INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
CÂMPUS PELOTAS - VISCONDE DA GRAÇA
CURSO TÉCNICO EM DESENVOLVIMENTO DE SISTEMA

PROGGAME: UMA PLATAFORMA GAMIFICADA PARA APRIMORAR HABILIDADES EM PROGRAMAÇÃO

Níkollas Moralles

Pelotas, 2024.

Níkollas Moralles

ProgGame: Uma Plataforma gamificada para aprimorar habilidades em Programação

Projeto de Desenvolvimento de Sistema apresentado como requisito na disciplina de Metodologia da Pesquisa II do curso Técnico em Desenvolvimento de Sistemas, do Instituto Federal Sul-rio-grandense – Campus Pelotas - Visconde da Graça.

Orientador: Prof. Dr. Raymundo Carlos Machado Ferreira Filho

Pelotas, 2024.

INDICE DE QUADROS

ÍNDICE DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

No mundo atual, a programação tornou-se uma habilidade essencial, abrindo portas para diversas áreas profissionais e impulsionando o desenvolvimento tecnológico. Nesse cenário, surge a necessidade de ferramentas para auxiliar no aprendizado e aprimoramento da prática de programação. Plataformas de estudos online para exercitar e praticar se apresentam como uma solução promissora, oferecendo um ambiente interativo e desafiador para programadores de todos os níveis.

O principal problema que este Trabalho de Conclusão de Curso (TCC) busca abordar, reside na dificuldade que muitos programadores, principalmente iniciantes, enfrentam em encontrar plataformas de estudos e exercícios práticos adequados às suas necessidades.

O trabalho proposto teve o objetivo de projetar e desenvolver uma plataforma de estudos e prática de programação e que ofereça uma boa experiência de aprendizado. A plataforma tem como foco principal atender às necessidades de programadores iniciantes, mas também fornece recursos para usuários de níveis mais avançados.

1.1 Tema

Tema: Plataforma de Prática de Programação
Delimitação:

- **Público-alvo:** Programadores iniciantes e intermediários.
- **Linguagens de programação:** Python.
- **Funcionalidades:** Desafios interativos e sistema de gamificação

1.2 Motivações

O desenvolvimento desta plataforma se justifica por diversos motivos. Em primeiro lugar, há uma necessidade de ferramentas de aprendizado, e a plataforma em questão fornece um ambiente para exercitar a programação de forma dinâmica, otimizando o processo de aprendizado.

Além disso, uma das motivações foi pessoal, pois percebi que era necessária a prática diária de programação para que houvesse uma melhora significativa no domínio de alguma linguagem. Outro motivo foi estimular o interesse pela programação, utilizando recursos como gamificação para tornar o aprendizado mais interessante. Por fim, a plataforma promove o desenvolvimento profissional, uma vez que o domínio da programação abre portas para diversas oportunidades no mercado de trabalho, contribuindo para a formação de profissionais qualificados.

1.3 Objetivos

Objetivo Geral:

Projetar e desenvolver uma plataforma de prática de programação, direcionada para programadores iniciantes e intermediários, com o objetivo de aprimorar suas habilidades e conhecimentos em linguagens como Python.

Objetivos Específicos:

a) realizar a análise dos requisitos necessários à implementação do sistema;
b) projetar a modelagem dos dados que serão necessários para o sistema;
c) prototipar e realizar uma avaliação sobre a proposta de layout da plataforma com os usuários pertencentes ao público alvo;
d) Utilizar elementos de jogos (*gamificação*) para tornar o aprendizado mais motivador e engajador, incentivando os usuários a progredirem na plataforma.

2 FUNDAMENTAÇÃO TEÓRICA

No Capítulo 2, intitulado "Fundamentação Teórica", abordamos dois eixos principais que sustentaram o desenvolvimento da plataforma de prática de programação: Engenharia de Software e Gamificação. Esses temas foram selecionados por sua relevância na construção de sistemas e na promoção de experiências mais ricas para os usuários.

Na subseção 2.1, Engenharia de Software, explorou-se os princípios e metodologias que estruturam o desenvolvimento de softwares. De acordo com Pressman (2011), a Engenharia de Software é uma abordagem sistemática e disciplinada para o desenvolvimento, operação e manutenção de sistemas, sendo essencial no contexto de plataformas com o propósito de ensinar. Apresentou-se etapas como a engenharia de requisitos, conforme detalhada por Sommerville (2011), bem como a arquitetura modular para garantir escalabilidade, facilitar a adição de novas funcionalidades e promover a manutenção contínua da plataforma.

Na subseção 2.2, Gamificação, apresentou-se os elementos de jogos integrados em contextos não lúdicos que podem aumentar a motivação e o engajamento dos usuários. Fundamentados em Werbach e Hunter (2012), explorou-se pontuação, níveis e recompensas para criar um ambiente motivador para o usuário. Apresentou-se ainda as teorias de motivação de Deci e Ryan (2000), que explicam como diferentes perfis de usuários podem ser incentivados a participar ativamente do aprendizado. Nesse sentido, a gamificação não apenas torna o processo de prática mais atrativo, mas também fornece feedback contínuo, promovendo autonomia e retenção de conhecimento.

Enquanto a Engenharia de Software oferece as bases técnicas e metodológicas, a Gamificação adiciona uma camada de engajamento e interatividade, essencial para manter os usuários comprometidos. Dessa forma, a fundamentação teórica apresentada neste capítulo fornece um embasamento para o desenvolvimento da plataforma, alinhando a técnica com experiências enriquecedoras para os usuários.

2.1 Engenharia de Software

A Engenharia de Software, de acordo com Pressman (2011) é uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, a operação e a manutenção de software. Podemos dizer que a Engenharia de Software aplica-se diretamente no contexto de desenvolvimento de sistemas complexos e de plataformas interativas, como uma plataforma de prática de programação. Esse campo é composto por processos e técnicas que suportam o ciclo de vida do software, desde a concepção inicial até a manutenção e evolução de sistemas em produção.

O desenvolvimento de uma plataforma de prática de programação requer uma base sólida em princípios de Engenharia de Software para lidar com desafios como a definição de requisitos, a modularidade do sistema, a facilidade de manutenção e a experiência do usuário. Nesse contexto, a Engenharia de Software fornece diretrizes que estruturam o desenvolvimento, facilitando a organização do código e permitindo uma expansão mais controlada do sistema conforme novas funcionalidades são exigidas. Além disso, as técnicas de Engenharia de Software contribuem para a organização e clareza no levantamento de requisitos, auxiliando na criação de uma solução que atenda de forma eficaz às necessidades dos usuários, que, neste caso, incluem estudantes e entusiastas da programação.

De acordo com Sommerville (2011), a Engenharia de Software é dividida em diversas etapas e processos fundamentais, sendo a engenharia de requisitos um dos primeiros passos cruciais para o sucesso de um projeto. Esta fase inclui o levantamento, a análise e a especificação dos requisitos, que são as funcionalidades e características desejadas pelos usuários. No caso de uma plataforma de prática de programação, os requisitos podem incluir aspectos como o cadastro de usuários, o sistema de feedback em tempo real e uma interface intuitiva para diferentes níveis de habilidade. Esse levantamento é essencial para assegurar que o sistema desenvolvido atenda às expectativas e contribua para uma curva de aprendizado positiva.

A plataforma desenvolvida também se beneficia dos conceitos de projeto e arquitetura de software, outra área de destaque na Engenharia de Software. Para garantir escalabilidade e modularidade, a plataforma é organizada em módulos funcionais, como cadastro, prática de exercícios e visualização de progresso. Esse tipo de organização modular facilita não apenas a implementação inicial, mas também futuras expansões e melhorias, como a adição de novos exercícios e funcionalidades sem impactar o sistema como um todo. Essa prática também está em conformidade com os padrões arquitetônicos promovidos pela Engenharia de Software, que visam à robustez e à manutenção do sistema.

Além do projeto inicial, a Engenharia de Software propõe práticas de validação e testes de software para garantir que cada componente do sistema funcione conforme o planejado. A plataforma de prática de programação implementa testes automatizados para verificar o correto funcionamento de exercícios, garantindo que os usuários recebam feedback imediato e preciso sobre seus códigos. Esse processo de validação e teste não apenas ajuda a manter a confiabilidade do sistema, mas também contribui para a satisfação dos usuários, uma vez que minimiza erros e possibilita uma experiência de uso contínua e sem interrupções.

Dessa forma, a Engenharia de Software fundamenta o desenvolvimento de sistemas complexos e desempenha um papel central na criação de uma plataforma de prática de programação. Com a aplicação de metodologias e práticas consagradas, é possível desenvolver uma solução robusta, eficiente e alinhada com as necessidades dos usuários, permitindo que o sistema atinja seu objetivo de facilitar o aprendizado de programação e acompanhar a evolução dos usuários de maneira contínua. Assim, os conceitos da Engenharia de Software permeiam todo o desenvolvimento da plataforma, desde a concepção inicial até a manutenção, garantindo que o produto final seja estável, expansível e de fácil uso.

2.2 Gamificação

A gamificação, definida como o uso de elementos de jogos em contextos não lúdicos, tem sido amplamente adotada para aumentar o engajamento, a motivação e o aprendizado em ambientes educacionais e corporativos. De acordo com Werbach e Hunter (2012), a gamificação é fundamentada na psicologia da motivação e utiliza mecânicas como recompensas, níveis, pontuação e desafios para incentivar os usuários a se envolverem de maneira mais ativa e significativa em atividades que, de outra forma, poderiam ser vistas como monótonas ou desmotivadoras. Ao integrar esses elementos, é possível criar um ambiente que promova a sensação de progresso e a autorrealização, essenciais para manter o usuário envolvido e comprometido com o processo.

No contexto educacional, a gamificação tem demonstrado ser uma ferramenta eficaz para melhorar o aprendizado e a retenção de conhecimento. Conforme Deterding *et al.* (2011), a gamificação possibilita uma abordagem de aprendizagem baseada em objetivos claros, feedback imediato e recompensas, elementos que incentivam a prática contínua e o aprimoramento de habilidades. Quando aplicada a plataformas de prática de programação, por exemplo, a gamificação pode ajudar a manter os estudantes envolvidos enquanto eles avançam por níveis de dificuldade, ganham pontos por cada exercício concluído e recebem recompensas virtuais, como medalhas e distintivos, pelo progresso alcançado.

A eficácia da gamificação em ambientes educacionais também é explicada pelo conceito de motivação intrínseca e extrínseca, conforme Deci e Ryan (2000). A motivação intrínseca ocorre quando a pessoa se engaja em uma atividade por pura satisfação, enquanto a extrínseca é incentivada por recompensas externas. No design de uma plataforma gamificada, ambos os tipos de motivação são explorados: a intrínseca, ao permitir que os estudantes vejam o próprio progresso e superem desafios, e a extrínseca, ao incluir pontuações, conquistas e recompensas. Segundo estudiosos como Hamari *et al.* (2014), esse equilíbrio é fundamental para o sucesso da gamificação, pois ele cria uma experiência de aprendizado gratificante que atende a diferentes perfis de usuários.

Para a criação de uma plataforma de prática de programação, a gamificação proporciona não apenas uma abordagem mais atrativa, mas também um sistema de feedback contínuo, um dos pilares da aprendizagem eficaz (Kapp, 2012). A possibilidade de receber um retorno instantâneo por meio de sistemas gamificados ajuda os estudantes a identificar rapidamente os erros e corrigi-los, promovendo uma experiência de aprendizado autônoma e interativa. Assim, a gamificação contribui para um ambiente de aprendizado onde os usuários sentem que estão progredindo, o que aumenta a retenção e o domínio das habilidades.

3 PERCURSO METODOLÓGICO

3.1 Especificação De Requisitos

Para coletar os requisitos da plataforma de prática de programação e garantir que ela atenda às necessidades dos usuários, foi utilizado o seguinte método:

Questionários com Usuários: Aplicação de questionários com usuários para:

- avaliar suas necessidades, expectativas e preferências em relação à plataforma;
- coletar feedback sobre a usabilidade, funcionalidade e efetividade da plataforma;
- identificar pontos de melhoria e oportunidades para otimizar a experiência do usuário.

O questionário foi realizado por alunos do curso Técnico em Desenvolvimento de Sistemas do Instituto Federal Sul-riograndense Campus Visconde da Graça.

As questões com usuários foram com relação a assiduidade com que exercitam a programação; as dificuldades encontradas, obstáculos e quais recursos costumam utilizar e se consideravam uma boa ideia uma plataforma. O link para questionário realizado: https://forms.gle/TzYuQ7MZ6tBhmppQ8.

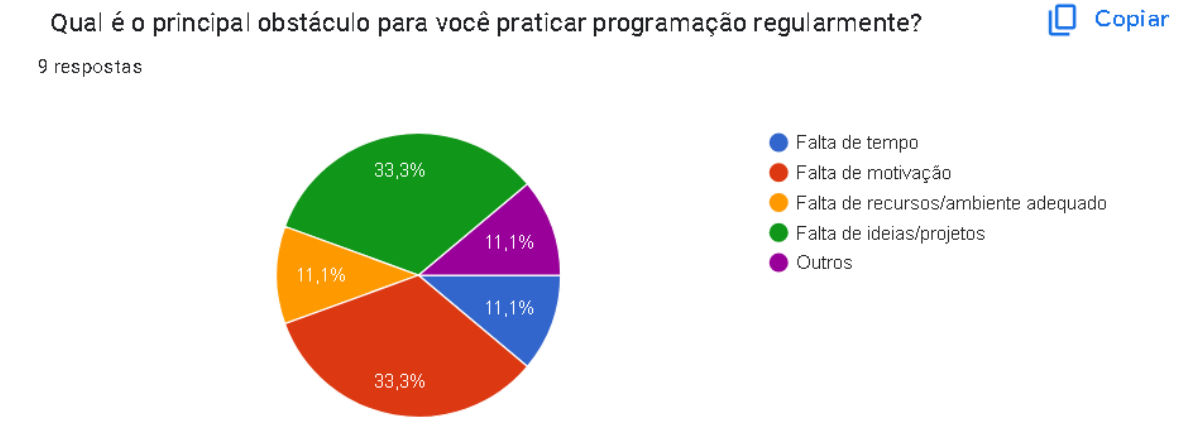
Figura 1: Gráfico de respostas da primeira questão do questionário

Fonte: autoria própria

Figura 2: Gráfico de respostas da segunda questão do questionário

Fonte: autoria própria

Figura 3: Gráfico de respostas da terceira questão do questionário

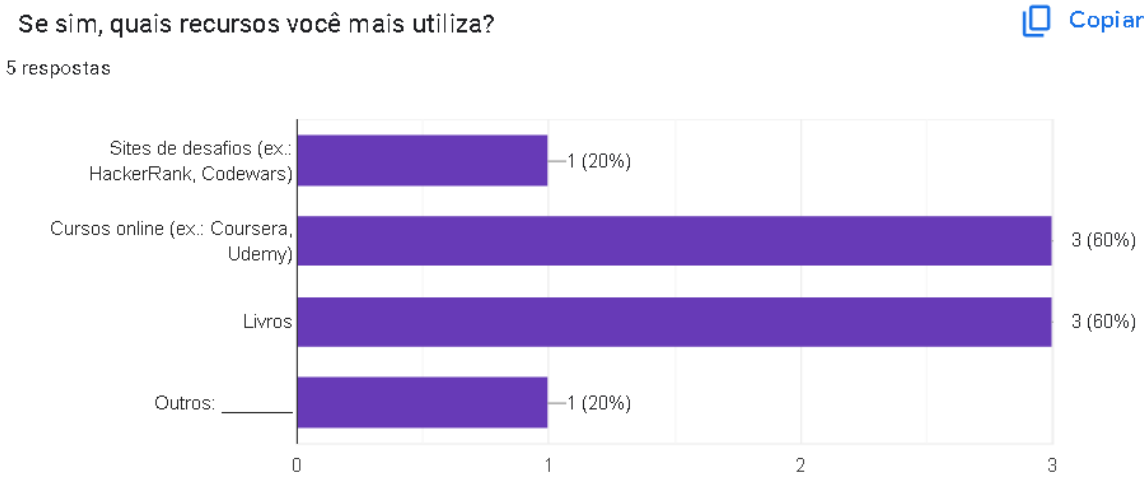


Fonte: autoria própria

Figura 4: Gráfico de respostas da quarta questão do questionário

Fonte: autoria própria

Figura 5: Gráfico de respostas da quinta questão do questionário

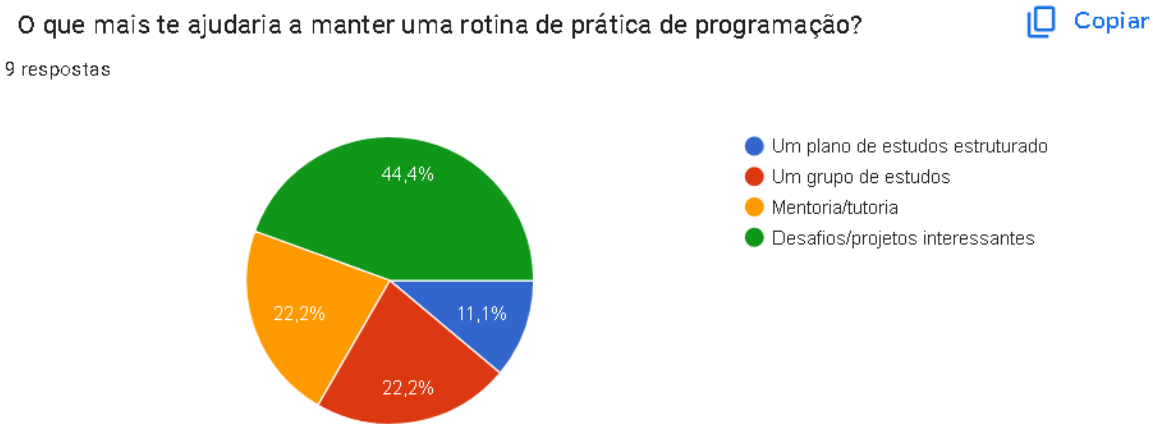


Fonte: autoria própria

Figura 6: Gráfico de respostas da sexta questão do questionário

Fonte: autoria própria

Figura 7: Gráfico de respostas da sétima questão do questionário

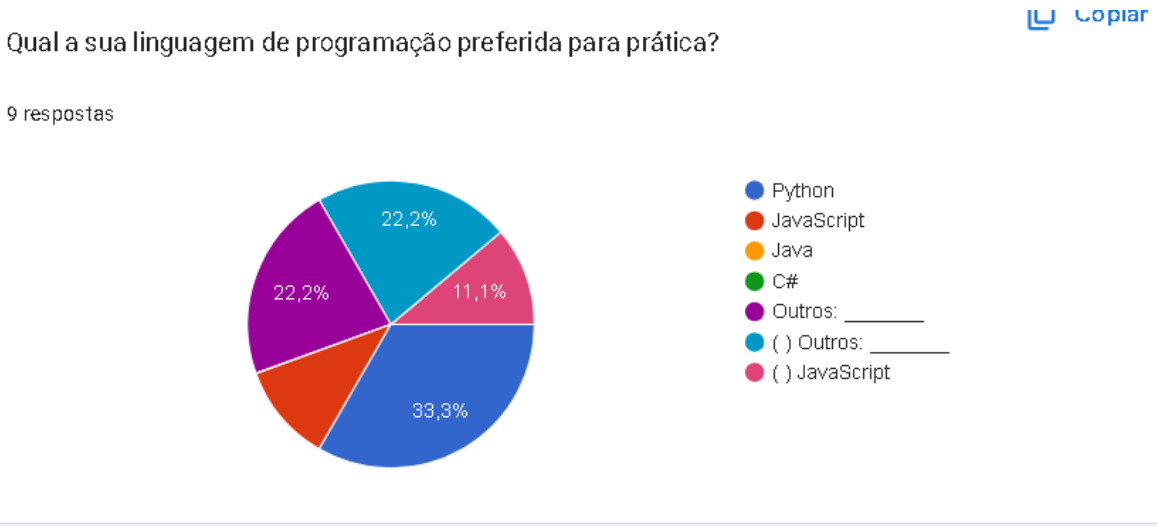


Fonte: autoria própria

Figura 8: Gráfico de respostas da oitava questão do questionário

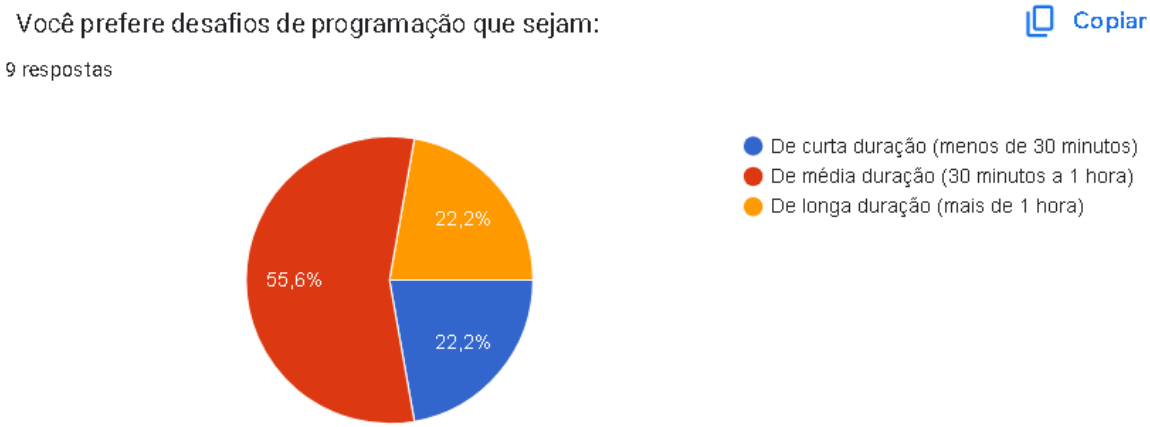
Fonte: autoria própria

Figura 9: Gráfico de respostas da nona questão do questionário



Fonte: autoria própria

Figura 10: Gráfico de respostas da décima questão do questionário



Fonte: autoria própria

3.2 Requisitos Funcionais

Os requisitos funcionais em uma modelagem de banco de dados são especificações que definem todas as etapas de elicitação de um sistema: Concepção, levantamento de requisitos, caso de uso e o sistema de banco de dados deve ser capaz de fazer. Eles descrevem as funções e operações que o banco de dados deve realizar para suportar as atividades e processos de negócios da organização. Os requisitos funcionais são fundamentais para garantir que o banco de dados desenvolvido suporte eficazmente as operações de negócio e atenda às necessidades dos usuários.

A partir das entrevistas e questionários realizados, elencou-se as principais funcionalidades que o sistema precisaria contemplar (Quadro 1).

Quadro 1: Tabela de Requisitos Funcionais

Requisito	Função	Descrição
REF 01	Gerenciar usuários	Permite que novos usuários se cadastrem na plataforma utilizando seus dados pessoais (nome, email, senha). Permite que os usuários cadastrados criem seus perfis na plataforma. Oferece opções para personalizar o perfil, incluindo foto, interesses e nível de conhecimento.
REF 02	Selecionar Linguagem de Programação	Permite que os usuários selecionem a linguagem de programação desejada para praticar, no momento apenas python estará disponível.
REF 03	Selecionar Dificuldade	Permite que os usuários selecionem o nível de dificuldade dos desafios que desejam completar: Iniciante, Intermediário ou Avançado.
REF 04	Escolher Desafios	Permite que o usuário escolha qual desafio ele deseja resolver.
REF 05	Acompanhar Progresso	Registra o progresso do usuário na plataforma, incluindo desafios completados, pontos conquistados e níveis alcançados.
REF 06	Atribuir Pontuação	Define a pontuação que cada desafio dará para o usuário e a quantidade de pontos necessários para passar de nível.
REF 07	Exibir pontos e nível	Exibe pontuação total e nível de usuário em seu perfil.

Fonte: autoria própria

3.3 Requisitos Não Funcionais

Requisitos não funcionais são especificações que definem critérios de funcionamento de um sistema, mas não estão diretamente relacionados às funcionalidades ou ao comportamento do sistema. Em vez disso, eles descrevem atributos de qualidade e restrições do sistema, garantindo que ele opere de maneira eficiente, segura e utilizável.

Um exemplo de requisito não funcional é o desempenho, que define a capacidade do sistema de realizar funções dentro de um tempo específico, abrangendo tempos de resposta, taxa de processamento e capacidade de throughput. Outro exemplo crucial é a segurança, que envolve medidas para proteger o sistema contra acessos não autorizados, violações de dados e outras ameaças de segurança.

Esses aspectos são apresentados de forma detalhada no Quadro 2.

Quadro 2: requisitos não-funcionais

Requisito	Classificação	Descrição
RNF 01	Usabilidade	A plataforma deve apresentar uma interface gráfica simples e intuitiva, com navegação fácil e menus claros.
RNF 02	Usabilidade	A plataforma deve ter uma experiência de usuário aprimorada para usuários com diferentes níveis de conhecimento técnico e habilidades
RNF 03	Usabilidade	A plataforma deve se adaptar automaticamente a diferentes tamanhos de tela, permitindo o acesso em computadores, tablets e smartphones.
RNF 04	Desempenho	A plataforma deve estar disponível para uso a maior parte do tempo, com interrupções mínimas para manutenção e atualizações.
RNF 05	Desempenho	A plataforma deve apresentar tempo de resposta rápido para todas as ações do usuário, minimizando atrasos e lentidão.
RNF 06	Desempenho	A plataforma deve ser escalável para suportar um grande número de usuários simultâneos, sem comprometer o desempenho.
RNF 07	Confiabilidade	A plataforma deve ser desenvolvida com tecnologias confiáveis e comprovadas, garantindo estabilidade e minimizando falhas.
RNF 08	Confiabilidade	A plataforma deve implementar um sistema de

		monitoramento e logs para identificar e solucionar problemas rapidamente.
RNF 09	Desenvolvimento	A aplicação será desenvolvida em: PHP, MySQL, HTML, CSS e JavaScript

Fonte: autoria própria

3.4 Modelagem do Banco de Dados

A modelagem de banco de dados é uma etapa crucial no desenvolvimento de sistemas, trazendo inúmeros benefícios que justificam sua importância. Conforme Heuser (2009), ela permite uma organização lógica e estruturada dos dados, facilitando o entendimento e a manipulação das informações. Com uma boa modelagem, evita-se a duplicação de dados, o que economiza espaço de armazenamento e melhora a eficiência das operações (Heuser, 2009).

Além disso, a modelagem garante a integridade e a consistência dos dados, aplicando regras de integridade referencial que evitam erros e inconsistências (Elmasri e Navathe, 2011). Isso é vital para a confiabilidade das informações armazenadas e para a precisão das operações realizadas sobre esses dados. Um banco de dados bem modelado também é mais fácil de manter e atualizar, pois mudanças podem ser feitas de forma mais controlada e com menos impacto nas aplicações que utilizam o banco (Heuser, 2009).

Outro ponto importante é a melhoria de desempenho que um design bem planejado pode proporcionar. A modelagem permite a otimização do uso de índices e outras técnicas de acesso, resultando em consultas e operações de banco de dados mais rápidas e eficientes (Elmasri e Navathe, 2011). Além disso, facilita a adaptação e evolução do sistema, permitindo que novas funcionalidades sejam incorporadas de maneira mais ágil e menos disruptiva (Heuser, 2009).

A modelagem de banco de dados também desempenha um papel crucial na estrutura e nas regras do banco de dados (Heuser, 2009).

3.5 Modelo de Casos de Uso

O modelo de casos de uso é uma ferramenta essencial no desenvolvimento de sistemas, desempenhando um papel crucial na definição e comunicação dos requisitos funcionais. Conforme Oliveira (2009), este modelo é importante por várias razões que beneficiam tanto os desenvolvedores quanto os stakeholders.

Em primeiro lugar, o modelo de casos de uso oferece uma maneira clara e compreensível de descrever as interações entre os usuários (atores) e o sistema. Pereira (2006) destaca que, ao utilizar uma linguagem simples e visual, ele facilita o entendimento das funcionalidades que o sistema deve oferecer, mesmo para aqueles que não possuem conhecimentos técnicos avançados. Isso é fundamental para assegurar que todas as partes interessadas tenham uma visão alinhada do que o sistema deve fazer.

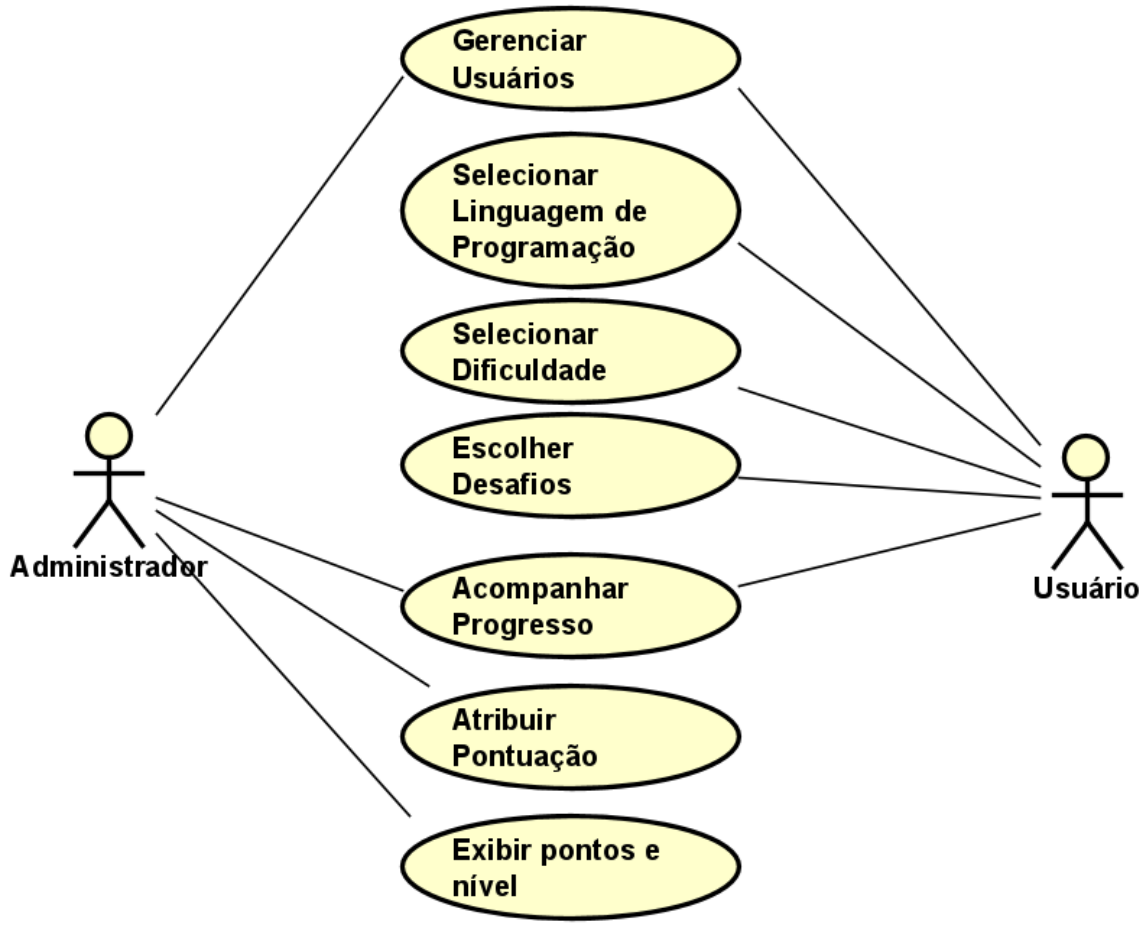
Além disso, o modelo de casos de uso ajuda a identificar e documentar os requisitos funcionais do sistema de forma sistemática. Cada caso de uso descreve um cenário específico de interação entre um ator e o sistema, detalhando as ações e as respostas esperadas. Oliveira (2009) afirma que isso proporciona uma base sólida para o desenvolvimento e a validação do sistema, garantindo que todos os requisitos importantes sejam considerados e implementados.

Outro benefício significativo é que o modelo de casos de uso facilita a priorização dos requisitos. Pereira (2006) menciona que, ao visualizar todos os casos de uso, os stakeholders podem determinar quais funcionalidades são mais críticas e devem ser desenvolvidas primeiro. Isso ajuda a direcionar os esforços de desenvolvimento para as áreas de maior impacto, otimizando o uso dos recursos disponíveis.

O modelo de casos de uso também promove a comunicação eficaz entre os membros da equipe de desenvolvimento. Ele serve como um ponto de referência comum, reduzindo ambiguidades e mal-entendidos sobre as funcionalidades do sistema (Oliveira, 2009). Isso é especialmente importante em projetos grandes e complexos, onde a coordenação entre diferentes equipes é essencial para o sucesso.

O Diagrama de Caso de Uso do sistema desenvolvido está apresentado na Figura 11.

Figura 11: Diagrama de Caso de uso

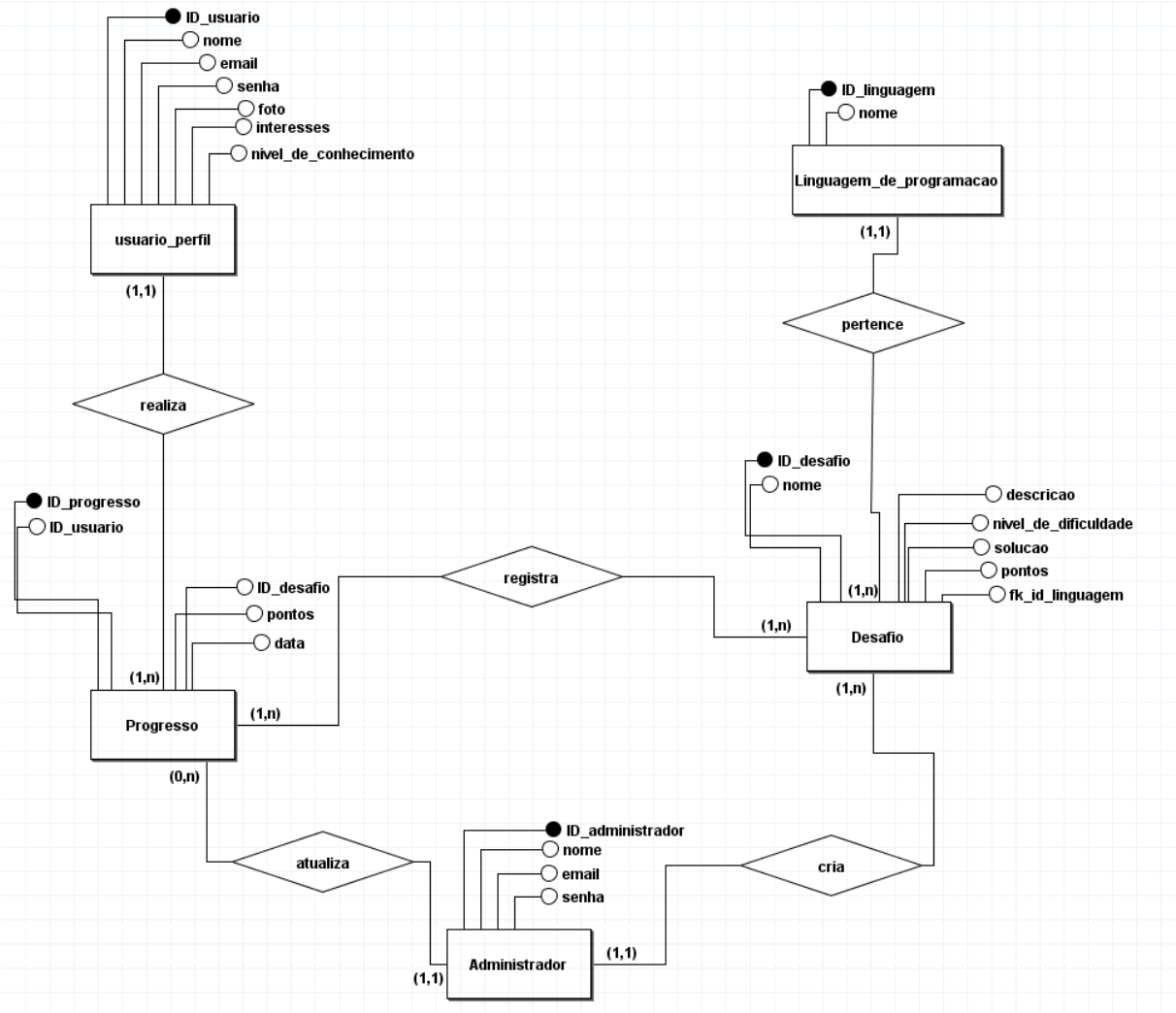


Fonte: autoria própria

3.6 Modelagem Conceitual do Banco de Dados

A modelagem de banco de dados é uma etapa essencial no desenvolvimento de sistemas, pois permite a organização estruturada e eficiente dos dados que serão armazenados e manipulados. O modelo Entidade-Relacionamento (E-R) é uma ferramenta fundamental nesse processo, pois facilita a visualização das entidades do sistema e seus relacionamentos, garantindo a integridade e a consistência dos dados.

Figura 12: Diagrama conceitual de Banco de Dados



Fonte: autoria própria

A entidade **usuário** representará todos os usuários cadastrados no sistema, como atributo eles terão: ID_ usuário (que será gerada ao usuário criar sua conta), nome, email e senha. Os usuários têm apenas um perfil, ou seja, eles possuem e podem personalizar apenas um perfil, podendo criar no mínimo um e apenas um perfil, representado pela cardinalidade (1,1). Além disso, o usuário pode escolher no mínimo uma linguagem de programação ou várias ao mesmo tempo, sendo denotado pela cardinalidade (1,n).

A entidade **Desafio** é a área onde o usuário irá escolher o desafio que deseja realizar, podendo escolher seu nível de dificuldade. Ao ser completado por algum

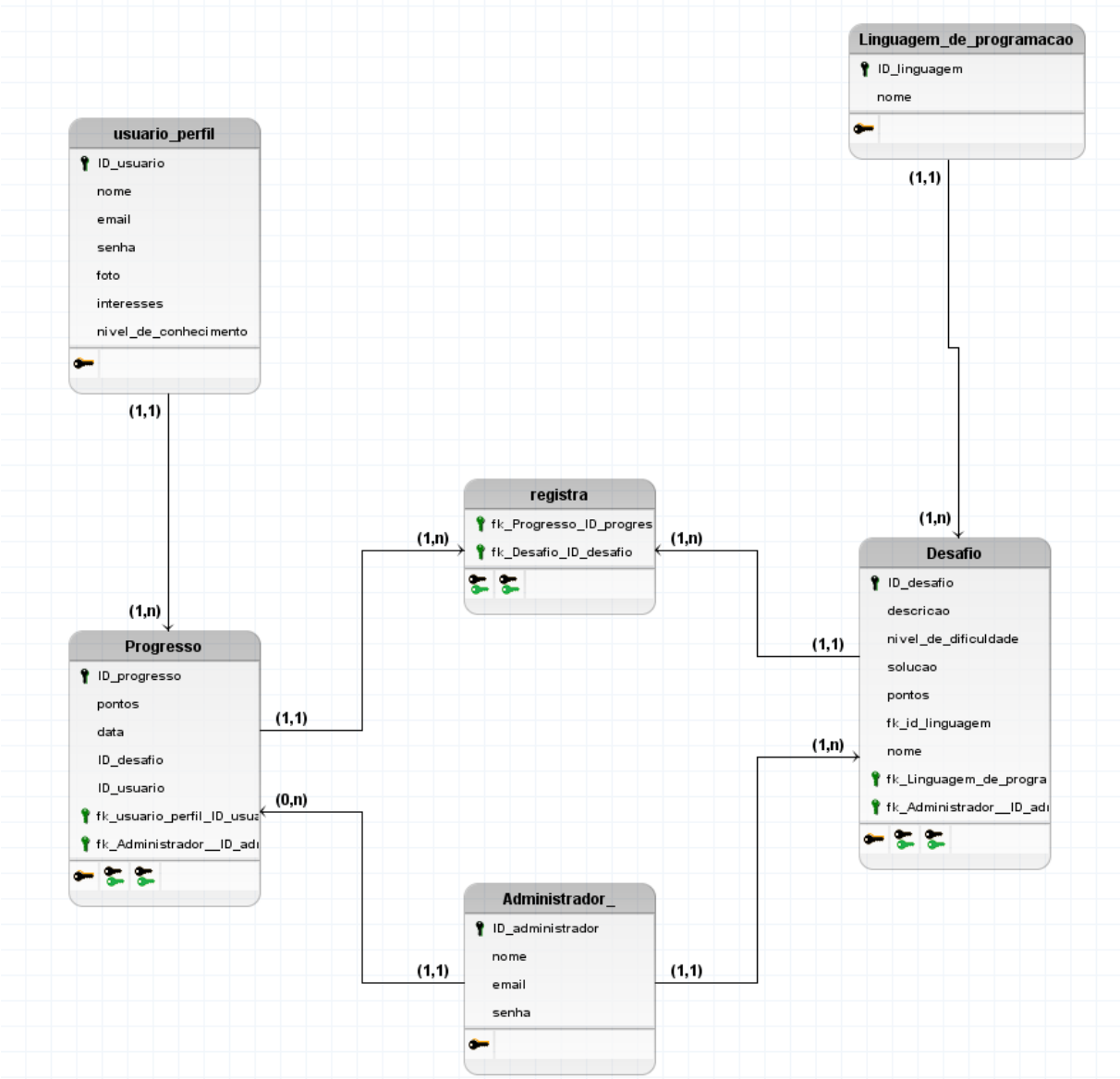
usuário os dados do desafio serão registrados na entidade **Progresso**, um desafio deverá ser registrado pelo menos uma vez e pode ser registrado inúmeras vezes, dependendo de quantos usuários o realizarem ou um mesmo usuário realiza-lo mais de uma vez, sendo representado pela cardinalidade (1, n).

A entidade **Administrador** é reservada para os administradores do banco de dados, eles têm a capacidade de criar desafios para a plataforma, tendo que criar no mínimo um e no máximo vários, conforme mostra a cardinalidade (1, n), e também, o administrador pode fazer atualizações nos registros de progressos, caso haja algum erro ou algo do gênero, podendo não realizar atualização alguma ou realizar várias (0, n).

3.7 Modelagem Lógica do Banco de Dados

A modelagem lógica do BD do sistema desenvolvido está apresentada na Figura 13.

Figura 13: Diagrama Lógico de Banco de Dados



Fonte: autoria própria

3.8 Tecnologias Utilizadas

Para o projeto do sistema SGBP, foram utilizadas as seguintes tecnologias:

- BrModelo: Ferramenta de modelagem de banco de dados, utilizada para desenvolver a modelagem conceitual, lógica e física;
- Astah UML: Linguagem gráfica utilizada para os diagramas de caso de uso;

Para o desenvolvimento do sistema foram utilizados as seguintes tecnologias:

- Visual Studio Code: Editor de Código fonte, utilizado para o desenvolvimento dos códigos do sistema;
- Xampp: Um pacote com os principais servidores de código aberto do mercado, utilizado para usar o sistema de forma local;
- PHP: uma linguagem de script open source de uso geral, muito utilizada, e especialmente adequada para o desenvolvimento web e que pode ser embutida dentro do HTML;
- JavaScript: Uma linguagem de programação de alto nível criada, a princípio, para ser executada em navegadores e manipular comportamentos de páginas web;
- CSS: Linguagem de estilo em cascata, utilizada para a estilização do sistema;
- HTML: Linguagem de marcação, utilizada para o desenvolvimento da marcação do layout do sistema.

4 DESCRIÇÃO DO SISTEMA

4.1 Cabeçalho da Plataforma

O cabeçalho (Figura 14) exibe uma barra de navegação horizontal com cinco opções principais. Essa barra permite que os usuários naveguem facilmente entre as diferentes seções, sendo elas: Home, Desafios, Suporte, Login e Cadastro promovendo uma experiência de uso intuitiva.

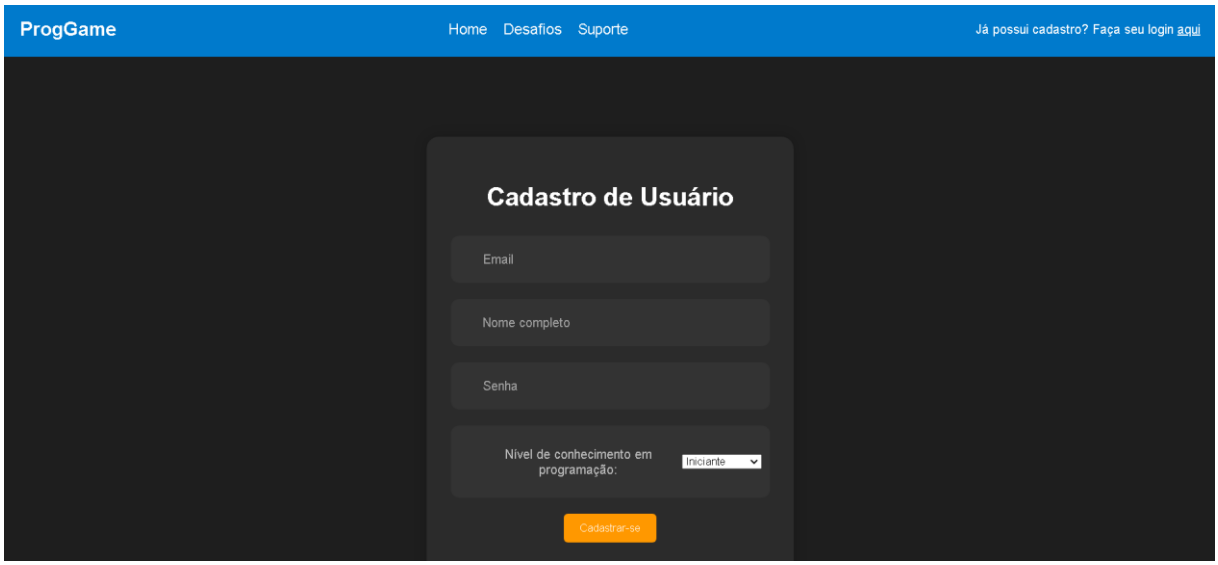
Figura 14: Cabeçalho da plataforma Proggame

Fonte: autoria própria

4.2 Tela de Cadastro

A tela da Figura 15 permite que novos usuários se cadastrem na plataforma. Os campos de entrada incluem informações básicas, como nome, email e senha. Após preencher o formulário, o usuário pode criar uma conta para acessar a plataforma e suas funcionalidades.

Figura 15: Tela de cadastro

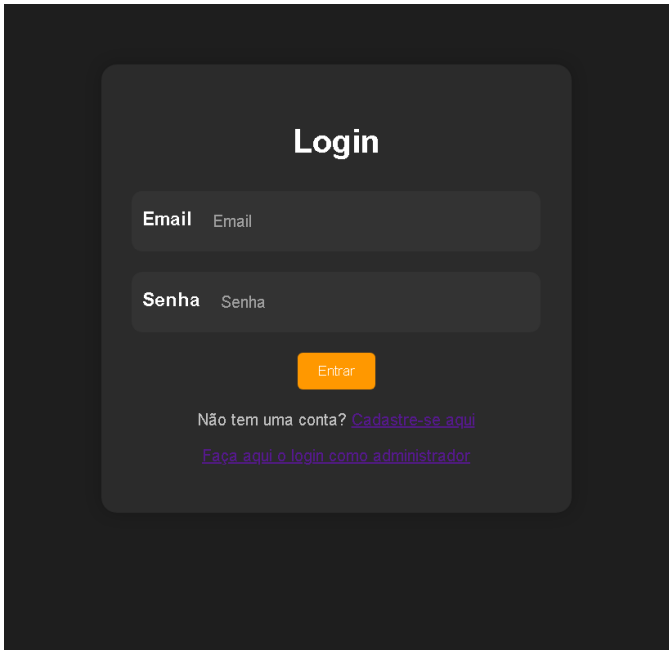


Fonte: autoria própria

4.3 Tela de Login

A tela de login (Figura 16) permite que os usuários já cadastrados acessem suas contas. Nela, os usuários devem inserir seu email e senha. Caso os dados estejam corretos, o sistema concede acesso à conta do usuário e redireciona para a página de perfil do usuário. Caso alguma informação esteja incorreta, o usuário deverá refazer o cadastro.

Figura 16: Tela de login

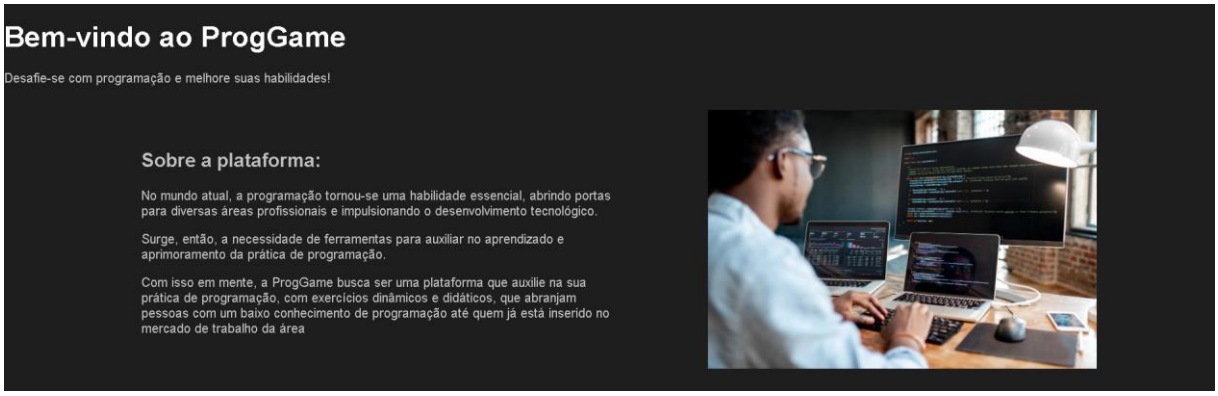


Fonte: autoria própria

4.4 Tela inicial

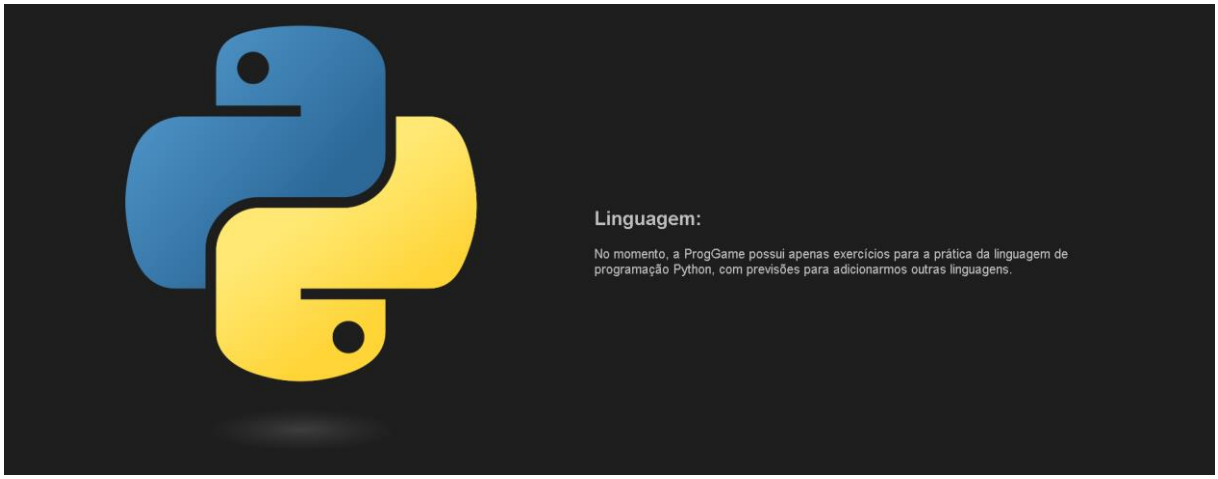
A tela inicial exibe uma visão geral da plataforma Proggame (Figuras 17 e 18). Nela, os usuários encontram atalhos para as seções de desafios, suporte e perfil, permitindo que comecem a explorar a plataforma de maneira eficiente e intuitiva.

Figura 17: Tela inicial da plataforma



Fonte: autoria própria

Figura 18: Tela inicial da plataforma 2



Fonte: autoria própria

4.5 Tela de Desafio sem login

A tela da Figura 19 apresenta uma mensagem informando que o login é necessário para acessar os desafios e participar da plataforma.

Figura 19: Tela de desafios sem o login realizado

Fonte: autoria própria

4.6 Tela de Suporte

A tela de suporte (Figura 20) permite que os usuários obtenham ajuda ou enviem dúvidas e solicitações. Nela, o usuário encontra uma opção de contato, por email, onde pode descrever seu problema ou pergunta, facilitando o suporte e atendimento.

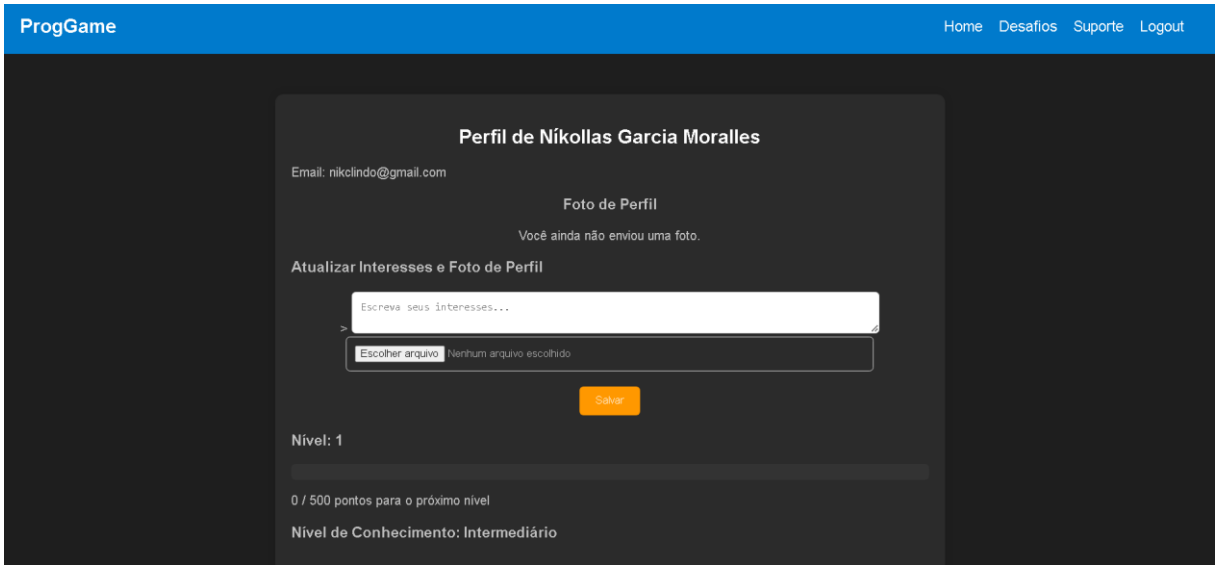
Figura 20: Tela de suporte

Fonte: autoria própria

4.7 Tela de Perfil

A tela de perfil (Figura 21) exibe as informações do usuário, como foto, interesses e barra de progresso de XP. O usuário pode atualizar seus interesses e acompanhar seu nível de conhecimento conforme completa desafios, criando uma experiência personalizada.

Figura 21: Tela de Perfil



Fonte: autoria própria

4.8 Tela de Desafios- Login realizado

Com o login realizado, o usuário tem acesso aos desafios da plataforma (Figura 22). A tela exibe uma lista de desafios disponíveis, classificados por dificuldade, permitindo uma escolha personalizada para a prática.

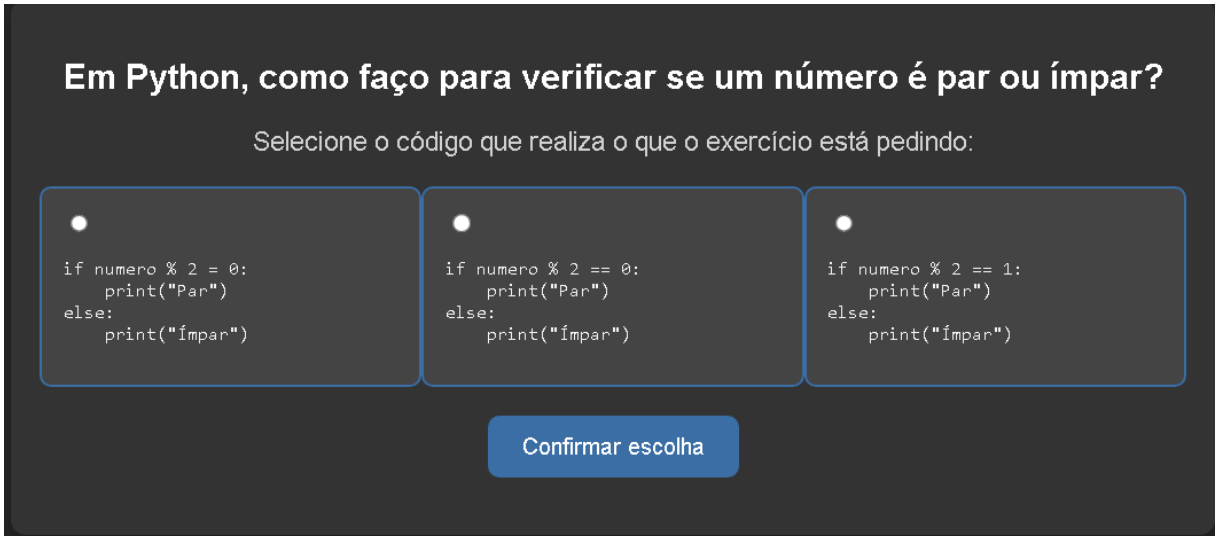
Figura 22: Tela de desafios com o login realizado

Fonte: autoria própria

4.9 Exemplos de desafios

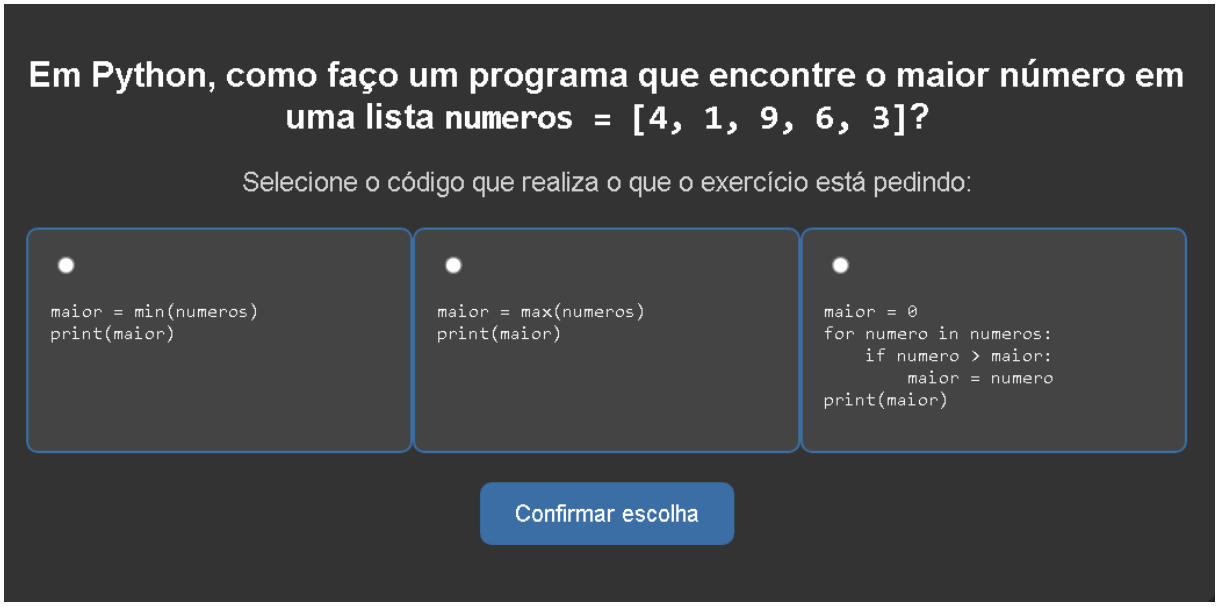
Ao selecionar algum dos desafios você será direcionado para outra tela onde terá o desafio para realizar, que pode ser em um dos 3 moldes propostos como demonstram: Figura 23, Figura 26, Figura 27.

Figura 23: Tela de desafio Par ou Ímpar



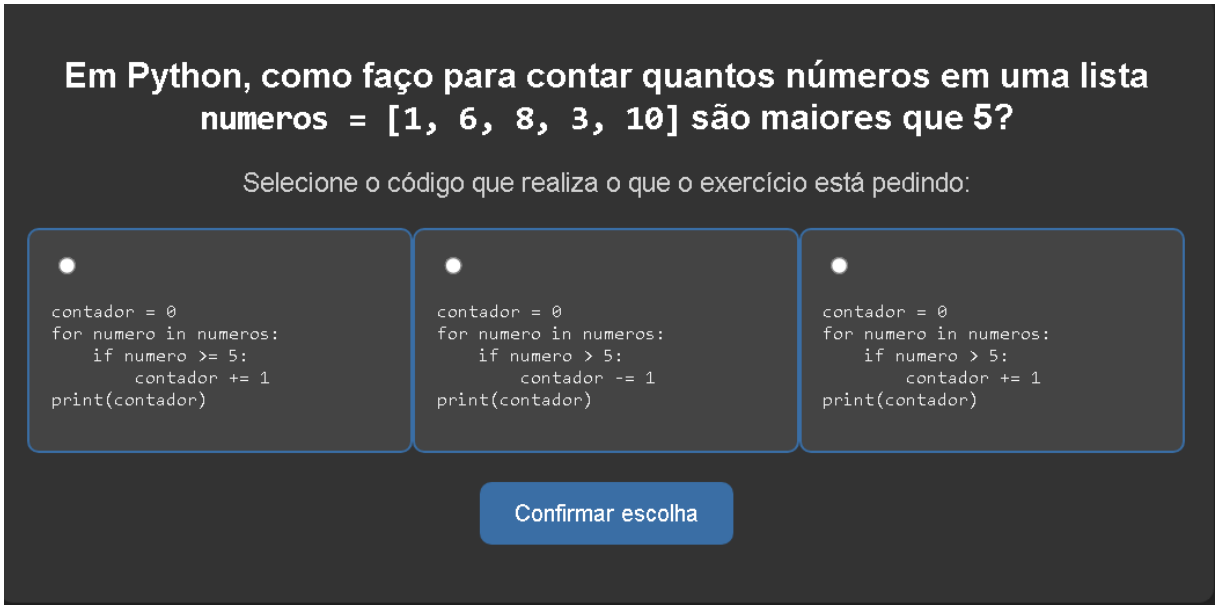
Fonte: autoria própria

Figura 24: Tela de desafio Maior número



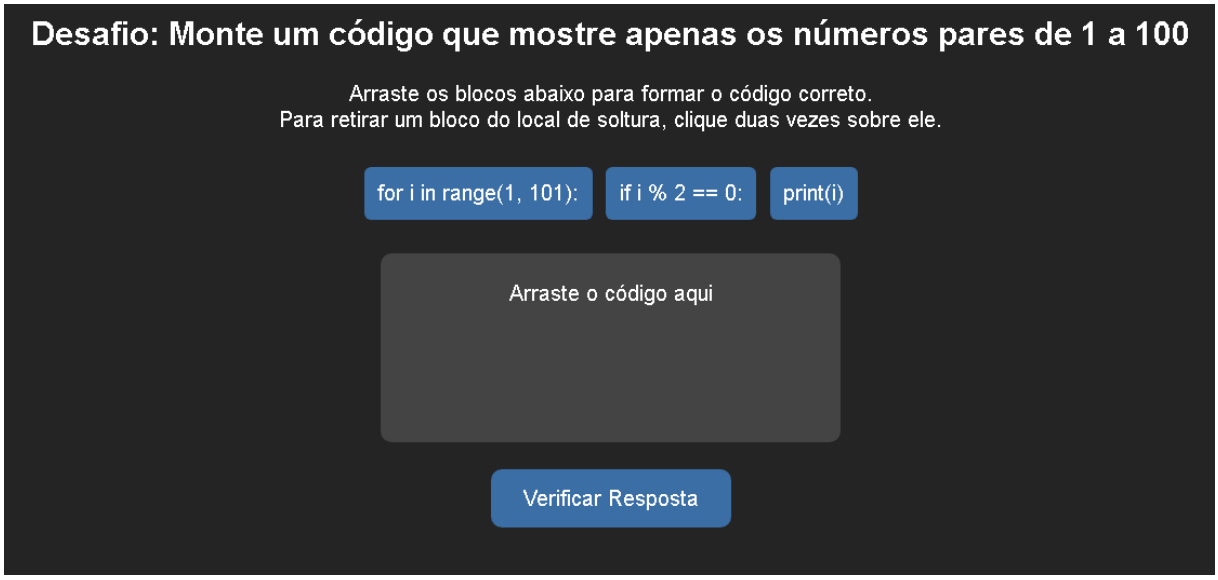
Fonte: autoria própria

Figura 25: Tela de desafio Números maiores que 5



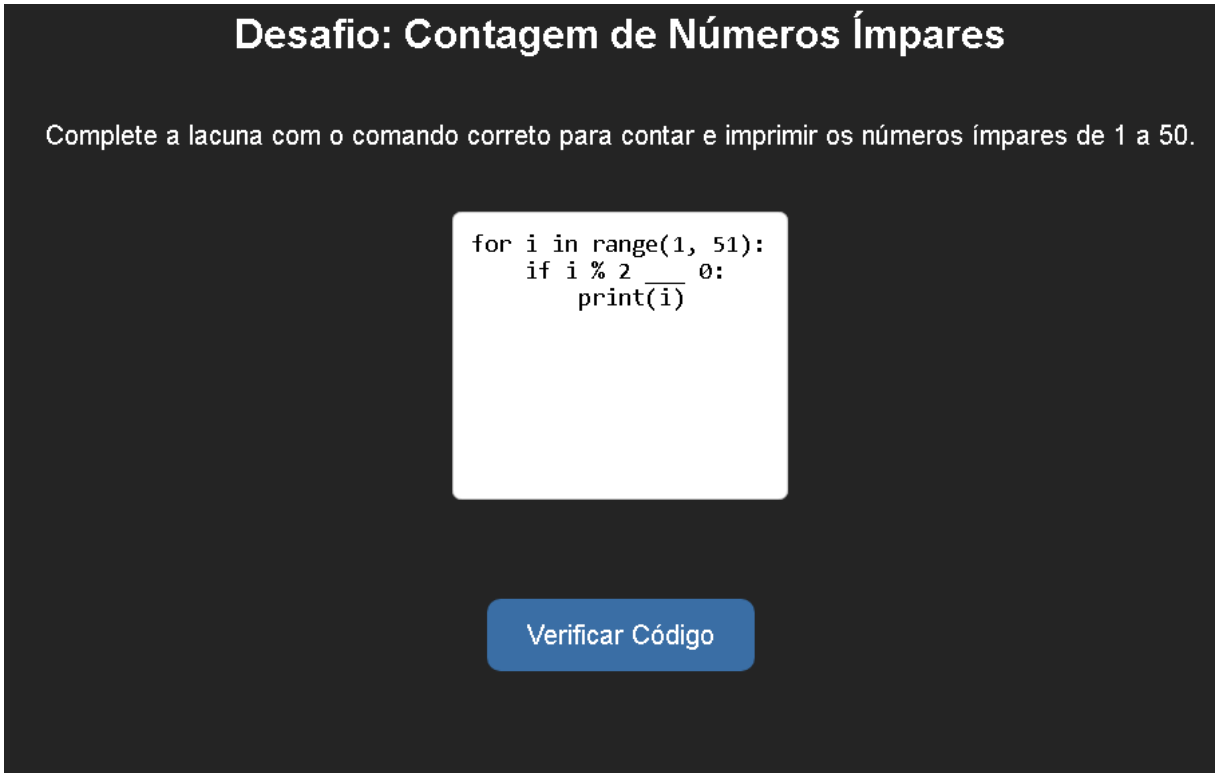
Fonte: autoria própria

Figura 26: Tela de desafio Números Pares



Fonte: autoria própria

Figura 27: Tela de desafio Contagem de Números Ímpares

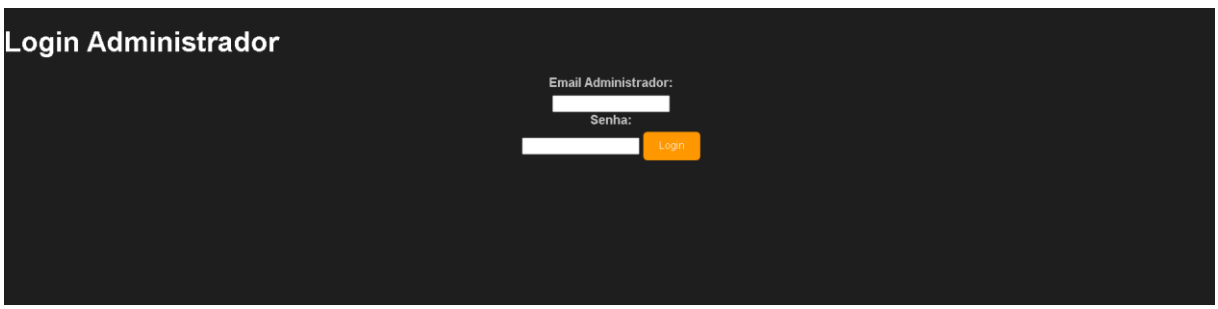


Fonte: autoria própria

4.10 Tela de Login do Administrador

Essa tela é dedicada ao acesso de administradores (Figura 28). Nela, o administrador deve inserir seu email e senha para acessar o painel de controle do sistema, onde terá acesso a funcionalidades de gestão de usuários, desafios e outras áreas administrativas.

Figura 28: Tela de login do administrador

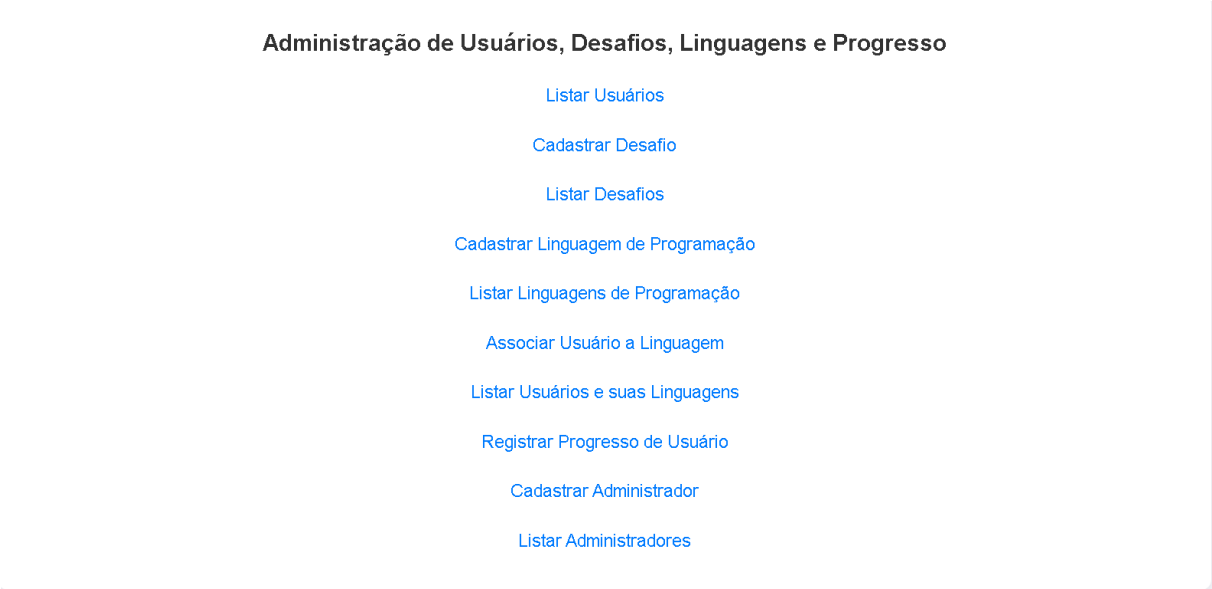


Fonte: autoria própria

4.11 Tela de Dashboard do Administrador

Essa tela (Figura 29) apresenta a visão geral do sistema para o administrador. Nela, o administrador pode acessar rapidamente funções essenciais como gestão de usuários, desafios e configurações.

Figura 29: Tela de dashboard para o Administrador



Fonte: autoria própria

4.12 Tela de Listagem de Usuários

A tela da Figura 30 permite que o administrador visualize todos os usuários cadastrados no sistema. Cada usuário está listado com informações básicas e opções para edição ou exclusão, garantindo uma administração eficiente e segura dos dados dos participantes.

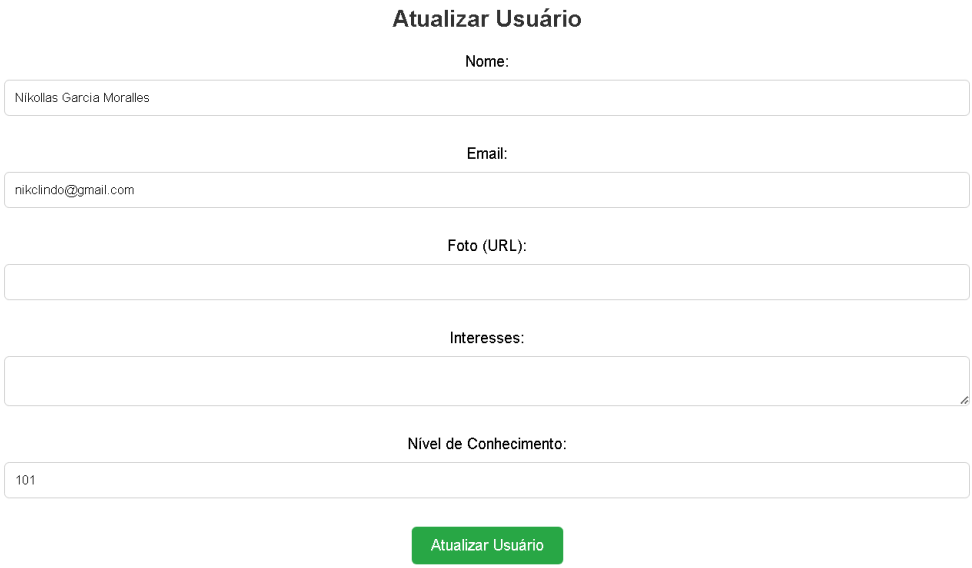
Figura 30: Tela de listagem de usuário

Fonte: autoria própria

4.13 Tela de Edição de Usuário

Nesta tela (Figura 31), o administrador pode alterar informações específicas dos usuários cadastrados. Os campos editáveis incluem nome, email, foto, interesses e nível de conhecimento, permitindo a atualização e correção de dados quando necessário.

Figura 31: Tela de edição do usuário

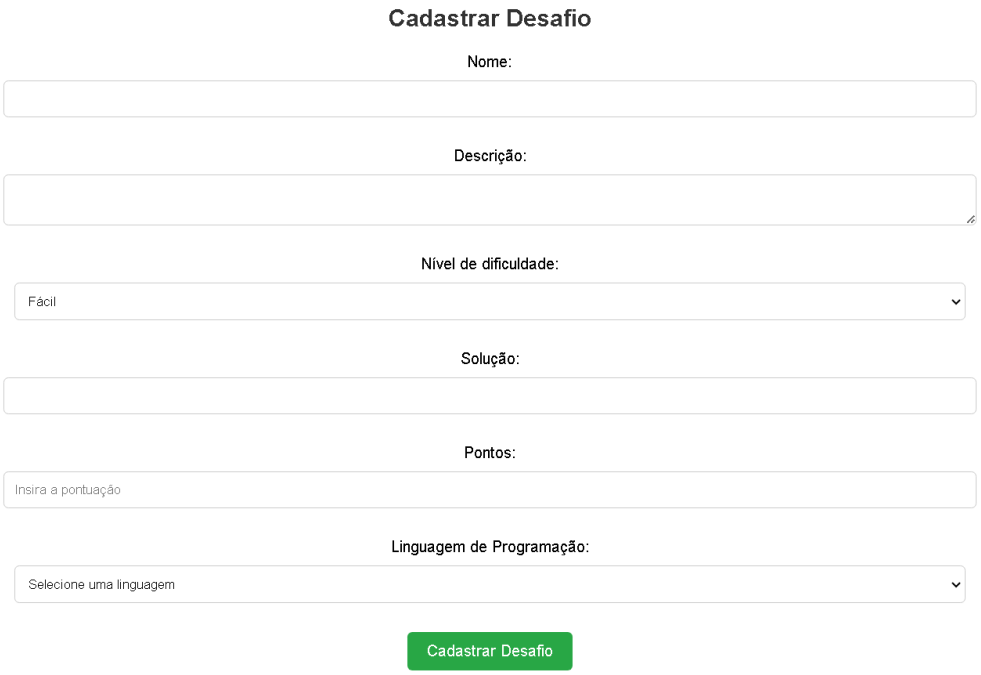


Fonte: autoria própria

4.14 Tela de Cadastro de Desafio

Essa tela permite ao administrador criar novos desafios para a plataforma. O administrador pode inserir informações como o nome do desafio, descrição, nível de dificuldade, a solução, a quantidade de pontos que o desafio dará e a linguagem de programação associada. Ao finalizar o cadastro, o novo desafio fica disponível para os usuários da plataforma.

Figura 32: Tela de cadastro de desafio



Fonte: autoria própria

4.15 Tela de Listagem de Desafios

A tela de listagem de desafios exibe todos os desafios cadastrados, organizados em uma tabela com colunas para nome, dificuldade e linguagem de programação. O administrador pode selecionar um desafio para edição ou exclusão, facilitando a gestão e manutenção dos conteúdos de prática.

Figura 33: Tela de listagem de desafio

Lista de Desafios

ID	Nome	Descrição	Nível	Linguagem	Ações
9	Par ou Ímpar	Em Python, como faço para verificar se um número é par ou ímpar?	1	4	Editar | Excluir
10	Maior número	Em Python, como faço um programa que encontre o maior número em uma lista numeros = [4, 1, 9, 6, 3]?	1	4	Editar | Excluir
11	Números maiores que 5	Em Python, como faço para contar quantos números em uma lista numeros = [1, 6, 8, 3, 10] são maiores que 5?	1	4	Editar | Excluir
12	Números Pares	Monte um código que mostre apenas os números pares de 1 a 100	2	4	Editar | Excluir
13	Soma de Números	Monte um código que some os números em uma lista	2	4	Editar | Excluir
14	Contagem de Vogais	Monte um código que conte a quantidade de vogais em uma string	2	4	Editar | Excluir
15	Contar Números Ímpares	Complete a lacuna com o comando correto para contar e imprimir os números ímpares de 1 a 50.	1	4	Editar | Excluir
16	Calcular Média de Notas	Complete a lacuna com o comando correto para calcular a média de uma lista de notas.	1	4	Editar | Excluir
17	Classificar Números	Complete a lacuna com o comando correto para classificar uma lista de números em ordem crescente.	1	4	Editar | Excluir

Fonte: autoria própria

4.16 Tela de Edição de Desafio

Essa tela permite ao administrador modificar informações dos desafios previamente cadastrados. Ao acessar essa tela, o administrador pode atualizar o nome, descrição, dificuldade e linguagem de programação do desafio selecionado. A atualização é salva e refletida imediatamente para os usuários.

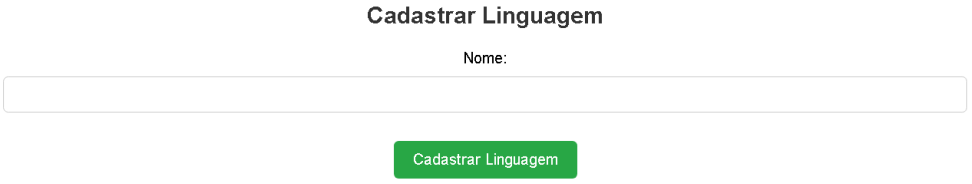
Figura 34: Tela de edição de desafio

Fonte: autoria própria

4.17 Tela de Cadastro de Linguagem de Programação

Nessa tela, o administrador pode cadastrar novas linguagens de programação na plataforma. Cada nova linguagem cadastrada fica disponível para associação com desafios e para seleção pelos usuários na escolha de práticas.

Figura 35: Tela de cadastro de linguagem



Fonte: autoria própria

4.18 Tela de Listagem de Linguagens de Programação

A tela de listagem de linguagens exibe todas as linguagens de programação disponíveis na plataforma. O administrador pode visualizar, editar ou excluir linguagens de programação conforme necessário, permitindo uma administração flexível das opções de prática para os usuários.

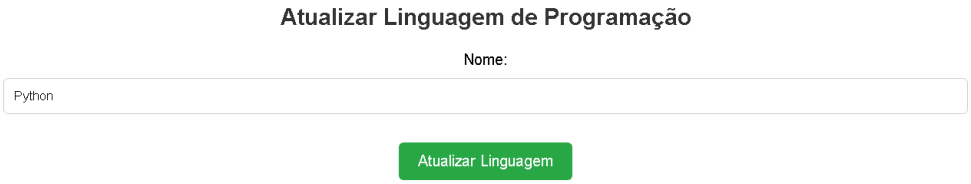
Figura 36: Tela de listagem de linguagens

Fonte: autoria própria

4.19 Tela de Edição de Linguagem de Programação

Essa tela permite que o administrador atualize o nome das linguagens de programação cadastradas. Qualquer alteração feita é refletida automaticamente nos desafios que utilizam essa linguagem, mantendo a consistência dos dados na plataforma.

Figura 37: Tela de edição de linguagem

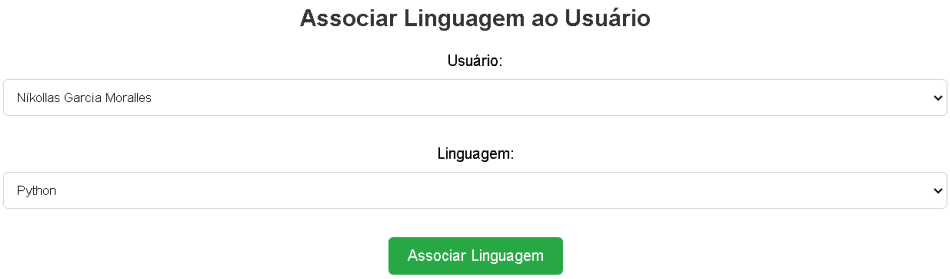


Fonte: autoria própria

4.20 Tela de Associação de Linguagem com Usuário

Nesta tela, o administrador pode associar uma linguagem de programação específica a um usuário. A associação permite personalizar as opções de prática, garantindo que o usuário tenha acesso apenas aos desafios da linguagem selecionada.

Figura 38: Tela de associação linguagem-usuário

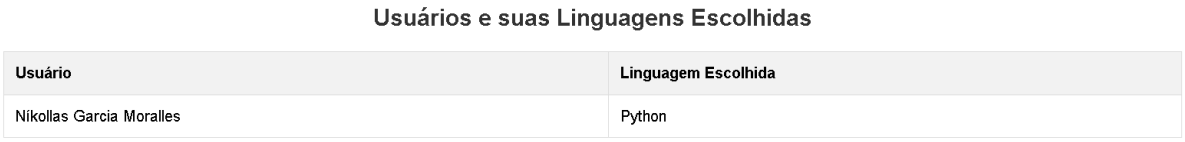


Fonte: autoria própria

4.21 Tela de Listagem de Usuários e Linguagens Escolhidas

A tela exibe uma tabela com os usuários e suas linguagens de programação associadas. Esse recurso permite que o administrador visualize as preferências dos usuários, facilitando a análise de dados e o monitoramento das práticas.

Figura 39: Tela de listagem de usuário e linguagem escolhida



Fonte: autoria própria

4.22 Tela de Registro de Progresso

Essa tela permite ao administrador registrar o progresso de um usuário caso necessário. Selecionando o usuário, qual desafio, quantidade de pontos e a data em que o desafio foi realizado.

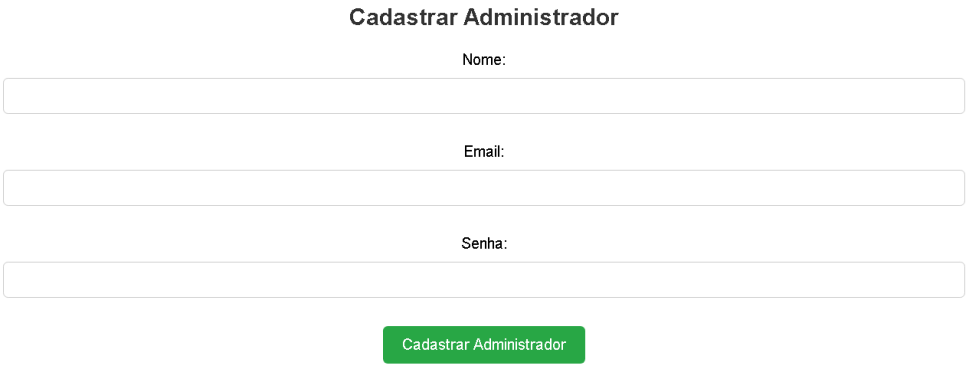
Figura 40: Tela de registro de progresso

Fonte: autoria própria

4.23 Tela de Cadastro de Administrador

Nessa tela, um administrador pode cadastrar novos administradores no sistema. Os campos incluem nome, email e senha, que permitem ao novo administrador acessar o painel de controle da plataforma para realizar tarefas administrativas.

Figura 41: Tela de cadastro de administrador



Fonte: autoria própria

4.24 Tela de Listagem de Administradores

A tela exibe todos os administradores cadastrados, organizados em uma lista com opções para editar ou excluir administradores. Essa funcionalidade facilita o

gerenciamento da equipe administrativa e garante o controle de acesso à plataforma.

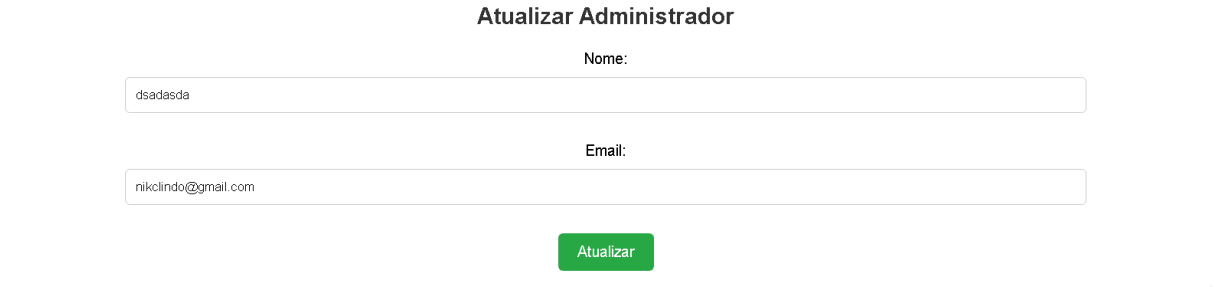
Figura 42: Tela de listagem de administrador

Fonte: autoria própria

4.25 Tela de Edição de Administrador

Essa tela permite que o administrador altere as informações dos administradores cadastrados, como nome e email.

Figura 43: Tela de edição do administrador



Fonte: autoria própria

5 CONSIDERAÇÕES FINAIS

No começo do terceiro semestre, foi dado início ao planejamento da plataforma Proggame, que começou a ser desenvolvida fisicamente a partir do quarto semestre. O sistema cumpre os requisitos propostos neste documento, com a exceção de alguns requisitos:

1) Selecionar Linguagem de Programação: Permite que os usuários selecionem a linguagem de programação desejada para praticar.

2) Selecionar Dificuldade: Permite que os usuários selecionem o nível de dificuldade dos desafios que desejam completar: Iniciante, Intermediário ou Avançado.

Estas funcionalidades não foram aplicadas por conta do sistema no momento, possui apenas foco na linguagem de programação Python e não possui diferenciação entre níveis de dificuldade propriamente desenvolvida, ficando para futuras atualizações da plataforma.

Concluímos este documento com a experiência de criar um sistema como projeto final de curso técnico, expressando satisfação pelo conhecimento obtido no ambiente acadêmico, além de um desenvolvimento pessoal e profissional.

6 REFERÊNCIAS

DECI, Edward L.; RYAN, Richard M. Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. Contemporary Educational Psychology, v. 25, n. 1, p. 54–67, 2000.

DETERDING, Sebastian; DIXON, Dan; KHALED, Rilla; NACKE, Lennart. From game design elements to gamefulness: defining "gamification". In: PROCEEDINGS OF THE 15th INTERNATIONAL ACADEMIC MINDTREK CONFERENCE: Envisioning Future Media Environments, 2011. p. 9-15.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 6. ed. São Paulo: Pearson Addison-Wesley, 2011.

HAMARI, Juho; KOIVISTO, Jonna; SARSA, Harri. Does gamification work?--A literature review of empirical studies on gamification. In: 2014 47th HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES. IEEE, 2014. p. 3025-3034.

HEUSER, Carlos Alberto. Projeto de Banco de Dados. 6. ed. Porto Alegre: Bookman, 2009.

KAPP, Karl M. The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education. San Francisco: Pfeiffer, 2012.

OLIVEIRA, Kelma V. N. de. Análise e Projeto de Sistemas de Informação Orientados a Objetos com UML. 1. ed. São Paulo: Elsevier, 2009.

PEREIRA, Reinaldo. UML 2.0 - Teoria e Prática. 1. ed. Rio de Janeiro: Brasport, 2006.

PRESSMAN, Roger S. Engenharia de Software: uma abordagem profissional. 7. ed. Porto Alegre: AMGH, 2011.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 2011.

VIEIRA, Francisco Giovanni David. Ensino de Marketing por meio de entrevista semi-estruturada. Revista Espaço Acadêmico, v. 17, n. 195, p. 01-08, 2017.

WERBACH, Kevin; HUNTER, Dan. For the Win: How Game Thinking Can Revolutionize Your Business. Wharton Digital Press, 2012.

APÊNDICE I - INSTRUÇÕES SQL PARA CRIAÇÃO DA BASE DE DADOS

```
--
--
Banco de dados: `proggame`

-- Estrutura da tabela `administrador`
CREATE TABLE `administrador` (
  `ID_administrador` int(11) NOT NULL,
  `nome` varchar(100) DEFAULT NULL,
  `email` varchar(100) DEFAULT NULL,
  `senha` varchar(255) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Estrutura da tabela `usuario_perfil`
CREATE TABLE `usuario_perfil` (
  `ID_usuario` int(11) NOT NULL,
  `nome` varchar(100) NOT NULL,
  `email` varchar(100) NOT NULL,
  `senha` varchar(255) NOT NULL,
  `foto` varchar(255) DEFAULT NULL,
  `interesses` text,
  `nivel_de_conhecimento` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Estrutura da tabela `linguagem_de_programacao`
CREATE TABLE `linguagem_de_programacao` (
  `ID_linguagem` int(11) NOT NULL,
  `nome` varchar(100) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

```
-- Estrutura da tabela `desafio`
CREATE TABLE `desafio` (
  `ID_desafio` int(11) NOT NULL,
  `nome` varchar(100) DEFAULT NULL,
  `descricao` text,
  `nivel_de_dificuldade` int(11) DEFAULT NULL,
  `solucao` text,
  `pontos` int(11) DEFAULT NULL,
  `fk_id_linguagem` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Estrutura da tabela `progresso`
CREATE TABLE `progresso` (
  `ID_progresso` int(11) NOT NULL,
  `ID_usuario` int(11) NOT NULL,
  `ID_desafio` int(11) NOT NULL,
  `pontos` int(11) DEFAULT NULL,
  `data` datetime DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;

-- Índices e restrições para a tabela `administrador`
ALTER TABLE `administrador`
  ADD PRIMARY KEY (`ID_administrador`);

-- Índices e restrições para a tabela `usuario_perfil`
ALTER TABLE `usuario_perfil`
  ADD PRIMARY KEY (`ID_usuario`);

-- Índices e restrições para a tabela `linguagem_de_programacao`
ALTER TABLE `linguagem_de_programacao`
  ADD PRIMARY KEY (`ID_linguagem`);
```

```
-- Índices e restrições para a tabela `desafio`
ALTER TABLE `desafio`
  ADD PRIMARY KEY (`ID_desafio`),
  ADD KEY `fk_id_linguagem` (`fk_id_linguagem`);

-- Índices e restrições para a tabela `progresso`
ALTER TABLE `progresso`
  ADD PRIMARY KEY (`ID_progresso`),
  ADD KEY `ID_usuario` (`ID_usuario`),
  ADD KEY `ID_desafio` (`ID_desafio`);

-- Ligando `fk_id_linguagem` da tabela `desafio` à tabela `linguagem_de_programacao`
ALTER TABLE `desafio`
  ADD CONSTRAINT `fk_id_linguagem` FOREIGN KEY (`fk_id_linguagem`) REFERENCES `linguagem_de_programacao` (`ID_linguagem`);

-- Ligando `ID_usuario` e `ID_desafio` da tabela `progresso` às tabelas correspondentes
ALTER TABLE `progresso`
  ADD CONSTRAINT `fk_progresso_usuario` FOREIGN KEY (`ID_usuario`) REFERENCES `usuario_perfil` (`ID_usuario`),
  ADD CONSTRAINT `fk_progresso_desafio` FOREIGN KEY (`ID_desafio`) REFERENCES `desafio` (`ID_desafio`);
```