CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA

Faculdade de Tecnologia Baixada Santista Rubens Lara

Curso Superior de Tecnologia em Sistemas para Internet

LUCAS GRADILONE VALIAS

CODEJOURNEY

Aplicativo educacional para prática de programação.

Santos, SP
2025

LUCAS GRADILONE VALIAS

CODEJOURNEY
Aplicativo educacional para prática de programação.

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia Rubens Lara, como exigência para a obtenção do Título de Tecnólogo em Sistemas para Internet.

Orientador: Profa. Ma. Rosana Cammarosano
Co-Orientador: Prof.Danilo Bezerra

Santos, SP
2025

RESUMO

O CodeJourney é um aplicativo desenvolvido com o objetivo de auxiliar estudantes e profissionais da área de tecnologia da informação no aprendizado de lógica e linguagens de programação. A proposta é tornar o processo de estudo mais interativo e envolvente, utilizando uma abordagem gamificada que estimula o progresso por meio de desafios e recompensas. A aplicação oferece trilhas de aprendizado que vão do nível básico ao avançado, permitindo que o usuário evolua conforme supera os obstáculos propostos. Além disso, busca incentivar a prática contínua e pode ser usada como ferramenta pedagógica em ambientes educacionais, promovendo maior engajamento em sala de aula.

Palavras-chaves: Aplicativo. Programação. Gamificação. Educação. Aprendizado interativo.

ABSTRACT

CodeJourney is an application developed with the goal of assisting students and IT professionals in learning programming logic and languages. The project aims to make the study process more interactive and engaging by using a gamified approach that encourages progress through challenges and rewards. The app offers learning paths ranging from beginner to advanced levels, allowing users to advance as they overcome proposed tasks. Additionally, it promotes continuous practice and can be used as an educational tool in classroom settings, fostering greater engagement among students. The experience is personalized, and the content is structured gradually to provide efficient, playful, and accessible learning.

Keywords: Application. Programming. Gamification. Education. Interactive learning.

LISTA DE ABREVIATURAS E SIGLAS

SAMESP – SINDICATO DAS ENTIDADES MANTENEDORAS DE ESTABELECIMENTOS DE ENSINO SUPERIOR NO ESTADO DE SÃO PAULO 15
HTML– HYPERTEXT MARKUP LANGUAGE 18
UML – UNIFIED MODELING LANGUAGE 22
RF – REQUISITOS FUNCIONAIS 23
MER– MODELO ENTIDADE RELACIONAMENTO 31
API – APPLICATION PROGRAMMING INTERFACE 32
JWT– JSON WEB TOKEN 32
DTO– DATA TRANSFER OBJECT 32
HTTP– HYPERTEXT TRANSFER PROTOCOL 33
WAMMI– WEBSITE ANALYSIS AND MEASUREMENT INVENTORY 42

LISTA DE ILUSTRAÇÕES

Ilustração 1 – Gráfico pesquisa 1.....16
Ilustração 2 – Gráfico pesquisa 2.....17
Ilustração 3 – Mimo (logo).....18
Ilustração 4 – LeetCode (logo).....19
Ilustração 5 – Scratch (logo).....19
Ilustração 6 – Unity (logo).....20
Ilustração 7 – C# (logo).....21
Ilustração 8 – .NET (logo).....22
Ilustração 9 – Diagrama de classes.....23
Ilustração 10 – Diagrama de casos de uso.....26
Ilustração 11 – MER.....31
Ilustração 12 – Código cadastro.....33
Ilustração 13 – Código login.....33
Ilustração 14 – Código jornada.....34
Ilustração 15– Tela de login.....35
Ilustração 16 – Tela de Jornadas.....35
Ilustração 17 – Menu de fases.....36
Ilustração 18 – Etapa questionário.....36
Ilustração 19 – Etapa fluxograma.....37
Ilustração 20 – Etapa script.....37
Ilustração 21 – Finalização fase.....38
Ilustração 22 – Respostas primeira pergunta.....41
Ilustração 23 – Respostas segunda pergunta.....42
Ilustração 24 – Respostas terceira pergunta.....42
Ilustração 25 – Respostas quarta pergunta.....43

LISTA DE TABELAS

TABELA 01 – TABELA DE REQUISITOS FUNCIONAIS 17
TABELA 02 – TABELA DE REQUISITOS NÃO FUNCIONAIS 19
TABELA 03 – TABELA DE TAREFAS DO TESTE 39

SUMÁRIO

1 INTRODUÇÃO 9
1.1 OBJETIVO 11
1.1.1 OBJETIVO GERAL 11
1.1.2 OBJETIVOS ESPECÍFICOS 11
1.2 ESTADO DA ARTE 12
2 DESENVOLVIMENTO 20
2.1 ANÁLISE DO SISTEMA 20
2.1.1 ANÁLISE DE REQUISITOS 23
2.1.2 DIAGRAMA DE CASO DE USO 25
2.1.3 FLUXO DE EVENTOS 26
2.2 BANCO DE DADOS 31
2.3 CAMADA DE NEGÓCIO 32
2.3.1 ARQUITETURA DA CAMADA DE NEGÓCIO 32
2.4 CAMADA DE APRESENTAÇÃO 35
3 RESULTADO 40
3.1 TESTES 40
3.2 QUESTIONÁRIO DE AVALIAÇÃO 42
3.3 CONCLUSÃO 45
REFERÊNCIAS BIBLIOGRÁFICAS 46
APÊNCIDE A 47

1 INTRODUÇÃO

A aprendizagem da programação representa um dos maiores desafios no campo da educação em tecnologia. Diferentemente de outras disciplinas, programar exige não apenas a assimilação de conceitos teóricos, mas também o desenvolvimento de habilidades práticas relacionadas ao pensamento lógico, abstração, resolução de problemas e domínio de linguagens formais. Essa combinação de competências torna o processo cognitiva e emocionalmente exigente, especialmente para iniciantes. A dificuldade em lidar com erros, interpretar mensagens técnicas e construir soluções coerentes desde as etapas iniciais contribui para altas taxas de frustração e evasão. Diante desse cenário, torna-se fundamental investigar os fatores que dificultam esse processo e buscar estratégias pedagógicas que tornem o ensino da programação mais acessível, eficaz e motivador.

Oram e Wilson (2010) ressaltam que “a dificuldade da programação não está em escrever código, mas em compreender e controlar a complexidade envolvida”. Isso reforça a ideia de que o maior desafio para quem aprende a programar está em organizar pensamentos complexos e traduzir problemas reais em soluções estruturadas. Por isso, é fundamental desenvolver estratégias pedagógicas que facilitem esse processo, tornando o aprendizado mais acessível e envolvente. Abordagens didáticas que atraiam e motivem os estudantes são essenciais para reduzir a taxa de desistência entre programadores iniciantes, garantindo uma formação mais sólida e duradoura. No Brasil, a situação é preocupante: dados do Instituto Semesp indicam que, em 2017, 66,5% dos alunos que ingressaram em cursos de Ciência da Computação, Design de Games ou Sistemas de Informação desistiram antes de concluir a graduação, uma taxa significativamente superior à média nacional de 55,5%, de acordo com o relatório do instituto.

Considerando as dificuldades e altas taxas de desistência no aprendizado de programação, a gamificação surge como uma estratégia pedagógica eficaz para melhorar o engajamento e a retenção dos estudantes. Ao incorporar elementos típicos de jogos, como recompensas, níveis, desafios e feedback imediato, essa abordagem torna o aprendizado mais motivador e dinâmico, facilitando a fixação de conceitos complexos. Diante disso, o presente trabalho apresenta o CodeJourney, um software

desenvolvido para proporcionar uma experiência de aprendizado diferenciada e gamificada em programação.

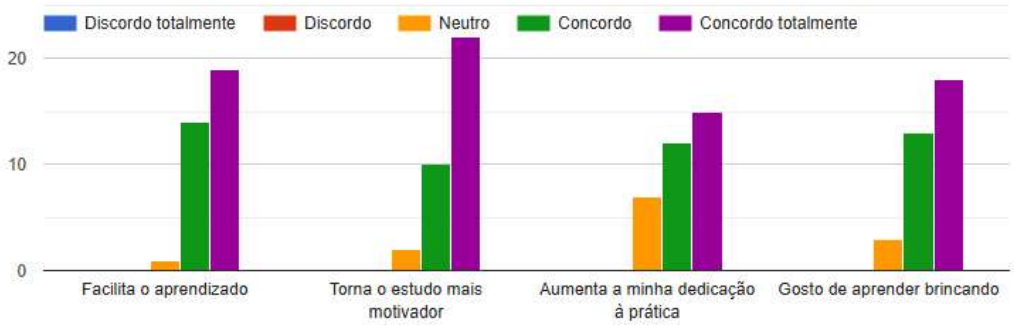
Para compreender melhor esse cenário, foi realizada uma pesquisa, para este projeto, com 34 estudantes de programação, na qual 91% dos participantes relataram considerar o aprendizado da programação uma tarefa difícil, como mostra a ilustração 1 abaixo.

Ilustração 1 – Gráfico pesquisa 1

Fonte: Autor, 2025

Além disso, a pesquisa incluiu perguntas sobre a percepção dos estudantes em relação a aspectos que poderiam melhorar seu aprendizado. Conforme mostra o gráfico da ilustração 2 a seguir, a maioria dos participantes concordou que certas estratégias podem facilitar o aprendizado, tornar o estudo mais motivador, aumentar a dedicação à prática e que aprender de forma lúdica é uma abordagem apreciada. Esses resultados indicam uma tendência clara de valorização de métodos que aliem motivação e engajamento, reforçando a importância de incorporar elementos gamificados e interativos no ensino de programação.

Ilustração 2 – Gráfico pesquisa 2



Fonte: Autor, 2025

1.1 OBJETIVO

Esta seção apresenta uma visão completa dos objetivos gerais e específicos do projeto, incluindo as métricas adotadas para sua realização.

1.1.1 OBJETIVO GERAL

O CodeJourney tem como objetivo principal oferecer uma plataforma interativa que auxilie estudantes e profissionais da área de Tecnologia da Informação no desenvolvimento e aprimoramento de habilidades em lógica e linguagens de programação. A proposta é abranger desde os conceitos mais básicos até os mais avançados, por meio de uma abordagem progressiva estruturada em fases e desafios gamificados. Cada etapa superada proporciona recompensas simbólicas, incentivando o engajamento contínuo do usuário.

Além de promover a prática autônoma, o CodeJourney também visa atuar como uma ferramenta pedagógica complementar em ambientes educacionais, contribuindo para a aprendizagem ativa. Através de sua metodologia lúdica e interativa, a plataforma busca despertar o interesse de novos estudantes, facilitando a assimilação de conteúdos complexos e promovendo um ambiente de estudo mais dinâmico e motivador. Dessa forma, pretende-se também fomentar o pensamento lógico, a resolução de problemas e a autonomia no processo de aprendizagem.

1.1.2 OBJETIVOS ESPECÍFICOS

Para garantir que a solução atenda às reais necessidades do público-alvo e seja eficaz tanto em ambientes de autoaprendizado quanto em contextos pedagógicos, foram definidos os seguintes objetivos específicos:

- Investigar e selecionar as metodologias de ensino de programação mais eficazes, com base em estudos teóricos e práticos, visando embasar o desenvolvimento pedagógico da plataforma;
- Analisar plataformas educacionais semelhantes, avaliando recursos e metodologias que possam ser adaptados ou aprimorados na proposta do CodeJourney;
- Desenvolver uma plataforma interativa com desafios gamificados, que estimulem o aprendizado progressivo de forma prática, acessível e motivadora;

1.2 ESTADO DA ARTE

Para apoiar o desenvolvimento do CodeJourney, foi feita uma pesquisa sobre aplicativos já existentes que ensinam programação. Essa análise ajuda a entender quais recursos e métodos são usados por outras plataformas e como eles podem inspirar novas ideias para o projeto.

Um exemplo é o aplicativo Mimo, como mostra a ilustração 3. Ele ensina programação de forma prática e acessível, especialmente para iniciantes. O conteúdo é dividido em trilhas com pequenos módulos que abordam lógica e linguagens como Python, JavaScript e HTML. O Mimo também inclui exercícios curtos ao longo das lições, permitindo que o aluno pratique logo após aprender. Além disso, usa gamificação para manter o usuário motivado, com pontos, níveis e conquistas. Esse tipo de abordagem interativa e dinâmica tem servido de referência para o desenvolvimento de ferramentas educacionais, como o CodeJourney.

Ilustração 3 – Mimo (logo)

Fonte: https://play.google.com/store/

A plataforma LeetCode, apresentada na ilustração 4 também reflete essa tendência ao transformar o estudo da programação em uma experiência mais interativa e desafiadora. Através de exercícios práticos organizados por níveis de dificuldade, rankings, conquistas e feedback imediato, o LeetCode consegue motivar os usuários a manterem uma rotina constante de prática. Embora não seja um jogo, o sistema de recompensas e progresso estimula o engajamento de maneira semelhante à gamificação. Esse tipo de abordagem mostra como elementos lúdicos e competitivos podem ser incorporados com sucesso no ensino de programação,

tornando o aprendizado mais atraente, mesmo para usuários com níveis variados de experiência.

Ilustração 4 – LeetCode (logo)

Fonte: https://leetcode.com/

Entre as funcionalidades presentes no CodeJourney, destaca-se o uso de uma abordagem visual inspirada na plataforma Scratch, como mostra a ilustração 5 a seguir, amplamente reconhecida por facilitar o ensino de lógica de programação por meio da montagem de fluxos e blocos de código. Assim como no Scratch, o CodeJourney permite que os usuários resolvam problemas estruturando algoritmos de forma interativa, conectando blocos que representam instruções lógicas. Essa metodologia torna o aprendizado mais acessível, especialmente nos estágios iniciais, ao abstrair a complexidade da sintaxe e focar no raciocínio computacional.

Ilustração 5 – Scratch (logo)

Fonte: https://scratchbrasil.net.br/

2 DESENVOLVIMENTO

Nesse capítulo será abordado as fases do desenvolvimento do sistema proposto. Para tanto, esse capítulo será subdividido nos seguintes itens; análise do sistema, a estrutura do Banco de Dados, as ferramentas utilizadas para o desenvolvimento do *front-end* e as principais rotinas implementadas no *back-end*.

2.1 ANÁLISE DO SISTEMA

Para o desenvolvimento de uma interface interativa e que entregue uma experiência gameficada satisfatória foram selecionadas as seguintes tecnologias:

O Unity, Ilustração 6, será a principal ferramenta utilizada no desenvolvimento deste projeto. Reconhecida mundialmente como uma das plataformas mais utilizadas na criação de jogos digitais, a *engine* destaca-se pela facilidade de uso, vasta documentação e uma comunidade ativa de suporte. Além disso, possibilita o desenvolvimento de experiências interativas e imersivas de forma intuitiva, oferecendo recursos visuais avançados e uma ampla biblioteca de ativos, atendendo perfeitamente às necessidades do aplicativo proposto.

Segundo Hocking (2018), "O Unity é uma engine que oferece uma infinidade de recursos úteis para diferentes jogos, permitindo adicionar apenas os elementos visuais e códigos específicos de cada projeto." Dessa forma, o Unity fornece recursos prontos, como física, renderização e suporte multiplataforma, ao mesmo tempo em que permite personalização, conexão servidores e banco de dados. Essa flexibilidade torna-o ideal para o desenvolvimento de soluções gamificadas, que exigem interatividade e adaptação às necessidades do usuário

Ilustração 6 – Unity (logo)

Fonte: https://unity.com

O *C#*, ilustração 7, é a linguagem de programação ideal para o desenvolvimento com Unity devido à sua forte orientação a objetos, que facilita a organização e modularização do código. Como observa Hocking (2018), " C# é a linguagem principal para Unity, oferecendo uma sintaxe poderosa, flexível e de fácil compreensão para os desenvolvedores criarem códigos estruturados e de fácil manutenção. " (p. 45). Sua sintaxe clara e recursos avançados permitem que os desenvolvedores manipulem de maneira eficiente comportamentos complexos e interações em tempo real.

Além disso, o C# se integra perfeitamente ao Unity, proporcionando uma plataforma robusta e flexível para criar experiências interativas e imersivas. Ele permite personalizar elementos visuais e códigos específicos de cada projeto, enquanto mantém a eficiência no gerenciamento de interações.

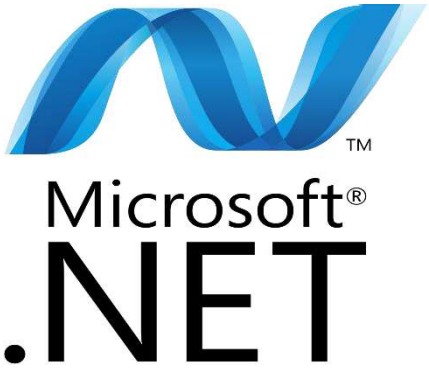
Ilustração 7 – C# (logo)

Fonte: https://itsoftware.com.co

O *.NET,* ilustração 8 é um *frameworker* eficiente e versátil para o desenvolvimento de servidores *backend*, proporcionando alto desempenho e integração simplificada com bancos de dados e serviços externos. Segundo a Microsoft (2025), o .NET é "uma plataforma de aplicativos segura, confiável e de alto desempenho". Essa capacidade de criar sistemas escaláveis e de fácil manutenção é essencial para o suporte de soluções gamificadas, que exigem alta interação e dinamismo.

Além disso, o .NET, com sua forte integração ao C#, oferece uma arquitetura orientada a objetos, permitindo maior organização e reutilização de código. Sua flexibilidade e compatibilidade com diversas tecnologias tornam-no ideal para criar backends robustos, que possam crescer e se adaptar às necessidades de projetos complexos e interativos.

Ilustração 8 – .NET (logo)



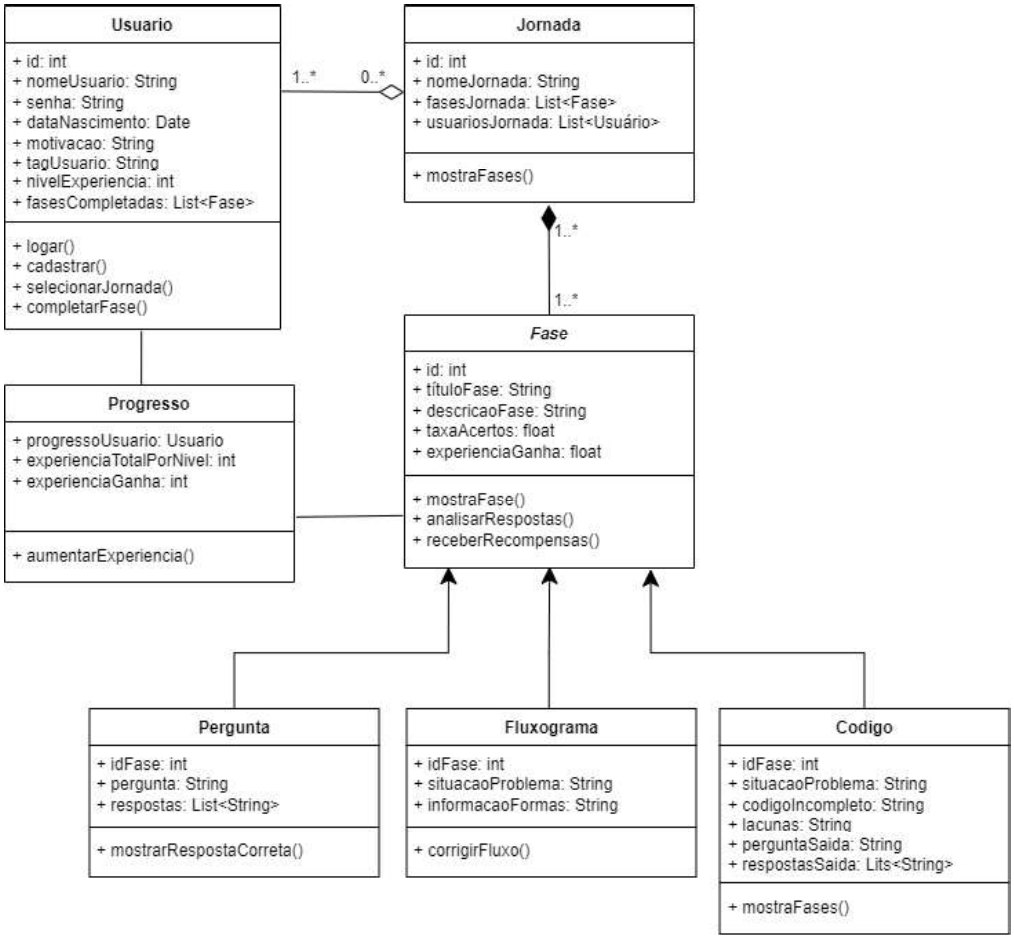
Fonte: https://dotnet.microsoft.com

Entre os diversos mecanismos disponíveis para a análise e modelagem no desenvolvimento de sistemas, optou-se pela utilização da *Unified Modeling Language* (UML). Trata-se de uma linguagem de modelagem padronizada, amplamente adotada na engenharia de software, cuja principal função é permitir a visualização, especificação, construção e documentação de sistemas de forma clara e organizada. A UML oferece um conjunto de notações gráficas que representam tanto a estrutura estática quanto o comportamento dinâmico dos sistemas, contribuindo significativamente para a comunicação entre membros da equipe de desenvolvimento e demais envolvidos no projeto.

Por meio de diagramas específicos, como os de classes, casos de uso, atividades, entre outros, é possível obter uma visão mais precisa dos componentes do sistema e de suas inter-relações. Isso facilita não apenas o planejamento e a implementação, mas também a manutenção e a escalabilidade do software ao longo do tempo.

> A UML – Unified Modeling Language ou Linguagem de Modelagem Unificada – é uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios de aplicação. Essa linguagem tornou-se, nos últimos anos, a linguagem-padrão de modelagem adotada internacionalmente pela indústria de engenharia de software. (GUEDES, 2009, p.19)

Iustração 9 – Diagrama de classes



Fonte – Autor, 2025

2.1.1 ANÁLISE DE REQUISITOS

Nesta fase ocorre a captura das intenções e as necessidades dos usuários a ser desenvolvido através dos requisitos funcionais (RF), apresentado na tabela 1.

> Requisitos são uma condição ou capacidade que deve ser alcançada ou possuída por um sistema, produto, serviço, resultado ou componente para satisfazer um contrato, padrão, especificação ou outro documento formalmente imposto. Requisitos incluem as necessidades quantificadas e documentadas, desejos e expectativas do patrocinador, clientes e outras partes interessadas. (VAZQUEZ; SIMÕES, 2016, p. 55)

Tabela 1 – Tabela de requisitos funcionais

ID	REQUISITOS FUNCIONAIS	DESCRIÇÃO
RF01	Cadastro	O usuário pode criar uma conta com nome de usuário e senha e definir seus dados e preferências
RF02	Login e Autenticação	O usuário pode acessar com seu nome de usuário e senha definidos no cadastro
RF03	Perfil de Usuário	O usuário pode criar, editar e excluir seu perfil com informações pessoais e preferências de aprendizado. O perfil exibe o progresso e estatísticas.
RF04	Configurações de conta	O usuário pode alterar suas configurações da conta editando suas credenciais, fazendo logout ou excluindo do sistema.
RF05	Seleção de Jornada de Aprendizado	O usuário escolhe entre diferentes jornadas de aprendizado, organizadas em níveis e fases.
RF06	Fases de Aprendizado	Cada fase contém exercícios interativos sobre tópicos de programação para o usuário realizar com feedback e correção
RF07	Sistema de Recompensas	O usuário ganha pontos de experiência, emblemas e recompensas ao completar fases e jornadas
RF08	Progresso	O aplicativo rastreia e exibe o progresso do usuário em cada jornada, com fases completadas e disponíveis.

Fonte: Autor, 2025

Para a criação do aplicativo se faz necessária também a análise dos requisitos não funcionais. Por definição de Vazques e Simões (2016, p. 167), requisitos não

funcionais indicam restrições de ordem geral que abordam aspectos relativos: ao ambiente, à organização, à implementação e à qualidade. A tabela 2 abaixo demonstra os requisitos não funcionais levantados para o aplicativo *CodeJourney.*

Tabela 2 – Tabela de requisitos não funcionais

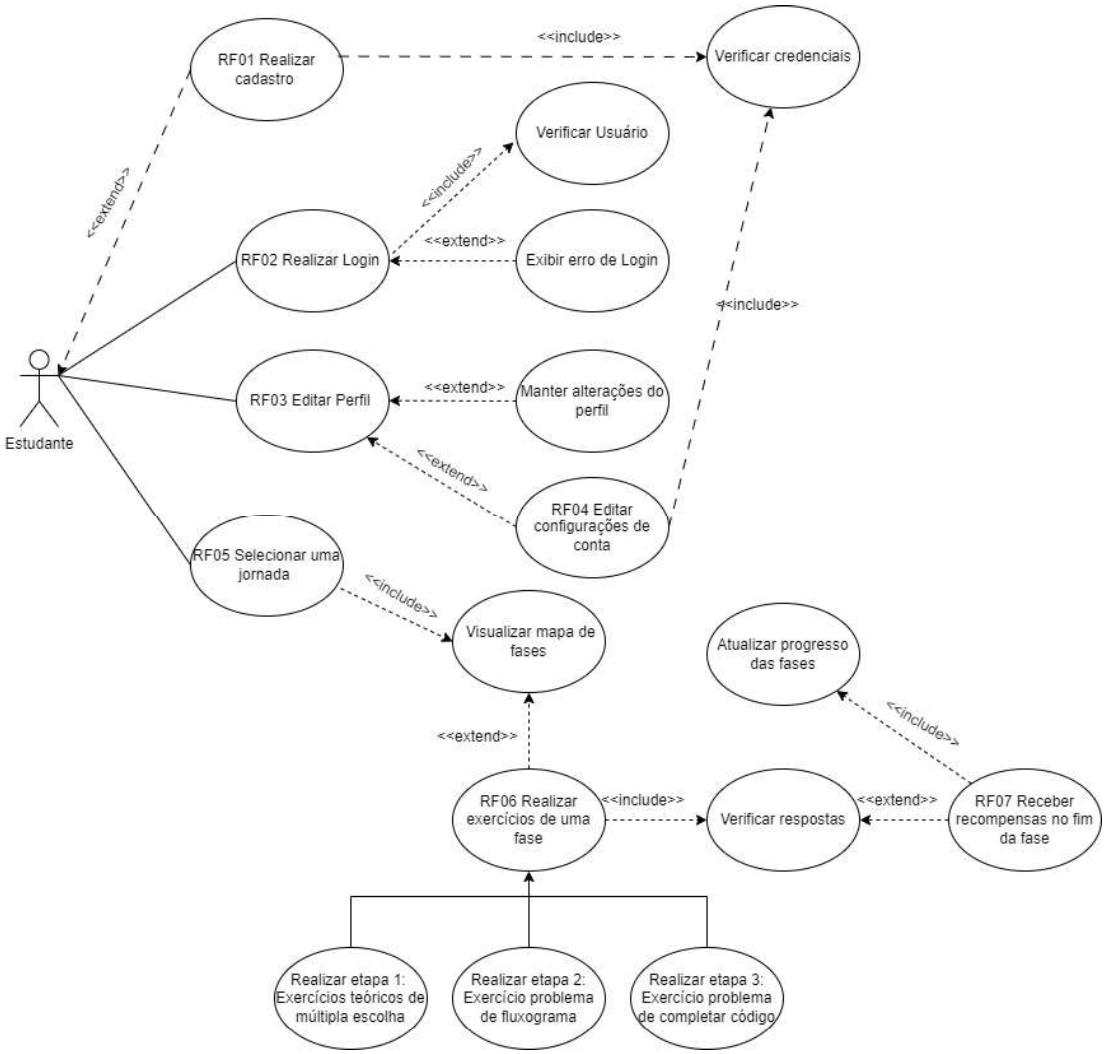
ID	REQUISITOS NÃO FUNCIONAIS	DESCRIÇÃO
RNF01	Usabilidade	A interface do usuário deve ser intuitiva e fácil de usar.
RNF02	Desempenho	O aplicativo deve ter um tempo de resposta rápido para interações e avaliações das fases
RNF03	Segurança	O aplicativo deve garantir a segurança dos dados do usuário
RNF04	Disponibilidade	O aplicativo deve garantir alta disponibilidade, assegurando que esteja acessível e operacional na maior parte do tempo.
RNF05	Escalabilidade	O aplicativo deve ser capaz de suportar um aumento no número de usuários e na quantidade de dados sem perda significativa de desempenho.

Fonte: Autor, 2025

2.1.2 DIAGRAMA DE CASO DE USO

Com os requisitos funcionais estabelecidos, o próximo passo é a criação do diagrama de caso de uso que tem como finalidade a representação gráfica e semântica da interação do usuário e o sistema. No diagrama mostrado na ilustração 10, os ícones representam os atores, como os usuários e sistemas externos, enquanto as ações possíveis do sistema são indicadas por elipses. As conexões entre os atores e os casos de uso destacam a interação entre eles, ajudando a entender quais atividades cada um realiza dentro do sistema.

Ilustração 10 – Diagrama de casos de uso



Fonte: Autor, 2025

2.1.3 FLUXO DE EVENTOS

O fluxo de eventos tem por intenção caracterizar o início e o término do caso de uso, estabelecer a interação dos atores no processo em descrição, gerando assim um fluxo principal e um fluxo alternativo do comportamento de cada caso de uso do sistema. Abaixo se encontra a descrição detalhada da descrição de cada um dos fluxos de eventos.

Caso de Uso 1: Realizar cadastro

Ator: Estudante

Fluxo Básico:

1. Estudante acessa o sistema e seleciona a opção de cadastro na tela de *login*

2. O aplicativo redireciona o usuário para a tela de cadastro e faz uma série de perguntas para criar o perfil do estudante
3. O estudante preenche um formulário com suas credenciais de nome de usuário e senha
4. O sistema emite mensagem de cadastro concluído e redireciona novamente para tela de *login*

Fluxo alternativo:

3. O sistema detecta que já existe um usuário com aquelas credenciais

3.1. O estudante preenche as informações novamente até as credenciais forem diferentes das registradas pelo sistema.

3.2. Retorna ao fluxo básico no passo 3

Caso de Uso 2: Realizar login

Ator: Estudante

Fluxo Básico:

1. O sistema solicita que o usuário insira seu nome de usuário e senha.
2. O estudante preenche os campos de nome de usuário e senha.
3. O estudante clica no botão "Entrar"
4. O sistema valida as credenciais no banco de dados.
5. Se as credenciais forem válidas, o sistema autentica o usuário e redireciona para a tela principal do aplicativo.

Fluxo alternativo:

5. O sistema detecta que alguma credencial está incorreta

5.1. O sistema exibe uma mensagem de erro e solicita que o estudante refaça suas credenciais

5.2. Retorna ao fluxo básico no passo 2

Caso de Uso 3: Editar Perfil

Ator: Estudante

Fluxo Básico:

1. O estudante navega até a tela de "Perfil" onde consegue visualizar as informações cadastradas do seu perfil e algumas estatísticas

2. O estudante altera os campos desejados selecionando a opção de edição
3. O estudante clica no botão "Salvar" para registar suas mudanças
4. O estudante é redirecionado para a tela de visualização do perfil com as alterações aplicadas

Caso de Uso 4: Configurações da conta

Ator: Estudante

Fluxo Básico:

1. O usuário seleciona a opção de configurações
2. O sistema redireciona para a página de configurações da conta
3. O estudante clica no botão de editar credenciais
4. O estudante altera suas credenciais
5. O sistema valida as credenciais no banco de dados.
6. O sistema exibe uma confirmação de alteração

Fluxo alternativo:

2A. O estudante clica no botão sair da conta

2.1A. O sistema exibe uma tela de confirmação

2.2A. O sistema faz logout da conta ativa

2.3A. O sistema redireciona o estudante para a tela de login

2B. O estudante clica no botão excluir conta

2.1B. O sistema exibe uma tela de confirmação

2.2B. O sistema excluí a conta do banco de dados

2.3B. O sistema redireciona o estudante para a tela de login

5. O sistema detecta que alguma credencial já está sendo utilizada

5.1. O sistema exibe uma mensagem de erro e solicita que o estudante reescreva novas credenciais válidas

5.2. Retorna ao fluxo básico no passo 2

Caso de Uso 5: Selecionar uma jornada

Ator: Estudante

Fluxo Básico:

1. O usuário visualiza a tela de Jornadas com opções de jornada de fases baseadas em linguagens de programação diferentes

2. O estudante seleciona uma Jornada
3. O estudante confirma sua seleção pressionando um botão de confirmação
4. O sistema redireciona a tela para o mapa de fases daquela jornada

Caso de Uso 6: Realizar exercícios de uma fase

Ator: Estudante

Fluxo Básico:

1. No mapa de fases o estudante seleciona uma das fases
2. Ao clicar, o sistema redireciona o estudante para a tela de fase
3. O sistema exibe para o estudante uma questão com quatro opções de respostas
4. O estudante seleciona uma das respostas
5. O estudante confirma sua resposta
6. O sistema analisa a resposta
7. O sistema exibe uma tela de resposta correta
8. O sistema exibe mais 3 questões para o estudante
9. Ao término das questões teóricas o sistema exibe uma nova tela com uma situação problema
10. O estudante clica sobre as setas de um fluxograma
11. O sistema exibe formas de um fluxograma
12. O estudante seleciona uma forma
13. O sistema instancia a forma a partir da extremidade da seta e cria uma nova seta começando do objeto
14. O estudante monta um fluxograma e ao término das operações confirma o fluxo montado
15. O sistema analisa a resposta do estudante
16. O sistema contorna de verde as formas corretamente posicionadas
17. O sistema exibe uma mensagem de acerto
18. O sistema redireciona o usuário para uma nova etapa com uma nova situação problema e um código incompleto
19. O estudante clica sobre as lacunas no código incompleto para editá-lo
20. O estudante digita a informação faltante
21. O estudante confirma suas modificações
22. O sistema analisa a resposta do estudante
23. O sistema exibe uma mensagem de acerto
24. O sistema exibe uma pergunta com 3 opções de resposta
25. O estudante seleciona uma das respostas
26. O estudante confirma sua resposta
27. O sistema analisa a resposta do estudante

28. O sistema exibe uma tela de acerto
29. O sistema marca a fase como finalizada

Fluxo alternativo:

*. O estudante clica no botão de voltar

*. O exercício é interrompido e o sistema retorna para o mapa de fases

7. O sistema exibe uma tela de resposta incorreta da questão de múltipla escolha

7.1 O sistema exibe a resposta correta esperada da questão

7.2 Retorna ao fluxo básico passo 8

16. O sistema contorna alguma das formas de vermelho

16.1 O sistema exibe uma tela de resposta incorreta

16.2 O sistema exibe o fluxo esperado

16.3 Retorna ao fluxo básico passo 18

23. O sistema exibe mensagem de resposta errada na análise do código

23.1 O sistema exibe as respostas esperadas da questão

22.2 Retorna ao fluxo básico passo 24

28. O sistema exibe uma mensagem de resposta errada na questão sobre o código

28.1 O sistema exibe a resposta correta esperada

28.2 Retorna ao fluxo básico passo 29

Caso de Uso 7: Receber recompensa de uma fase

Ator: Estudante

Fluxo Básico:

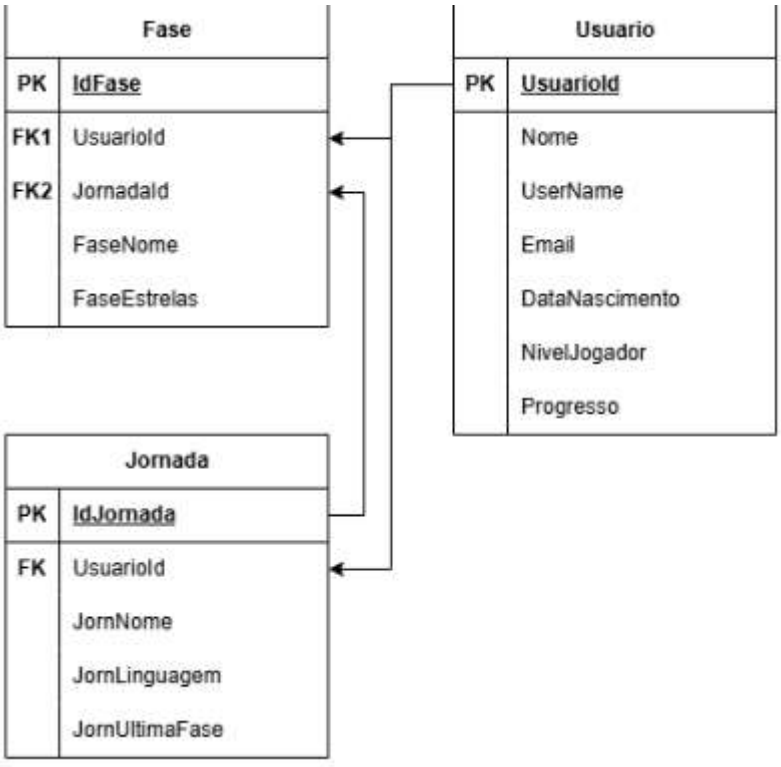
1. O estudante faz as 3 etapas de exercícios propostos
2. O sistema computa a porcentagem de acerto das respostas do usuário para atribuir uma nota de 1 a 3 estrelas
3. O sistema computa baseado na quantidade de respostas acertos a quantidade de experiencia a ser ganha pelo estudante
4. O sistema exibe a tela de recompensa com a quantidade de experiência obtida e a nota pelos acertos
5. O estudante clica em um botão para obter recompensas
6. O sistema aumenta a quantidade de experiência do jogador na barra de experiência
7. O sistema atualiza o progresso das fases

8. O sistema redireciona o usuário para o mapa de fases

2.2 BANCO DE DADOS

O principal objetivo desta seção é apresentar a modelagem dos dados por meio de tabelas e detalhar o correspondente Modelo Entidade-Relacionamento (MER). Para isso, foi adotada como exemplo uma estrutura de dados bem definida, com tabelas organizadas e esquematizadas que servem de base para a representação do MER. Essa abordagem visa proporcionar uma compreensão clara e abrangente das interações e relações existentes entre as entidades do sistema. Através da ilustração 11, é possível visualizar de forma objetiva como os dados estão estruturados e como se interconectam, facilitando o entendimento da lógica por trás da organização do banco de dados.

Ilustração 11 – MER



Fonte: Autor, 2025

O modelo proposto é composto por três entidades principais: Usuário, Jornada e Fase. A entidade Usuário possui uma chave primária (PK) que é referenciada como chave estrangeira (FK) na entidade Jornada, representando a relação direta entre um usuário e as jornadas que ele pode selecionar. A entidade Fase, por sua vez, possui duas chaves estrangeiras: uma referenciando a Jornada e outra referenciando o Usuário. Essa estrutura relacional reflete o fluxo do sistema, onde o usuário seleciona

uma jornada específica e, a partir dessa escolha, passa a ter acesso a um conjunto de fases organizadas em um mapa vinculado à jornada.

2.3 CAMADA DE NEGÓCIO

Para tornar a experiência do usuário única e personalizada, foi implementado para o CodeJourney um sistema de login de acesso, além de uma estrutura para capturar o progresso e salvar as informações de cadastro do usuário. Essas funcionalidades permitem que cada pessoa tenha seu próprio perfil, com dados individuais de desempenho, fases concluídas, recompensas obtidas e preferências de aprendizado. Na API, essas informações são armazenadas em um banco de dados utilizando da biblioteca SQLite, que fornece um sistema leve e embutido para gerenciamento de banco de dados, sendo recuperadas sempre que o usuário realiza o login. A autenticação foi desenvolvida visando confiabilidade e proteção dos dados, utilizando práticas recomendadas de segurança, como a autenticação JWT. Além disso, a API gerencia as interações entre o usuário e o conteúdo gamificado, garantindo que os avanços sejam devidamente registrados e sincronizados com a aplicação.

> Uma API é a interface que um programa de software apresenta a outros programas, a humanos e, no caso das APIs web, ao mundo via internet. O design de uma API revela muito sobre o programa por trás dela — modelo de negócio, funcionalidades do produto, e até bugs ocasionais. Embora as APIs sejam projetadas para funcionar com outros programas, elas são principalmente feitas para serem entendidas e utilizadas por humanos que escrevem esses outros programas. (JIN; LAMMON; FINK, 2018, p. 3)

2.3.1 ARQUITETURA DA CAMADA DE NEGÓCIO

Cada camada da API desempenha um papel específico, garantindo a organização, manutenibilidade e segurança do sistema. Ela é organizada em várias partes que facilitam o desenvolvimento e a manutenção do sistema. A camada de *Authorization* cuida da autenticação e garante que apenas usuários com permissão acessem certas funcionalidades. As *Controllers* recebem as requisições feitas pelo aplicativo e direcionam as ações para os serviços corretos. A pasta *Data* é responsável pela conexão com o banco de dados e por gerenciar os dados salvos. Os *DTOs (Data Transfer Objects)* servem para transportar as informações de forma

segura entre as diferentes partes da aplicação, sem expor diretamente os dados internos. A camada de *Models* define as estruturas dos dados que representam as informações principais do sistema. Já a camada *Repository* organiza as regras de acesso aos dados, facilitando consultas e atualizações no banco. Essa divisão por camadas deixa o projeto mais claro, organizado e fácil de expandir no futuro.

Os métodos responsáveis pelo cadastro e *login* dos usuários estão implementados dentro da *controller* principal que faz parte da entidade Usuario, utilizando requisições HTTP do tipo POST. O método de cadastro permite que um novo usuário seja registrado no sistema. Antes de efetuar o cadastro, o sistema verifica se já existe algum usuário com o mesmo nome ou e-mail. Caso exista, uma mensagem de erro é retornada; caso contrário, os dados são salvos no banco por meio do repositório, como mostra a ilustração 12.

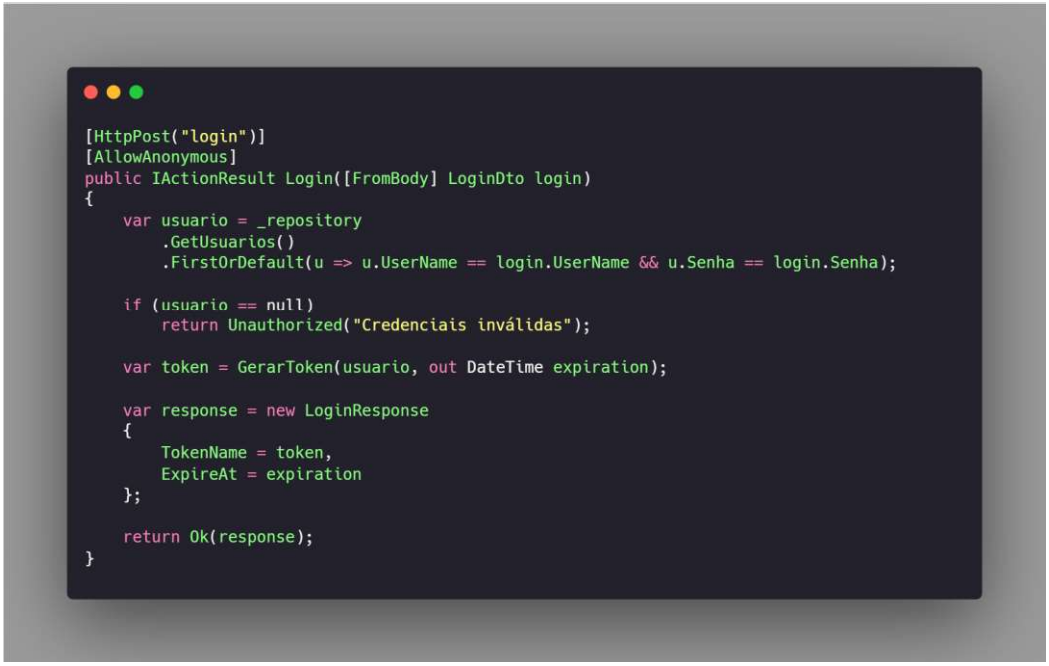
Ilustração 12 – Código cadastro

```
[HttpPost("cadastro")]
[AllowAnonymous]
public IActionResult Insert([FromBody] Usuario usuario)
{
    var usuarioExistente = _repository
        .GetUsuarios()
        .FirstOrDefault(u => u.UserName == usuario.UserName || u.Email == usuario.Email);

    if (usuarioExistente != null)
    {
        return BadRequest("Já existe um usuário com esse nome de usuário.");
    }

    _repository.Insert(usuario);
    return Ok(usuario);
}
```

Fonte: Autor, 2025

Já o método de *login,* da ilustração 13, busca validar as credenciais informadas pelo usuário. Se os dados estiverem corretos, é gerado um token de autenticação com tempo de expiração, que será utilizado para acessar funcionalidades protegidas da aplicação. Ambos os métodos são decorados com o atributo *[AllowAnonymous],* permitindo que possam ser acessados sem autenticação prévia.

Ilustração 13 – Código login



```
[HttpPost("login")]
[AllowAnonymous]
public IActionResult Login([FromBody] LoginDto login)
{
    var usuario = _repository
        .GetUsuarios()
        .FirstOrDefault(u => u.UserName == login.UserName && u.Senha == login.Senha);

    if (usuario == null)
        return Unauthorized("Credenciais inválidas");

    var token = GerarToken(usuario, out DateTime expiration);

    var response = new LoginResponse
    {
        TokenName = token,
        ExpireAt = expiration
    };

    return Ok(response);
}
```

Fonte: Autor, 2025

Além dos métodos de cadastro e *login*, a *controller* da aplicação CodeJourney também conta com outros métodos que seguem o padrão CRUD *(Create, Read, Update e Delete)*. Esses métodos permitem, por exemplo, consultar usuários cadastrados, atualizar informações e excluir registros. Com isso, é possível manter o controle dos dados de forma organizada e funcional, garantindo que o sistema ofereça uma base sólida para a manipulação das informações dos usuários dentro d plataforma.

Para salvar o progresso do usuário é usada a entidade Jornada. Nesse contexto dois métodos se destacam pela sua funcionalidade essencial na plataforma. O primeiro, responsável por consultar as fases das jornadas, permite que sejam buscadas com base no nome e no identificador do usuário, retornando todas as fases que correspondem aos critérios informados. Essa funcionalidade facilita o acesso segmentado às informações, permitindo que cada usuário visualize apenas as fases relacionadas a ele para a jornada escolhida. O segundo método possibilita a atualização de uma jornada existente passando dados como as fases alcançadas e recompensas obtidas em cada fase, identificada pelo id do usuário. Dessa forma, o sistema assegura que as informações armazenadas reflitam com precisão o progresso ou as alterações realizadas pelo usuário. Como mostra a ilustração 14.

Ilustração 14 – Código jornada

```
[HttpGet("pegar")]
public IActionResult GetJornadasPorNome([FromQuery] string jornNome, [FromQuery] int
idUsuario)
{
    var jornadas = _jornadaRepository.GetJornadas(jornNome, idUsuario);
    if (jornadas == null || !jornadas.Any())
    {
        return NotFound("Nenhuma jornada encontrada com esse nome.");
    }

    return Ok(jornadas);
}

[HttpPut("atualizar/{id}")]
public IActionResult AtualizarJornada(int id, [FromBody] Jornadas jornadaAtualizada)
{
    if (jornadaAtualizada == null)
    {
        return BadRequest("Dados da jornada inválidos.");
    }

    _jornadaRepository.UpdateJornada(id, jornadaAtualizada);
    return Ok("Jornada atualizada com sucesso.");
}
    _repository.Insert(usuario);
    return Ok(usuario);
}
```

Fonte: Autor, 2025

2.4 CAMADA DE APRESENTAÇÃO

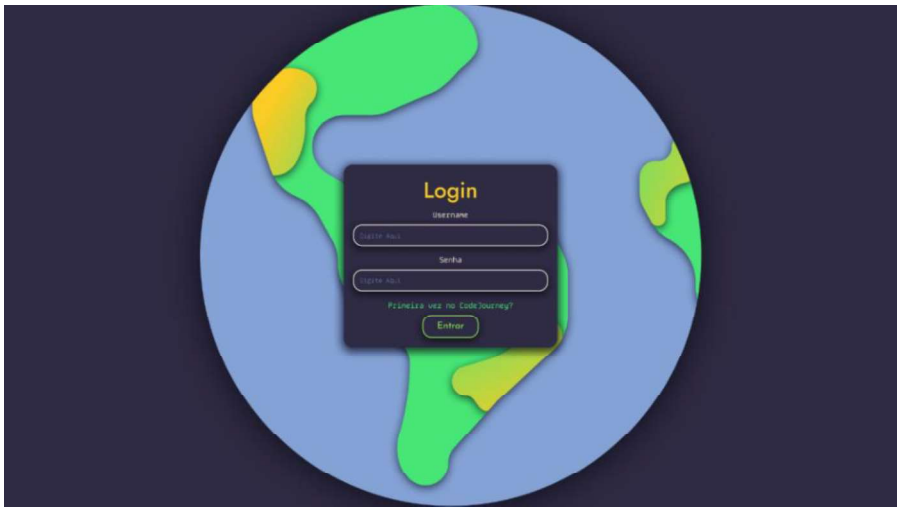
Neste tópico, serão detalhadas as etapas de desenvolvimento das telas do aplicativo. O design das interfaces foi elaborado no *Figma*, uma ferramenta amplamente reconhecida por sua eficiência na criação de protótipos interativos e layouts intuitivos. Para a realização do *front-end*, foi utilizada a plataforma Unity, por ser altamente recomendada na criação de aplicativos gamificados. O Unity oferece recursos visuais avançados, suporte a animações, interatividade em tempo real e grande flexibilidade para customização, permitindo transformar o design em experiências dinâmicas e envolventes, características essenciais para um projeto que busca engajar o usuário por meio da gamificação.

O aplicativo é responsivo para diferentes tamanhos de tela, mas tem como resolução padrão 1920 x 1080. A plataforma foi configurada para ajustar

dinamicamente os elementos da interface, adaptando-os a diversas resoluções, sem comprometer a integridade do design ou a usabilidade.

Ao acessar o aplicativo o usuário será apresentado a tela de *login* onde deverá colocar suas credenciais para acessar sua conta ou clicar em “Primeira vez no CodeJourney?”, conforme ilustração 15, para se cadastrar.

Ilustração 15 – Tela de Login



Fonte: Autor, 2025

Após realizar o login na plataforma, o usuário será direcionado para uma tela com diferentes jornadas de aprendizado sobre linguagens de programação. Nessa tela, poderá selecionar a jornada desejada para acessar o menu de fases correspondente a cada linguagem, como é possível observar na ilustração 16 abaixo.

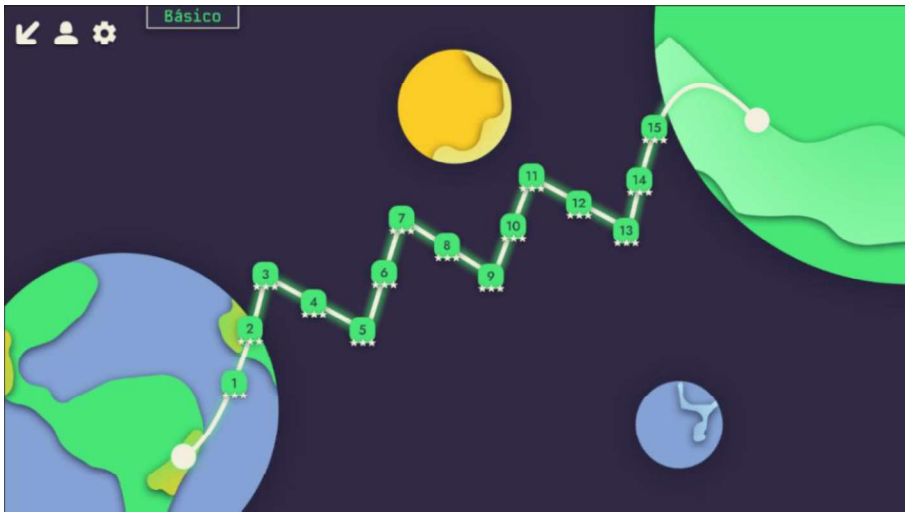
Ilustração 16 – Tela de Jornadas

Fonte: Autor, 2025

No menu de fases o usuário poderá visualizar informações do perfil do jogador, acessar o menu de configurações para editar esses dados e selecionar um dos elementos destacados na trilha, numerados de 1 a 15, organizados nos níveis básico,

intermediário e avançado de cada jornada, para acessar os desafios, conforme a ilustração 17 apresenta.

Ilustração 17 – Menu de fases



Fonte: Autor, 2025

Ao acessar uma fase, o usuário será apresentado a uma sequência de perguntas sobre programação. Cada questão terá alternativas destacadas, sendo apenas uma a correta. Para avançar, o usuário deverá selecionar a resposta correta, exibindo uma mensagem de feedback a cada acerto ou erro, conforme ilustrado na imagem 18.

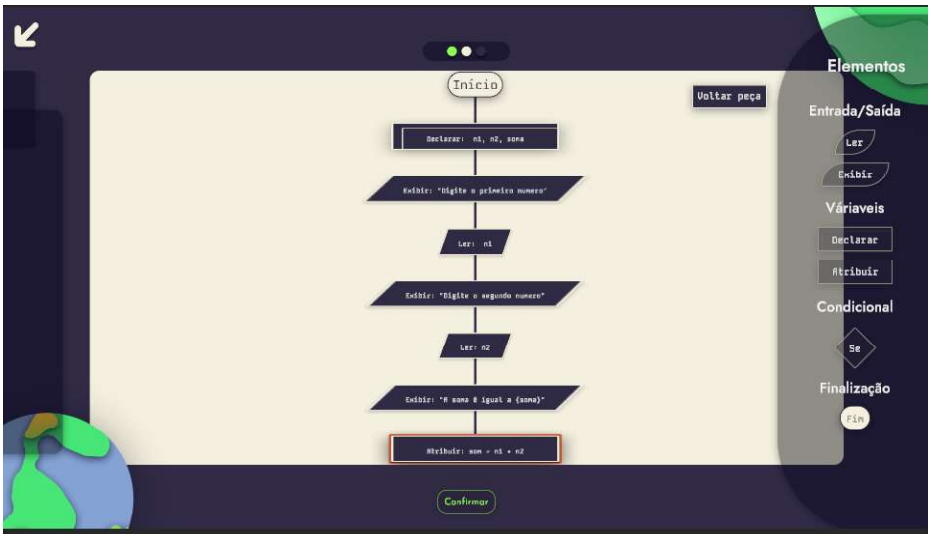
Ilustração 18 – Etapa questionário

Fonte: Autor, 2025

Ao finalizar a sessão anterior, o usuário avançará para uma nova etapa, onde será apresentado a um problema lógico. Para resolvê-lo, deverá selecionar as peças de um fluxograma e verificar o algoritmo criado. Se a ordem das peças estiver incorreta será destacado os elementos nas posições erradas a serem corrigidos

podendo remover as peças clicando no botão “Voltar peça”, assim como apresentado na imagem 19. Caso tudo esteja correto o sistema exibe uma mensagem de confirmação e transaciona para a próxima etapa da fase

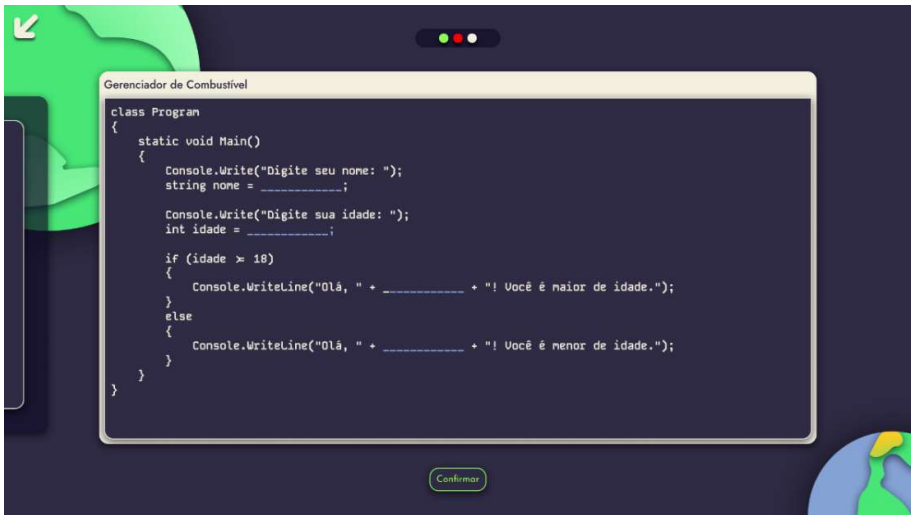
Ilustração 19 – Etapa Fluxograma



Fonte: Autor, 2025

Por fim, um novo problema será apresentado ao usuário, conforme ilustrado na imagem 20. Desta vez, ele precisará preencher as lacunas do código, inserindo as informações ausentes para corrigir a lógica do *script*. Em caso de erro, o aplicativo notificará o usuário e exibirá o feedback correspondente.

Ilustração 20 – Etapa Script

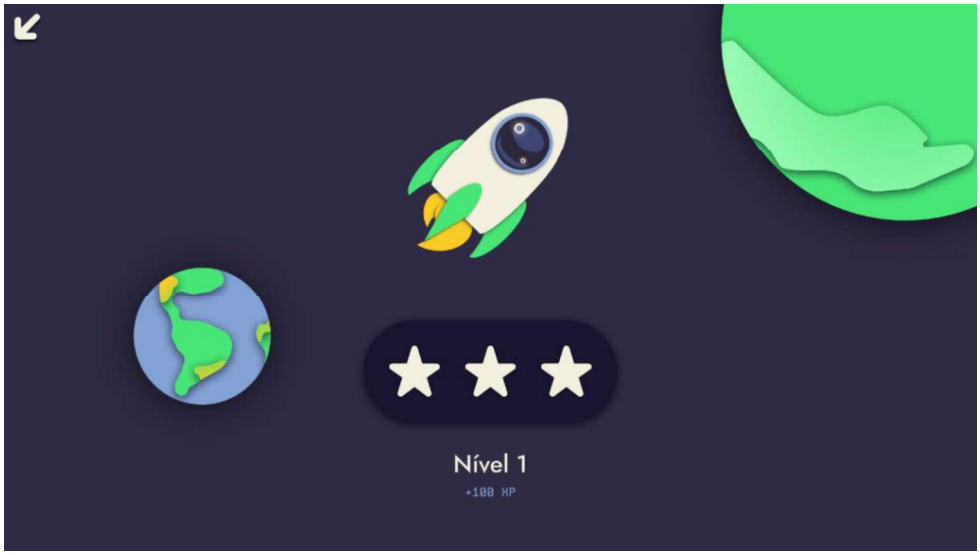


Fonte: Autor, 2025

Após a conclusão de todas as etapas, o aplicativo avaliará o desempenho do usuário, atribuindo uma nota de uma a três estrelas e concedendo uma quantidade de progresso de nível de experiência. Em seguida, o ciclo será finalizado, e o usuário

será redirecionado ao menu de fases, onde poderá iniciar um novo desafio, conforme ilustrado na imagem 21.

Ilustração 21 – Finalização fase



Fonte: Autor, 2025

3. RESULTADO

Este capítulo final aborda a fase conclusiva do desenvolvimento do sistema, enfatizando a aplicação prática por meio de testes com usuários representativos do público-alvo. Essa etapa tem como foco central a obtenção de percepções autênticas dos usuários finais, cujos comentários são cruciais para refinar e ajustar a solução desenvolvida.

A metodologia adotada consistiu na realização de testes de usabilidade, nos quais os participantes foram convidados a executar tarefas previamente definidas em contextos simulados, permitindo que suas interações com a interface fossem observadas e analisadas. Essa técnica é particularmente eficaz por evidenciar aspectos da experiência do usuário que não são perceptíveis apenas pela análise técnica, como frustrações, confusões ou comportamentos inesperados.

De acordo com Krug (2014), a observação direta de usuários reais utilizando um sistema é uma das ferramentas mais poderosas para identificar pontos problemáticos em interfaces digitais. Isso porque os testes revelam tanto os obstáculos enfrentados pelos usuários quanto os elementos que funcionam de maneira fluida e natural. Com essas informações, é possível propor melhorias orientadas, seja no *layout* da interface, nos fluxos de navegação ou na lógica das funcionalidades.

Portanto, essa etapa é mais do que uma verificação final: ela é parte integrante do ciclo de desenvolvimento centrado no usuário, proporcionando não só a detecção de erros, mas também *insights* valiosos para tornar a experiência mais eficaz e agradável. O *feedback* direto do usuário não apenas valida decisões anteriores, como também direciona futuras evoluções da aplicação.

3.1 TESTES

Para mensurar os resultados obtidos, o aplicativo CodeJourney foi disponibilizado a um grupo de 5 estudantes de programação, com instruções descrevendo tarefas que representam um ciclo de utilidade do sistema e um questionário de avaliação a ser respondido ao final para que os usuários testes possam avaliar as suas experiencias sobre o uso do aplicativo.

Ao longo da avaliação, cada usuário teve que realizar oito atividades distintas. Essas atividades foram escolhidas de forma criteriosa por refletirem as funcionalidades centrais da plataforma. A Tabela 3, a seguir, apresenta a relação completa dessas tarefas.

Tabela 3 – Tabela de tarefas do teste

Nº	TITULO	TAREFA
01	Cadastro	Criar uma conta com nome de usuário e senha e definir alguns dados.
02	Login e Autenticação	Acessar com seu nome de usuário e senha definidos no cadastro
03	Acessar Trilha	Acessar uma das três trilhas pré-definidas
04	Acessar fase	Acessar uma das fases no mapa de fases
05	Etapa Quiz	Finalizar a etapa de questionário da fase
06	Etapa Fluxo	Finalizar a etapa de criar o fluxograma da fase
07	Etapa Código	Finalizar a etapa de completar o código da fase
08	Remover conta	Excluir a conta nas configurações

Fonte: Autor, 2025

O teste foi realizado na Baixada Santista, região do estado de São Paulo, durante o mês de abril de 2025. Participaram da avaliação estudantes universitários de programação, de diferentes gêneros, com idades entre 20 e 25 anos. Todos já possuíam experiência prévia com tecnologia e linguagens de programação, o que contribuiu para uma compreensão facilitada da aplicação e minimizou a necessidade de explicações detalhadas antes do início.

Ao executarem as atividades propostas, os participantes apresentaram comportamentos semelhantes, indicando familiaridade com a interface e fluidez na navegação. Todas as tarefas foram concluídas com sucesso, sem qualquer falha que comprometesse sua realização. O tempo médio para a finalização das tarefas foi de cerca de 8 minutos, refletindo tanto o bom desempenho do sistema quanto a rápida adaptação dos usuários à plataforma.

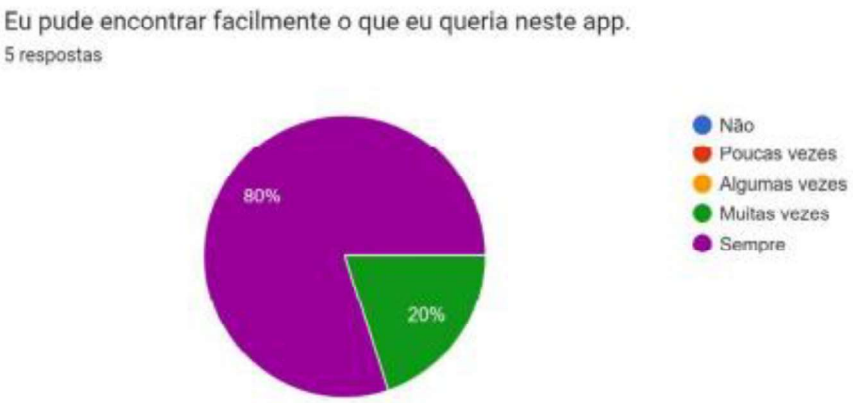
3.2 QUESTIONÁRIO DE AVALIAÇÃO

Concluída a etapa dos testes de usabilidade, foi aplicada uma pesquisa anônima com os participantes, a fim de reunir impressões detalhadas sobre a experiência com a aplicação. Para estruturar esse levantamento, utilizou-se como base o modelo WAMMI (*Website Analysis and Measurement Inventory*), reconhecido por avaliar a usabilidade de interfaces digitais sob a ótica do usuário.

O questionário foi elaborado e distribuído através da plataforma *Google Forms*, uma ferramenta online que permite a criação e disseminação de formulários personalizados. Além de facilitar a participação remota dos respondentes, essa tecnologia também contribuiu para a organização dos dados coletados. Ao final do processo, os resultados foram automaticamente compilados em gráficos visuais, o que favoreceu uma análise rápida e objetiva das percepções dos usuários.

A ilustração 22 abaixo mostra respostas dos usuários relacionados a facilidade que tiveram em navegar pelo aplicativo baseado na seguinte afirmação: “Eu pude encontrar facilmente o que queria neste app”. Ao analisar o gráfico se nota que 80% dos participantes afirmaram “Sempre” e 20% “Muitas vezes” terem encontrado o que busca, sugerindo boa navegabilidade do sistema.

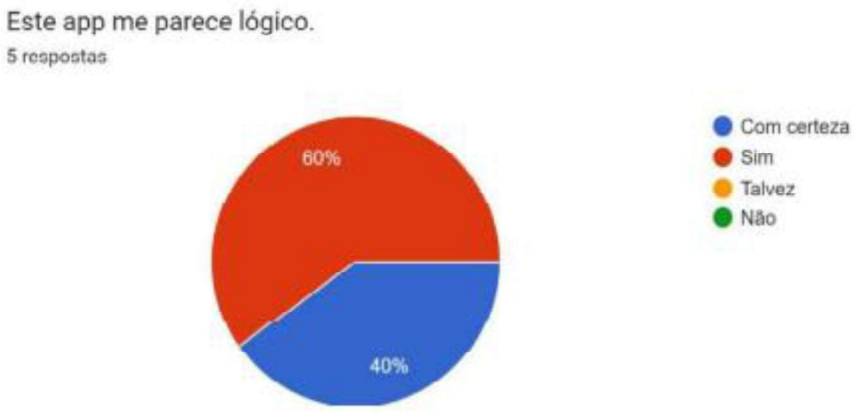
Ilustração 22 – Gráfico de respostas primeira pergunta



Fonte: Autor, 2025

Conforme mostra a ilustração 23, 60% dos participantes afirmaram "Com certeza" ao avaliar a lógica da aplicação, enquanto os 40% restantes escolheram a opção "Sim", confirmando a clareza e coerência da ferramenta. Com base nas respostas dos usuários, conclui-se que a estrutura e o funcionamento do sistema foram bem compreendidos.

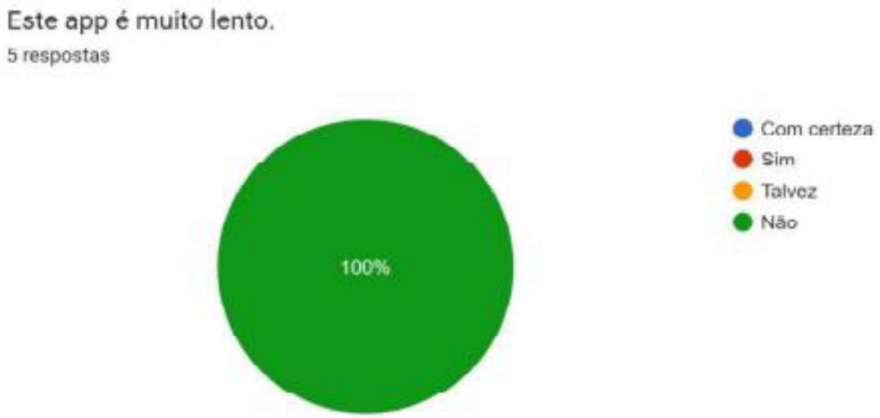
Ilustração 23 – Gráfico de respostas segunda pergunta



Fonte: Autor, 2025

A Ilustração 24 evidencia que 100% dos participantes afirmaram não considerar a aplicação lenta, o que demonstra que o sistema apresentou a velocidade de resposta esperada. Esse resultado confirma que a aplicação oferece uma experiência fluida, contribuindo para o engajamento e a satisfação dos usuários.

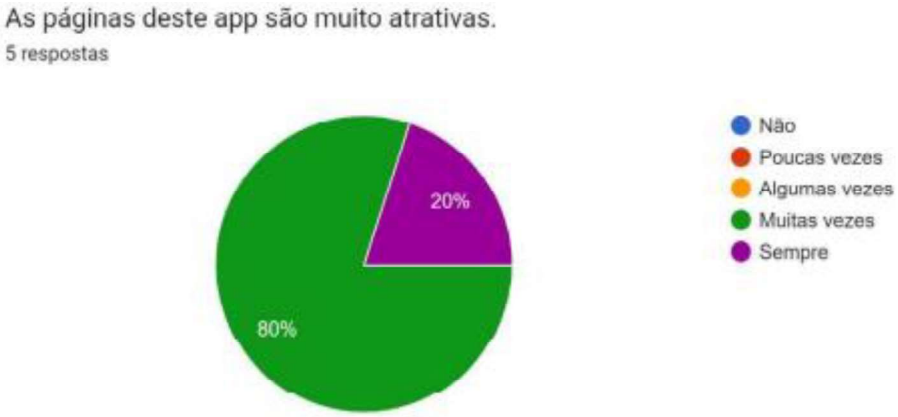
Ilustração 24 – Gráfico de respostas terceira pergunta



Fonte: Autor, 2025

A ilustração 25, relativa ao design do aplicativo demonstra que 80% dos usuários acharam a página muito atrativa e 20% gostaram sempre.

Ilustração 25 – Gráfico de respostas quarta pergunta



Fonte: Autor, 2025

Com base nos resultados obtidos nas quatro ilustrações apresentadas, é possível concluir que o sistema foi bem aceito pelos usuários em aspectos fundamentais como navegabilidade, lógica de funcionamento, desempenho e design. A maioria absoluta dos participantes demonstrou facilidade em utilizar o aplicativo, compreendeu sua estrutura com clareza, percebeu rapidez nas respostas e teve uma boa impressão visual, o que reforça a eficácia da solução desenvolvida e seu alinhamento com as expectativas do público-alvo.

3.3 CONCLUSÃO

O desenvolvimento do aplicativo CodeJourney teve como foco principal a criação de uma solução educacional inovadora voltada ao ensino de lógica de programação, utilizando elementos de gamificação para tornar o aprendizado mais acessível, dinâmico e atrativo. Direcionado especialmente a jovens estudantes da área de tecnologia, o projeto buscou estimular o engajamento por meio de desafios, recompensas e uma interface intuitiva.

Com base nos testes realizados, constatou-se que o aplicativo atendeu plenamente aos objetivos propostos, demonstrando bons resultados em aspectos como usabilidade, clareza na lógica de funcionamento, desempenho e atratividade visual. Os dados obtidos reforçam a viabilidade e o potencial da proposta como uma ferramenta complementar no processo de aprendizado de programação.

Além disso, identificou-se uma possibilidade relevante de aprimoramento: a criação de um acesso personalizado para contas de professores. Essa funcionalidade permitiria que a plataforma cumprisse de maneira ainda mais ativa seu papel didático em sala de aula, potencializando o acompanhamento do progresso dos alunos e a personalização dos conteúdos conforme as necessidades pedagógicas.

Dessa forma, o CodeJourney se consolida como uma base sólida para o desenvolvimento contínuo de soluções educacionais interativas, com potencial para evoluir e expandir seu impacto no ensino de programação.

REFERÊNCIAS BIBLIOGRÁFICAS

BRASILPOPULAR. Quase 70% dos estudantes da área de TI não concluem a faculdade. Disponível em: https://brasilpopular.com/quase-70-dos-estudantes-da-area-de-ti-nao-concluem-a-faculdade/. Acesso em: 14 maio 2025.

CNN BRASIL. 55,5% dos alunos desistem antes de completar ensino superior, aponta relatório. Disponível em: https://www.cnnbrasil.com.br/nacional/555-dos-alunos-desistem-antes-de-completar-ensino-superior-aponta-relatorio/. Acesso em: 14 maio 2025.

GUEDES, Gilleanes Thorwald Araujo. **UML 2 - Uma Abordagem Prática**. 1. ed. São Paulo: Novatec Editora, 2009.

HOCKING, Joseph. **Unity in Action: Multiplatform Game Development in C# with Unity 2018**. 2. ed. Manning Publications, 2018.

JIN, Brenda; SAHNI, Saurabh; SHEVAT, Amir. **Designing Web APIs: Building APIs That Developers Love.** 1. ed. Sebastopol: O'Reilly Media, 2018.

MICROSOFT. O que é o .NET?. Disponível em: https://dotnet.microsoft.com/pt-br/learn/dotnet/what-is-dotnet. Acesso em: 14 maio 2025.

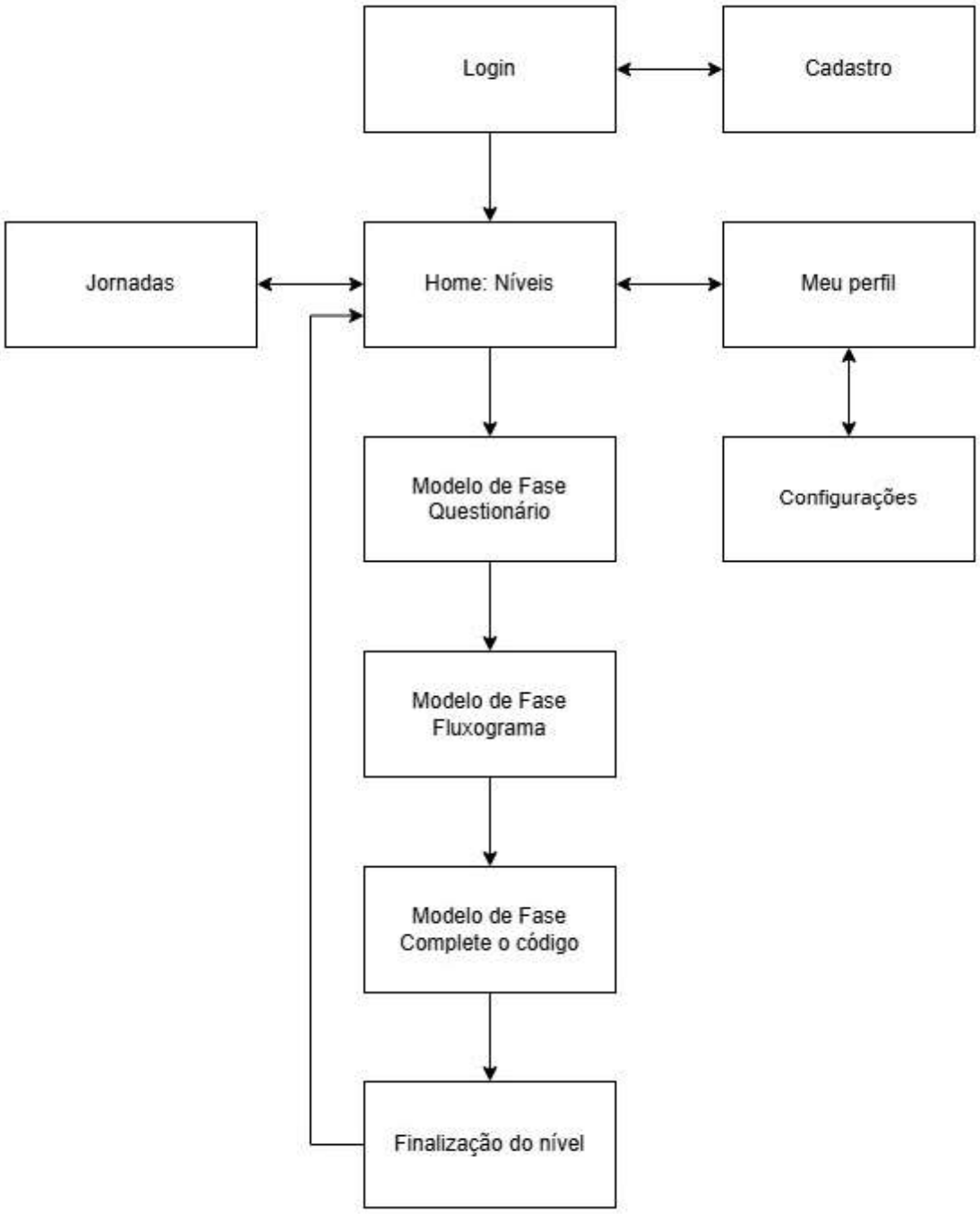
ORAM, Andy; WILSON, Greg. **Making Software: What Really Works, and Why We Believe It**. O'Reilly Media, 2010.

KRUG, Steve**. Não me faça pensar!: uma abordagem de bom senso à usabilidade na web.** 3. ed. Rio de Janeiro: Alta Books, 2014.

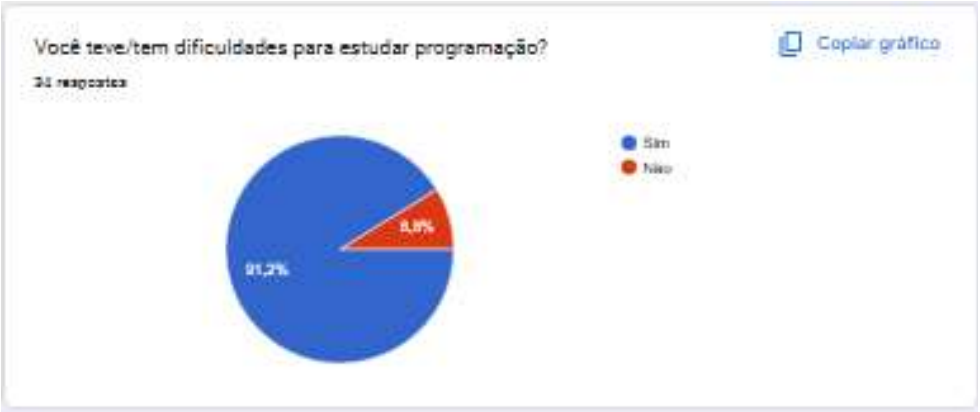
VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira**. Engenharia de Requisitos: Software Orientado ao Negócio**. São Paulo: Editora Novatec, 2017.

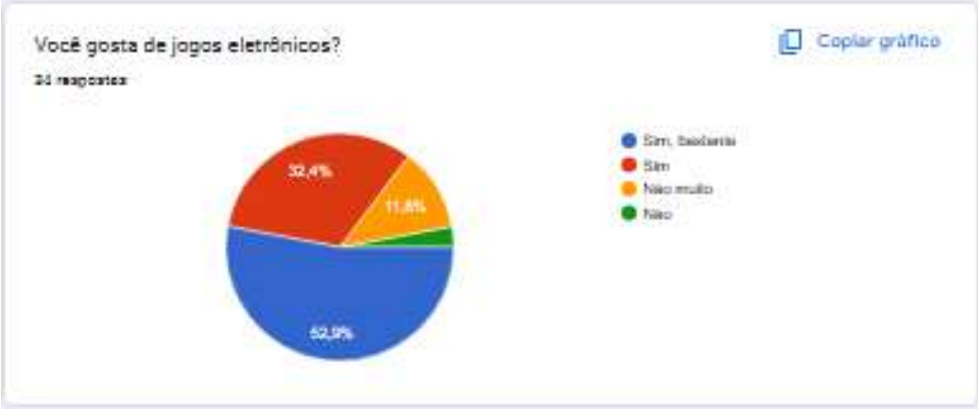
APÊNDICE A

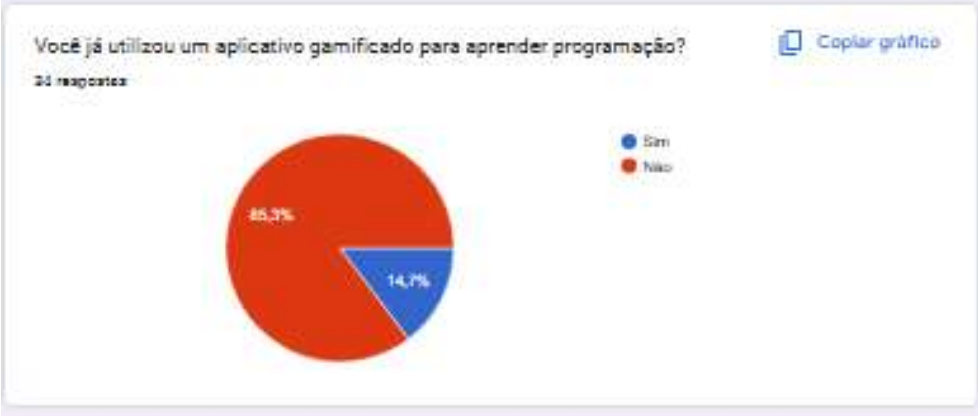
Fluxo do sistema



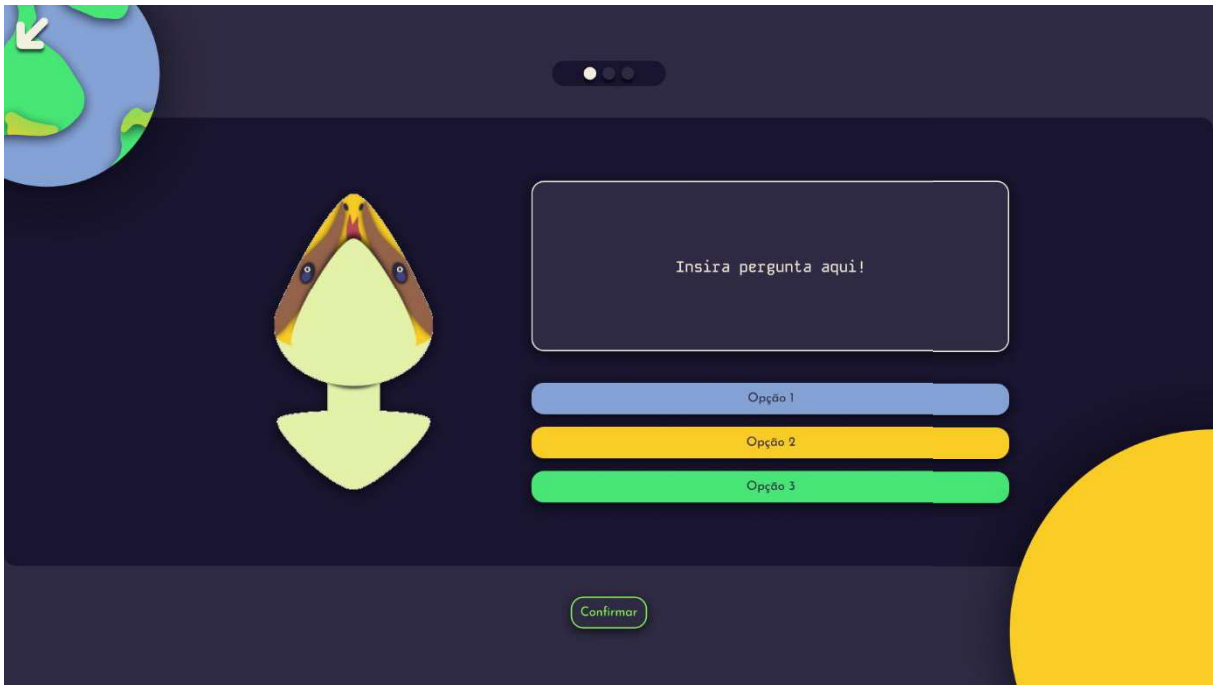
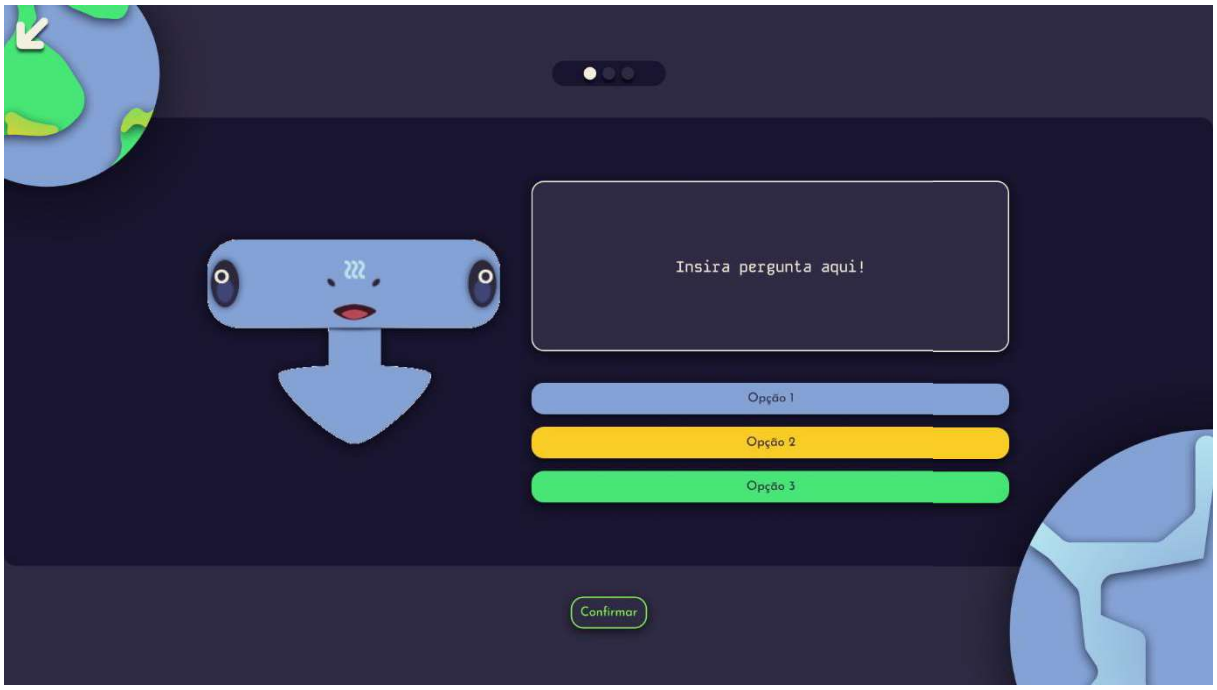
Respostas do formulário de pesquisa







Modelo de fase Python



Modelo de fase Javascript