA
JULHO, 2026

Scriptonauta - Ensino de Javascript através de uma plataforma lúdica

Nilton Rêgo Teixeira

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Rodrigo Luis de Souza da Silva

JUIZ DE FORA

JULHO, 2026

Scriptonauta - Ensino de Javascript através de uma plataforma lúdica

Nilton Rêgo Teixeira

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Rodrigo Luis de Souza da Silva
Doutor em Engenharia Civil

Marcelo Caniato Renhe
Doutor em Engenharia de Sistemas e Computação

Igor de Oliveira Knop
Doutor em Modelagem Computacional

JUIZ DE FORA

6 DE JULHO, 2026

Aos meus pais, pelo apoio e sustento.

Resumo

Este trabalho apresenta o desenvolvimento e a avaliação do Scriptonauta, uma plataforma educacional gamificada voltada ao ensino de programação em JavaScript para estudantes iniciantes. A proposta tem como objetivo apoiar o processo de aprendizagem por meio de atividades interativas, *feedback* automático e elementos de gamificação, como progressão por níveis, narrativa espacial e sistema de vidas. A plataforma foi desenvolvida como uma aplicação *web* utilizando tecnologias modernas, incluindo TypeScript, Next.js, React e PostgreSQL, permitindo acesso multiplataforma e facilitando sua utilização em diferentes dispositivos. O sistema foi estruturado em planetas de aprendizagem, cada um representando um conjunto progressivo de conteúdos, abordando desde conceitos básicos até estruturas mais avançadas da linguagem JavaScript. A avaliação da plataforma foi realizada com estudantes da disciplina de Algoritmos II da Universidade Federal de Juiz de Fora, totalizando 78 participantes. Os resultados obtidos indicam uma percepção positiva em relação à organização pedagógica, ao potencial educacional e aos elementos de gamificação. Apesar de problemas iniciais de desempenho e estabilidade, melhorias implementadas entre as aplicações resultaram em um aumento significativo das médias de avaliação. De forma geral, os resultados sugerem que o Scriptonauta possui potencial para apoiar o ensino introdutório de programação, oferecendo uma experiência de aprendizagem mais engajante, interativa e estruturada.

Palavras-chave: Software Educacional, Ensino de Programação; JavaScript, Gamificação.

Abstract

This work presents the development and evaluation of Scriptonauta, a gamified educational platform designed to support the teaching of JavaScript programming to beginner students. The proposal aims to enhance the learning process through interactive activities, automatic feedback, and gamification elements such as level progression, narrative-based structure, and a life system. The platform was developed as a web application using modern technologies, including TypeScript, Next.js, React, and PostgreSQL, enabling cross-device access and ease of use across different environments. The system is organized into learning planets, each representing a progressive set of programming concepts ranging from basic fundamentals to more advanced JavaScript structures. The platform was evaluated with students enrolled in the Algorithms II course at the Federal University of Juiz de Fora, totaling 78 participants. The results indicate positive perceptions regarding pedagogical organization, educational potential, and gamification features. Although initial performance and stability issues were identified, improvements implemented between evaluation phases led to significant increases in the obtained ratings. Overall, the findings suggest that Scriptonauta has the potential to support introductory programming education by providing a more engaging, interactive, and structured learning experience.

Keywords: Educational Software, Programming Education, JavaScript, Gamification.

Agradecimentos

Aos meus pais, Magali de Souza e Nilton Teixeira por todo o encorajamento e apoio. Sem vocês, não teria conseguido embarcar nessa jornada.

Ao professor Rodrigo pela orientação, companheirismo e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Aos meus amigos e companheiros que reuni nessa jornada, compartilhando momentos, risadas e angustias. Obrigado por tornarem essa jornada mais leve.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"O homem não teria alcançado o possível se, repetidas vezes, não tivesse tentado o impossível".

Max Weber

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

DCC	Departamento de Ciência da Computução
UFJF	Universidade Federal de Juiz de Fora
AST	Árvore Sintática Abstrata (Abstract Syntax Tree)
BNCC	Base Nacional Comum Curricular

1 Introdução

1.1 Apresentação do Tema

A programação de computadores tem se consolidado como uma competência fundamental na sociedade contemporânea, marcada pela intensa presença das tecnologias digitais em diferentes setores. Além de constituir uma habilidade técnica relevante para o mercado de trabalho, a programação contribui para o desenvolvimento de capacidades cognitivas relacionadas à resolução de problemas, ao raciocínio lógico e à construção de soluções estruturadas para desafios complexos.

Nesse contexto, o ensino de programação vem adquirindo crescente importância nos ambientes educacionais. Segundo Wing (2006), o aprendizado de programação favorece o desenvolvimento do pensamento computacional, definido como uma forma de raciocínio baseada na formulação de problemas e na elaboração de soluções que possam ser executadas por sistemas computacionais. Essa competência possui caráter interdisciplinar, podendo ser aplicada em diversas áreas do conhecimento.

Além disso, aprender a programar permite que os estudantes assumam um papel mais ativo na utilização da tecnologia. Para Resnick (2017), a programação possibilita que indivíduos deixem de ser apenas consumidores de recursos digitais e passem a atuar como criadores de soluções, ampliando sua autonomia, criatividade e capacidade de expressão. Dessa forma, o ensino de programação transcende a formação técnica, contribuindo também para o desenvolvimento de habilidades essenciais para a participação crítica e produtiva na sociedade digital.

1.2 Contextualização

Nos últimos anos, o ensino de programação tem ganhado espaço em diferentes níveis educacionais, impulsionado pela transformação digital da sociedade e pela crescente necessidade de desenvolver competências relacionadas ao pensamento computacional. Diversos

países passaram a incorporar conteúdos de computação em seus currículos escolares, reconhecendo a programação não apenas como uma habilidade técnica, mas também como uma ferramenta para estimular o raciocínio lógico, a criatividade e a resolução de problemas (WING, 2006).

No Brasil, esse movimento também tem se fortalecido. A Base Nacional Comum Curricular (BNCC) passou a incentivar o desenvolvimento de competências relacionadas ao pensamento computacional e ao uso crítico das tecnologias digitais ao longo da Educação Básica (Brasil, 2018). Além disso, a Sociedade Brasileira de Computação propôs diretrizes para o ensino de Computação na Educação Básica, contribuindo para a consolidação da área e servindo de referência para políticas públicas e iniciativas educacionais (Sociedade Brasileira de Computação, 2019). Esse cenário evidencia a crescente importância do desenvolvimento de ferramentas e metodologias capazes de apoiar o ensino de programação desde os anos iniciais da formação escolar.

Nesse contexto, o uso de tecnologias educacionais tem se consolidado como uma estratégia para apoiar o processo de ensino-aprendizagem. Ambientes virtuais de aprendizagem, plataformas de exercícios e sistemas de correção automática permitem que estudantes pratiquem conteúdos de forma autônoma, recebendo retorno sobre seu desempenho de maneira mais rápida e frequente. Segundo Papert (1980), a aprendizagem tende a ser mais significativa quando o estudante participa ativamente da construção do conhecimento, experimentando e testando suas próprias soluções.

Paralelamente, abordagens baseadas em gamificação vêm sendo exploradas para aumentar o engajamento dos alunos em ambientes educacionais. De acordo com Deterding et al. (2011), a gamificação consiste na utilização de elementos característicos dos jogos em contextos não relacionados a jogos, buscando incentivar a participação e a motivação dos usuários. No ensino de programação, esses elementos podem incluir sistemas de pontuação, níveis, recompensas e desafios progressivos, contribuindo para tornar o aprendizado mais atrativo e menos intimidador para iniciantes.

Além disso, a linguagem JavaScript tem se destacado como uma das mais utilizadas no desenvolvimento de aplicações *web* modernas. Sua ampla adoção pela indústria de software e sua capacidade de execução diretamente nos navegadores tornam seu aprendi-

zado particularmente relevante para estudantes que desejam ingressar na área de desenvolvimento de *software*. Dessa forma, a criação de ferramentas educacionais voltadas ao ensino de JavaScript representa uma oportunidade de aproximar os estudantes de uma tecnologia amplamente empregada no mercado de trabalho.

1.3 Descrição do Problema

Apesar da crescente disponibilidade de recursos para o ensino de programação, as taxas de dificuldade e evasão em cursos introdutórios permanecem elevadas. Segundo Robins, Rountree e Rountree (2003), muitos estudantes enfrentam obstáculos relacionados à abstração dos conceitos fundamentais, à interpretação de mensagens de erro e à dificuldade em compreender a lógica necessária para construir algoritmos. Esses fatores frequentemente resultam em frustração e desmotivação, especialmente durante os primeiros contatos com uma linguagem de programação.

Outro aspecto relevante é a necessidade de *feedback* rápido e compreensível. Conforme destacado por Jenkins (2002), quando o estudante não recebe orientações claras sobre seus erros, torna-se mais difícil identificar falhas de raciocínio e corrigir equívocos de aprendizagem. Em muitos ambientes de ensino tradicionais, o retorno sobre atividades práticas depende da correção manual realizada pelo professor, o que pode atrasar significativamente o processo de aprendizagem.

Embora existam plataformas de ensino de programação amplamente utilizadas, muitas delas concentram-se na execução de exercícios com respostas pré-definidas ou em conteúdos predominantemente expositivos. Em diversos casos, a avaliação automática limita-se à comparação do resultado produzido pelo código, sem analisar aspectos relacionados à lógica empregada pelo estudante ou fornecer explicações contextualizadas sobre possíveis erros. Como consequência, o aluno pode concluir uma atividade sem compreender adequadamente os conceitos envolvidos.

Além disso, nem todas as plataformas oferecem mecanismos de engajamento capazes de manter o interesse dos estudantes ao longo do processo de aprendizagem. A ausência de elementos lúdicos, desafios progressivos e recompensas pode reduzir a motivação dos usuários, especialmente daqueles que estão iniciando seus estudos em programação. Dessa

forma, identifica-se a necessidade de uma solução que combine ensino estruturado, atividades práticas, correção automática inteligente e elementos de gamificação, proporcionando uma experiência de aprendizagem mais dinâmica e efetiva.

Embora o ensino de programação seja relevante em diferentes níveis educacionais, este trabalho concentra-se em estudantes entre 10 e 13 anos de idade. Essa faixa etária corresponde a um período em que os alunos já possuem condições de desenvolver habilidades relacionadas ao pensamento computacional e tem sido frequentemente considerada em pesquisas sobre ensino introdutório de programação e gamificação. Dessa forma, o Scriptonauta foi concebido levando em consideração as características desse público, especialmente quanto à linguagem utilizada, à organização pedagógica dos conteúdos e aos elementos visuais e de gamificação empregados na plataforma. A escolha dessa faixa etária não implica que a plataforma seja restrita a esse público. Ela foi definida por representar o público-alvo predominante nos estudos analisados nesta revisão da literatura e por corresponder ao grupo para o qual se pretende realizar avaliações experimentais em trabalhos futuros.

1.4 Justificativa

Diante dos desafios persistentes no ensino de programação e das limitações observadas nas plataformas existentes, justifica-se o desenvolvimento de uma ferramenta educacional voltada ao ensino introdutório de JavaScript para estudantes de 10 a 13 anos. A escolha desse público fundamenta-se no fato de que essa faixa etária representa um momento importante para o desenvolvimento do pensamento computacional e corresponde ao público-alvo adotado por diversas pesquisas sobre ensino de programação presentes na literatura revisada. Além disso, essa definição orientou decisões de projeto relacionadas à linguagem utilizada, aos elementos de gamificação e à organização pedagógica da plataforma.

1.5 Objetivos

Este projeto tem como objetivo geral apresentar um *software* educacional de apoio ao ensino de JavaScript, oferecendo atividades interativas que facilitem o aprendizado de forma clara e atrativa. O sistema foi desenvolvido para execução em navegadores *web*, permitindo o acesso por diferentes dispositivos, como *desktops*, *smartphones* e *tablets*, exigindo apenas uma conexão com a Internet.

Para alcançar esse objetivo geral, foram definidos os seguintes objetivos específicos:

- Realizar um estudo experimental do *software*, por meio da coleta e análise qualitativa de dados obtidos durante sua utilização, visando avaliar sua contribuição para o processo de aprendizagem.
- Disponibilizar o sistema ao público como uma ferramenta educacional de uso real, ampliando seu alcance para além do contexto experimental e possibilitando sua utilização por estudantes e professores interessados no aprendizado de JavaScript.

1.6 Metodologia

Este estudo possui caráter aplicado, visando à criação e à análise de uma solução prática para o problema apresentado. Para averiguar a eficácia da solução proposta, foi realizada uma avaliação quantitativa, fundamentada na aplicação de um questionário estruturado e na análise estatística dos dados coletados, permitindo mensurar a aceitação do software pelos participantes e identificar seus principais pontos fortes e limitações.

2 Revisão da Literatura

Nos últimos anos, diversos estudos têm investigado o uso de estratégias de gamificação e jogos educacionais para apoiar o ensino de programação e o desenvolvimento do pensamento computacional. Esses trabalhos buscam tornar o processo de aprendizagem mais atrativo e motivador, especialmente para estudantes da educação básica.

O estudo de Sun e Liu (2025) analisou os impactos do uso das plataformas CodeCombat e Scratch no desenvolvimento do pensamento computacional de alunos do ensino fundamental. A avaliação foi conduzida por meio de uma intervenção educacional com estudantes, comparando o desempenho dos participantes antes e após a utilização das plataformas, além de analisar diferenças relacionadas ao gênero e à experiência prévia em programação. Os resultados indicaram que ambas as abordagens contribuem positivamente para a aprendizagem, embora apresentem diferenças relacionadas à experiência prévia dos estudantes e ao gênero dos participantes.

No trabalho de Choi e Choi (2024), os autores investigaram a influência do currículo baseado em blocos da plataforma Code.org na motivação dos estudantes. A avaliação foi realizada por meio da aplicação de questionários de motivação antes e após a utilização da plataforma, complementados pela análise da participação dos estudantes nas atividades propostas. Os resultados demonstraram aumento do engajamento durante as atividades, evidenciando o potencial das abordagens baseadas em programação em blocos para o ensino introdutório de programação.

O trabalho de Kiraly et al. (2025) explorou a combinação entre tutoriais gamificados e a metodologia de sala de aula invertida para o ensino de SQL. Para avaliar a proposta, os autores compararam o desempenho acadêmico dos estudantes em atividades e avaliações práticas, além de coletarem percepções dos participantes por meio de questionários. Os resultados indicaram melhorias no desempenho acadêmico e no envolvimento dos alunos, sugerindo que a integração entre metodologias ativas e gamificação pode favorecer o processo de aprendizagem.

No trabalho de Thanyaphongphat, Thongkoo e Daungcharone (2023), foi pro-

posta uma abordagem educacional baseada em robótica e competições utilizando MicroPython. A avaliação ocorreu durante atividades práticas desenvolvidas com estudantes, considerando tanto o desempenho nas tarefas quanto questionários de percepção sobre motivação e interesse pela programação. Os resultados mostraram ganhos na aprendizagem e no interesse dos estudantes, demonstrando que atividades práticas associadas à programação podem contribuir para o desenvolvimento de competências computacionais.

O estudo de Nobnop et al. (2023) avaliou o uso de um jogo educacional inspirado no gênero *Battle Royale* para o ensino de programação. A pesquisa concentrou-se principalmente na avaliação da usabilidade do sistema, utilizando questionários respondidos pelos estudantes após a utilização da plataforma. Os resultados demonstraram boa aceitação por parte dos estudantes e indicaram potencial para aumentar o interesse pela área por meio da utilização de mecânicas de jogos.

No trabalho de Liu, Bai e Xia (2023), os autores investigaram o uso de atividades gamificadas envolvendo programação textual para o desenvolvimento do pensamento computacional. A avaliação foi baseada na comparação do desempenho dos estudantes em atividades de programação e em instrumentos para mensuração das habilidades de pensamento computacional antes e após a intervenção. Os resultados evidenciaram melhorias significativas nas habilidades dos estudantes, reforçando o potencial da gamificação mesmo em ambientes que utilizam linguagens textuais.

O trabalho de Zhan et al. (2022) realizou uma meta-análise de estudos sobre gamificação aplicada ao ensino de programação. Diferentemente dos demais trabalhos, não houve experimentação com estudantes; os autores sintetizaram resultados de pesquisas previamente publicadas, analisando estatisticamente os efeitos da gamificação sobre motivação, engajamento e desempenho. Os resultados indicaram que elementos como recompensas, desafios, sistemas de progressão e *feedback* tendem a impactar positivamente esses aspectos.

Por fim, o estudo de Sun e Liu (2023) investigou os efeitos da gamificação no ensino de Python para estudantes do ensino fundamental. A avaliação utilizou testes de pensamento computacional e atividades práticas de programação, comparando o desempenho dos participantes antes e após a intervenção. Os autores observaram melhorias

nas habilidades de pensamento computacional e identificaram diferenças de desempenho relacionadas ao gênero dos participantes.

De modo geral, observa-se que a maior parte dos trabalhos avaliou suas propostas por meio de estudos experimentais com estudantes, utilizando uma combinação de testes de desempenho, questionários de percepção e análises comparativas antes e após a intervenção. Os resultados indicam que a gamificação e os jogos educacionais podem contribuir significativamente para o ensino de programação e para o desenvolvimento do pensamento computacional. Entretanto, observa-se que muitas das soluções existentes concentram-se em linguagens de programação em blocos (CHOI; CHOI, 2024), em linguagens distintas do JavaScript, como Python (SUN; LIU, 2023) e SQL (KIRALY et al., 2025), ou em plataformas já consolidadas, como Scratch e CodeCombat (SUN; LIU, 2025). Além disso, algumas propostas dependem de recursos específicos, como kits de robótica (THANYAPHONGPHAT; THONGKOO; DAUNGCHARONE, 2023), enquanto outras priorizam aspectos isolados, como a usabilidade de jogos educacionais (NOBNOP et al., 2023) ou a avaliação de estratégias de gamificação (LIU; BAI; XIA, 2023; ZHAN et al., 2022). Nesse contexto, o Scriptonauta busca contribuir com uma abordagem acessível e totalmente baseada na *web*, voltada ao ensino introdutório de JavaScript por meio de programação textual. A plataforma integra elementos de gamificação, narrativa, progressão pedagógica em fases, sistema de vidas e *feedback* imediato, reunindo em uma única solução características que, nos trabalhos analisados, geralmente são exploradas de forma independente.

3 Abordagem Proposta

Neste capítulo é apresentada a abordagem proposta de desenvolvimento de uma plataforma de ensino de JavaScript.

3.1 Modelo de Aprendizado

O modelo de aprendizado do Scriptonauta foi desenvolvido com o objetivo de organizar o ensino de programação de maneira gradual, estruturada e motivadora. A proposta considera que estudantes iniciantes frequentemente apresentam dificuldades na compreensão de conceitos abstratos e na associação entre teoria e prática, especialmente nos primeiros contatos com uma linguagem real de programação. Dessa forma, o sistema adota uma estrutura de progressão baseada em níveis e missões, permitindo que os conteúdos sejam introduzidos de forma incremental. Cada etapa do aprendizado representa um conjunto específico de habilidades e conhecimentos, reduzindo a sobrecarga cognitiva e favorecendo a consolidação gradual dos conceitos. No Scriptonauta, a progressão ocorre por meio de planetas e missões temáticas inseridos em uma narrativa espacial. Cada planeta corresponde a um módulo de aprendizagem, contendo desafios relacionados a determinados conteúdos de programação. O estudante avança conforme conclui atividades e demonstra domínio mínimo dos conceitos apresentados. Como forma de estimular a atenção e o engajamento dos alunos durante as atividades, a plataforma implementa um sistema de vidas. Cada erro cometido ao longo da execução de um planeta reduz em uma unidade a quantidade de vidas disponíveis. Inicialmente, o aluno possui cinco vidas por planeta; ao esgotá-las, a atividade deve ser reiniciada desde o início. A organização dos conteúdos segue uma sequência pedagógica progressiva, na qual habilidades mais simples servem como base para conteúdos posteriores. A descrição geral do conteúdo e o objetivo de cada planeta podem ser vistos na Tabela 3.1.

Portanto, o modelo de progressão de aprendizado do Scriptonauta procura unir organização pedagógica, progressão gradual de dificuldade e elementos motivacionais, ofe-

Tabela 3.1: Planetas, conteúdos e objetivos de aprendizagem do Scriptonauta

Planeta	**Conteúdo Principal**	**Objetivos de Aprendizagem**
Variáveis	Declaração e utilização de variáveis (`let`, `const`)	Familiarizar o estudante com o ambiente de programação e introduzir o conceito de armazenamento de informações.
Números	Tipos numéricos e operações aritméticas (`+`, `-`, `*`, `/`, `%`)	Compreender como representar e manipular valores numéricos em *JavaScript*.
Leitura de Dados	Entrada de dados por meio da função `prompt()`	Aprender a receber informações fornecidas pelo usuário durante a execução do programa.
Operadores	Operadores de comparação (`===`, `!==`, `<`, `>`, `<=`, `>=`)	Entender como comparar valores e produzir resultados lógicos para tomada de decisão.
Condicionais	Estruturas `if` e `else`	Desenvolver a capacidade de criar fluxos de execução baseados em condições.
Operadores Condicionais	Operadores lógicos (`&&`, `||`, `!`)	Construir expressões condicionais mais complexas utilizando proposições compostas.
Loops	Estruturas de repetição `while` e `do...while`	Compreender o conceito de repetição e sua aplicação na automação de tarefas.
Mais Loops	Estrutura de repetição `for`	Utilizar laços de repetição com controle explícito de inicialização, condição e incremento.
Strings	Cadeias de caracteres e métodos básicos de manipulação	Aprender a representar, concatenar e manipular textos em *JavaScript*.
Funções	Declaração, parâmetros, retorno e reutilização de código	Desenvolver programas modulares por meio da criação e utilização de funções.

recendo um ambiente estruturado para apoiar o desenvolvimento das habilidades iniciais de programação.

3.2 Tecnologias Utilizadas

O Scriptonauta foi desenvolvido como uma aplicação *web* utilizando tecnologias amplamente empregadas no desenvolvimento de sistemas modernos. A implementação foi realizada em **TypeScript**, enquanto a interface foi construída com **Next.js**, **React** e **Tailwind CSS**, permitindo o desenvolvimento de uma aplicação responsiva, modular

e de fácil manutenção.

O gerenciamento de autenticação e usuários foi realizado por meio da plataforma **Clerk**, responsável pelos processos de cadastro, autenticação e controle de sessões. Para a persistência dos dados, foi utilizado o sistema gerenciador de banco de dados **PostgreSQL**, com acesso às informações realizado por meio do **Drizzle ORM**, que simplifica a comunicação entre a aplicação e o banco de dados.

Além disso, foram utilizados mecanismos de armazenamento local do navegador para preservar temporariamente informações do usuário, contribuindo para uma experiência mais contínua em situações de instabilidade de conexão.

A combinação dessas tecnologias possibilitou o desenvolvimento de uma plataforma *web* moderna, capaz de atender aos requisitos funcionais do Scriptonauta e fornecer uma base sólida para a implementação dos recursos educacionais e das mecânicas de gamificação propostas.

3.3 Apresentação da Plataforma

Esta seção apresenta as principais interfaces e funcionalidades do Scriptonauta, destacando os elementos responsáveis pela navegação, execução das atividades e acompanhamento do progresso dos estudantes. O sistema foi desenvolvido com uma temática espacial, buscando proporcionar uma experiência mais envolvente durante o processo de aprendizagem de programação.

3.3.1 Tela de Seleção de Planetas

A tela de seleção de planetas constitui o ambiente principal de navegação do sistema. Nela, os estudantes podem visualizar os planetas disponíveis, identificar seu progresso atual e acessar os conteúdos correspondentes a cada etapa de aprendizagem.

Cada planeta representa um conjunto de conteúdos relacionados a um determinado tópico de programação. À medida que o estudante conclui as atividades propostas, novos planetas são desbloqueados, permitindo a continuidade da jornada de aprendizagem.

Além de funcionar como mecanismo de organização pedagógica, a utilização dos planetas reforça a narrativa espacial do Scriptonauta, transformando a progressão dos conteúdos em uma experiência de exploração. A Figura 3.1 apresenta a tela de seleção de planetas da plataforma.

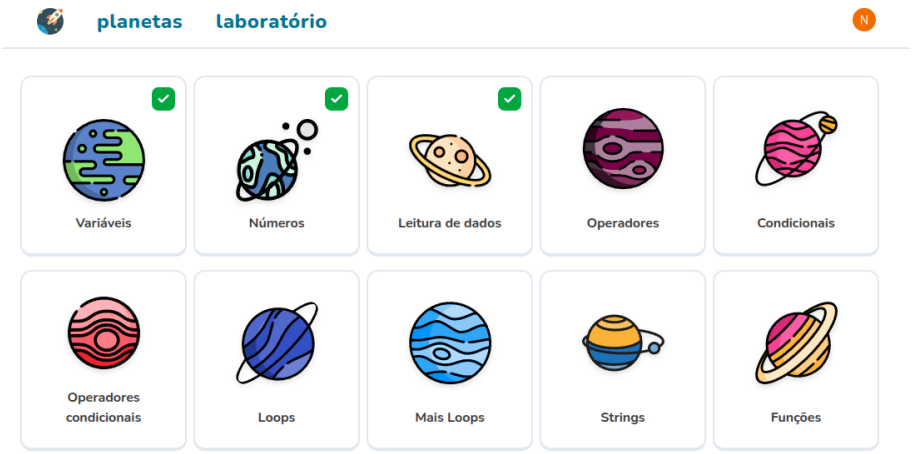

Figura 3.1: Tela de seleção de planetas do Scriptonauta.

3.3.2 Tipos de Atividades

O Scriptonauta foi projetado para oferecer diferentes formas de interação com os conteúdos de programação. Para isso, o sistema disponibiliza três tipos principais de atividades, cada uma com objetivos pedagógicos específicos.

As questões expositivas são utilizadas como momentos de instrução ao longo da jornada de aprendizagem, permitindo a introdução de explicações, exemplos e orientações relacionadas aos tópicos abordados em cada planeta. Dessa forma, elas desempenham um papel semelhante ao de pequenos módulos de conteúdo, preparando o estudante para a realização das atividades práticas subsequentes.

Durante essas interações, o usuário pode visualizar informações teóricas, exemplos de código e explicações sobre conceitos fundamentais da programação. O avanço ocorre mediante a leitura do conteúdo apresentado, sem a necessidade de validação automática de respostas pelo sistema. A Figura 3.2 mostra a interface de uma atividade expostiva.

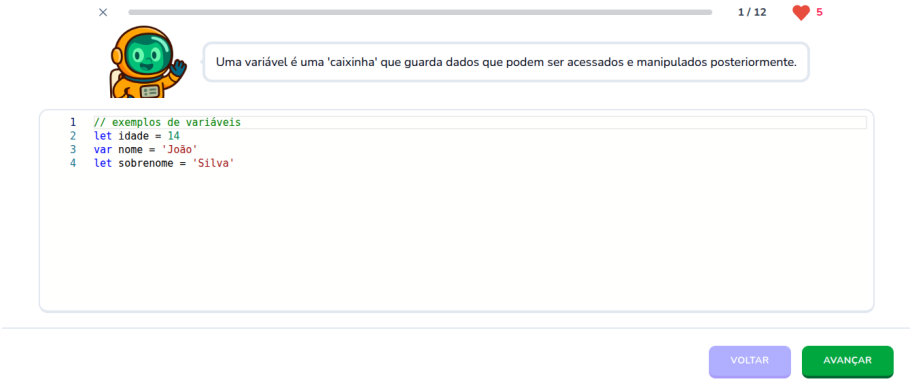

Figura 3.2: Atividade expositiva.

As atividades de múltipla escolha apresentam questões acompanhadas de alternativas, nas quais o estudante deve selecionar a resposta correta.

Esse formato é utilizado principalmente para verificar a compreensão de conceitos teóricos, sintaxe de comandos e interpretação de trechos de código. Além disso, permite a obtenção de *feedback* imediato após cada resposta, auxiliando o estudante na identificação de possíveis equívocos. A Figura 3.3 mostra um exemplo de questão de múltipla escolha.

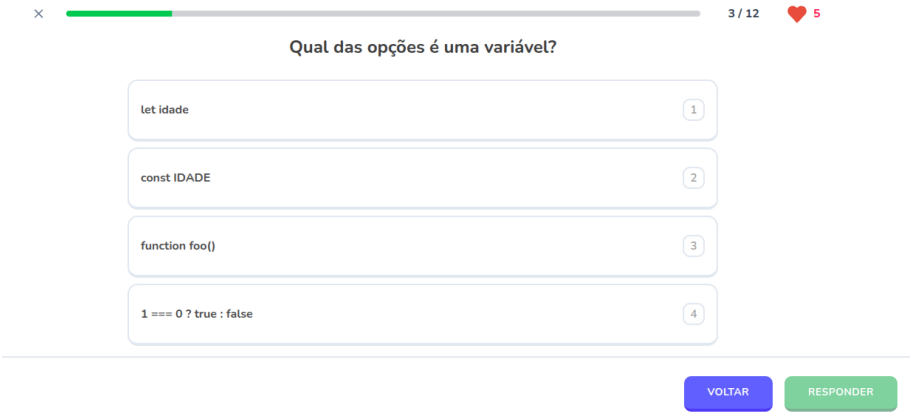

Figura 3.3: Atividade de múltipla escolha.

A interface desse tipo de atividade oferece um *feedback* imediato, informando o aluno se a questão foi respondida corretamente ou não. A Figura 3.4 mostra a mudança da interface caso a resposta esteja correta.

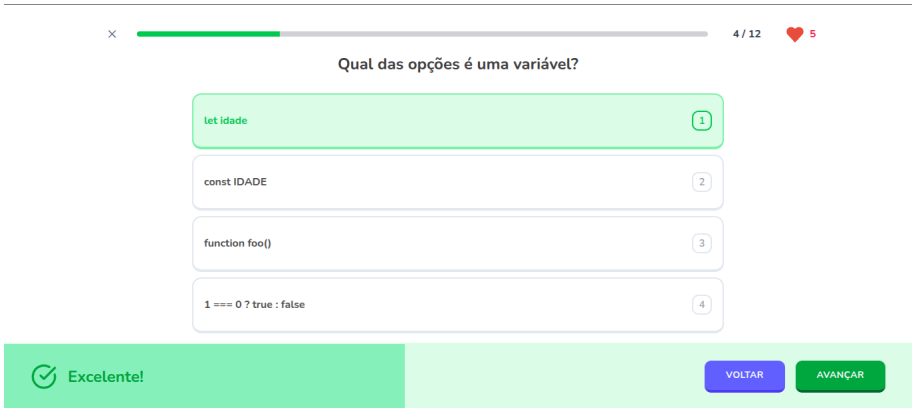

Figura 3.4: Atividade de múltipla escolha respondida corretamente.

A Figura 3.5 mostra a interface caso a resposta esteja errada. Nesse caso, o aluno pode responder novamente caso ainda tenha vidas disponíveis.

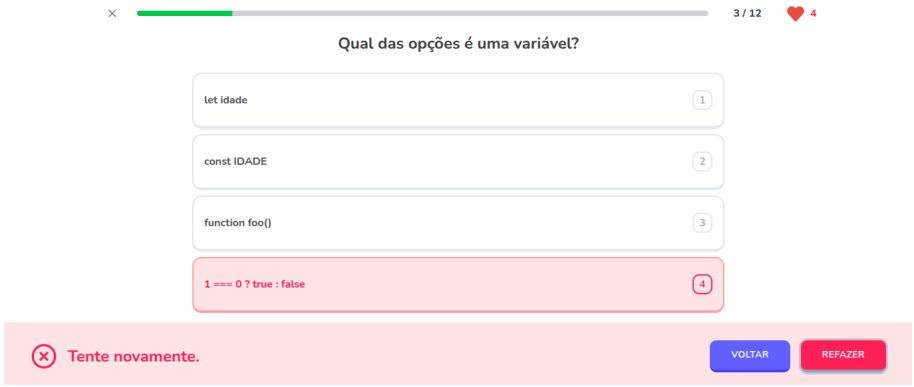

Figura 3.5: Atividade de múltipla escolha respondida de forma incorreta.

As atividades de escrita de código, ou atividades abertas, representam o nível mais avançado de interação disponível na plataforma. Nesse formato, o estudante deve desenvolver uma solução completa para um problema proposto, utilizando os conhecimentos adquiridos ao longo das etapas anteriores.

A conclusão da atividade ocorre quando todas as tarefas propostas são finalizadas. Para isso, a plataforma avalia o código desenvolvido pelo estudante, ao clicar em executar ou em responder, identificando quais requisitos foram atendidos e marcando as respectivas tarefas como concluídas. As atividades abertas contam também com dicas, auxiliando no desenvolvimento da solução proposta.

Esse modelo busca estimular a autonomia do estudante e aproximar as atividades das situações reais encontradas no desenvolvimento de software. A Figura 3.6 mostra a

interface de uma atividade aberta.

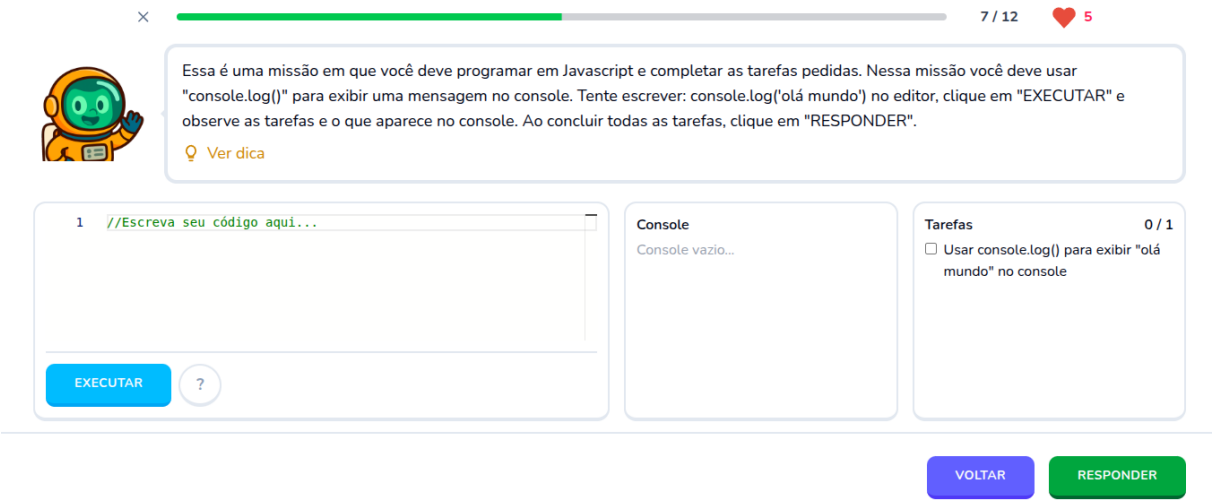

Figura 3.6: Atividade aberta.

Ao concluir todas as atividades do planeta, a plataforma exibe uma mensagem de parabenização (Figura 3.7) e retorna para a tela de seleção de planetas.

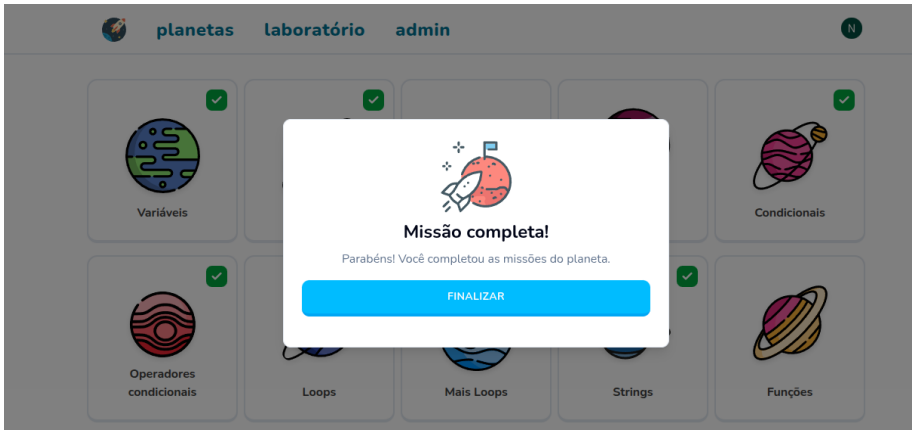

Figura 3.7: Mensagem de parabenização por completar um planeta.

3.3.3 Laboratório

Além das atividades estruturadas, o Scriptonauta disponibiliza a funcionalidade Laboratório, acessível a partir da tela de seleção de planetas. Esse ambiente foi desenvolvido com o objetivo de oferecer aos estudantes um espaço livre para experimentação e prática de programação.

No Laboratório, os usuários podem escrever e executar códigos em JavaScript sem

a necessidade de seguir um enunciado específico ou cumprir tarefas predefinidas. Dessa forma, o ambiente funciona como um espaço de exploração, permitindo que os estudantes testem conceitos, realizem experimentos e pratiquem livremente os conteúdos aprendidos durante as missões.

A interface do Laboratório, apresentada na Figura 3.8, é semelhante à utilizada nas atividades de escrita de código, porém sem a presença de objetivos, instruções ou mecanismos de avaliação. Essa abordagem busca incentivar a autonomia dos estudantes, oferecendo um ambiente seguro para a construção e validação de soluções próprias.

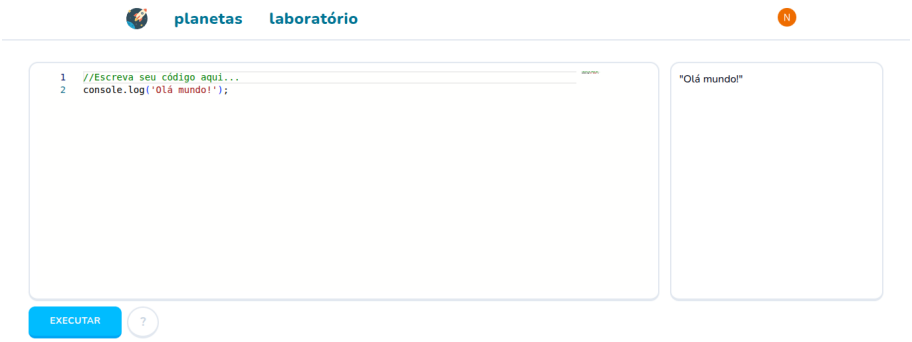

Figura 3.8: Interface do laboratório.

3.3.4 Administração

Além das funcionalidades destinadas aos estudantes, o Scriptonauta possui uma área de administração acessível a partir da tela de seleção de planetas. Essa funcionalidade é restrita a usuários com permissões especiais, responsáveis pela manutenção e gerenciamento dos conteúdos disponíveis na plataforma. A principal finalidade da área de administração é fornecer uma interface que permita realizar operações de criação, consulta, atualização e remoção *(Create, Read, Update and Delete – CRUD)* dos elementos que compõem o ambiente educacional. Dessa forma, novos conteúdos podem ser adicionados e modificados sem a necessidade de alterações diretas no código-fonte da aplicação, facilitando a expansão e manutenção da plataforma. A interface administrativa, exibida na Figura 3.9 foi organizada em quatro abas principais: **Planetas**, **Lições**, **Tarefas** e **Testes**, cada uma destinada a um conjunto específico de funcionalidades.

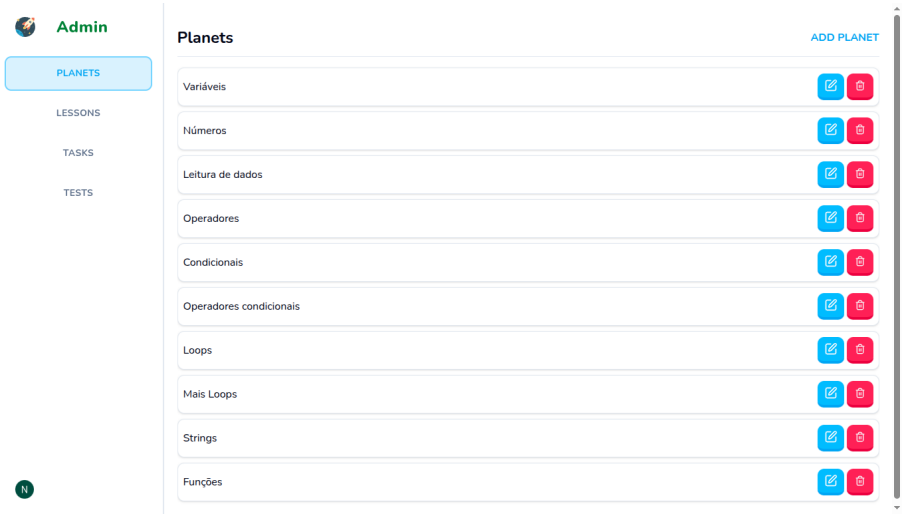

Figura 3.9: Interface de edição de planetas.

A aba de planetas permite o gerenciamento dos módulos de aprendizagem presentes no sistema. Por meio dessa interface, administradores podem adicionar novos planetas, editar informações de planetas existentes ou removê-los quando necessário.

Além das operações básicas de gerenciamento, essa aba também possibilita definir a ordem de exibição dos planetas na tela principal da plataforma. Essa funcionalidade é importante para garantir que a sequência de conteúdos siga a progressão pedagógica planejada para o processo de aprendizagem.

A aba de lições (Figura 3.10) é responsável pelo gerenciamento dos conteúdos internos de cada planeta.

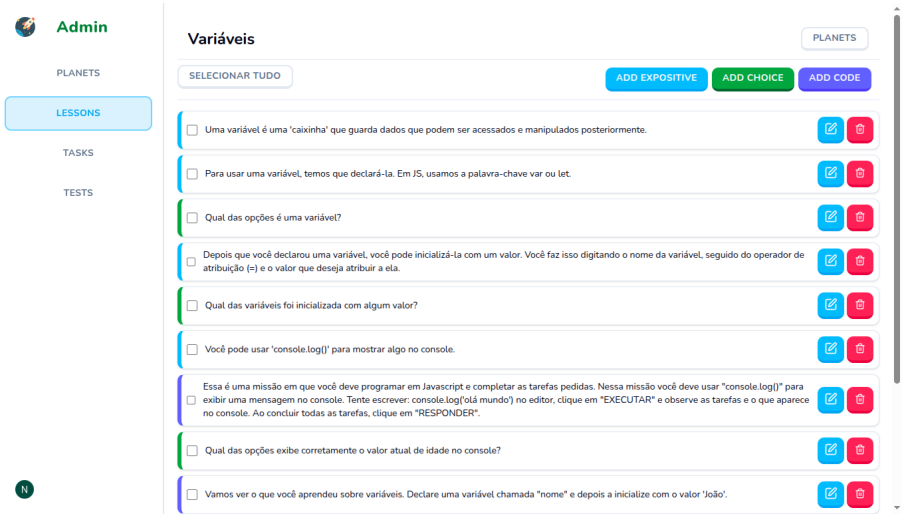

Figura 3.10: Interface da edição de lições.

Nessa interface, administradores podem criar novas lições, editar conteúdos existentes e remover materiais que não sejam mais necessários. Por meio dessa funcionalidade, é possível organizar a sequência de atividades e conteúdos associados a cada planeta, permitindo a constante atualização do material disponibilizado aos estudantes.

A aba de tarefas foi desenvolvida especificamente para o gerenciamento das tarefas associadas às lições do tipo aberta. Durante o desenvolvimento da plataforma, verificou-se a necessidade de uma interface dedicada para essa finalidade, uma vez que as tarefas possuem características e parâmetros próprios que demandam mecanismos específicos de edição.

Nessa seção, os administradores podem criar, modificar e remover tarefas, bem como configurar os critérios utilizados pelo sistema para verificar sua conclusão. A existência de uma área exclusiva para esse gerenciamento contribui para uma manutenção mais eficiente dos conteúdos avaliativos da plataforma. A Figura 3.11 mostra a interface da aba de tarefas.

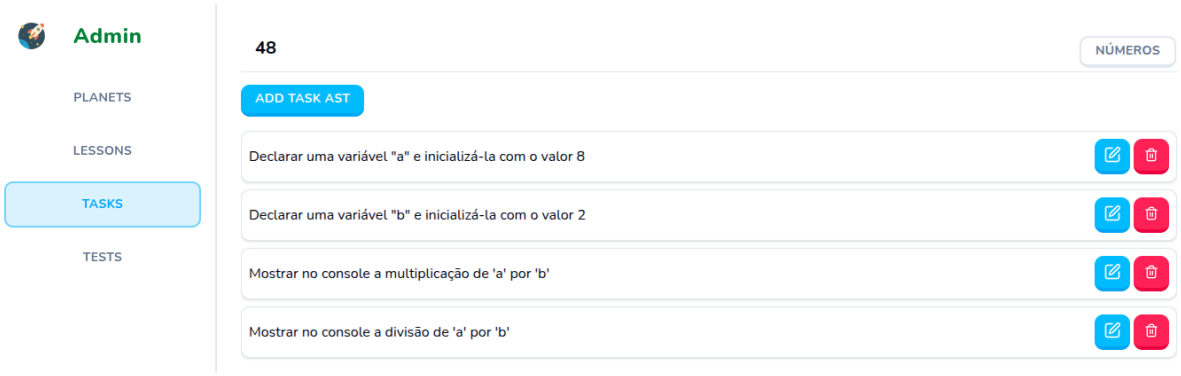

Figura 3.11: Interface de edição de tarefas.

A aba de testes foi criada com o objetivo de agilizar o processo de validação das lições do tipo aberta. Por meio dessa funcionalidade, os administradores podem selecionar individualmente qualquer lição cadastrada na plataforma e executá-la em um ambiente de teste.

Essa abordagem permite verificar o funcionamento das tarefas, validar critérios de avaliação e identificar possíveis inconsistências antes que os conteúdos sejam disponibilizados aos estudantes. Dessa forma, a aba de testes, observada na Figura 3.12, atua como uma ferramenta de apoio ao processo de manutenção e garantia da qualidade dos

materiais educacionais presentes no Scriptonauta.

Figura 3.12: Interface da área de testes.

A área de administração desempenha um papel fundamental na manutenção e evolução da plataforma, permitindo que novos conteúdos sejam incorporados de forma ágil e organizada. Além disso, sua implementação reduz a dependência de intervenções diretas no código do sistema, tornando o processo de atualização mais acessível para usuários responsáveis pela gestão pedagógica do Scriptonauta.

3.4 Análise de código

Um dos principais desafios no desenvolvimento do Scriptonauta foi a implementação de mecanismos capazes de avaliar automaticamente as soluções produzidas pelos estudantes nas atividades abertas. Diferentemente das questões de múltipla escolha, esse tipo de atividade exige que a plataforma interprete e analise códigos escritos livremente pelos usuários, verificando não apenas o resultado produzido, mas também determinadas características estruturais da solução.

Com o objetivo de oferecer uma avaliação mais flexível e adequada aos diferentes tipos de desafios propostos, foram implementadas duas abordagens complementares de

análise de código: a análise baseada em Árvore Sintática Abstrata (Abstract Syntax Tree – AST) e a análise baseada em testes automatizados.

3.4.1 Análise Baseada em *AST*

A primeira abordagem utiliza a representação do código na forma de uma Árvore Sintática Abstrata (*AST*). Nesse modelo, o código-fonte escrito pelo estudante é convertido em uma estrutura hierárquica composta por nós que representam os diferentes elementos da linguagem de programação, como declarações de variáveis, estruturas condicionais, laços de repetição e chamadas de funções.

A utilização da *AST* permite que o sistema avalie características estruturais da solução sem depender exclusivamente do resultado final produzido pelo programa. Dessa forma, é possível verificar se o estudante utilizou determinados conceitos exigidos pela atividade.

Por exemplo, em um exercício cujo objetivo seja praticar estruturas de repetição, a plataforma pode analisar a árvore sintática em busca da presença de comandos como *for* ou *while*. De maneira semelhante, atividades voltadas ao uso de condicionais podem exigir a utilização de instruções *if* ou *switch*.

Essa abordagem é especialmente útil em contextos educacionais, pois permite avaliar o processo de construção da solução e não apenas sua saída final. Além disso, contribui para evitar situações em que o estudante obtenha o resultado correto utilizando mecanismos diferentes daqueles que a atividade pretende ensinar.

3.4.2 Análise Baseada em Testes

A segunda abordagem emprega testes automatizados para validar o comportamento do código submetido pelo estudante. Nesse modelo, a solução desenvolvida é executada em um ambiente controlado e submetida a um conjunto de verificações previamente definidas para cada atividade.

Cada teste representa um requisito específico que deve ser atendido pela solução. Durante a execução, o sistema compara os resultados produzidos pelo código com os resultados esperados para diferentes cenários de entrada.

Quando um teste é aprovado, a plataforma considera que o requisito correspondente foi satisfeito. Caso contrário, o estudante recebe *feedback* indicando que ainda existem aspectos da solução que precisam ser corrigidos.

Essa estratégia permite validar funcionalidades de forma objetiva, garantindo que o programa produzido apresente o comportamento esperado independentemente dos valores utilizados durante a execução.

Além disso, a abordagem baseada em testes possibilita a implementação das tarefas presentes nas atividades abertas. Cada tarefa pode ser associada a um ou mais testes específicos, sendo marcada automaticamente como concluída quando os critérios correspondentes são satisfeitos.

3.4.3 Conclusão

A adoção dessas técnicas permitiu que o Scriptonauta oferecesse correção automática para atividades abertas, fornecendo *feedback* imediato aos estudantes e reduzindo a necessidade de avaliação manual por parte dos professores. Dessa forma, a plataforma consegue apoiar o processo de aprendizagem de programação de maneira escalável e alinhada aos objetivos educacionais de cada atividade.

4 Avaliação

Este capítulo apresenta o processo de avaliação da plataforma desenvolvida, descrevendo os participantes envolvidos, o método adotado para coleta de dados e os resultados obtidos. O objetivo da avaliação foi investigar a percepção dos usuários sobre a experiência de utilização da plataforma, bem como identificar seus pontos fortes e oportunidades de melhoria.

4.1 Caracterização dos participantes

A avaliação foi realizada com estudantes matriculados em quatro turmas da disciplina DCC200 – Algoritmos II, oferecidas pelo Departamento de Ciência da Computação (DCC) da Universidade Federal de Juiz de Fora (UFJF).

A escolha desse público foi motivada pela proximidade entre o conteúdo abordado na disciplina e os objetivos da plataforma desenvolvida. Os estudantes de DCC200 já possuem conhecimentos introdutórios de programação, uma vez que a disciplina DCC199 – Algoritmos é pré-requisito obrigatório. Nessa disciplina introdutória, os alunos desenvolvem competências relacionadas aos fundamentos da programação, incluindo estruturas de controle, variáveis, funções, vetores e resolução de problemas computacionais utilizando a linguagem C++.

Dessa forma, espera-se que os participantes possuam experiência suficiente para interagir com os desafios e recursos oferecidos pela plataforma, permitindo uma avaliação mais consistente sobre sua usabilidade, utilidade pedagógica e potencial de apoio ao aprendizado de programação.

4.2 Método de avaliação

A avaliação foi conduzida por meio de uma abordagem qualitativa baseada na percepção dos usuários. Após utilizarem a plataforma, os participantes responderam a um ques-

tionário estruturado contendo afirmações relacionadas à sua experiência de uso.

Por se tratar de uma avaliação da plataforma voltada exclusivamente à coleta da percepção dos participantes, sem identificação dos respondentes, sem intervenção sobre os indivíduos e sem coleta de dados pessoais ou sensíveis, os participantes não foram submetidos à assinatura do Termo de Consentimento Livre e Esclarecido (TCLE).

Para mensurar o grau de concordância dos participantes com cada afirmação, foi utilizada uma escala Likert de cinco pontos. Para fins de análise quantitativa, as respostas foram convertidas em valores numéricos de 1 a 5, em que **1** corresponde a *Discordo totalmente*, **2** a *Discordo parcialmente*, **3** a *Nem concordo nem discordo*, **4** a *Concordo parcialmente* e **5** a *Concordo totalmente*.

A utilização da escala Likert permite capturar a percepção dos usuários de forma simples e amplamente aceita em pesquisas de avaliação de sistemas educacionais, possibilitando identificar tendências de satisfação, usabilidade e engajamento.

O questionário foi elaborado para avaliar diferentes aspectos da plataforma, incluindo facilidade de uso, clareza da interface, percepção de aprendizagem e elementos de gamificação, sendo organizado em cinco categorias de análise, descritas a seguir.

4.2.1 Perfil dos Participantes

Inicialmente, foram coletadas informações para caracterizar a amostra avaliada, incluindo:

- Curso atual do participante;
- Período acadêmico;
- Nível de familiaridade com a linguagem JavaScript;
- Nível de compreensão de lógica de programação.

Essas informações permitiram contextualizar os resultados obtidos e verificar se os participantes possuíam experiência suficiente para avaliar a plataforma.

4.2.2 Organização Pedagógica do Conteúdo

Esta categoria buscou avaliar a forma como os conteúdos foram estruturados e apresentados ao usuário. As seguintes afirmações foram analisadas:

- A divisão do conteúdo de JavaScript em planetas facilita a compreensão progressiva da linguagem;
- Os textos utilizados para explicar os conceitos de programação são adequados para a capacidade cognitiva de uma criança de 10 a 13 anos;
- As mecânicas, ferramentas e desafios permitem o aprendizado de forma leve e gradual.

4.2.3 Potencial Educacional

Esta categoria teve como objetivo avaliar a percepção dos participantes sobre a capacidade da plataforma de auxiliar no ensino de programação para o público-alvo proposto.

Foi considerada a seguinte afirmação:

- O uso deste sistema gamificado tem potencial para melhorar a compreensão de lógica de programação por crianças de 10 a 13 anos.

4.2.4 Gamificação e Engajamento

Esta categoria avaliou o potencial dos elementos de gamificação para incentivar a participação contínua dos usuários.

Foi considerada a seguinte afirmação:

- A mecânica de gamificação implementada tem um forte potencial para manter os usuários engajados.

4.2.5 Usabilidade e Experiência de Uso

Esta categoria avaliou aspectos relacionados à interação com o sistema e à qualidade da interface.

Foram consideradas as seguintes afirmações:

- A interface do aplicativo é intuitiva e a navegação é clara;
- O design visual é atrativo e adequado para o público infantil;
- O sistema tem um bom funcionamento, sem erros aparentes e bom tempo de resposta;
- O Scriptonauta é uma plataforma fácil de utilizar, sendo acessível para todos os públicos.

4.2.6 Questões Abertas

Ao final do questionário, foram disponibilizadas questões discursivas com o objetivo de coletar opiniões detalhadas dos participantes sobre a plataforma:

- Quais foram os pontos fortes que você identificou no sistema?
- Quais foram os pontos fracos que você identificou no sistema (incluindo bugs ou dificuldades de uso)?
- Quais funcionalidades você acredita que estão faltando ou precisam ser melhoradas na plataforma?
- Deixe aqui outras sugestões e comentários.

As respostas abertas foram analisadas qualitativamente, permitindo identificar padrões de percepção, sugestões de melhoria e possíveis limitações da plataforma que não seriam capturadas apenas pelas questões objetivas.

4.3 Resultados

Nesta seção são apresentados os resultados obtidos a partir da avaliação do Scriptonauta. Ao todo, foram coletadas 78 respostas de estudantes da disciplina DCC200 – Algoritmos II da Universidade Federal de Juiz de Fora.

A aplicação do questionário ocorreu em dois momentos distintos. Na primeira aplicação, realizada em três turmas da disciplina, foram obtidas 67 respostas. Após a

análise inicial dos resultados e dos problemas relatados pelos participantes, foram realizadas melhorias na plataforma, especialmente relacionadas à estabilidade do sistema e ao suporte a múltiplas conexões simultâneas. Uma semana depois, uma nova aplicação foi conduzida em uma quarta turma, resultando em mais 11 respostas.

Essa estratégia permitiu não apenas avaliar a percepção geral dos participantes, mas também observar o impacto das correções implementadas entre as duas etapas da avaliação.

4.3.1 Perfil dos participantes

A análise do perfil dos participantes permitiu compreender melhor o contexto acadêmico e o nível de experiência prévia com programação dos respondentes. Em relação à formação acadêmica, observou-se predominância dos cursos de Ciência da Computação e Ciências Exatas. Além desses, também foram citados, em menor frequência, os cursos de Estatística e Engenharia Computacional. Esse resultado indica uma amostra majoritariamente composta por estudantes de áreas diretamente relacionadas à computação e ao raciocínio quantitativo.

No que se refere ao nível de familiaridade com a linguagem JavaScript, observa-se uma predominância de participantes com conhecimentos iniciais. A maior parte dos respondentes declarou nível básico (48,7%), seguida por aqueles que não possuem familiaridade com a linguagem (33,3%). Uma parcela menor indicou nível intermediário (15,4%) e apenas 2,6% afirmou possuir nível avançado.

Quanto à compreensão geral de lógica de programação, os dados indicam um perfil mais distribuído em níveis intermediários. A maioria dos participantes declarou nível intermediário (60,3%), seguida por nível básico (29,5%) e nível avançado (10,3%).

4.3.2 Resultados da Avaliação

A Figura 4.1 apresenta as médias obtidas para cada questão do questionário de avaliação.

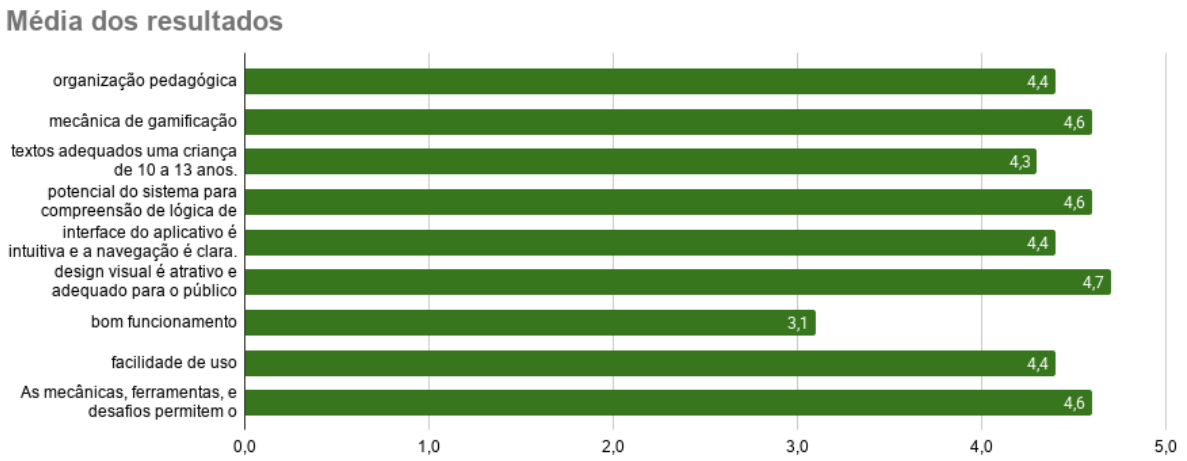

Figura 4.1: Medias das questões

De forma geral, os resultados indicam uma percepção bastante positiva dos participantes em relação ao Scriptonauta. Todas as questões, com exceção do funcionamento da plataforma, obtiveram médias superiores a 4,0, evidenciando elevado grau de concordância com as afirmações avaliadas.

Os aspectos mais bem avaliados foram o *design* visual da plataforma (4,7), o potencial educacional (4,6), os elementos de gamificação (4,6) e a organização do conteúdo em uma progressão gradual (4,6). Esses resultados sugerem que os participantes perceberam a plataforma como atrativa, motivadora e adequada para apoiar o ensino introdutório de programação.

A organização dos conteúdos em planetas (4,4), a interface intuitiva (4,4), a facilidade de uso (4,4) e a adequação dos textos ao público-alvo (4,3) também receberam avaliações positivas, indicando que a proposta pedagógica e a experiência de utilização foram bem aceitas pelos participantes.

O único aspecto com avaliação inferior foi o funcionamento da plataforma (3,1). Durante a primeira aplicação foram observados problemas de instabilidade da rede e limitações relacionadas ao suporte a múltiplos acessos simultâneos, o que ocasionou lentidão e falhas durante a utilização.

4.3.3 Comparação entre as Aplicações

Com o objetivo de avaliar o impacto das modificações implementadas após a primeira aplicação, foi realizada uma segunda rodada de avaliações com os participantes. A Figura 4.2 apresenta a comparação das médias obtidas em cada questão nas duas aplicações.

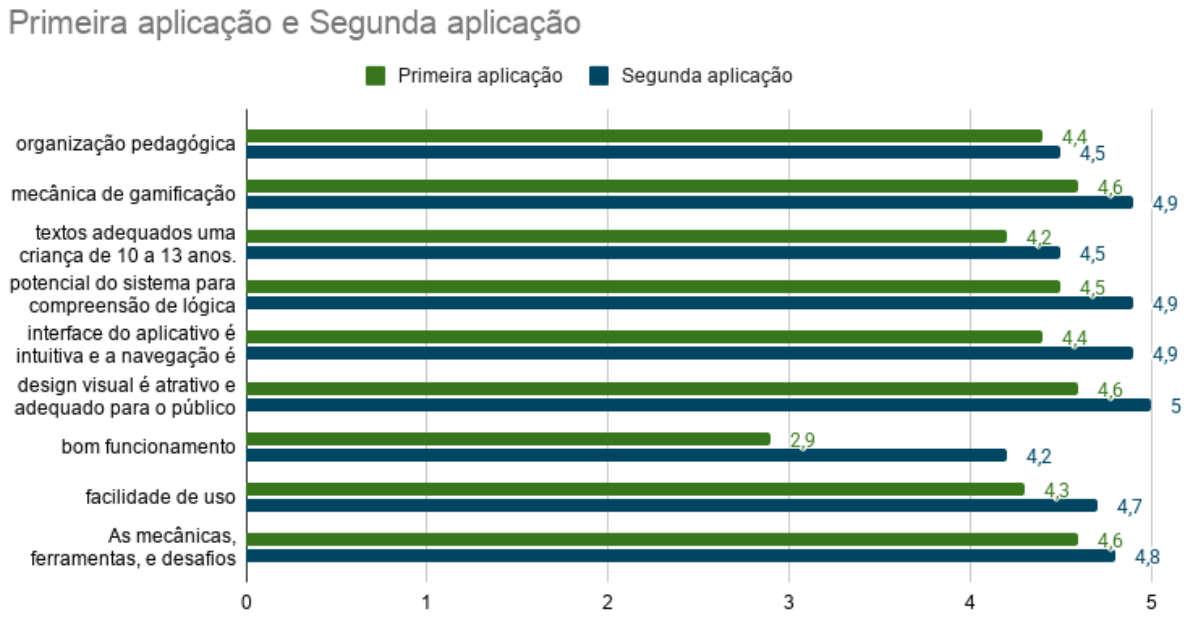

Figura 4.2: Comparação entre as duas aplicações

Após a apresentação dos resultados, observa-se que praticamente todas as questões apresentaram aumento em suas médias. A maior variação ocorreu na questão relacionada ao funcionamento da plataforma, cuja média passou de 2,9 na primeira aplicação para 4,2 na segunda. Esse resultado indica que as correções realizadas entre as avaliações foram eficazes para solucionar os principais problemas identificados pelos usuários durante a etapa inicial.

Também foram observadas melhorias na percepção do potencial educacional da plataforma, cuja média evoluiu de 4,5 para 4,9, bem como na avaliação dos elementos de gamificação, que passou de 4,6 para 4,9. Da mesma forma, a adequação dos conteúdos para o aprendizado gradual apresentou aumento de 4,6 para 4,8.

Outro resultado relevante refere-se ao *design* visual da plataforma, que atingiu média máxima (5,0) na segunda aplicação, reforçando a boa aceitação da interface desenvolvida.

De maneira geral, a comparação entre as duas aplicações evidencia que as melhorias implementadas contribuíram para elevar a percepção dos usuários quanto à qualidade da plataforma, sobretudo nos aspectos relacionados ao funcionamento, à estabilidade e à experiência de uso.

4.3.4 Síntese dos Resultados

Os resultados obtidos demonstram que o Scriptonauta foi bem recebido pelos participantes, alcançando médias superiores a 4,0 em oito das nove questões avaliadas. Os dados sugerem que a plataforma possui potencial para apoiar o ensino de programação para crianças, oferecendo uma experiência considerada intuitiva, visualmente atrativa e capaz de manter os usuários engajados.

Embora problemas técnicos tenham sido identificados durante a primeira aplicação, as melhorias realizadas posteriormente resultaram em avanços significativos na avaliação do sistema. Dessa forma, os resultados indicam que a plataforma apresenta características promissoras para utilização em contextos educacionais voltados ao ensino introdutório de programação.

5 Conclusão

Este trabalho apresentou o desenvolvimento, implementação e avaliação do Scriptonauta, uma plataforma educacional gamificada voltada ao ensino introdutório de programação em JavaScript. A proposta foi motivada pelas dificuldades enfrentadas por estudantes iniciantes na área de programação, especialmente relacionadas à abstração de conceitos, à interpretação de erros e à falta de *feedback* imediato durante o processo de aprendizagem.

A solução desenvolvida buscou enfrentar esses desafios por meio da integração de elementos pedagógicos e lúdicos, estruturando o conteúdo em uma narrativa espacial composta por planetas de aprendizagem. Essa organização permitiu a apresentação progressiva dos conceitos, reduzindo a carga cognitiva e favorecendo a assimilação gradual dos conteúdos. Além disso, a plataforma incorporou mecanismos de gamificação, como sistema de vidas, desafios progressivos e *feedback* automático, com o objetivo de aumentar o engajamento dos estudantes.

Do ponto de vista técnico, o Scriptonauta foi implementado como uma aplicação *web* utilizando tecnologias modernas, o que possibilitou sua execução em diferentes dispositivos e facilitou seu acesso pelos usuários. A inclusão de mecanismos de avaliação automática, baseados em análise de AST e testes automatizados, contribuiu para a oferta de *feedback* imediato e para a verificação objetiva das soluções desenvolvidas pelos estudantes.

A avaliação realizada com estudantes da disciplina de Algoritmos II indicou uma percepção geral positiva em relação à plataforma. Os resultados demonstraram boa aceitação quanto à organização pedagógica, ao potencial educacional e aos elementos de gamificação. Apesar disso, foram identificados problemas iniciais de desempenho e estabilidade, os quais impactaram a experiência de uso em uma primeira etapa da avaliação. Após a implementação de melhorias, observou-se uma evolução significativa nas avaliações, especialmente nos aspectos técnicos e de usabilidade.

De forma geral, os resultados obtidos indicam que o Scriptonauta possui potencial para contribuir com o ensino introdutório de programação, oferecendo uma abordagem

mais interativa, estruturada e motivadora. Como trabalhos futuros, pretende-se realizar a aplicação da plataforma com estudantes de 10 a 13 anos, público-alvo inicialmente definido para o sistema, a fim de avaliar sua efetividade no contexto para o qual foi concebido. Além disso, sugere-se a ampliação da plataforma com novos conteúdos, a realização de estudos com públicos mais diversos e a exploração de técnicas mais avançadas de análise automática de código para aprimorar ainda mais o processo de avaliação das atividades.

Bibliografia

Brasil. *Base Nacional Comum Curricular*. Brasília: Ministério da Educação, 2018.

CHOI, W. C.; CHOI, I. C. Exploring the impact of code.org's block-based coding curriculum on student motivation in k-12 education. In: *2024 12th International Conference on Information and Education Technology (ICIET)*. [S.l.: s.n.], 2024. p. 93–97.

DETERDING, S. et al. From game design elements to gamefulness: Defining gamification. *Proceedings of the 15th International Academic MindTrek Conference*, p. 9–15, 2011.

JENKINS, T. Teaching programming: A journey from teacher to motivator. *Proceedings of the 2nd Annual LTSN-ICS Conference*, p. 1–5, 2002.

KIRALY, S. et al. Enhancing sql learning: Gamified tutorials and flipped classroom synergy. *Social Sciences & Humanities Open*, v. 12, p. 101762, 2025. ISSN 2590-2911. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S2590291125004905⟩.

LIU, J.; BAI, Y.; XIA, X. Research on developing computational thinking of middle school students based on gamified text programming teaching activities. In: *2023 5th International Conference on Computer Science and Technologies in Education (CSTE)*. [S.l.: s.n.], 2023. p. 175–179.

NOBNOP, R. et al. Evaluating learner's usability using codenite educational battle royal programming language game. In: *2023 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*. [S.l.: s.n.], 2023. p. 7–10.

PAPERT, S. *Mindstorms: Children, Computers, and Powerful Ideas*. [S.l.]: Basic Books, 1980.

RESNICK, M. *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*. [S.l.]: MIT Press, 2017.

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

Sociedade Brasileira de Computação. *Diretrizes para Ensino de Computação na Educação Básica*. Porto Alegre, 2019.

SUN, L.; LIU, J. Effects of gamified python programming on primary school students' computational thinking skills: A differential analysis of gender. *Journal of Educational Computing Research*, v. 62, 12 2023.

SUN, L.; LIU, J. Comparative study of codecombat-based python programming and scratch programming effects on sixth graders' computational thinking: interaction effects of gender and programming experience. *Thinking Skills and Creativity*, v. 57, p. 101852, 2025. ISSN 1871-1871. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S1871187125001014⟩.

THANYAPHONGPHAT, J.; THONGKOO, K.; DAUNGCHARONE, K. A micropython-based educational robotic maze approach: Learning improvement to competition. In: *2023 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunications Engineering (ECTI DAMT & NCON)*. [S.l.: s.n.], 2023. p. 138–142.

WING, J. M. Computational thinking. *Communications of the ACM*, ACM, v. 49, n. 3, p. 33–35, 2006.

ZHAN, Z. et al. The effectiveness of gamification in programming education: Evidence from a meta-analysis. *Computers and Education: Artificial Intelligence*, v. 3, p. 100096, 2022. ISSN 2666-920X. Disponível em: ⟨https://www.sciencedirect.com/science/article/pii/S2666920X22000510⟩.