Thiago Diaz Virginio da Silva

Desenvolvimento de um aplicativo móvel para ensino de Python

São Carlos - SP, Brasil
2025

Thiago Diaz Virginio da Silva

Desenvolvimento de um aplicativo móvel para ensino de Python

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Universidade Federal de São Carlos como requisito parcial para a obtenção do título de Bacharel em Engenharia de Computação.

Universidade Federal de São Carlos – UFSCar
Centro de Ciências Exatas e de Tecnologia – CCET
Departamento de Computação – DC
Graduação em Engenharia de Computação

Orientador: Prof. Dr. André Ricardo Backes

São Carlos - SP, Brasil
2025

Dedico este trabalho primeiramente à minha família. À minha mãe, por ser minha base, e ao meu pai e avós (in memoriam), por iluminarem meu caminho. Em especial, agradeço à república Capo de Fusca e aos meus amigos, que foram meu alicerce e alegria durante toda a trajetória acadêmica.

Agradecimentos

Este trabalho é o resultado do apoio de muitas pessoas especiais, a quem expresso minha eterna gratidão.

À minha mãe, Mari Franci da Silva Diaz, obrigado por ser minha base, por cada palavra de conforto e por acreditar em mim mais do que eu mesmo. Sem você, nada disso seria possível.

Ao meu pai, Severino Virginio da Silva Júnior, que mesmo ausente fisicamente, deixou um legado de força que me impulsionou em cada desafio. Esta conquista também é sua.

Agradeço imensamente ao meu orientador, professor André Ricardo Backes. Sua confiança no meu potencial e sua disposição para ajudar foram fundamentais para que esta pesquisa tomasse forma. Obrigado pela paciência e pela guia segura.

À república Capo de Fusca e aos meus amigos: vocês foram meu refúgio e minha alegria. As risadas, o apoio e a certeza de ter com quem contar fizeram toda a diferença nos dias mais pesados.

Obrigado a todos por fazerem parte desta história!

"A única maneira de fazer um ótimo trabalho é amar o que você faz."
— Steve Jobs

Resumo

O presente trabalho apresenta o desenvolvimento de um aplicativo móvel educacional, projetado para o ensino da linguagem de programação Python, com foco em jovens iniciantes. Utilizando o *framework* Flutter, foi criada uma plataforma que combina lições teóricas e exercícios interativos em uma experiência gamificada, incorporando mecânicas como sistema de vidas, pontos de experiência (XP) e desbloqueio progressivo de módulos. A arquitetura do aplicativo foi baseada no padrão de repositório, garantindo escalabilidade e manutenibilidade, enquanto o *backend* foi implementado com Firebase Authentication e Cloud Firestore. O protótipo desenvolvido demonstra que a abordagem proposta é eficaz para engajar o público-alvo e proporcionar uma jornada de aprendizado mais dinâmica e motivadora, atendendo aos objetivos pedagógicos e tecnológicos estabelecidos.

Palavras-chave: Ensino de Programação. Python. Desenvolvimento Móvel. Flutter. Gamificação.

Abstract

This work presents the development of an educational mobile application designed to teach the Python programming language, focusing on young beginners. Using the Flutter framework, a cross-platform solution was created that combines theoretical lessons and interactive exercises in a gamified experience, incorporating mechanics such as a lives system, experience points (XP), and progressive module unlocking. The application's architecture was based on the repository pattern, ensuring scalability and maintainability, while the backend was implemented with Firebase Authentication and Cloud Firestore. The developed prototype demonstrates that the proposed approach is effective in engaging the target audience and providing a more dynamic and motivating learning journey, fulfilling the established pedagogical and technological objectives.

Keywords: Programming Education. Python. Mobile Development. Flutter. Gamification.

Lista de ilustrações

Lista de abreviaturas e siglas

API	*Application Programming Interface*
UX	*User Experience*
UI	*User Interface*
XP	*Experience Points*
FCM	*Firebase Cloud Messaging*
RNF	Requisitos Não Funcionais
RF	Requisitos Funcionais
SDK	*Software Development Kit*
UID	*User Identifier* (Identificador do Usuário)
BaaS	*Backend as a Service*

Sumário

1 Introdução

No contexto atual da era digital, o aprendizado de programação tornou-se uma competência fundamental, especialmente com a tecnologia permeando todos os aspectos da vida cotidiana [1]. Dentre as diversas linguagens, o Python destaca-se por sua sintaxe clara e versatilidade, sendo amplamente utilizado em áreas como desenvolvimento web, ciência de dados e inteligência artificial [2].

Contudo, os métodos de ensino tradicionais frequentemente enfrentam dificuldades em manter o engajamento dos estudantes, especialmente os mais jovens, diante de uma geração cada vez mais conectada [5]. Nesse cenário, o uso de aplicativos móveis surge como uma alternativa pedagógica eficaz, favorecendo a aprendizagem ativa por meio de cursos interativos acessíveis a qualquer momento e lugar [9].

Além disso, a crescente demanda por profissionais da área de tecnologia, evidenciada por um aumento de 34,3% nas vagas do setor em 2022 [8], reforça a necessidade de novas ferramentas educacionais. A programação é considerada hoje uma habilidade essencial para a competitividade no mercado de trabalho [7].

Este trabalho insere-se nesse contexto ao propor o desenvolvimento de um aplicativo educativo que une a simplicidade do Python com a interatividade e o poder de engajamento da gamificação. A abordagem adota o conceito de *mobile learning* para oferecer uma formação acessível quando e onde o usuário desejar.

É importante ressaltar que, embora o escopo deste trabalho se concentre no ensino de Python, a arquitetura da aplicação foi projetada de forma modular e escalável, com o potencial de se tornar uma plataforma de aprendizado colaborativa para a comunidade acadêmica da UFSCar.

1.1 Objetivos

O objetivo geral deste trabalho é desenvolver um aplicativo móvel educativo em Flutter, focado no ensino da linguagem de programação Python para jovens, facilitando o aprendizado por meio de uma abordagem interativa e gamificada.

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

- Desenvolver uma plataforma móvel que integre materiais teóricos com exercícios práticos interativos, como múltipla escolha e verdadeiro/falso, permitindo a aplicação imediata dos conceitos de Python;
- Implementar um sistema de gamificação robusto para aumentar o engajamento e a motivação dos usuários, incluindo mecânicas de vidas, pontos de experiência (XP), sequência de dias de estudo (*streak*) e desbloqueio progressivo de conteúdo;

- Estruturar a aplicação com uma arquitetura escalável e um modelo de dados que permitam sua futura publicação em lojas de aplicativos e a avaliação contínua de sua eficácia no processo de aprendizagem dos jovens.

2 Fundamentação Teórica

2.1 Ensino de Programação e Gamificação

A era digital sublinha a importância do ensino de programação como uma competência essencial, que transcende o campo da tecnologia e impacta diversas áreas do conhecimento [1]. O aprendizado de programação, especialmente para os jovens, traz benefícios cognitivos comprovados, como o aprimoramento do raciocínio lógico, do pensamento crítico e da capacidade de solucionar problemas complexos de forma estruturada [6].

Contudo, o ensino tradicional enfrenta desafios significativos que frequentemente resultam em desmotivação e evasão. A natureza abstrata dos conceitos, a sintaxe rigorosa das linguagens e a ausência de feedback imediato podem dificultar o processo de aprendizagem e gerar frustração nos alunos iniciantes [5].

Para contornar essas dificuldades, têm sido exploradas metodologias inovadoras de ensino, entre as quais se destaca a gamificação. Esse conceito consiste na aplicação de elementos e mecânicas de jogos em contextos não lúdicos, como a educação, com o objetivo de aumentar o engajamento e a motivação dos usuários. Elementos como sistemas de pontos (XP), medalhas (*achievements*) e *rankings* (*leaderboards*) tornam a experiência de aprendizado mais interativa e recompensadora. A gamificação promove um ciclo de *feedback* positivo, em que o aluno é imediatamente reconhecido pelo seu progresso, incentivando-o a persistir diante dos desafios.

Paralelamente, a ascensão dos dispositivos móveis consolidou o *mobile learning* como uma nova modalidade de ensino. O uso de aplicativos em smartphones e tablets permite um aprendizado autônomo, flexível e personalizado, no qual o estudante aprende em seu próprio ritmo e no momento mais conveniente. Essa abordagem *just-in-time* é especialmente eficaz para o ensino de habilidades práticas como programação, onde a prática frequente em sessões curtas tende a ser mais produtiva do que longas exposições teóricas [4].

Este trabalho se posiciona na interseção dessas duas tendências, utilizando a gamificação como motor de engajamento e o *mobile learning* como meio de entrega de conteúdo, com o intuito de oferecer uma solução sinérgica para os desafios do ensino de programação.

2.2 Trabalhos Correlatos

Diversas soluções de ensino de programação através de aplicativos e plataformas web já foram propostas, cada uma com abordagens metodológicas, níveis de interatividade e focos educacionais distintos. Para contextualizar este trabalho, foram analisadas algumas plataformas proeminentes que oferecem cursos de programação para iniciantes: Codecademy,

Grasshopper, Alura, Mimo, SoloLearn e Programming Hub.

2.2.1 Codecademy

O Codecademy é uma das plataformas online mais conhecidas para o aprendizado interativo de programação. Sua metodologia é fortemente centrada na prática (*learning by doing*), oferecendo um ambiente que combina uma breve explicação teórica com um editor de código interativo, onde o aluno escreve e executa o código diretamente no navegador. A plataforma cobre uma vasta gama de linguagens, incluindo Python, JavaScript, e SQL, e organiza seu conteúdo em trilhas de carreira e cursos baseados em projetos. Embora sua experiência interativa seja robusta no ambiente web, ela pode não ser totalmente otimizada ou apresentar a mesma fluidez em dispositivos móveis, dependendo da complexidade da lição. A Figura 1 ilustra sua interface de aprendizado baseada na web [10].

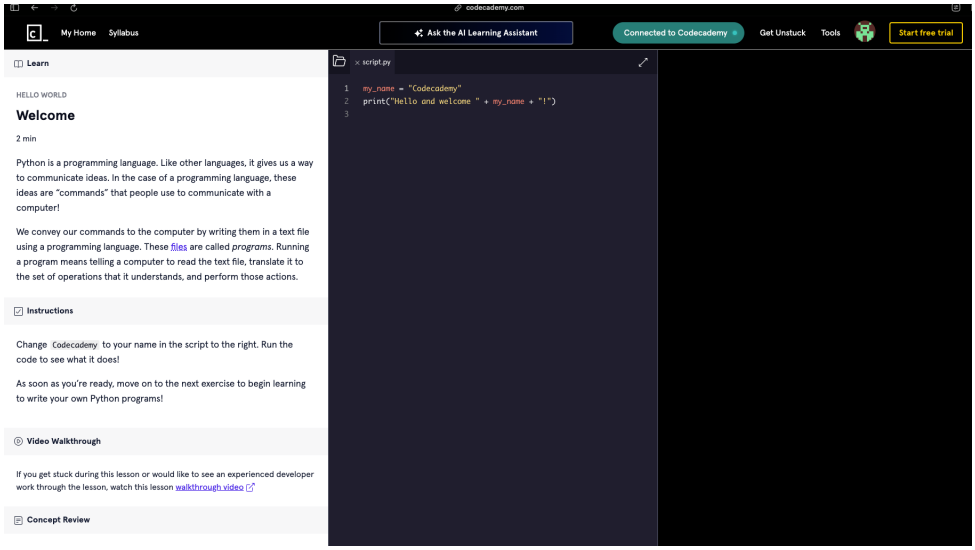


Figura 1 – Interface de uma lição interativa no site Codecademy.

2.2.2 Grasshopper

O Grasshopper era um aplicativo gratuito desenvolvido pelo Google que ensinava programação de forma interativa e gamificada, especialmente para iniciantes em JavaScript. O aplicativo utilizava lições e desafios práticos, com uma interface visual de blocos, para ensinar conceitos como variáveis, loops e condicionais. Apesar de sua popularidade e de ter sido lançado em português, o Grasshopper foi descontinuado pelo Google em junho de 2023. Sua abordagem é relevante para este trabalho por seu forte apelo gamificado, mas sua limitação era o foco exclusivo em JavaScript, não servindo como uma ferramenta para quem desejava aprender Python. A Figura 2 demonstra o estilo de lição baseado em quebra-cabeças que a plataforma utilizava [11].

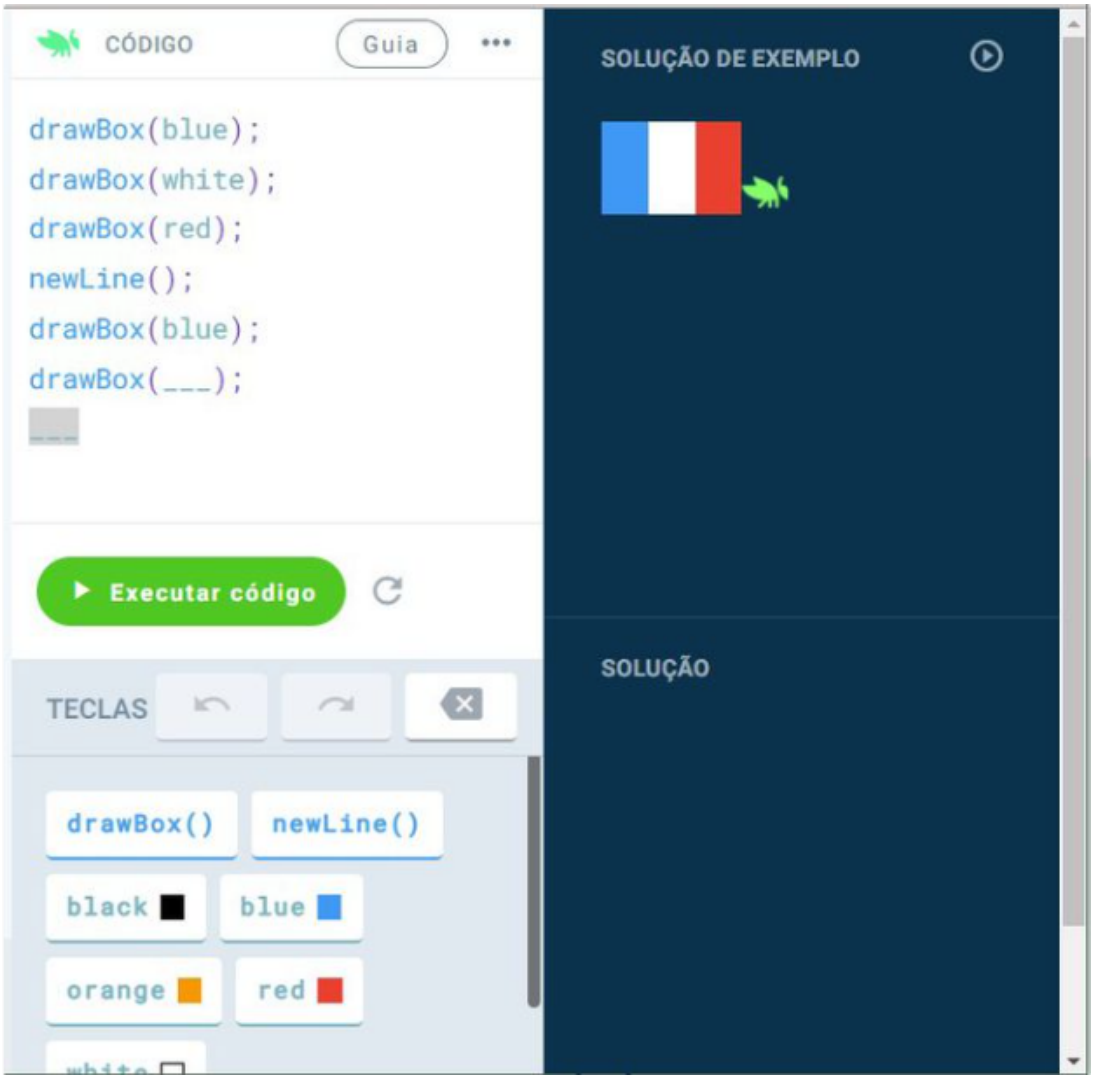


Figura 2 – Interface de uma lição interativa no aplicativo Grasshopper.

2.2.3 Alura

A Alura é uma das maiores plataformas de educação em tecnologia do Brasil, oferecendo um extenso catálogo de cursos que vão muito além da programação, abrangendo áreas como Design, Marketing Digital e Gestão. Sua metodologia é mais próxima de um ambiente de sala de aula online tradicional, baseada primariamente em videoaulas ministradas por instrutores, complementadas por apostilas para leitura e exercícios de fixação. Um de seus pontos fortes é a comunidade ativa em fóruns de discussão, onde os alunos podem interagir e tirar dúvidas. Apesar de sua abrangência e qualidade de conteúdo, a experiência de aprendizado pode ser considerada menos interativa em comparação com aplicativos que oferecem feedback em tempo real dentro de um editor de código integrado. A Figura 3 mostra a estrutura de um de seus cursos [12].

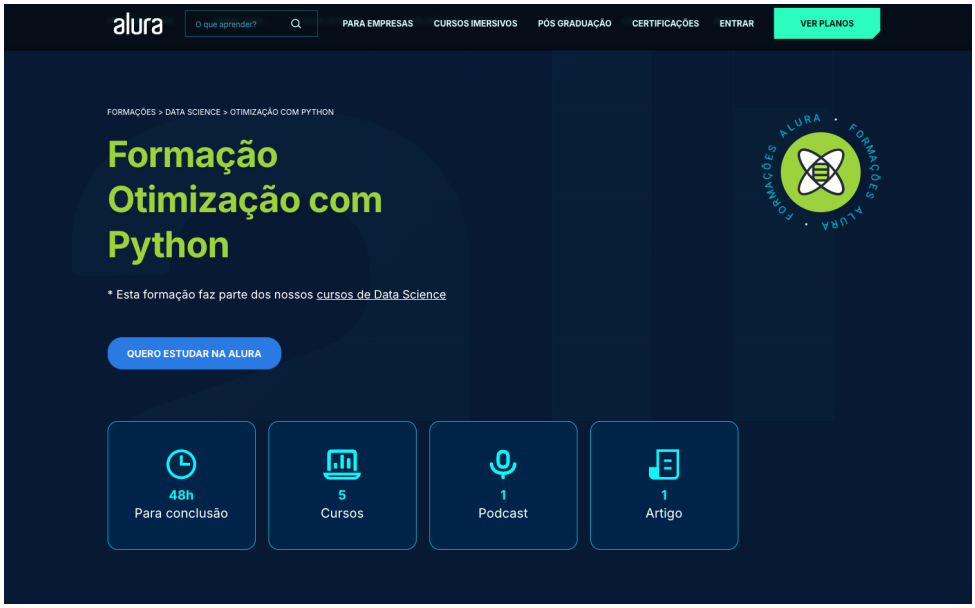


Figura 3 – Interface de início de curso no site Alura.

2.2.4 Programming Hub

O Programming Hub é um aplicativo móvel que se destaca por sua abordagem de microlearning, oferecendo lições curtas e focadas em mais de 50 linguagens e tecnologias de programação. A plataforma utiliza uma combinação de teoria, exemplos interativos e quizzes para facilitar a retenção do conhecimento. Uma de suas características notáveis é a forma como o aprendizado se assemelha a uma história, tornando a experiência mais engajadora. Ao final dos cursos, também oferece certificados. Embora seja prático para um aprendizado rápido, a profundidade do conteúdo e a complexidade dos exercícios podem não ser suficientes para usuários que buscam um conhecimento mais aprofundado e prático sobre desenvolvimento de software. A Figura 4 mostra a estrutura de um de seus cursos [13].

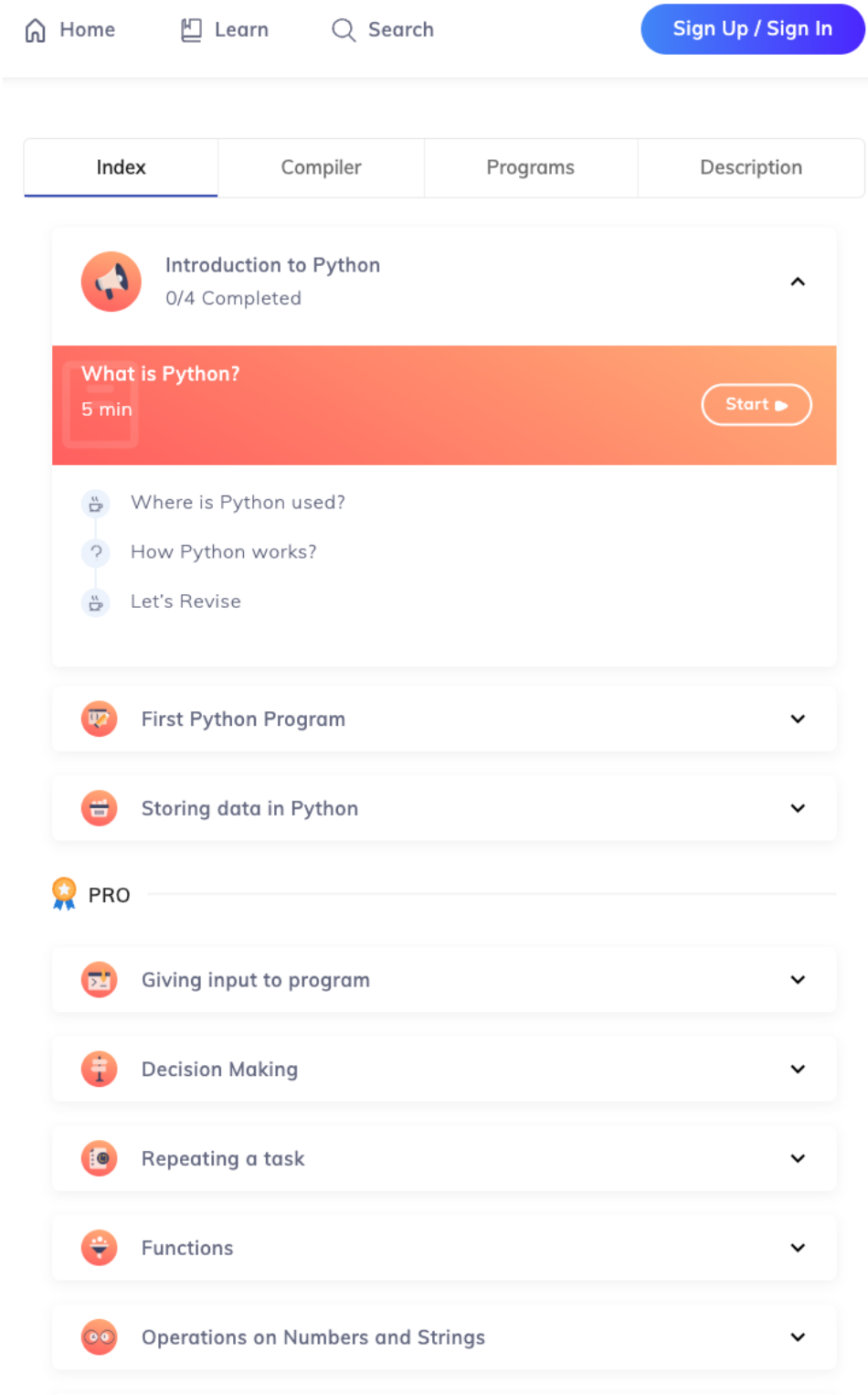


Figura 4 – Interface do curso de Python no aplicativo Programming Hub.

2.2.5 Mimo

Este aplicativo se destaca por sua abordagem de microlearning, oferecendo lições que podem ser concluídas em poucos minutos por dia. Cobrindo uma variedade de tecnologias, incluindo Python, JavaScript e HTML, o Mimo utiliza uma interface fortemente inspirada

no Duolingo, com exercícios interativos e quizzes curtos para ensinar conceitos básicos de programação. Embora seja frequentemente elogiado por iniciantes por sua simplicidade e por manter a motivação, a plataforma é geralmente vista como um recurso suplementar, ideal para revisar conceitos ou para quem está começando do zero, mas que pode não ter a profundidade necessária para a formação de um desenvolvedor pronto para o mercado. A Figura 5 ilustra a interface de uma de suas lições [14]

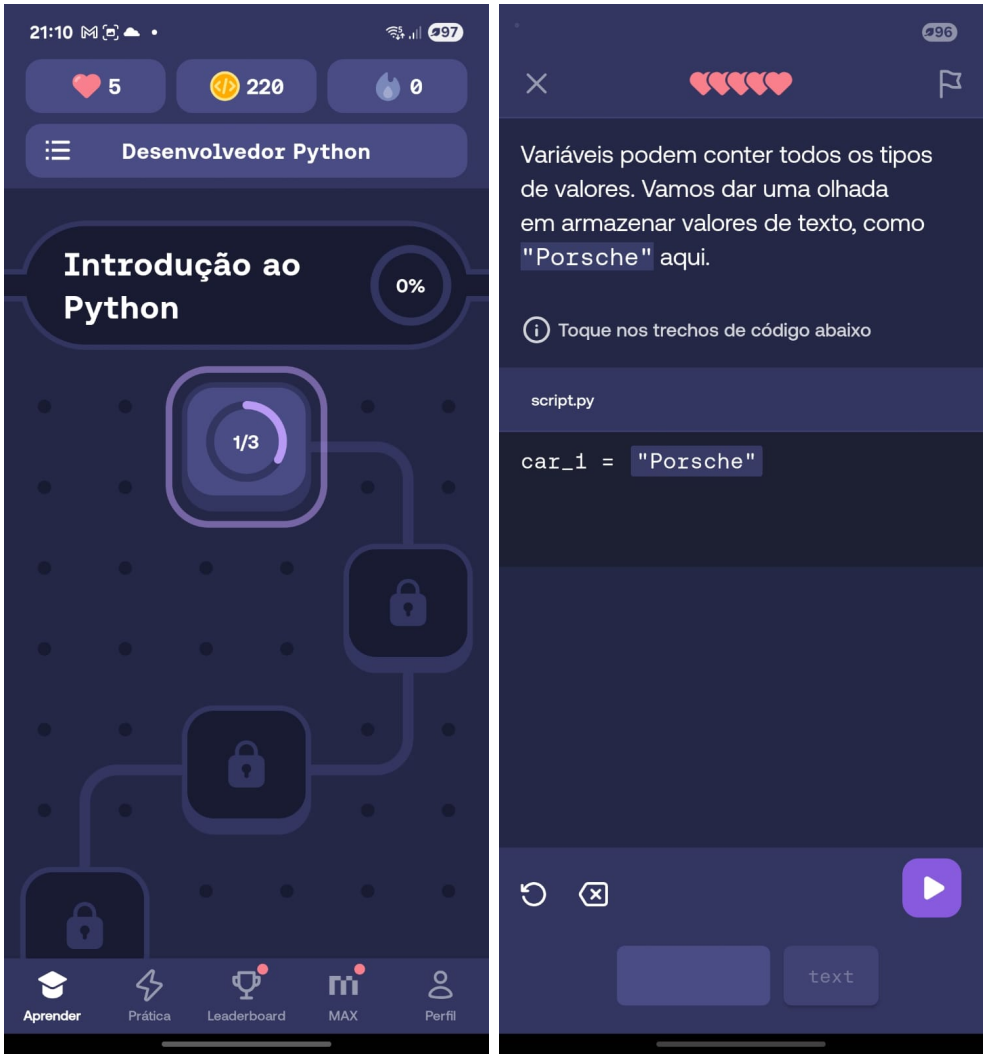


Figura 5 – Interface do curso de Python no aplicativo Mimo.

2.2.6 Sololearn

O Sololearn se posiciona como uma plataforma de aprendizado móvel com um forte componente social e uma vasta gama de cursos, cobrindo mais de 20 linguagens de programação como Python, JavaScript, C e SQL. Além das lições e quizzes interativos, um de seus maiores diferenciais é o *Code Playground*, um ambiente onde os usuários podem escrever, executar e compartilhar seus próprios projetos de código dentro do aplicativo, e também ver os projetos de outros usuários como é possível ver na figura 6. A plataforma também possui fóruns de discussão ativos para cada curso. Embora muito completo, alguns

usuários apontam que a dependência de exercícios de múltipla escolha pode não preparar totalmente o aluno para a escrita de código do zero, sendo mais eficaz quando combinado com outras formas de prática mais livres.

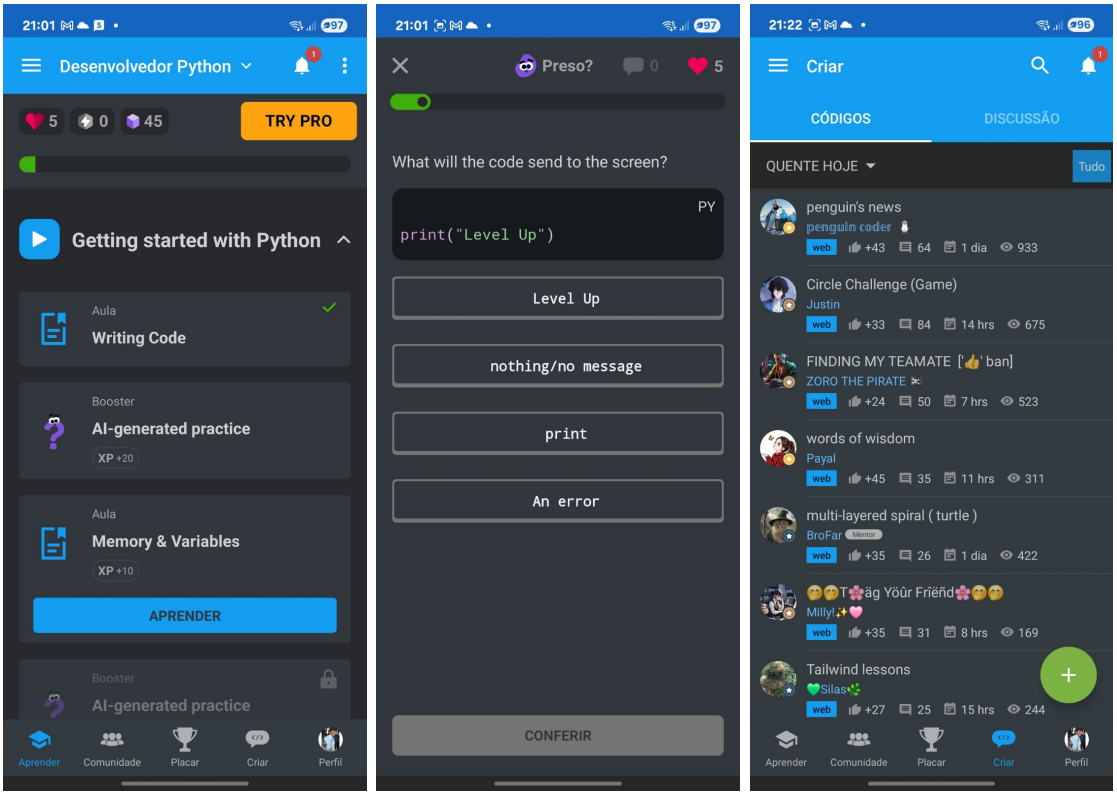


Figura 6 – Interface do curso de Python no aplicativo SoloLearn.

Para sintetizar a análise, a Tabela 1 apresenta um comparativo direto entre as plataformas estudadas e a proposta desenvolvida neste trabalho, destacando as principais características e diferenciais.

Tabela 1 – Tabela comparativa entre plataformas de ensino de programação.

Plataforma	Foco Principal	Metodologia	Gamificação	Diferencial da Proposta
Codecademy	Múltiplas linguagens (Web)	Prática em editor de código	Baixa (barras de progresso)	Foco em uma experiência móvel nativa e gamificação mais profunda
Grasshopper	JavaScript (Básico)	Resolução de puzzles visuais	Alta (base do app)	Foco na linguagem Python e em conteúdo com curadoria acadêmica
Alura	Múltiplas tecnologias	Videoaulas, textos e exercícios de fixação	Baixa (sistema de pontos)	Aprendizado interativo com feedback imediato, em vez de passivo
Mimo	Web/Python (Básico)	Microlearning com lições curtas e quizzes	Alta (estilo Duolingo)	Conteúdo alinhado ao ambiente acadêmico da UFSCar e totalmente gratuito
Sololearn	Múltiplas linguagens	Lições interativas com forte componente social	Média (XP, ranking, desafios)	Foco em uma jornada de aprendizado mais guiada e com desbloqueio sequencial
GuaxiCode	**Python (Iniciante)**	**Teoria + Exercícios Interativos**	**Alta (Vidas, XP, Streak, Ranking)**	**Conteúdo acadêmico da UFSCar e gratuito**

2.3 Tecnologias Utilizadas

A seleção das tecnologias foi guiada pela necessidade de um desenvolvimento rápido, multiplataforma e com uma experiência de usuário rica e interativa, ideal para um aplicativo educacional.

2.3.1 Flutter e Dart

Para o desenvolvimento do aplicativo, foi escolhido o *framework* **Flutter**, mantido pelo Google. A principal vantagem do Flutter é sua capacidade de criar aplicativos para Android e iOS a partir de um único código-base, utilizando a linguagem de programação **Dart** [3]. Essa abordagem reduz significativamente o tempo e o custo de desenvolvimento.

A arquitetura de widgets do Flutter permite a criação de interfaces de usuário complexas e fluidas, com controle total sobre cada pixel da tela. Isso foi fundamental para projetar uma experiência de aprendizado visualmente agradável e engajadora, com animações e transições suaves que contribuem para a retenção do usuário.

2.3.2 Firebase

Como plataforma de backend (Backend-as-a-Service - BaaS), foi utilizado o **Firebase**, também do Google. A escolha se deu por sua integração simplificada com o Flutter e pelo conjunto robusto de serviços que atendem a todas as necessidades do projeto:

- **Firebase Authentication:** Utilizado para gerenciar todo o fluxo de autenticação, incluindo cadastro por e-mail e senha e *login* social com o Google. Ele fornece uma solução segura e pronta para o gerenciamento de usuários.
- **Cloud Firestore:** É o banco de dados NoSQL utilizado para armazenar todos os dados da aplicação. Sua natureza flexível e escalável foi ideal para modelar as estruturas de cursos, módulos, lições e perfis de usuário. A capacidade de sincronização em tempo real, embora não explorada ativamente em sua totalidade, garante uma base sólida para futuras funcionalidades colaborativas.

3 Metodologia e Desenvolvimento

Este capítulo detalha a metodologia de desenvolvimento adotada e a implementação técnica do aplicativo proposto. Abordam-se as tecnologias selecionadas, a arquitetura de software, o modelo de dados e a implementação das funcionalidades centrais que compõem a experiência de aprendizado gamificada.

3.1 Análise de Requisitos e Modelagem

Para que o desenvolvimento da aplicação fosse iniciado de forma estruturada, a primeira etapa consistiu na análise e modelagem dos requisitos do sistema. Similarmente à abordagem de trabalhos correlatos na área de desenvolvimento de software, foi utilizado um Diagrama de Caso de Uso apresentado na figura 7 para mapear as interações do ator principal, o **Estudante**, com as funcionalidades do aplicativo. Este diagrama, que serviu como guia para o desenvolvimento, estabeleceu que todas as funcionalidades centrais, como acessar um módulo ou visualizar o perfil, exigiriam que o usuário estivesse autenticado no sistema.

A partir desta análise, foram definidos os seguintes requisitos funcionais (RF) para o projeto:

- **RF01:** O sistema deve permitir que o usuário se cadastre e realize *login* utilizando e-mail/senha e também através de sua conta Google.
- **RF02:** O sistema deve exibir os cursos e seus respectivos módulos de forma organizada.
- **RF03:** O sistema deve implementar uma lógica de desbloqueio sequencial, onde um módulo só se torna acessível após a conclusão do anterior.
- **RF04:** A plataforma deve apresentar lições em múltiplos formatos, incluindo teoria e exercícios interativos (múltipla escolha, verdadeiro/falso).
- **RF05:** O sistema deve gerenciar um sistema de vidas por módulo, deduzindo uma vida a cada resposta incorreta do usuário. Caso as vidas se esgotem, o progresso no módulo é interrompido.
- **RF06:** O progresso do usuário, incluindo Pontos de Experiência (XP), Sequência de Dias (Streak), lições e módulos concluídos, deve ser registrado de forma persistente.
- **RF07:** O usuário deve poder visualizar seu progresso e estatísticas de gamificação em uma tela de perfil dedicada.

- **RF08:** O usuário deve ter a capacidade de editar informações básicas do seu perfil, como o nome de exibição.

Além dos requisitos funcionais, foram estabelecidos requisitos não funcionais (RNF) para guiar as decisões de tecnologia e design:

- **RNF01:** A aplicação deve ser multiplataforma, com suporte para os sistemas operacionais Android e iOS, a partir de um único código-base.
- **RNF02:** A interface deve ser intuitiva, responsiva e com *feedback* visual claro para as ações do usuário.
- **RNF03:** Os dados pessoais e de progresso dos usuários devem ser armazenados de forma segura, com acesso restrito apenas ao próprio usuário.

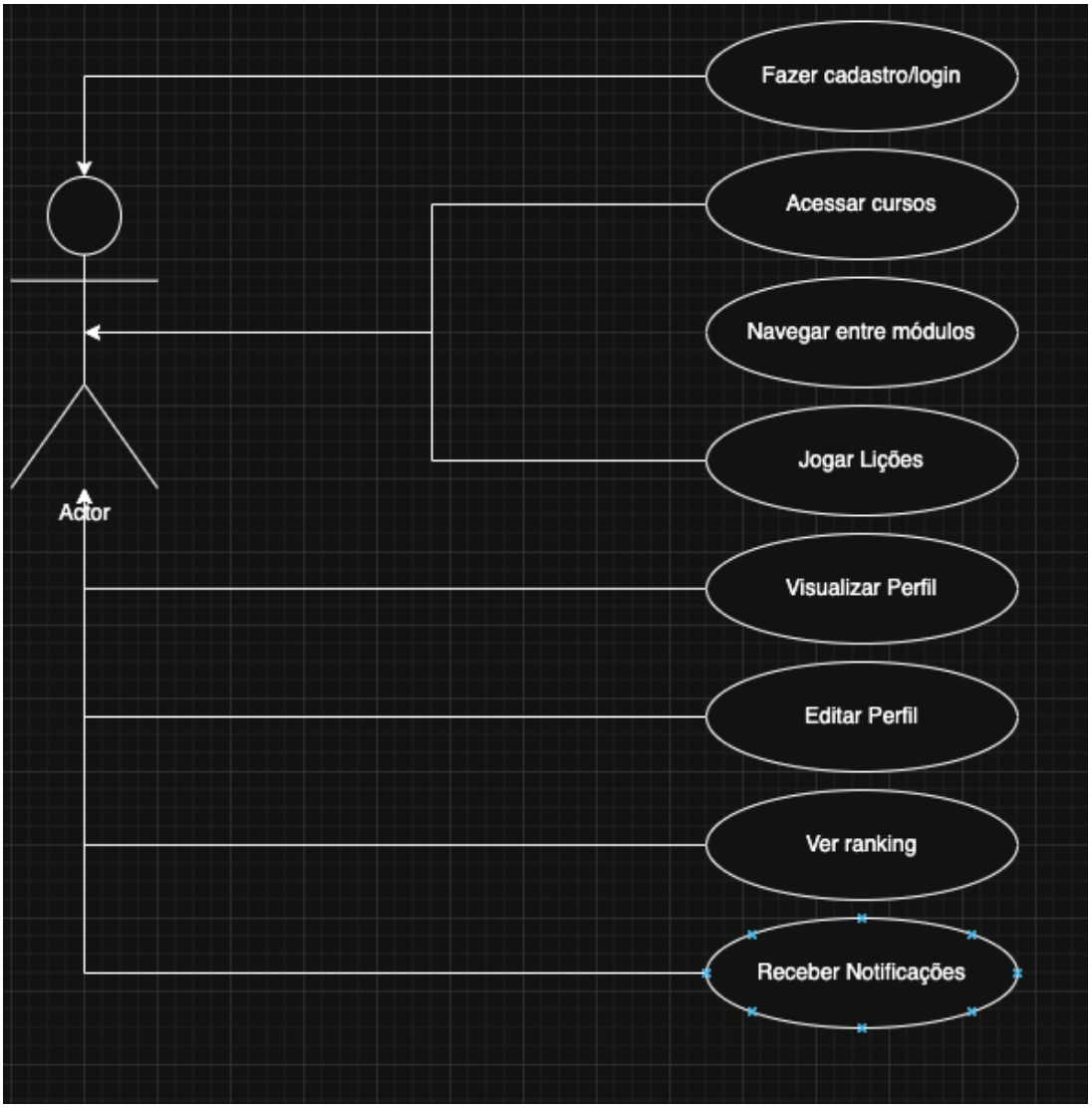


Figura 7 – Diagrama de caso de uso representando as interações do Estudante com as funcionalidades do aplicativo.

3.2 Arquitetura da Aplicação

O aplicativo foi estruturado seguindo o **Repository Pattern** (Padrão de Repositório), visando uma clara separação de responsabilidades e a manutenibilidade do código. A lógica foi dividida em três camadas principais:

- **Camada de Apresentação (UI):** Composta pelas páginas (widgets do Flutter, como *LessonPlayerPage*, *ProfilePage*, etc.). Sua única responsabilidade é exibir os dados e capturar as interações do usuário, delegando todas as operações de lógica de negócio e acesso a dados para os repositórios.

- **Camada de Dados (Repositórios):** Classes como *UserRepository* e *LessonsRepository* atuam como a única ponte entre a UI e o Firestore. Elas abstraem toda a complexidade das chamadas ao banco de dados. Por exemplo, para buscar o perfil de um usuário, a UI simplesmente chama *userRepository.getUserProfile*(), sem precisar saber os detalhes de coleções ou documentos do Firestore.

- **Camada de Modelo (*Models*):** Classes simples de dados (*UserProfile*, *Lesson*, *CourseModule*, etc.) que definem a estrutura dos objetos utilizados no aplicativo. Elas incluem métodos *fromMap* para converter os dados vindos do Firestore em objetos Dart fortemente tipados, garantindo a segurança e consistência dos dados em todo o app.

Para o gerenciamento de estado local das páginas, foi utilizada a abordagem nativa do Flutter com *StatefulWidget* e o método *setState*(), que se mostrou suficiente e performática para a complexidade do projeto.

3.3 Design e Experiência do Usuário (UI/UX)

Em grande parte do desenvolvimento, houve uma preocupação com os conceitos de Usabilidade e Experiência do Usuário (UX), pois um aplicativo educacional, especialmente voltado ao público jovem, depende de uma interface clara e engajadora para ser eficaz. O objetivo foi criar um design limpo, que evitasse distrações e proporcionasse um fluxo de aprendizado fluido.

Seguindo a metodologia de projetos de interfaces digitais, a fase inicial de design envolveu a prototipagem de telas de alta fidelidade utilizando a ferramenta online **Figma**. Isso permitiu a visualização e iteração rápida sobre os *layouts* e a navegabilidade da aplicação antes mesmo da escrita do código.

As principais decisões de design para promover uma boa experiência de usuário foram:

- **Navegação Principal:** A adoção de uma *BottomNavigationBar* fixa com os ícones de **Cursos** e **Perfil** oferece ao usuário acesso constante e imediato às duas seções mais importantes do aplicativo.
- **Ambiente de Lição Focado:** A *LessonPlayerPage* foi projetada para funcionar como um *player* imersivo. Ao entrar em um módulo, a navegação principal desaparece, permitindo que o usuário se concentre totalmente na lição, seja ela teórica ou prática. A presença constante da contagem de vidas na *AppBar* serve como um elemento de gamificação que mantém o usuário ciente de seu status no desafio.
- **Feedback Imediato e Visual:** Para reforçar o aprendizado, a *ExercisePage* foi construída para fornecer *feedback* instantâneo. Após a verificação de uma resposta, um rodapé *color-coded* (verde para acerto, vermelho para erro) aparece, informando não apenas o resultado, mas também um texto explicativo. Esta resposta imediata é crucial no processo de gamificação para manter o usuário motivado.
- **Visualização de Progresso:** A página de perfil foi projetada para ser mais do que um local de edição de dados. Ela funciona como um painel de conquistas, onde o usuário pode visualizar de forma clara e objetiva suas estatísticas (*XP*, Nível, *Streak*), transformando seu esforço em recompensas visíveis e tangíveis.

3.4 Modelo de Dados no Firestore

A estrutura do banco de dados no Firestore foi projetada para ser escalável, permitindo a adição de novos cursos no futuro. Ela se baseia em duas coleções principais na raiz:

- **Coleção *courses*:** Contém os documentos de cada curso (ex: *python course*). Cada documento de curso possui uma subcoleção *modules*, que por sua vez contém os documentos de cada módulo. Dentro de cada módulo, uma subcoleção *lessons* armazena todas as lições (teóricas e de exercício) daquele módulo. Como apresentado na figura 8.
- **Coleção *users*:** O ID de cada documento nesta coleção é o UID do usuário do Firebase Authentication. Este documento armazena todas as informações pessoais e de progresso, como:
  - Dados do perfil: *displayName*, *email*, *createdAt*.
  - Dados de gamificação: *experiencePoints*, *currentStreak*, *lastStudyDate*.
  - Dados de progresso: Um mapa *courseProgress* para os módulos concluídos e um mapa *completedLessons* para as lições já finalizadas, ambos organizados por curso para suportar múltiplos cursos no futuro.

O modelo de dados da coleção *users* está visualmente representado na figura 9.

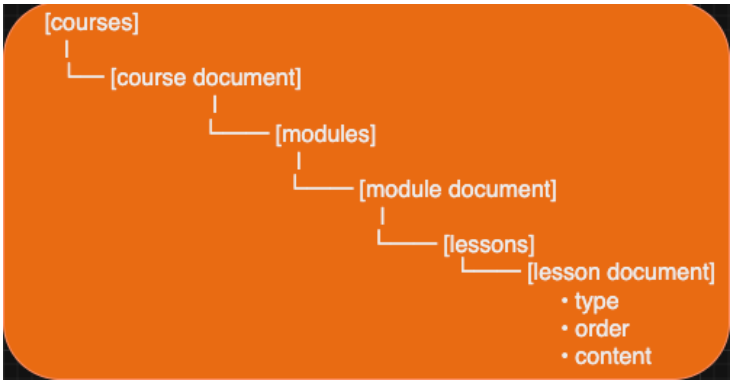


Figura 8 – Modelo de dados da coleção *courses*, mostrando a hierarquia de módulos e lições em cada curso.

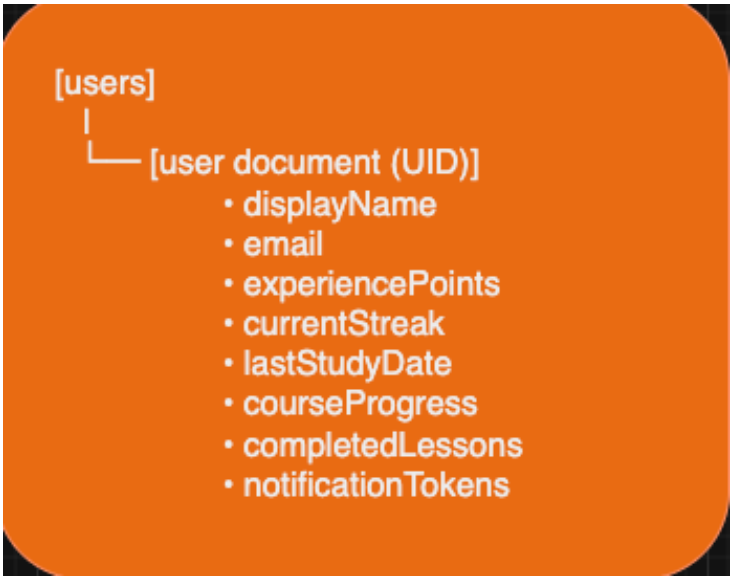


Figura 9 – Modelo de dados da coleção *users*, com campos que armazenam informações do perfil, progresso e dados de gamificação.

3.5 Implementação das Funcionalidades Principais

Com a arquitetura e o modelo de dados definidos, as funcionalidades de gamificação e progressão foram implementadas.

3.5.1 Sistema de Gamificação e Progressão

O núcleo da experiência do usuário é gerenciado pela *LessonPlayerPage*, que orquestra o fluxo de aprendizado e as mecânicas de jogo:

- **Sistema de Vidas:** O usuário inicia cada módulo com 5 vidas. Uma vida é deduzida a cada resposta incorreta em um exercício. Se as vidas chegam a zero, o progresso no módulo é interrompido e o usuário é incentivado a recomeçar.

- **Pontos de Experiência (XP):** O ganho de XP foi projetado para recompensar o esforço e o aprendizado, e não apenas a repetição. O sistema só concede XP na primeira vez que uma lição é completada, com valores diferenciados: uma quantidade menor para lições teóricas e uma maior para exercícios concluídos com sucesso.

- **Sequência de Dias (Streak):** O sistema incentiva o engajamento diário, rastreando e atualizando a sequência de dias consecutivos de estudo do usuário.

- **Desbloqueio de Módulos:** A *ModuleGridPage* calcula dinamicamente se um módulo deve ser desbloqueado, verificando se o módulo anterior consta na lista de módulos concluídos do usuário para aquele curso específico. Isso cria uma jornada de aprendizado linear e estruturada.

3.5.2 Sistema de Ranking

Para aumentar o engajamento e introduzir um elemento de competição saudável, foi implementado um sistema de ranking de usuários. A funcionalidade foi encapsulada em uma tela dedicada, a `RankingPage`, acessível através do menu de navegação principal.

A interface exibe uma lista dos 50 usuários com maior pontuação de experiência (XP), mostrando a posição, avatar, nome de exibição e a pontuação de cada um. Os três primeiros colocados recebem um destaque visual com ícones e cores que remetem a medalhas de ouro, prata e bronze. Uma característica importante para a experiência do usuário é um card fixo na parte inferior da tela, que exibe a posição do próprio usuário logado, tornando o ranking pessoalmente relevante mesmo que ele não esteja entre os primeiros colocados.

A implementação técnica no `UserRepository` exigiu a criação de dois métodos distintos para otimizar as consultas ao Firestore:

- ***fetchTopUsers***(): Utiliza uma consulta com `orderBy('experiencePoints', descending: true)` e `limit(50)` para buscar de forma eficiente apenas os usuários do topo da classificação.

- ***fetchUserRank***(): Como o Firestore não oferece uma função direta para obter a posição de um documento em uma consulta ordenada, foi implementada uma solução customizada. O método primeiro busca o XP do usuário atual e, em seguida, executa uma consulta de agregação (`.count()`) para contar quantos outros usuários possuem um `experiencePoints` maior. O ranking do usuário é, então, o resultado dessa contagem mais um.

Para o funcionamento destas consultas, foi necessária a criação de um índice composto no Firestore, conforme solicitado pela plataforma em sua primeira execução.

3.5.3 Serviço de Notificações *Push*

Com o objetivo de aumentar a retenção e o reengajamento dos usuários, foi implementado um sistema de notificações *push* utilizando o serviço de **Firebase Cloud Messaging (FCM)**. Este sistema permite ao aplicativo enviar mensagens informativas aos usuários mesmo quando eles não estão ativamente usando-o.

A implementação no lado do cliente foi centralizada em uma classe de serviço, `NotificationService`. O fluxo de inicialização consiste em:

1. **Solicitação de Permissão:** Ao iniciar, o aplicativo solicita ao usuário permissão para receber notificações, uma exigência dos sistemas operacionais modernos como Android e iOS.

2. **Geração e Armazenamento de *Token*:** Uma vez que a permissão é concedida, o FCM gera um *token* de registro único para o dispositivo. Este *token* é então salvo em uma lista no documento do perfil do usuário no Firestore, permitindo o envio de notificações direcionadas.

3. **Tratamento de Notificações:** A aplicação foi configurada para lidar com mensagens recebidas em dois cenários distintos. Se o aplicativo estiver em segundo plano ou fechado, a notificação é entregue diretamente pelo sistema operacional na bandeja de notificações. Se o aplicativo estiver aberto e em primeiro plano, o listener `FirebaseMessaging.onMessage` captura a notificação e exibe seu conteúdo em uma `SnackBar` não intrusiva, informando o usuário sem interromper sua atividade atual.

Para o escopo deste trabalho, o envio de notificações pode ser realizado através do painel do Cloud Messaging no Console do Firebase, uma ferramenta eficaz para enviar anúncios e novidades para todos os usuários da aplicação. A implementação de notificações automáticas e personalizadas (como lembretes de *streak*) é proposta como um trabalho futuro.

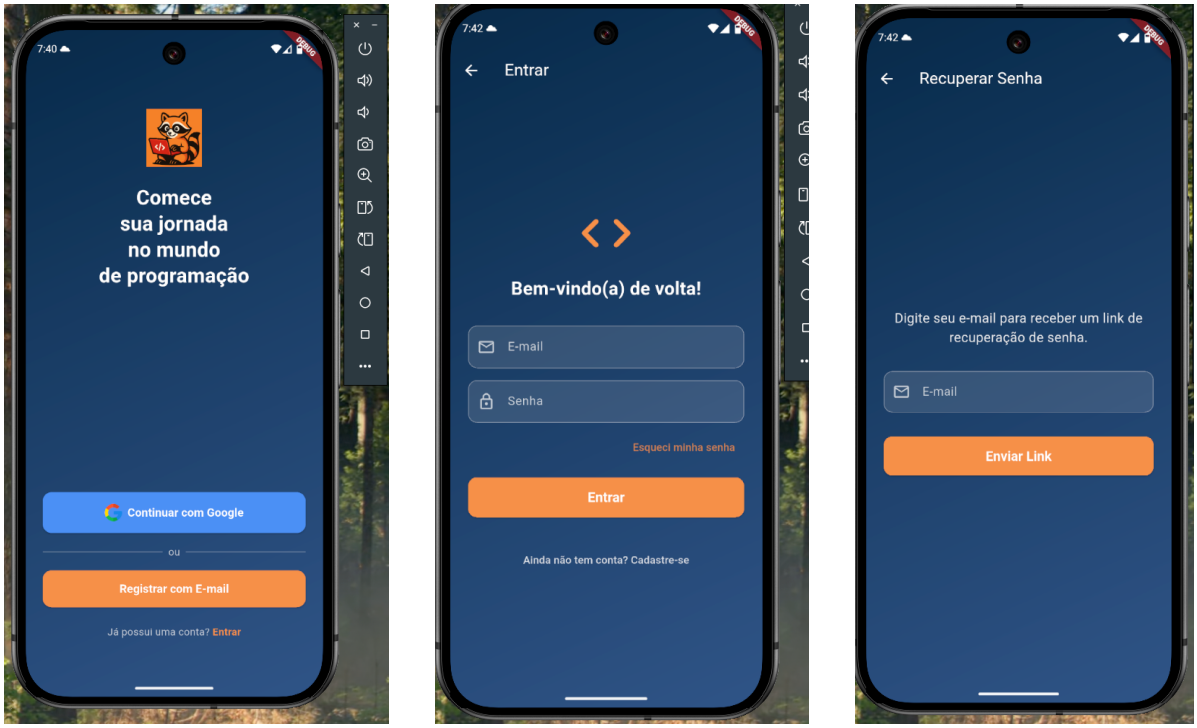
4 Resultados Obtidos

Este capítulo apresenta o resultado prático do desenvolvimento, demonstrando as principais telas e funcionalidades do aplicativo educacional de Python. O objetivo é ilustrar como a metodologia e a arquitetura descritas no capítulo anterior se materializaram em uma experiência de usuário coesa e funcional, que atende aos objetivos propostos por este trabalho.

A aplicação foi projetada com foco em uma interface limpa e uma navegação intuitiva, visando engajar o público jovem. O fluxo do usuário foi pensado para ser direto, desde o primeiro acesso até a conclusão das lições, incorporando os elementos de gamificação de forma integrada à jornada de aprendizado.

4.1 Fluxo de Autenticação e Navegação Principal

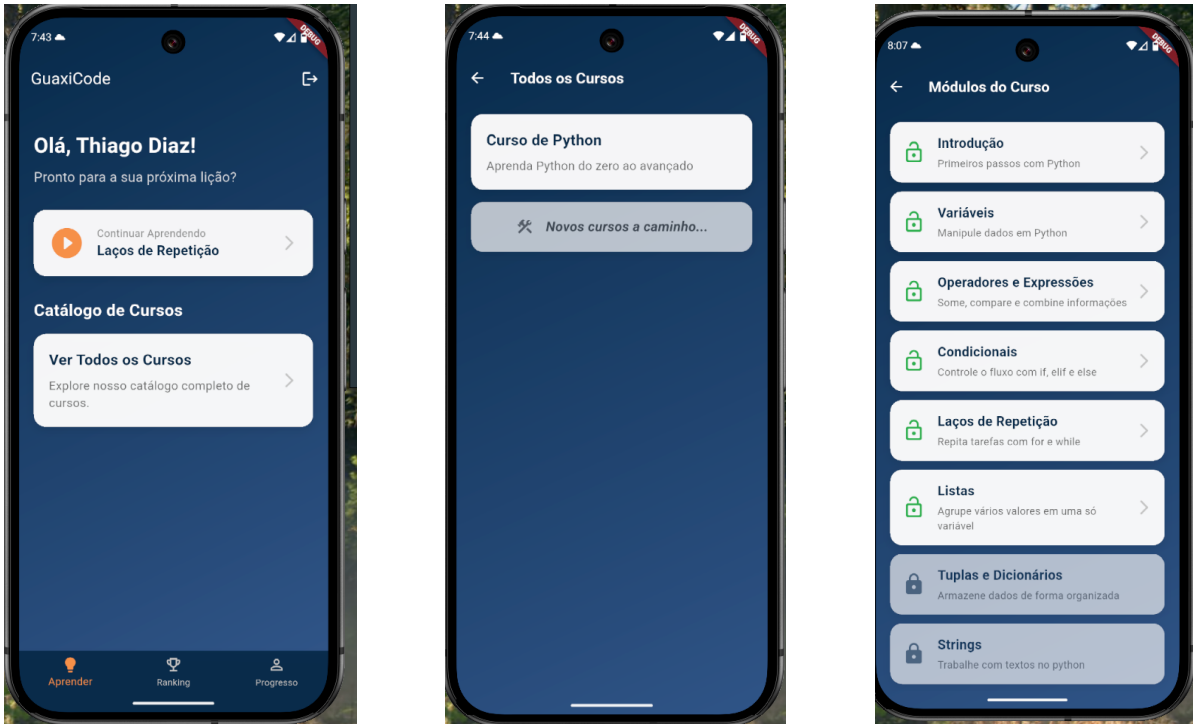
Ao iniciar o aplicativo, o usuário é recebido por uma tela de *splashscreen* que estabelece a identidade visual do projeto. Em seguida, ele é direcionado para a tela de cadastro e *login*, que serve como portal de entrada. Conforme detalhado na Figura 10, o usuário pode optar por se autenticar com uma conta Google ou criar uma conta utilizando e-mail e senha, com uma tela dedicada também para a recuperação de senha. No momento do primeiro acesso, uma entrada correspondente é criada na coleção *users* do Firestore, inicializando os dados de perfil e progresso do usuário.



(a) Opções de Cadastro (b) Login com E-mail (c) Recuperação de Senha

Figura 10 – Telas do fluxo de autenticação do usuário.

Após a autenticação, o usuário é direcionado para a tela principal, ilustrada na Figura 11, que utiliza uma barra de navegação inferior com três seções: Aprender, Ranking e Progresso. A tela inicial de **Aprender** funciona como um *dashboard*, apresentando um atalho para o próximo módulo a ser estudado e um acesso ao catálogo completo de cursos. Ao navegar para a lista de cursos e selecionar o curso de Python, o usuário visualiza a tela de módulos, que exibe de forma dinâmica quais estão bloqueados ou desbloqueados com base em seu progresso individual.



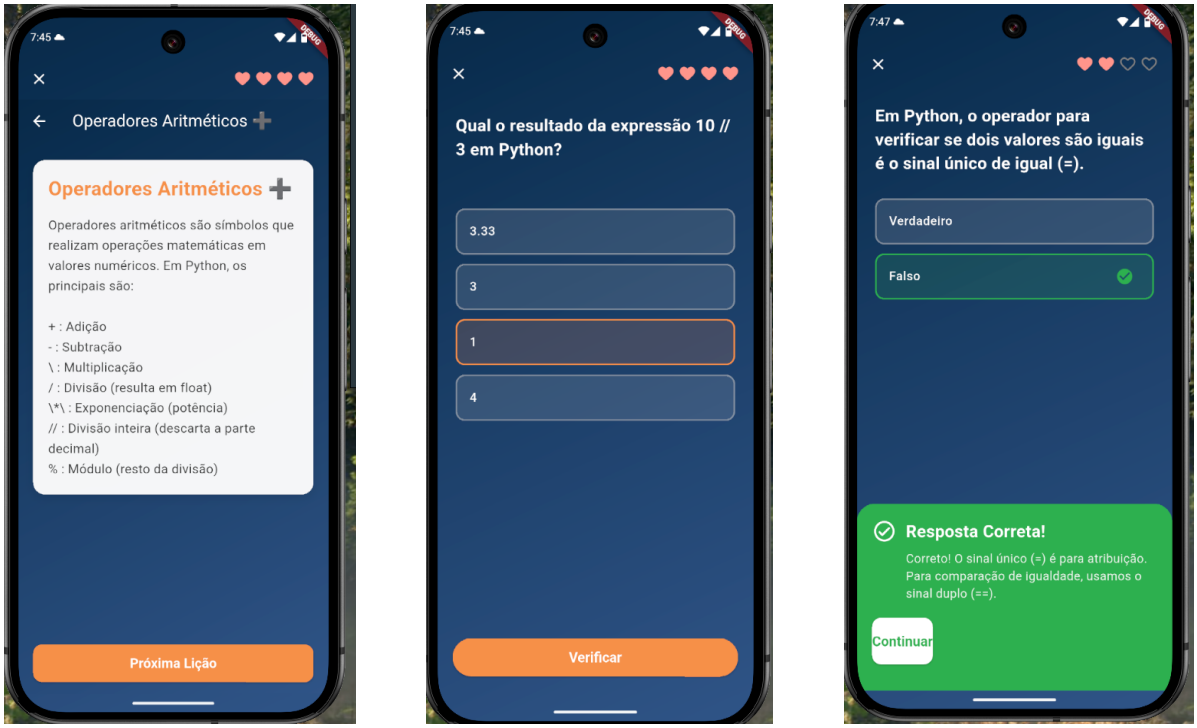
(a) Tela Home (b) Lista de Cursos (c) Módulos do Curso

Figura 11 – Navegação principal: (a) Tela Home, (b) Lista de Cursos e (c) Módulos do Curso.

4.2 Player de Lições e Gamificação

A experiência central de aprendizado ocorre na tela denominada *LessonPlayerPage*. Este componente atua como um orquestrador, apresentando o conteúdo sequencial de um módulo e gerenciando as mecânicas de gamificação. Conforme ilustrado na Figura 12, o *player* possui uma barra superior persistente que exibe o sistema de vidas do usuário.

O corpo da página exibe dinamicamente a lição, que pode ser de teoria ou de exercício. Nas lições de exercício, o usuário seleciona uma resposta e recebe um *feedback* visual imediato, verde para acerto e vermelho para erro, sendo informado sobre a perda de uma vida neste último caso. Caso todas as vidas se esgotem, um diálogo de **Fim de Jogo** é apresentado. Ao avançar, uma *SnackBar* informa o usuário sobre os pontos de experiência (XP) ganhos, reforçando o ciclo de feedback positivo.



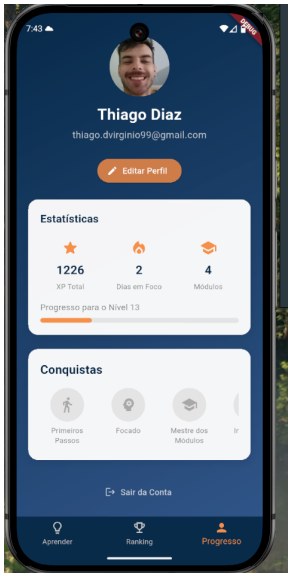
(a) Tela de Teoria (b) Exercício Interativo (c) Feedback de Resposta

Figura 12 – Ciclo de aprendizado no Player de Lições.

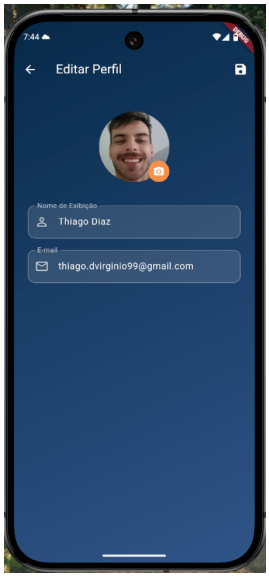
4.3 Perfil, Progresso e Notificações

Para materializar o progresso do usuário, a seção de **Progresso** exibe a *ProfilePage*. Como visto na Figura 13, esta tela funciona como um painel pessoal onde o usuário visualiza suas estatísticas (XP, nível, streak), edita seu nome e avatar, e acessa o ranking geral. O ranking exibe os melhores jogadores e destaca a posição do próprio usuário, incentivando a competição saudável.

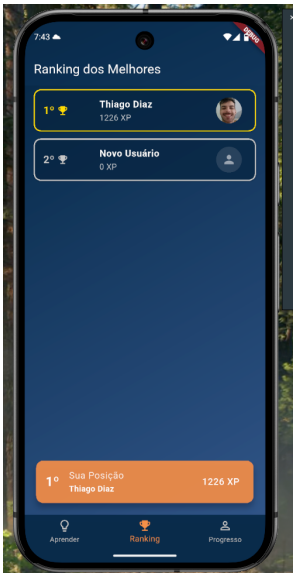
Adicionalmente, para aumentar a retenção, foi implementado um sistema de notificações *push* via Firebase Cloud Messaging (FCM). Quando o aplicativo está em primeiro plano, as notificações são recebidas de forma não intrusiva através de uma *SnackBar*, como demonstra a Figura 14.



(a) Perfil do Usuário



(b) Edição de Perfil



(c) Ranking de Jogadores

Figura 13 – Telas de progresso: (a) Perfil do Usuário, (b) Edição de Perfil e (c) Ranking.

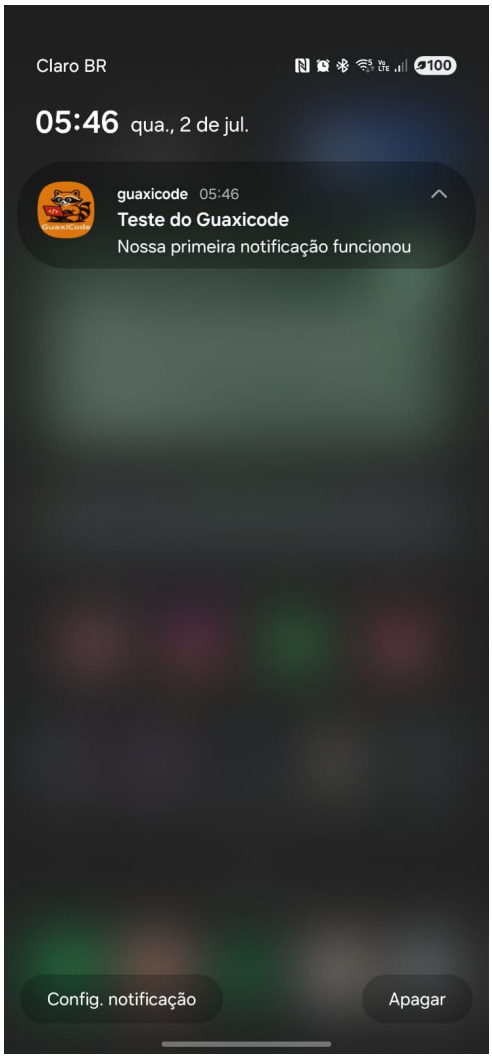


Figura 14 – Exemplo de notificação recebida com o aplicativo em primeiro plano.

5 Conclusão

O desenvolvimento de software educacional representa um desafio que transcende a mera implementação técnica, exigindo uma fusão entre pedagogia, design de experiência do usuário e engenharia de software. Este Trabalho de Conclusão de Curso documentou o processo completo de concepção, planejamento e desenvolvimento de um aplicativo móvel gamificado para o ensino da linguagem de programação Python, utilizando tecnologias modernas como o framework Flutter e a plataforma Firebase.

O objetivo geral de desenvolver um aplicativo educativo, interativo e gamificado foi plenamente alcançado. A materialização deste objetivo se deu através da implementação de uma plataforma robusta que apresenta conteúdo teórico e exercícios práticos de forma sequencial e focada. Para aumentar o engajamento, foram integradas mecânicas de gamificação essenciais, como um sistema de vidas, a atribuição de pontos de experiência (XP) condicionada ao desempenho e um sistema de desbloqueio progressivo de conteúdo. A sustentação técnica dessas funcionalidades foi garantida por uma arquitetura de software escalável, baseada no *Repository Pattern*, que assegura uma clara separação entre a interface do usuário e a lógica de acesso a dados.

Conclui-se que o protótipo desenvolvido é uma solução viável e promissora para os desafios do ensino de programação para iniciantes, combinando conteúdo estruturado com uma experiência de usuário interativa e motivadora. O projeto não apenas atingiu seus objetivos técnicos, mas também representou uma valiosa experiência de aprendizado acadêmico e profissional, solidificando conhecimentos em desenvolvimento móvel, design de sistemas e gerenciamento de projetos de software.

5.1 Trabalhos Futuros

Embora o protótipo desenvolvido seja funcional e atenda aos objetivos propostos, o projeto GuaxiCode foi concebido como uma plataforma com grande potencial de expansão. Como próximos passos para a continuidade do trabalho, são propostas as seguintes frentes de atuação:

5.1.1 Validação com Usuários e Publicação

O próximo passo imediato consiste na validação da proposta com usuários reais. A estratégia inicial será a distribuição do arquivo de instalação do aplicativo (*.apk*) em um ambiente controlado, para estudantes do Departamento de Computação da UFSCar. O objetivo desta fase de testes é coletar *feedback* qualitativo sobre a usabilidade, a eficácia do método de ensino e o engajamento gerado pelas mecânicas de gamificação.

Com base nos dados e sugestões coletadas nesta validação inicial, o aplicativo será refinado e, subsequentemente, preparado para a publicação oficial na Google Play Store. Este lançamento permitirá alcançar um público mais amplo e validar a solução em um ambiente fora do contexto acadêmico, além de possibilitar a coleta de métricas de uso em larga escala.

5.1.2 Evolução da Gamificação e Conteúdo

Uma análise crítica do sistema de gamificação atual aponta para uma melhoria necessária na mecânica de vidas. No modelo implementado, a quantidade fixa de cinco vidas pode gerar uma experiência excessivamente punitiva em módulos muito extensos, desmotivando o aluno que, ao cometer um erro próximo ao final de uma longa sequência, é forçado a reiniciar todo o seu progresso. Para refinar essa mecânica, sugere-se como trabalho futuro a implementação de um sistema mais dinâmico. Uma abordagem seria tornar o número inicial de vidas proporcional à quantidade de exercícios no módulo. Outra, ainda mais robusta, seria permitir que o usuário recupere uma vida a cada sequência de acertos consecutivos, recompensando o bom desempenho e aumentando a resiliência do estudante.

Adicionalmente, a gamificação pode ser aprofundada com a implementação da lógica para a concessão de conquistas (*badges*) e a expansão do sistema de ranking com classificações semanais ou mensais para manter a competitividade a longo prazo. A experiência de aprendizado também pode ser enriquecida com o desenvolvimento de novos tipos de exercícios interativos, como o de seleção de blocos de código.

5.1.3 Expansão da Plataforma

Para transformar o aplicativo em um verdadeiro ecossistema de aprendizado para a comunidade acadêmica, os trabalhos futuros preveem também a expansão da plataforma para que outros professores e grupos da UFSCar possam criar e publicar seus próprios cursos. Isso seria suportado pelo desenvolvimento de um painel administrativo web, que facilitaria o gerenciamento de conteúdo sem a necessidade de conhecimento técnico sobre o banco de dados, e, por fim, a publicação do aplicativo na App Store para dispositivos iOS.

Referências

[1] JOHNSON, L.; ADAMS, S.; CUMMINS, M. *The NMC horizon report: 2012 higher education edition*. Austin, Texas: The New Media Consortium, 2012.

[2] BORGES, Luiz Eduardo. *Python para desenvolvedores: aborda Python 3.3*. Novatec Editora, 2014.

[3] GOOGLE. *Flutter UI Documentation*. Disponível em: https://docs.flutter.dev/ui. Acesso em: 01 de julho de 2025.

[4] SINGH, Y.; SURI, P.K. An empirical analysis of mobile learning app usage experience. *Technology in Society*, v. 68, p. 101929, 2022.

[5] CROMPTON, H.; BURKE, D. The use of mobile learning in higher education: A systematic review. *Computers & Education*, v. 123, p. 53-64, 2018.

[6] PIOTROWSKI, J.T.; MEESTER, L. Can apps support creativity in middle childhood?. *Computers in Human Behavior*, v. 85, p. 23-33, 2018.

[7] REDE NACIONAL DE ENSINO E PESQUISA. *A importância da programação para o futuro do trabalho*. Disponível em: https://esr.rnp.br/desenvolvimento-de-sistemas/a-importancia-da-programacao-para-o-futuro-do-trabalho/. Acesso em: 01 de julho de 2025.

[8] G1. *Entenda se programação ainda vale a pena; profissionais contam como está o setor*. Disponível em: https://g1.globo.com/tecnologia/noticia/2023/05/09/entenda-se-programacao-ainda-vale-a-pena-profissionais-contam-como-esta-o-setor
ghtml. Acesso em: 01 de julho de 2025.

[9] MOURA, Adelina. Mobile learning: Tendências tecnológicas emergentes. In: *Aprender na era digital: Jogos e Mobile-Learning*. Edições Sílabo, 2012. p. 127-147.

[10] CODECADEMY. *Codecademy*. Disponível em: https://www.codecademy.com/. Acesso em: 07 de julho de 2025.

[11] GOOGLE. *Grasshopper: Learn to Code*. Disponível em: https://grasshopper.app/. Acesso em: 07 de julho de 2025.

[12] ALURA. *Alura Cursos Online de Tecnologia*. Disponível em: https://www.alura.com.br/. Acesso em: 07 de julho de 2025.

[13] PROGRAMMING HUB. *Programming Hub: Learn to Code*. Disponível em: https://programminghub.io/. Acesso em: 07 de julho de 2025.

[14] MIMO. *Mimo: Learn to Code*. Disponível em: https://mimo.org/. Acesso em: 07 de julho de 2025.

[15] SOLOLEARN. *Sololearn: Learn to Code*. Disponível em: https://www.sololearn.com/. Acesso em: 07 de julho de 2025.