INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO
RIO GRANDE DO SUL
CAMPUS CANOAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

PEDRO HENRIQUE KIELING DAGOSTINI

Protáthlima: Uma aplicação web para criação de campeonatos amadores do jogo virtual League Of Legends.

Canoas, 13 de dezembro de 2024.

Pedro Henrique Kieling Dagostini

Protáthlima: Uma aplicação web para criação de campeonatos amadores do jogo virtual League Of Legends.

Trabalho de Conclusão de Curso apresentado como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul – Campus Canoas.

Prof. Dr. Dieison Soares Silveira

Orientador

Canoas, 13 de dezembro de 2024.

Ministério da Educação
Secretaria de Educação Profissional, Científica e Tecnológica
Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul
Campus Canoas

ATA DE DEFESA PÚBLICA DO TRABALHO DE CONCLUSÃO DE CURSO

Aos 09 dias do mês de dezembro de 2024, às 16 horas, em sessão pública na sala E10 do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul, Campus Canoas, na presença da Banca Examinadora presidida pelo Professor:

Dr. Dieison Soares Silveira

e composta pelos examinadores:

1. **Ma. Denise Regina Pechmann**
2. **Me. Igor Lorenzato Almeida,**

o aluno **Pedro Henrique Kieling Dagostini** apresentou o Trabalho de Conclusão de Curso intitulado: **Protáthlima: Uma aplicação web para criação de campeonatos amadores do jogo virtual League Of Legends,** como requisito curricular indispensável para a integralização do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Após reunião em sessão reservada, a Banca Examinadora deliberou e decidiu pela APROVAÇÃO do referido trabalho, divulgando o resultado formalmente ao aluno e demais presentes e eu, na qualidade de Presidente da Banca, lavrei a presente ata que será assinada por mim, pelos demais examinadores e pelo aluno.

Presidente da Banca Examinadora

Examinador 01

Examinador 02

Aluno

RESUMO

Este trabalho apresenta o desenvolvimento e a documentação da aplicação web Protáthlima (πρωτάθλημα) que significa campeonato em grego, destinada à criação e organização de campeonatos amadores do jogo League of Legends. A plataforma foi projetada com funcionalidades como a integração com a API da Riot Games, que possibilita a validação automática de perfis e o balanceamento de equipes baseado em rankings. Além disso, inclui um sistema de gamificação que incentiva a participação dos jogadores e promove o engajamento na comunidade gamer. O desenvolvimento envolveu o uso de tecnologias como PHP, MySQL/MariaDB, HTML, CSS e JavaScript moderno para a implementação do front-end e back-end, além de metodologias estruturadas, como modelagem com diagramas UML e levantamento de requisitos por meio de entrevistas com usuários. O trabalho também aborda os desafios enfrentados, como a manipulação de dados em tempo real e a criação de uma interface responsiva e intuitiva. Os resultados obtidos destacam o potencial da aplicação em atender às demandas da comunidade de jogadores, proporcionando uma solução eficiente e escalável para organização de torneios. Como trabalhos futuros, estão a expansão para outros jogos, melhorias na escalabilidade e a integração com redes sociais para ampliação da visibilidade dos eventos.

ABSTRACT

This study presents the development and documentation of the Protáthlima web application, designed for creating and organizing amateur tournaments for the game League of Legends. The platform incorporates features such as integration with the Riot Games API, enabling automatic profile validation and team balancing based on player rankings. Additionally, it includes a gamification system that encourages player participation and fosters engagement within the gaming community. The development process utilized technologies such as PHP, MySQL/MariaDB, HTML, CSS, and modern JavaScript for front-end and back-end implementation, along with structured methodologies such as UML diagrams and user requirement gathering through interviews. The study also addresses challenges faced, including real-time data handling and the creation of a responsive and intuitive interface. The results highlight the application's potential to meet the demands of the gaming community by providing an efficient and scalable solution for tournament management. Future recommendations include expanding support to other games, improving scalability, and integrating social media features to enhance event visibility.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
ELO	*Indicador de habilidade de jogadores em sistemas de ranqueamento*
ES6	*ECMAScript 6*
HTML	*Hypertext Markup Language*
JSON	*JavaScript Object Notation*
LoL	*League of Legends*
PHP	*Hypertext Preprocessor*
RN	*Regras de negócio*
RF	*Requisitos Funcionais*
RNF	*Requisitos Não Funcionais*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*

SUMÁRIO

1. INTRODUÇÃO

Segundo Giovana Bratti (Open Leaders, 2022) “...a competição na raça humana já existia na pré-história, quando os homens das cavernas disputavam alimentos para garantir a sobrevivência…”. Hoje, a competição está presente em diversas áreas da vida moderna, como status, fama e bens materiais, e nos jogos digitais não é diferente. Nos últimos anos, os jogos eletrônicos evoluíram de simples formas de entretenimento para um dos maiores cenários competitivos globais, os esportes eletrônicos (eSports). Esse fenômeno criou oportunidades tanto para jogadores quanto para empresas e organizações.

No contexto dos eSports, o League of Legends (LoL) é um dos títulos mais proeminentes e influentes. Ele é um jogo de estratégia em tempo real e arena de batalha multijogador online, onde duas equipes de cinco jogadores competem para destruir a base inimiga. Cada jogador controla um personagem, conhecido como campeão, com habilidades únicas que evoluem durante o jogo, e deve trabalhar em equipe para obter vantagem estratégica, derrubar torres, derrotar oponentes e conquistar objetivos no mapa. O jogo requer comunicação, coordenação e conhecimento profundo das mecânicas para alcançar a vitória.

O jogo possui uma comunidade global de milhões de jogadores (activeplayer.io 2022), gerando um cenário competitivo robusto, alimentado por torneios amadores e profissionais. Grandes marcas, como MasterCard, Cisco, KIA e Mercedes-Benz, já patrocinaram campeonatos oficiais de LoL, evidenciando a importância e o impacto do jogo no mercado. Entretanto, muitos jogadores que desejam aprimorar suas habilidades e se aproximar do ambiente competitivo encontram dificuldade em se desenvolver devido à falta de uma plataforma de fácil acesso para jogadores de League of Legends participarem de campeonatos amadores, pois se vê necessário uma conta válida no jogo para realizar o cadastro na plataforma.

Para atender a essa demanda, foi desenvolvido o "Protáthlima", uma aplicação web que facilitará a criação de campeonatos amadores de League of Legends. A plataforma permite que os usuários organizem suas próprias competições, formando equipes e campeonatos.

1.1 MOTIVAÇÃO

O cenário competitivo dos eSports tem se expandido rapidamente, atraindo milhões de jogadores e espectadores em todo o mundo. Segundo uma pesquisa realizada pela Newzoo[1] e publicada na reportagem do Matheus Dal Prá (Ge Globo 2021) a final do mundial de League of Legends (LoL) de 2019 alcançou cerca de 99,6 milhões de telespectadores, com 21,2 milhões apenas no Brasil, que ocupa a terceira posição no ranking global. Este aumento no interesse não apenas mostra a popularidade do jogo, mas também destaca o potencial crescimento de uma nova geração de jogadores.

Entretanto, muitos desses jogadores amadores enfrentam desafios para se inserir no ambiente competitivo. A falta de plataformas que facilitem a formação de equipes e a organização de campeonatos torna difícil para esses jogadores aprimorar suas habilidades e conquistar experiências valiosas. Para atender a essa demanda, o "Protáthlima" mostra-se como uma solução inovadora.

A aplicação permite que os usuários busquem jogadores por meio da API da RIOT, obtendo informações dos jogadores como campeões mais jogados, taxas de vitórias e derrotas. Com isso, os usuários podem formar equipes de maneira mais eficaz, já que a criação de equipes é feita com base em um critério de ranking, garantindo que os membros tenham níveis semelhantes de habilidade. Os responsáveis pelas equipes têm autonomia para editá-las, promovendo uma dinâmica colaborativa.

Além disso, o "Protáthlima" oferece uma funcionalidade de busca de equipes onde o usuário poderá verificar todas as equipes cadastradas na aplicação. O responsável por cada equipe será destacado, permitindo que os usuários encontrem e se conectem com outras equipes por meio do chat do próprio jogo. Devido às limitações da API, a organização de campeonatos exigirá uma gestão manual dos resultados dos jogos por parte do responsável pelo campeonato, a progressão dos times nas chaves dos campeonatos deverá ser feita manualmente. Além disso, foi implementado um sistema de gamificação que recompensa os jogadores com medalhas quando suas equipes vencem, incentivando a participação e o engajamento.

Dessa forma, o "Protáthlima" não só facilita a entrada de jogadores no cenário competitivo, mas também promove um ambiente de competição saudável e divertido, onde todos podem aprimorar suas habilidades e interagir com outros entusiastas do LoL.

[1] Newzoo é uma empresa de dados, pesquisas e consultorias.

1.2. OBJETIVOS

1.2.1 OBJETIVO GERAL

Desenvolver uma aplicação web que permita a criação e gestão de campeonatos amadores de League of Legends, facilitando a formação de equipes e a descoberta de novos jogadores.

1.2.2 OBJETIVOS ESPECÍFICOS

- Implementar funcionalidades de cadastro e login de usuários, garantindo a segurança e a privacidade dos dados.
- Desenvolver um sistema de busca de jogadores utilizando a API da RIOT, permitindo acesso a informações relevantes, como campeões mais jogados e porcentagens de vitórias.
- Criar um módulo de formação e gestão de equipes, que valide a diferença de ranque entre os jogadores.
- Implementar uma funcionalidade de busca e visualização de equipes, facilitando a conexão entre jogadores.
- Desenvolver um sistema de criação e organização de campeonatos, permitindo a gestão manual das chaves e resultados.
- Implementar mecanismos de feedback.
- Realizar testes unitários e de integração para assegurar a funcionalidade e a usabilidade da aplicação

2. REFERENCIAL TEÓRICO

A aplicação Protáthlima tem como objetivo simplificar a criação e gestão de campeonatos amadores do jogo League of Legends (LoL), proporcionando aos usuários ferramentas para organização de equipes e torneios, além de explorar os dados de jogadores por meio da API da Riot Games. Durante o desenvolvimento, foram utilizadas diversas tecnologias, incluindo PHP, PHPUnit, MariaDB, JSON, HTML5, CSS3, JavaScript moderno e a API da Riot Games. Além disso, diagramas UML foram empregados para representar os casos de uso e a estrutura do banco de dados. Este capítulo apresenta as principais tecnologias e ferramentas empregadas no projeto.

2.1 UML

Segundo o Lucidchart[2] A UML (Unified Modeling Language) é uma linguagem gráfica utilizada para modelar e visualizar sistemas de software. Ela fornece uma maneira padronizada de representar os diferentes aspectos de um sistema, facilitando a comunicação entre desenvolvedores, analistas e outros envolvidos no processo de desenvolvimento. A UML inclui vários tipos de diagramas, sendo os mais comuns os diagramas de caso de uso e de classes.

No contexto deste projeto, foram utilizados diagramas UML para representar a estrutura do sistema e a organização do banco de dados, facilitando a compreensão das interações entre os componentes do sistema e a estrutura de armazenamento de dados. Esses diagramas auxiliam na documentação e no planejamento da implementação, garantindo uma visão clara da arquitetura da aplicação.

2.1 HTML E CSS

O HTML é responsável por estruturar o conteúdo das páginas web, utilizando elementos como <h1> para cabeçalhos, <p> para parágrafos e <img> para imagens. Ele define a semântica e organização do conteúdo, permitindo que navegadores interpretem e exibam as informações de forma funcional (Duckett, 2011). O CSS complementa o HTML, controlando a aparência e o layout das páginas. Com ele, é possível aplicar estilos como cores, fontes, espaçamentos e posicionamentos, além de criar layouts que funcionam em diferentes dispositivos. A interação entre HTML e CSS permite que páginas sejam visualmente atraentes e organizadas (Duckett, 2011).

[2] Disponivel em: <https://www.lucidchart.com/pages/pt/o-que-e-uml>

Como explicam Meyer e Weyl em CSS: The Definitive Guide: Web Layout and Presentation (2023), "HTML e CSS juntos criam a base de qualquer página web, combinando estrutura semântica com estilização sofisticada para melhorar a usabilidade e o design"

2.2 MariaDB

Segundo sua página oficial[3] indica, o MariaDB é um sistema de gerenciamento de bancos de dados de código aberto. Desenvolvido por Michael "Monty" Widenius, criador do MySQL, o MariaDB mantém compatibilidade total com seu predecessor, permitindo substituição sem alterações significativas nas aplicações existentes. Ele traz melhorias em desempenho, confiabilidade e novas funcionalidades.

Como destaca Forta (2011) no MariaDB Crash Course, "MariaDB é uma evolução do MySQL, construída com o objetivo de manter compatibilidade e introduzir novos recursos mais rapidamente, atendendo às necessidades de desenvolvedores que exigem mais controle e eficiência em suas aplicações".

2.4 PHP

PHP é uma linguagem de script de uso geral, de código aberto, projetada para desenvolvimento web e pode ser embutida em HTML. Seu código é executado no servidor, gerando páginas dinâmicas para o cliente enquanto mantém o código original oculto (PHP Manual, 2024). Além disso, o PHP é conhecido por sua simplicidade e suporte a diversos bancos de dados, como o MySQL/MariaDB, o que a torna uma escolha popular no desenvolvimento web. Devido a essa grande popularidade, fácil acesso a muitas informações sobre ele, não cogitei a escolhê-lo como a linguagem de programação do back-end desta aplicação.

2.5 JSON

JSON (JavaScript Object Notation) é um formato de dados leve, fácil de ler e escrever, utilizado para a troca de informações entre sistemas. Sua estrutura baseada em pares chave-valor torna a manipulação e a transmissão de dados entre servidores e clientes muito eficientes (JSON, 2024). Por ser amplamente adotado em APIs e outras tecnologias web, JSON facilita a integração de diferentes

[3] Disponível em: < https://mariadb.org/about/>

plataformas, principalmente em sistemas distribuídos como aplicações web, micro serviços e bancos de dados.

2.6 JAVASCRIPT MODERNO

O JavaScript moderno é baseado no ECMAScript 2015 e inclui melhorias significativas, como funções assíncronas e módulos, que facilitam o desenvolvimento de aplicações complexas (MDN Web Docs, 2024).

Essas melhorias no JavaScript permitem uma programação mais funcional, modular e limpa, facilitando a manutenção e a legibilidade do código. Como afirma Flanagan (2020), o JavaScript moderno melhorou consideravelmente a forma como lidamos com operações assíncronas e a organização do código, tornando-o mais poderoso e fácil de trabalhar, especialmente em grandes projetos.

2.7 API DA RIOT GAMES

Uma API é um conjunto de regras e definições que permite que diferentes softwares se comuniquem entre si. Através das APIs, diferentes sistemas podem compartilhar dados e funcionalidades sem que o usuário final precise entender como o sistema funciona internamente (Fielding, 2000). Nesse projeto é feito o uso da API da Riot Games para validação de conta e informações referentes aos jogadores e usuários da aplicação.

2.8 PHPUNIT

PHPUnit é uma framework de testes unitários para PHP que facilita a escrita e execução de testes automatizados, ajudando desenvolvedores a garantir que seu código funcione corretamente e atenda aos requisitos definidos (Sebastian, 2024). Por este motivo foi escolhido para realizar os testes unitários.

3. ESTADO DA ARTE

Durante a construção do projeto, foram realizadas pesquisas de aplicações web presentes no mercado com proposta igual ao Protáthlima. A pesquisa serviu para melhor entendimento dos requisitos do projeto.

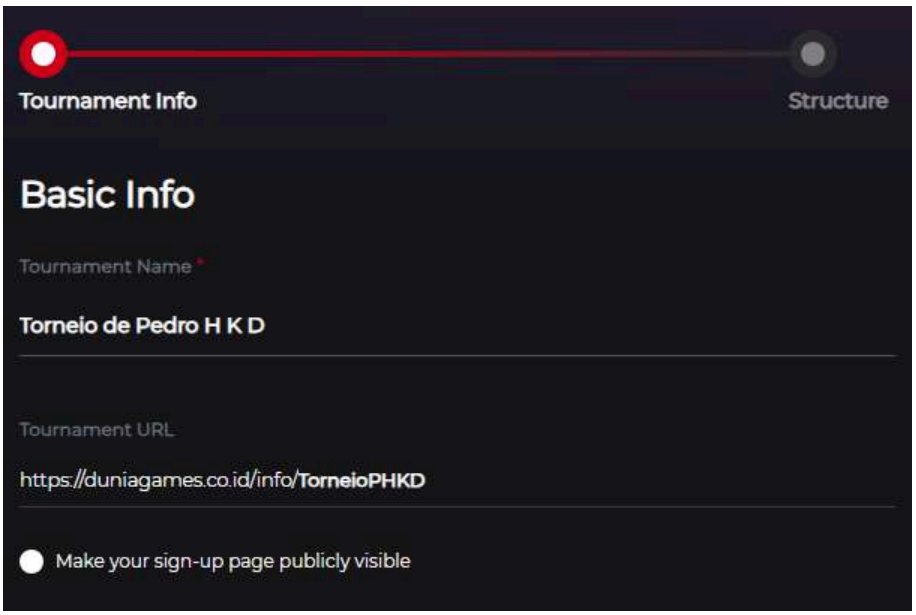
3.1 DUNIA GAMES

Atendendo o mercado de eSports da região sul e leste do continente asiático[4] Dunia Games é um portal informativo do segmento de jogos digitais criado na Indonésia. Dentro do portfólio de serviços oferecidos pela empresa se encontra uma plataforma de organização de torneios dos mais variados jogos disponíveis.

Os recursos oferecidos pela aplicação de torneiros da plataforma permitem o usuário criar, gerenciar e promover um torneio personalizado com sua marca pessoal, apresentando as funcionalidades:

Tela de edição de nomenclatura: permite ao usuário atribuir um título ao seu torneio e criar uma URL personalizada para divulgação de seu torneio.

Figura 1: Tela de edição de nomenclatura Dunia Games.

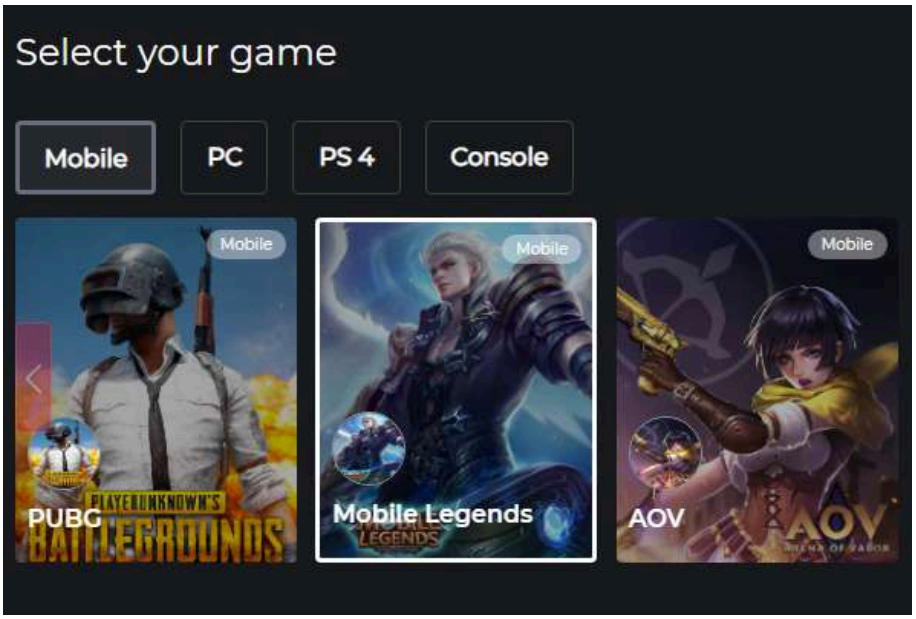


Fonte: Dunia Games (2024)

Seleção de jogo: permite ao usuário selecionar o jogo digital utilizado como plataforma para as partidas do torneio.

[4] Disponível em <duniagames.co.id>

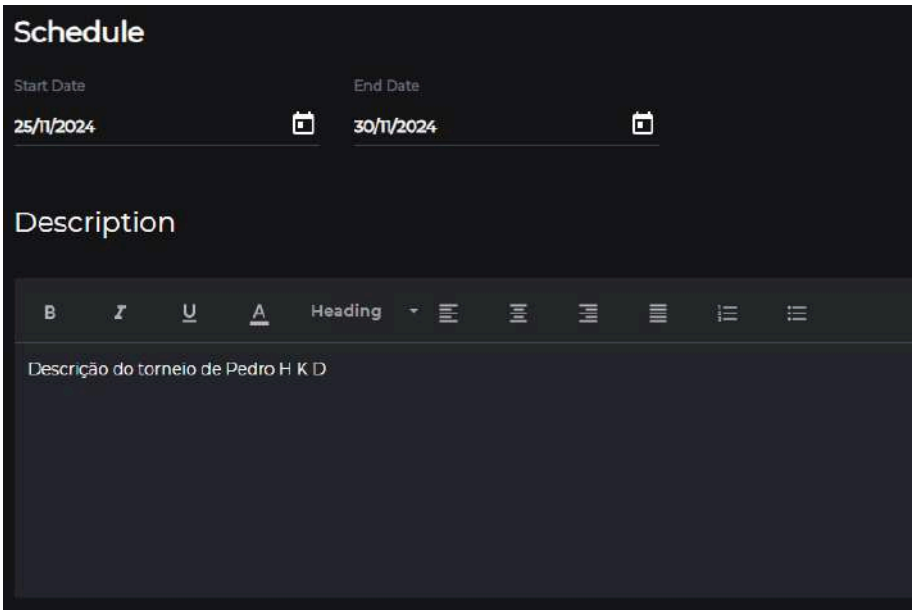
Figura 2: Seleção de jogo Dunia Games.



Fonte: Dunia Games (2024)

Agendamento e descrição: permite ao usuário programar a data de realização das etapas do torneio, com um campo para colocação de informações pertinentes.

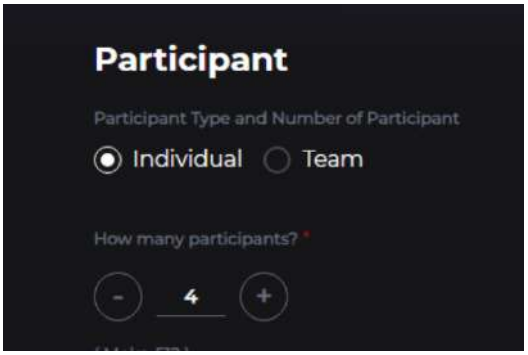
Figura 3: Agendamento e descrição Dunia Games



Fonte: Dunia Games (2024)

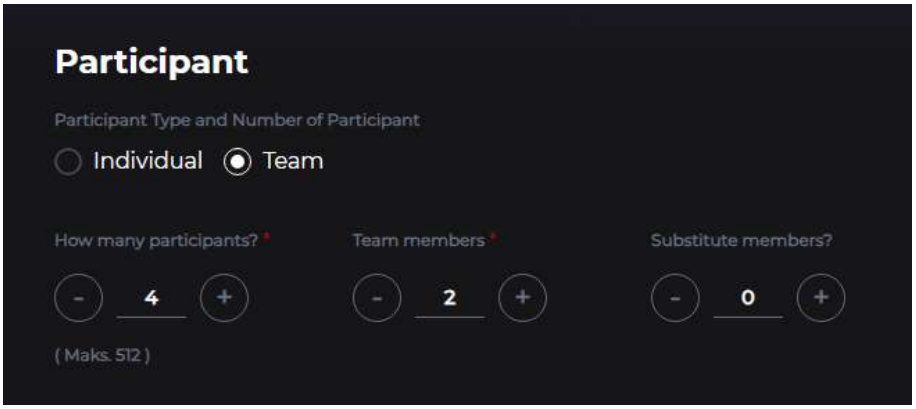
Configuração de cadastro de participantes: o site permite a seleção do modo de cadastro dos participantes, com opções de seleção de quantidade de jogadores individuais, quantidade de equipes, jogadores na equipe e possíveis suplentes.

Figura 4: Configuração de cadastro de participantes da Dunia Games.



Fonte: Dunia Games (2024)

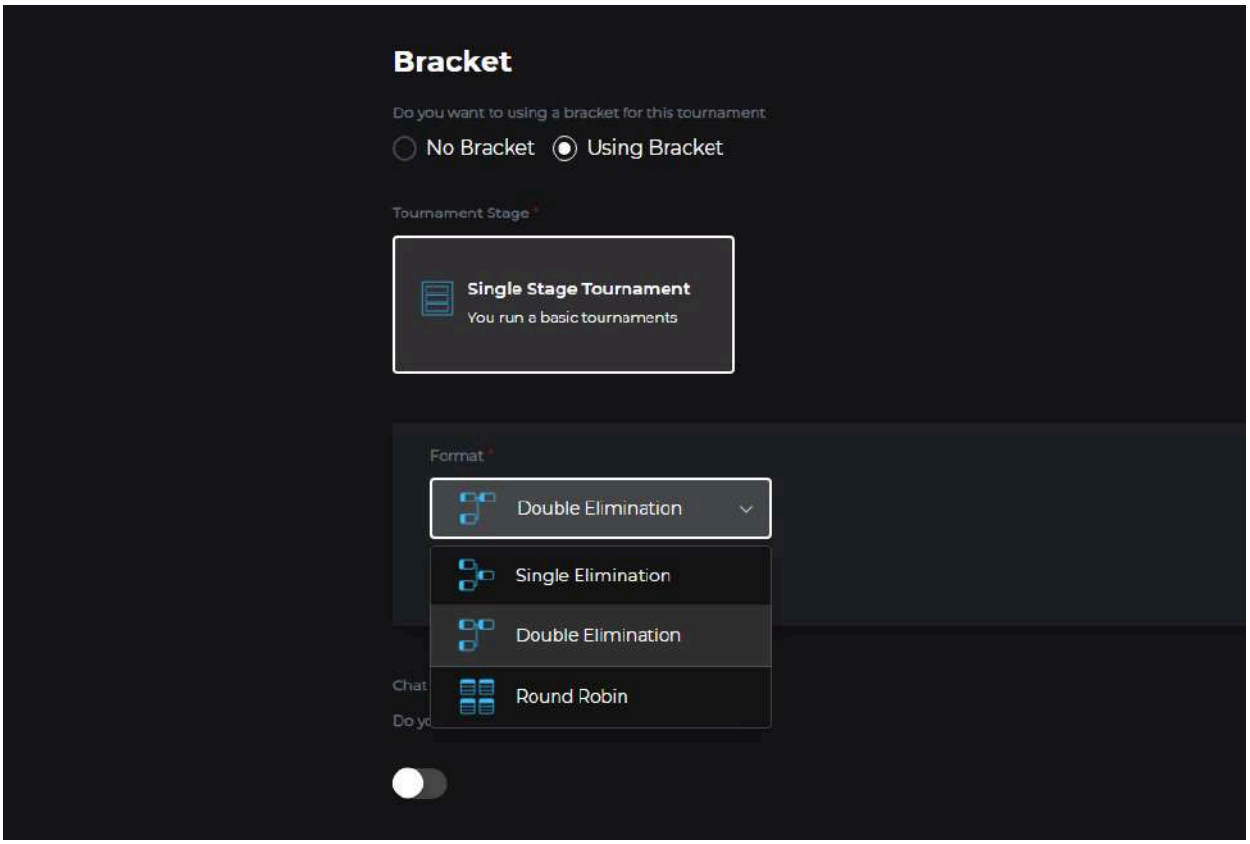
Figura 5 Configuração de cadastro de participantes Dunia Games.



Fonte: Dunia Games (2024)

Chaves: permitem organização das chaves de pareamento dos participantes.

Figura 6: Chaves de campeonato Dunia Games.



Fonte: Dunia Games (2024)

3.2 CHALLENGE.PLACE

Challenge.Place é uma ferramenta de gerenciamento de campeonatos sofisticada. Seus recursos permitem controlar competições em diversas modalidades de esportes, tanto eletrônicos como digitais. A plataforma permite ao organizador gerenciar todas as instâncias do campeonato, ao contrário da ferramenta ofertada pela Dunia Games.

Os recursos disponíveis na aplicação são:

Seleção de modalidade: permite ao usuário selecionar a modalidade utilizada em seu torneio.

Figura 7: Seleção de modalidade Challenge Place.

Fonte: Challenge Place (2024)

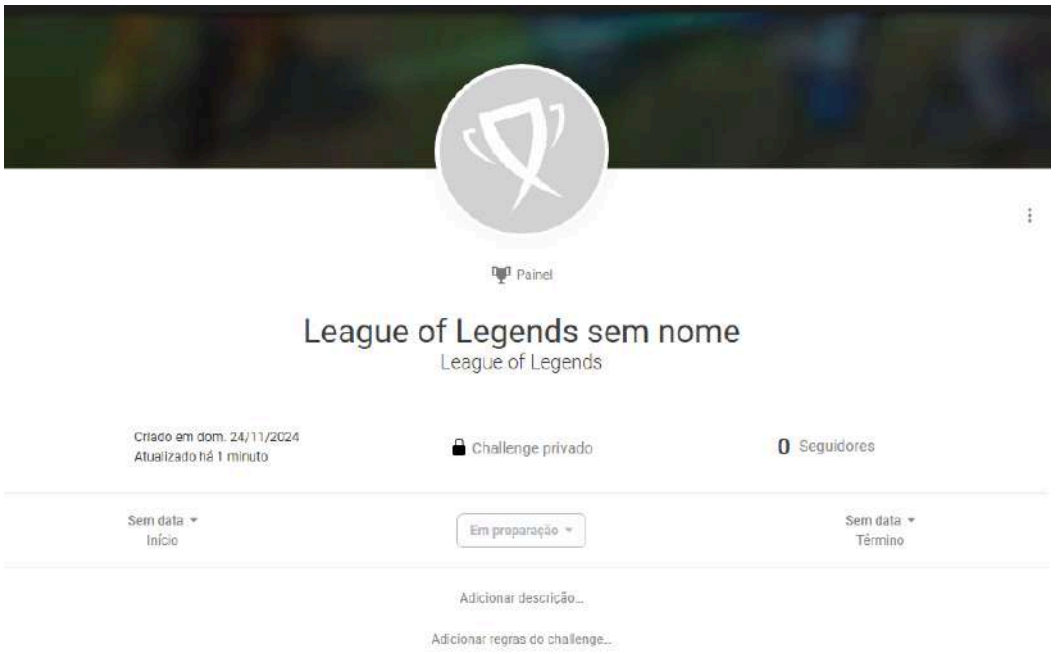
Configuração de inclusão de participantes: definição da quantidade de participantes que compõem um espaço de competidor nas chaves.

Figura 8: Configuração de participantes Challenge Place.

Fonte: Challenge Place (2024)

Agendamento: seleção de data de início e término do torneio criado. A ferramenta também possui um campo de seleção de status do torneio, permitindo sinalizar se está “em preparação”, “em progresso” ou “finalizado".

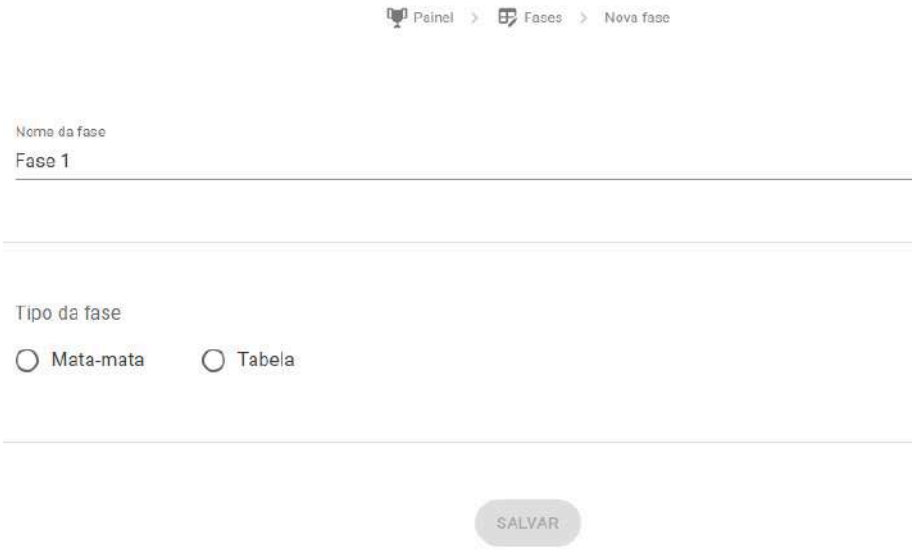
Figura 9: Agendamento de campeonato Challenge Place.



Fonte: Challenge Place (2024)

Definição de fases: é possível criar e organizar etapas no torneio, bem como a utilização do formato de tabelas de pontuação.

Figura 10: Definição de fases Challenge Place.



Fonte: Challenge Place (2024)

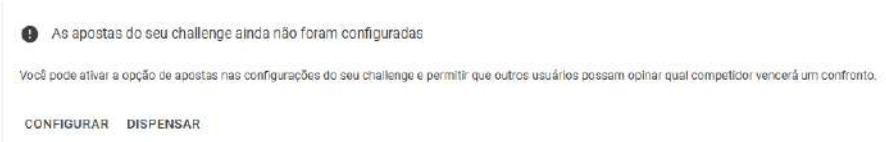
Criação de competidores: diferente da ferramenta ofertada pela Dunia Games, não há aba de inscrição de participantes. Os organizadores do evento possuem total controle sobre o cadastro dos usuários.

Figura 11: Criação de competidores Challenge Place.

Fonte: Challenge Place (2024)

Apostas: há um sistema próprio de gerenciamento de apostas, que permite ao organizador definir valores mínimos e máximos, quem está habilitado a apostar, e o valor recolhido pelo torneio como taxa sobre as apostas.

Figura 12: Apostas Challenge Place.



Fonte: Challenge Place (2024)

Divulgação: a plataforma gera um endereço para o torneio, sem a opção de personalização da URL.

Figura 13: Divulgação Challenge Place

Fonte: Challenge Place (2024)

Portal de notícias: atualizações sobre o torneio podem ser gerenciadas através da própria plataforma.

Figura 14: Portal de notícias Challenge Place

Fonte: Challenge Place (2024)

3.3 COMPARATIVO

A ferramenta proposta neste trabalho inclui recursos necessários para o gerenciamento de um torneio, com a modalidade sendo exclusivamente no jogo digital League of Legends podendo criar torneios com ou sem balanceamento de equipes.

Tabela 1: Comparativo entre as plataformas avaliadas.

Recursos	Dunia Games	Challenge.Place	Protáthlima
Edição de informações	✓	✓	✓
Seleção de jogos	✓	✓	x
Autoinscrição participantes	✓	x	x
Agendamento	✓	✓	✓
Configuração de chaves	✓	✓	✓
Link personalizado	✓	x	x
Portal de atualizações	x	✓	x
Apostas	x	✓	x
Pesquisa dos jogadores na plataforma	x	x	✓
Opção de balanceamento	x	x	✓

Fonte: Elaborado pelo autor (2024)

A análise comparativa entre o Protáthlima e outras plataformas destaca suas peculiaridades. A exclusividade para o League of Legends permite integração direta à API da Riot Games, recurso que as outras plataformas não possuem. Protáthlima se destaca ao permitir a pesquisa detalhada de jogadores, exibindo informações relevantes como taxa de vitória, ranque e campeões mais jogados.

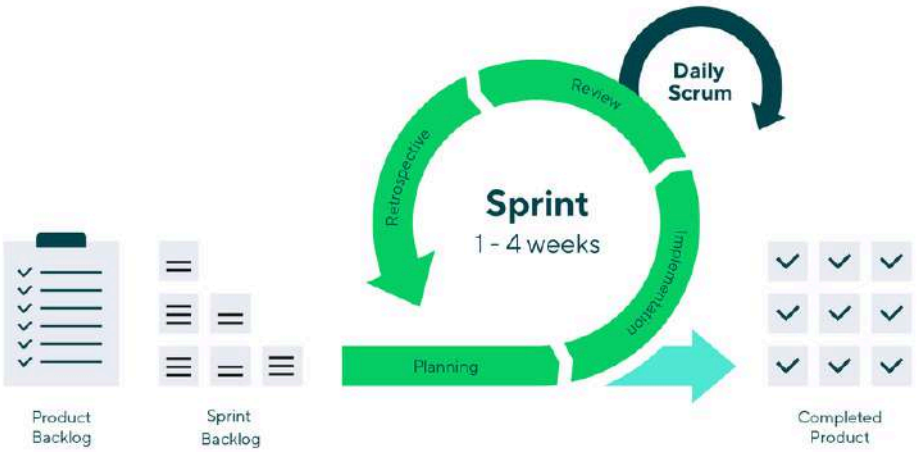
Essa funcionalidade agrega valor à plataforma e preenche lacunas deixadas pelas demais, evidenciando seu foco em oferecer uma experiência personalizada e exclusiva aos jogadores de League of Legends.

4. METODOLOGIA

Segundo Gerhardt (2009, p. 35), a pesquisa de natureza lógica “Objetiva gerar conhecimentos para aplicação prática, dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.”

A metodologia adotada para o desenvolvimento do "Protáthlima" foi uma abordagem ágil utilizando a variante solo[5] do framework Scrum, que permitiu um desenvolvimento incremental e iterativo. O processo foi dividido em sprints[6], que são ciclos curtos de desenvolvimento. Cada sprint incluiu o planejamento, execução e revisão, permitindo ajustes constantes.

Figura 15: Exemplo da sprint da metodologia scrum



Fonte: Wrike (2024)

A análise e modelagem do sistema foram realizadas através da elaboração de diagramas UML[7], como casos de uso, classes e atividades. Esses diagramas permitiram a visualização dos requisitos e da estrutura do sistema, garantindo que todas as funcionalidades necessárias sejam contempladas.

[5] Disponível em: <https://engenhariasoftware.wordpress.com/wp-content/uploads/2016/04/scrum-solo.pdf>

[6] “São eventos de duração fixa de um mês ou menos para criar consistência. Uma nova Sprint começa imediatamente após a conclusão da Sprint anterior.” Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-3.0.pdf>

[7] Linguagem de modelagem unificada(Unified Modeling Language). Disponível em: <https://www.lucidchart.com/pages/pt/o-que-e-uml>.

Para a implementação do "Protáthlima", a interface da aplicação foi desenvolvida utilizando HTML[8] e CSS[9], priorizando uma experiência de usuário intuitiva e responsiva. O design da interface é focado na usabilidade, facilitando a navegação e interação dos usuários.

No back-end[10], a linguagem PHP foi utilizada para o desenvolvimento. Esta escolha se deve à sua flexibilidade e à sua ampla aplicação em projetos web. A aplicação possui integração com a API[11] da RIOT, permitindo a busca de informações sobre jogadores e suas estatísticas. A validação do ranking dos jogadores é feita com base nos dados obtidos da API, assegurando que as equipes formadas tenham uma composição equilibrada.

Após o desenvolvimento das funcionalidades, foram realizados testes exploratórios para garantir a qualidade do software. Os testes exploratórios serviram para verificar a funcionalidade de cada módulo isoladamente, e asseguraram que os módulos funcionem em conjunto de forma eficaz. Além disso, foi realizada uma bateria de testes práticos com usuários voluntários para coletar feedback sobre a usabilidade, design e funcionalidades da aplicação. Essa etapa foi crucial para identificar áreas de melhoria e adaptar a aplicação às necessidades dos usuários.

[8] Linguagem de Marcação de HiperTexto
[9] Cascading Style Sheets ou Folhas de Estilo em Cascata
[10] São as engrenagens do site, que o usuário não tem acesso direto e não consegue interagir
[11] Application Programming Interface (Interface de Programação de Aplicação)

5. APLICAÇÃO PROTÁTHLIMA

Nesta seção são apresentados os detalhes do desenvolvimento da aplicação Protáthlima, incluindo o levantamento de requisitos e as funcionalidades implementadas para atender às necessidades dos jogadores de League of Legends.

5.1. LEVANTAMENTO DE REQUISITOS

Para o levantamento de requisitos, foram realizadas entrevistas diretas com jogadores de League of Legends durante as partidas, caracterizando uma pesquisa qualitativa. Nesse processo, utilizou-se uma abordagem de entrevista semiestruturada, onde os jogadores foram questionados sobre suas necessidades e expectativas em relação à plataforma, jogadores de fácil acesso ao autor foram questionados oralmente enquanto jogadores dentro das partidas de League of Legends foram questionados de forma descritiva, perguntas como:

- Você utilizaria uma plataforma para criar e gerenciar campeonatos de League of Legends?
- Gostaria de ter um mecanismo de busca de jogadores nessa plataforma?
- Gostaria de ter a opção de balanceamento nos campeonatos?
- O que acha sobre a validação de ranque nas equipes?

O formato semiestruturado permitiu que os entrevistados expressassem livremente suas opiniões, ao mesmo tempo em que garantiu que temas chave relacionados à validação do ranqueamento, gestão de usuários, equipes e torneios, e visualização de informações fossem abordados. Não foi utilizado um formulário formal, pois a pesquisa focou em captar insights imediatos e relevantes para o desenvolvimento da aplicação. Foram realizadas seis entrevistas no total, todas realizadas com jogadores de League of Legends.

Com base nesses objetivos, os requisitos funcionais da aplicação foram definidos e classificados nas seguintes áreas:

5.1.1. REQUISITOS FUNCIONAIS (RF)

Gestão de Usuários

- RF01: O sistema deve permitir o cadastro de usuários com email, senha e ID do Riot Games.
- RF02: O sistema deve autenticar usuários através de login com email e senha.
- RF03: O sistema deve permitir que usuários façam logout.

- RF04: O sistema deve validar o ID do Riot Games durante o cadastro.

Gestão de Equipes

- RF05: O sistema deve permitir a criação de equipes com:
    - Nome da equipe;
    - 5 jogadores titulares;
    - Responsável pela equipe.
- RF06: O sistema deve calcular e armazenar o tier médio da equipe.
- RF07: O sistema deve permitir a edição da composição da equipe.
- RF08: O sistema deve permitir a exclusão de equipes.
- RF09: O sistema deve exibir uma lista de equipes com paginação.

Gestão de Torneios

- RF10: O sistema deve permitir a criação de torneios com:
    - Nome do torneio;
    - Data de início;
    - Número de equipes (4 ou 8);
    - Opção de balanceamento de ELO.
- RF11: O sistema deve gerenciar inscrições de equipes nos torneios.
- RF12: O sistema deve validar o balanceamento de ELO entre equipes (máximo 1 tier de diferença).
- RF13: O sistema deve gerar automaticamente o chaveamento do torneio.
- RF14: O sistema deve permitir o registro de resultados das partidas.
- RF15: O sistema deve atualizar automaticamente o status do torneio (ABERTO, EM_ANDAMENTO, FINALIZADO).
- RF16: O sistema deve distribuir medalhas aos jogadores da equipe vencedora.

Busca e Visualização

- RF17: O sistema deve permitir a busca de jogadores.
- RF18: O sistema deve exibir informações detalhadas dos jogadores.
- RF19: O sistema deve mostrar os torneios ativos na página inicial.
- RF20: O sistema deve permitir visualizar torneios em andamento e seus chaveamentos.

5.1.2. REQUISITOS NÃO FUNCIONAIS (RNF)

Integração

- RNF11: O sistema deve integrar-se com a API do Riot Games.
- RNF12: O sistema deve tratar erros de comunicação com APIs externas.
- RNF13: O sistema deve validar dados recebidos de APIs externas.

5.1.3. REGRAS DE NEGÓCIO (RN)

- RN01: Uma equipe deve ter exatamente 5 jogadores titulares.
- RN02: Um torneio só pode ser iniciado com 4 ou 8 equipes.
- RN03: Com balanceamento de ELO ativo, as equipes só podem ter diferença máxima de 1 tier.
- RN04: Apenas o organizador pode gerenciar seu próprio torneio.
- RN05: Um jogador só pode estar em uma equipe por vez.
- RN06: Torneios em andamento não podem ter equipes removidas.
- RN07: Apenas usuários autenticados podem criar torneios.
- RN08: Equipes vencedoras recebem medalhas automaticamente.
- RN09: O chaveamento é gerado aleatoriamente.

Esses requisitos funcionais, regras de negócio e requisitos não funcionais foram essenciais para nortear o desenvolvimento da aplicação Protáthlima, garantindo que as funcionalidades atendessem às demandas identificadas nas entrevistas com os jogadores.

5.1.4 DIAGRAMA DE CASOS DE USO

Para melhor visualização do projeto, foi criado um diagrama de caso de uso que engloba as funcionalidades do sistema conforme na Figura 16. "Um diagrama de casos de uso é uma representação gráfica que ilustra as interações entre os atores e o sistema, descrevendo os casos de uso que o sistema deve fornecer para atender às necessidades dos usuários" (Booch, Rumbaugh & Jacobson, 1999).

A Figura 16 apresenta o diagrama de casos de uso, detalhando as atividades que podem ser desempenhadas pelos atores, usuário não autenticado, usuário autenticado e o sistema Riot Games.

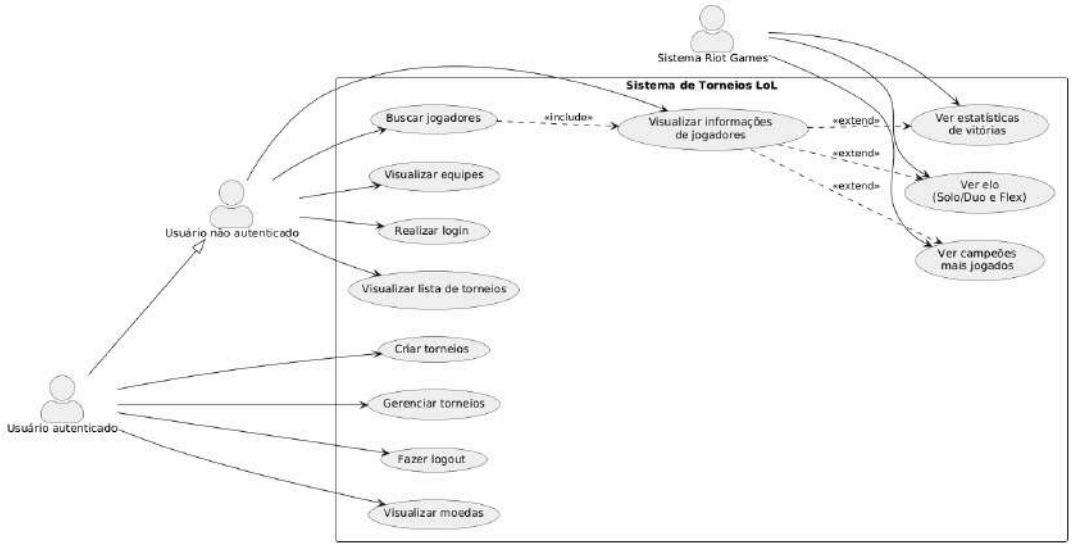
- Usuário não autenticado pode buscar jogadores (integrado ao sistema Riot Games), visualizar equipes, lista de torneios e realizar login.
- Usuário autenticado tem acesso a funções adicionais, como criar e gerenciar torneios, visualizar medalhas acumuladas e realizar logout.

- O sistema Riot Games fornece informações detalhadas sobre jogadores, incluindo estatísticas de vitórias, elos competitivos e campeões mais jogados

Os casos de uso incluem:

- Buscar jogadores: Disponível para todos usuários, com um relacionamento de inclusão («include») com o caso de uso "Visualizar informações de jogadores".
- Visualizar informações de jogadores: Permite acessar dados detalhados, com relacionamentos de extensão («extend») para "Ver estatísticas de vitórias", "Ver elo (Solo/Duo e Flex)" e "Ver campeões mais jogados".
- Visualizar equipes: Disponível para todos usuários, eles conseguem visualizar as equipes cadastradas na aplicação e seus respectivos membros.
- Criar torneios e Gerenciar torneios: Exclusivos para usuários autenticados.
- Visualizar moedas: Também restrito a usuários autenticados.
- Fazer logout: Disponível apenas para usuários autenticados para encerrar a sessão.

Figura 16: Diagrama de casos de uso



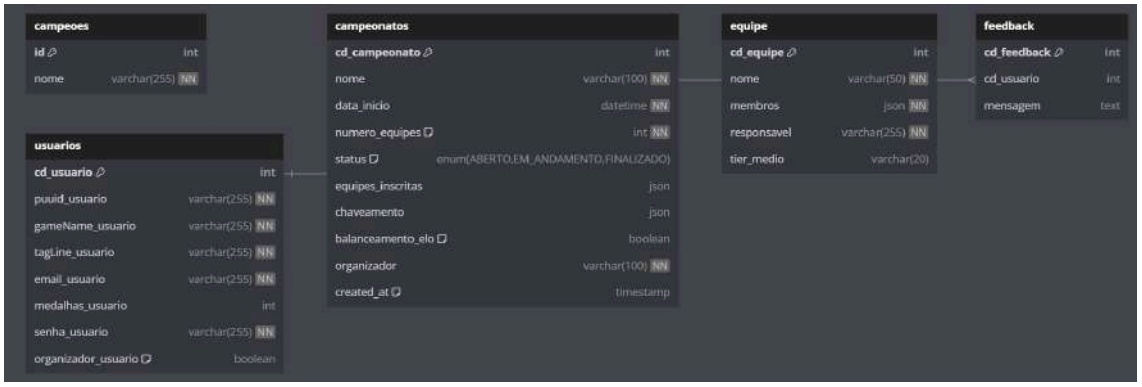
Fonte: Elaborado pelo autor (2024)

Esse diagrama organiza as funcionalidades do sistema com foco em acessibilidade (autenticados e não autenticados) e integração com a API da Riot Games, garantindo que os dados exibidos sejam úteis e confiáveis para os usuários. A separação dos casos de uso assegura clareza no papel de cada ator e a hierarquia das funcionalidades.

5.2 MODELAGEM DO BANCO DE DADOS

A modelagem do banco de dados foi projetada para gerenciar os principais aspectos da aplicação Protáthlima, abordando as entidades relacionadas a jogadores, equipes, campeonatos e dados associados ao League of Legends. A estrutura do banco garante a integridade e consistência das informações, facilitando a integração com a API da Riot Games e o atendimento aos requisitos levantados. Na Figura 17, podemos verificar o diagrama ER[12] do banco de dados. "Um diagrama de entidade-relacionamento (ER) é um modelo visual que descreve as entidades de um sistema e os relacionamentos entre elas. Ele é usado para projetar e estruturar dados em um banco de dados relacional, mostrando as dependências e conexões entre diferentes tipos de dados" (Elmasri & Navathe, 2015).

Figura 17: Modelagem do banco de dados



Fonte: Elaborado pelo autor (2024)

Na Figura 17 apresenta o modelo relacional do banco de dados desenvolvido para o sistema, detalhando as tabelas, seus atributos e as relações entre elas. A estrutura foi projetada para atender às funcionalidades do sistema descritas previamente.

- Tabela **usuarios**:
    - Descrição: Armazena os dados dos usuários do sistema.
    - Atributos:
        - **cd_usuario** (chave primária): Identificador único do usuário.
        - **puuid_usuario**: Identificador universal de usuário fornecido pela Riot API.
        - **gameName_usuario**: Nome de invocador do jogador no League of Legends.
        - **tagLine_usuario**: Identificador adicional do jogador.
        - **email_usuario**: Endereço de e-mail do usuário.

[12] Diagrama de Entidade-Relacionamento

        - **medalhas_usuario**: Quantidade de medalhas conquistadas pelo usuário no sistema.
        - **senha_usuario**: Senha do usuário (armazenada de forma segura).
        - **organizador_usuario**: Indica se o usuário é organizador de torneios (booleano).
- Tabela **campeonatos**:
    - Descrição: Armazena informações sobre os campeonatos criados no sistema.
    - Atributos:
        - **cd_campeonato** (chave primária): Identificador único do campeonato.
        - **nome**: Nome do campeonato.
        - **data_inicio**: Data de início do campeonato.
        - **numero_equipes**: Número de equipes participantes.
        - **status**: Estado atual do campeonato (ABERTO, EM_ANDAMENTO, FINALIZADO).
        - **equipes_inscritas**: Lista das equipes inscritas (armazenada em formato JSON).
        - **chaveamento**: Estrutura do chaveamento do torneio (armazenada em JSON).
        - **balanceamento_elo**: Indica se há balanceamento por ranque entre as equipes (booleano).
        - **organizador**: Nome do organizador do campeonato.
        - **created_at**: Timestamp de criação do campeonato.
- Tabela **equipe**:
    - Descrição: Contém informações das equipes criadas pelos usuários.
    - Atributos:
        - **cd_equipe** (chave primária): Identificador único da equipe.
        - **nome**: Nome da equipe.
        - **membros**: Lista dos membros da equipe (armazenada em formato JSON).
        - **responsavel**: Usuário responsável pela equipe.
        - **tier_medio**: Ranque médio da equipe.
- Tabela **feedback**:
    - Descrição: Armazena feedbacks enviados pelos usuários.
    - Atributos:
        - **cd_feedback** (chave primária): Identificador único do feedback.
        - **cd_usuario**: Referência ao identificador do usuário que enviou o feedback.

        - **mensagem**: Mensagem enviada no feedback.
- Tabela **campeoes**:
    - Descrição: Armazena dados sobre campeões do jogo League of Legends.
    - Atributos:
        - **id** (chave primária): Identificador único do campeão.
        - **nome**: Nome do campeão.
- Relação entre **usuarios** e **campeonatos**:
    - O campo organizador na tabela campeonatos refere-se ao **cd_usuario** na tabela **usuarios**. Essa relação identifica qual usuário criou e é responsável pelo campeonato.
- Relação entre **usuarios** e **equipe**:
    - O campo **responsavel** na tabela equipe refere-se ao **cd_usuario** na tabela **usuarios**. Essa relação identifica o usuário que é responsável pela gestão da equipe.
- Relação entre **usuarios** e **feedback**:
    - O campo **cd_usuario** na tabela **feedback** refere-se ao **cd_usuario** na tabela **usuarios**, permitindo rastrear qual usuário enviou cada feedback.
- Outras associações:
    - Embora não representadas diretamente com chaves estrangeiras no diagrama, os campos **equipes_inscritas** e **chaveamento** na tabela campeonatos armazenam dados estruturados (JSON) que representam equipes associadas a um campeonato. Isso cria uma relação lógica com a tabela **equipe**.

Com essa modelagem, a aplicação Protáthlima é capaz de organizar e gerenciar de forma eficiente as interações entre jogadores, equipes e torneios, mantendo a escalabilidade necessária para o aumento de usuários e dados no sistema.

5.3. DESENVOLVIMENTO

Nesta seção iremos discutir o desenvolvimento da aplicação.

5.3.1 HOSPEDAGEM

Para a hospedagem da aplicação, foi escolhida a plataforma InfinityFree[13], ela é uma plataforma de webhosting gratuita e sem anúncios. Apesar de ser gratuita ela oferece vários recursos como:

- Banda ilimitada
- Compatibilidade com PHP 8.2 e MySQL 8.0/MariaDB 10.6

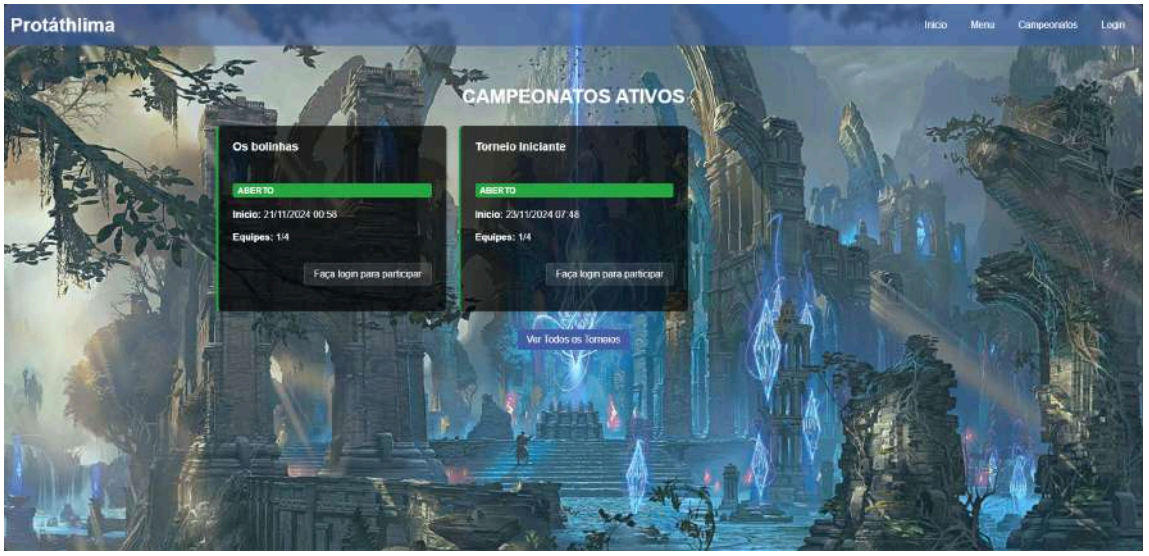
[13] Disponível em: <https://www.infinityfree.com/>

- 5 GB de espaço de disco

5.3.2 TELA INICIAL

Na figura 18, podemos ver a tela inicial da aplicação. É fácil notar que é uma tela bem minimalista, ela mostra os campeonatos ativos com um botão redirecionando para a tela de campeonatos. O header é o mesmo para todas as telas, possuindo um menu de navegação.

Figura 18: Tela inicial do Protathlima

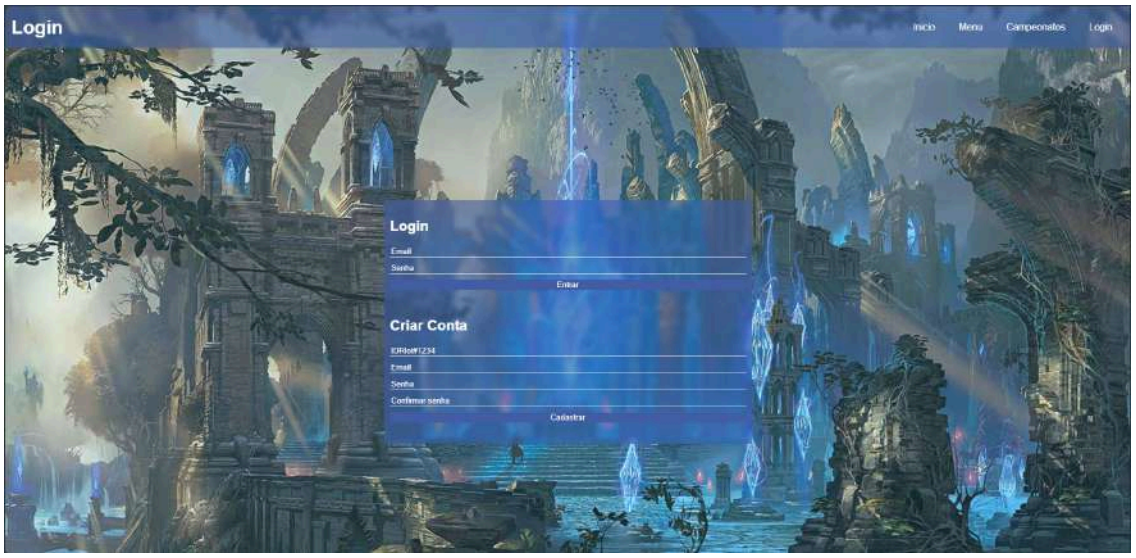


Fonte: Elaborado pelo autor (2024)

5.3.3 TELA DE LOGIN/CADASTRO

Na Figura 19, podemos ver a tela de login/cadastro. É uma tela bem simples com dois formulários, um de login e outro de cadastro. As informações de cadastro são enviadas para o arquivo php de cadastro onde são feitas as validações podemos verificar pela Figura 20 que primeiro é validado o campo do e-mail, depois se as senhas coincidem e por último validamos se o RiotID informado é válido conforme pode ser observado nas Figuras 20 e 21, fazendo isso economizamos consultas desnecessárias na API, somente após esse processo é realizada a consulta na API como indica a Figura 22. Na parte do login é mais simples, apenas consulta o banco de dados e verifica se existe um e-mail cadastrado e se a senha informada confere com o que está no banco de dados, podendo ser visualizado na Figura 23.

Figura 19: Tela de login/cadastro do Protathlima



Fonte: Elaborado pelo autor (2024)

Na Figura 20 podemos verificar o processo de cadastro da aplicação, neste código estamos chamando funções que validam se o RiotID[14] e o e-mail são válidos, se a senha e confirmação de senha conferem por fim é feita a consulta na API retornando os dados da conta.

Figura 20: Validações iniciais para o cadastro do Protathlima

```php
$username = $_POST['signup-username'];
$password_usuario = filter_var($_POST['signup-password'], FILTER_SANITIZE_FULL_SPECIAL_CHARS);
$confirm_password = filter_var($_POST['signup-confirm-password'], FILTER_SANITIZE_FULL_SPECIAL_CHARS);
$email = filter_var($_POST['signup-email'], FILTER_SANITIZE_EMAIL);

if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    header("Location: login.php?error=" . urlencode("Email inválido"));
    exit();
}

if ($password_usuario !== $confirm_password) {
    header("Location: login.php?error=" . urlencode("As senhas não coincidem"));
    exit();
}

if (!validaRiotID($username)) {
    header("Location: login.php?error=" . urlencode("RIOT ID inválido"));
    exit();
}

list($gameName, $tagLine) = explode('#', $username);
$resultado = consultarRiotID($gameName, $tagLine, $apiKey);
```

Fonte: Elaborado pelo autor (2024)

[14] ID da conta Riot

O RiotID é composto por duas partes, o gameName e a tagLine que são separados pelo caractere "#", na Figura 21 podemos verificar a separação do RiotID em duas variáveis a validação delas via regex[15].

Figura 21: Validação do Riot ID

```php
function validaRiotID($riotID)
{
    $parts = explode('#', $riotID);

    if (count($parts) !== 2) {
        return false;
    }

    list($gameName, $tagLine) = $parts;

    if (preg_match('/^[\w\s]{3,16}$/u', $gameName) && preg_match('/^[a-zA-Z0-9]{3,16}$/', $tagLine)) {
        return true;
    } else {
        return false;
    }
}
```

Fonte: Elaborado pelo autor (2024)

Na Figura 22 podemos ver a função "consultaRiotID" mandando uma requisição para a API passando as variáveis como parâmetro e esperando o retorno de um JSON.

Figura 22: Consulta de informações via API da riot games

```php
function consultarRiotID($gameName, $tagLine, $apiKey)
{
    $encodedGameName = urlencode($gameName);
    $link = "https://americas.api.riotgames.com/riot/account/v1/accounts/by-riot-id/$encodedGameName/$tagLine?api_key=$apiKey";
    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, $link);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/json',
        'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36
        Edg/125.0.0.0'
    ]);

    $response = curl_exec($ch);

    if ($response === false) {
        $error = curl_error($ch);
        curl_close($ch);
        return ["error" => "cURL Error: $error $link"];
    }

    curl_close($ch);

    $data = json_decode($response, true);

    if (json_last_error() !== JSON_ERROR_NONE) {
        return ["error" => "JSON Decode Error: " . json_last_error_msg()];
    }

    return $data;
}
```

Fonte: Elaborado pelo autor (2024)

[15] Sequência de caracteres usada para definir padrões de busca e manipulação de texto.

Na Figura 23, podemos ver a autenticação para login sendo feita a partir de um select no banco de dados. Em caso de sucesso é iniciada a sessão com as informações pertinentes.

Figura 23: Autenticação de login

```php
$sql = "SELECT * FROM usuarios WHERE email_usuario = :email AND senha_usuario = :password_usuario";
$stmt = $pdo->prepare($sql);
$stmt->execute([
    ':email' => $email,
    ':password_usuario' => $password_usuario
]);

if ($stmt->rowCount() > 0) {

    if (session_status() == PHP_SESSION_NONE) {
        session_start();
    }

    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    $_SESSION['loggedin'] = true;
    $_SESSION['cd_usuario'] = $row['cd_usuario'];
    $_SESSION['gameName_usuario'] = $row['gameName_usuario'];
    $_SESSION['tagLine_usuario'] = $row['tagLine_usuario'];
    $_SESSION['puuid_usuario'] = $row['puuid_usuario'];
    $_SESSION['medalhas_usuario'] = $row['medalhas_usuario'];

    header("Location: index.php?success=1&success_message=" . urlencode("Login realizado com sucesso!"));
} else {
    header("Location: login.php?error=" . urlencode("Email ou senha incorretos"));
}
exit();
```

Fonte: Elaborado pelo autor (2024)

5.3.4 BUSCA DE JOGADORES

A tela de busca de jogadores possui dois modelos, o primeiro é apenas uma barra de busca como destaca a Figura 24 e o segundo contém as informações retornadas pela busca, fazendo dessa página uma página dinâmica. A pesquisa consiste em verificar na API algumas características do RiotID informado como ranque na fila solo e na fila flex, pontos de liga, taxa de vitória e campeões mais jogados podendo ser observadas na Figura 25. Da mesma maneira que no cadastro, nesta tela também é validado o RiotID antes de fazer a consulta na API. Essa consulta é feita em quatro partes:

- Consulta do puuID[16]. (Mesma consulta da Figura 22 porém apenas é salvo o puuID em uma variável)
- Consulta do summonerID[17]. (Figura 26)

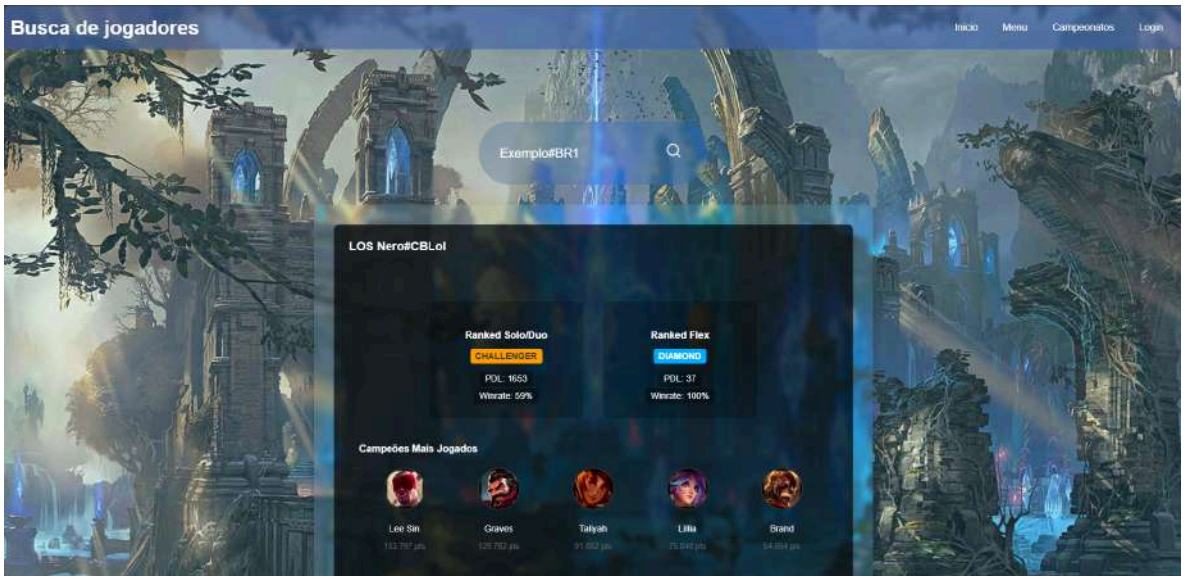
[16] Código identificador interno da Riot Games, único para cada conta.
[17] Código identificador interno referente a partidas das contas do League of Legends.

- Consulta do ranque. (Figura 27)
- Consulta dos campeões mais jogados. (Figura 28)

Figura 24: Busca de jogadores

Fonte: Elaborado pelo autor (2024)

Figura 25: Estatísticas do jogador



Fonte: Elaborado pelo autor (2024)

Na Figura 26 podemos ver a consulta do summonerID via API, é passado o puuid (adquirido na consulta do RiotID) e a chave da API e esperado o retorno do summonerID.

Figura 26: Consulta do summonerID pela API

```php
function consultarSummonerID($puuid, $apiKey)
{
    $link = "https://br1.api.riotgames.com/lol/summoner/v4/summoners/by-puuid/$puuid?api_key=$apiKey";

    $ch = curl_init();

    curl_setopt($ch, CURLOPT_URL, $link);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/json',
        'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36
        Edg/125.0.0.0'
    ]);

    $response = curl_exec($ch);

    if ($response === false) {
        $error = curl_error($ch);
        curl_close($ch);
        return ["error" => "cURL Error: $error $link"];
    }

    curl_close($ch);
    $data = json_decode($response, true);
    if (json_last_error() !== JSON_ERROR_NONE) {
        return ["error" => "JSON Decode Error: " . json_last_error_msg()];
    }
    $summonerID = $data['id'];
    return $summonerID;
}
```

Fonte: Elaborado pelo autor (2024)

Figura 27: Consulta do elo pela API

```php
$data = json_decode($response, true);

foreach ($data as $entry) {
    if ($entry['queueType'] == 'RANKED_SOLO_5x5') {
        $rankedSolo = [
            'tier' => $entry['tier'],
            'rank' => $entry['rank'],
            'wins' => $entry['wins'],
            'losses' => $entry['losses'],
            'leaguePoints' => $entry['leaguePoints']
        ];
    } elseif ($entry['queueType'] == 'RANKED_FLEX_SR') {
        $rankedFlex = [
            'tier' => $entry['tier'],
            'rank' => $entry['rank'],
            'wins' => $entry['wins'],
            'losses' => $entry['losses'],
            'leaguePoints' => $entry['leaguePoints']
        ];
    }
}

if (json_last_error() !== JSON_ERROR_NONE) {
    return ["error" => "JSON Decode Error: " . json_last_error_msg()];
}

return ['rankedSolo' => $rankedSolo, 'rankedFlex' => $rankedFlex];
```

Fonte: Elaborado pelo autor (2024)

Na Figura 27 podemos ver como são tratados os dados referentes ao ranque do jogador, guardamos os valores necessários para mostrar os dados da consulta de jogador.

Figura 28: Consulta de campeões pela API

```php
function consultarCampeoes($puuid, $apiKey)
{
    $link = "https://br1.api.riotgames.com/lol/champion-mastery/v4/champion-masteries/by-puuid/$puuid/top?count=5&api_key=$apiKey";
    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $link);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Content-Type: application/json',
        'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Safari/537.36
        Edg/125.0.0.0'
    ]);
    $response = curl_exec($ch);
    if ($response === false) {
        $error = curl_error($ch);
        curl_close($ch);
        return ["error" => "cURL Error: $error $link"];
    }
    curl_close($ch);
    $data = json_decode($response, true);
    if (json_last_error() !== JSON_ERROR_NONE) {
        return ["error" => "JSON Decode Error: " . json_last_error_msg()];
    }
    $championData = [];
    foreach ($data as $entry) {
        $championData[] = [
            'championId' => $entry['championId'],
            'championPoints' => $entry['championPoints']
        ];
    }
    return $championData;
}
```

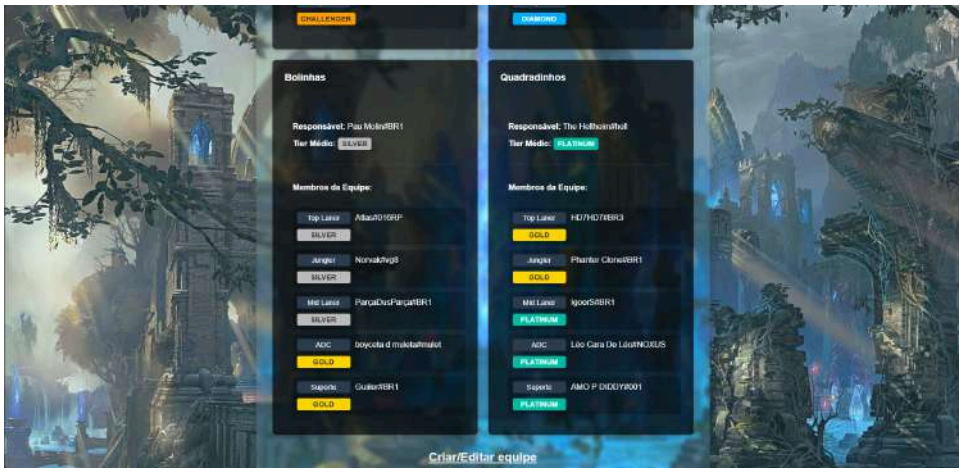
Fonte: Elaborado pelo autor (2024)

Na Figura 28 é chamada a API para retornar os dados dos cinco campeões mais jogados do jogador, guardamos as variáveis para “championId” para pegar a imagem do campeão, e a “championPoints” que mostra a quantia de pontos de maestria.

5.3.5 EQUIPES

A tela de equipes representada na Figura 29 lista todas as equipes cadastradas no sistema através de consulta no banco de dados. Ela traz as seguintes informações: nome, responsável, ranque médio e os membros da equipe com seus respectivos ranques. A tela também conta um com botão de criar e editar a equipe que o usuário é responsável.

Figura 29: Tela de equipes cadastradas no sistema

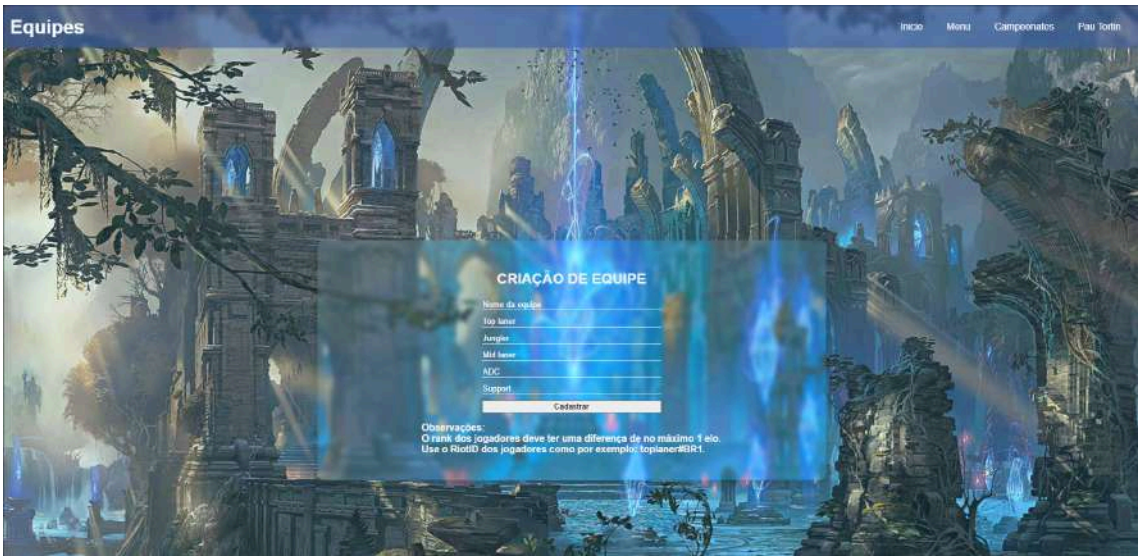


Fonte: Elaborado pelo autor (2024)

5.3.6 CRIAÇÃO/EDIÇÃO DE EQUIPE

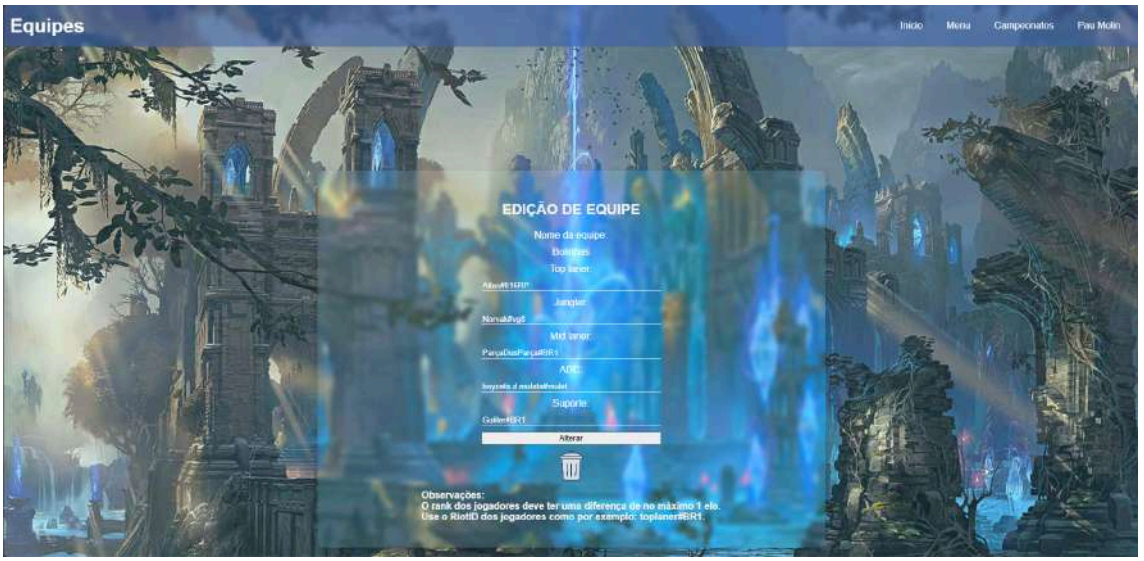
A tela de criação representada na Figura 30 de equipe é dinâmica e permite edição como podemos ver na Figura 31 da equipe caso o usuário seja responsável por ela. Ela possui um formulário bem simples sem validações que redireciona para o arquivo de criação da equipe onde é feita a validação dos Riot IDs informados, e calculado o ranque médio conforme a Figura 32 mostra e se a diferença de ranque dos jogadores é maior que 1 como podemos observar na Figura 33.

Figura 30: Tela de criação de equipe



Fonte: Elaborado pelo autor (2024)

Figura 31: Tela de edição de equipe



Fonte: Elaborado pelo autor (2024)

Figura 32: Cálculo do tier médio

```php
function calcularTierMedio($elos, $tiers) {
    $soma = 0;
    $count = 0;

    foreach ($elos as $elo) {
        $index = array_search($elo, $tiers);
        if ($index !== false) {
            $soma += $index;
            $count++;
        }
    }

    if ($count === 0) return 'UNRANKED';

    $media = $soma / $count;


    $menor_diferenca = PHP_FLOAT_MAX;
    $tier_medio = 'UNRANKED';

    foreach ($tiers as $index => $tier) {
        $diferenca = abs($index - $media);
        if ($diferenca < $menor_diferenca) {
            $menor_diferenca = $diferenca;
            $tier_medio = $tier;
        }
    }

    return $tier_medio;
}
```

Fonte: Elaborado pelo autor (2024)

Na Figura 32 podemos ver como é feito o cálculo do tier médio, a função soma os índices dos elos em um array e calcula a média desses índices. Se não tiver dados é retornado o valor de "UNRANKED". O tier médio é encontrado verificando o índice mais próximo do valor da média.

A Figura 33 é o código que garante que se houver algum jogador com o elo maior que um tier não deixa ser cadastrada a equipe.

Figura 33: Validação de diferença entre os elos

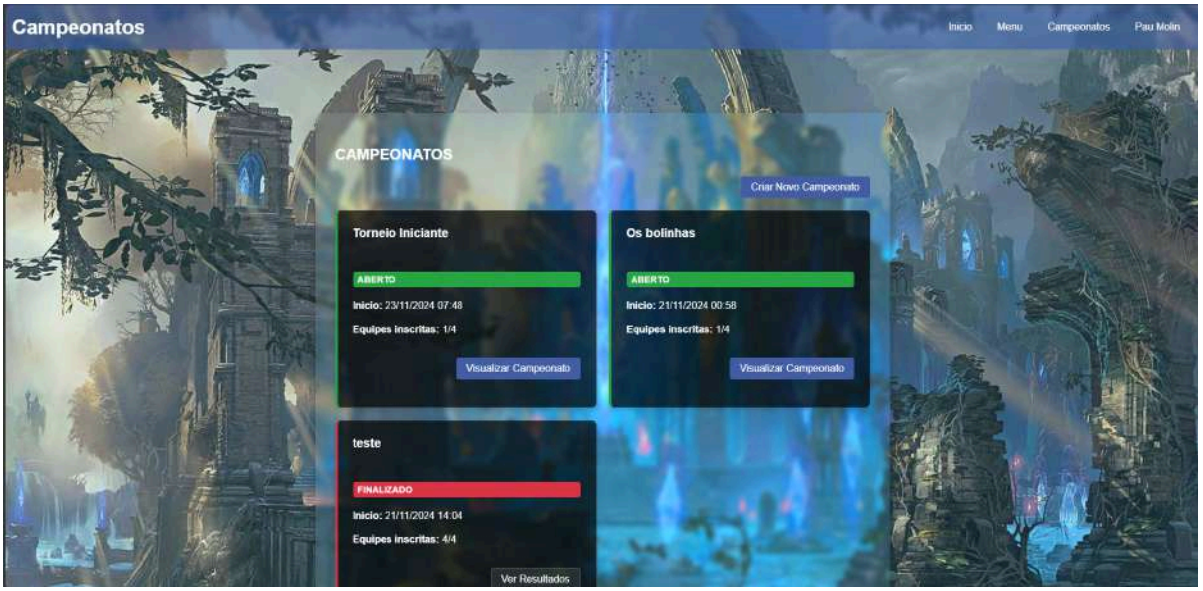
```
foreach ($elos as $elo1) {
    foreach ($elos as $elo2) {
        if (diffTiers($elo1, $elo2, $tiers) > 1) {
            header("refresh:0;url=cadastraequipe.php"); ?>
            <script>
                alert("A diferença de elo entre os jogadores é maior que um tier")
            </script>
            <?php
            exit();
        }
    }
}
```

Fonte: Elaborado pelo autor (2024)

5.3.7 TELA DE CAMPEONATOS

A tela de campeonatos representada na Figura 34 é bem simples, ela apenas lista os campeonatos cadastrados no sistema com informações básicas sobre eles. Ela também possui botões para criar um novo campeonato e para visualizar os já existentes.

Figura 34: Tela de campeonatos

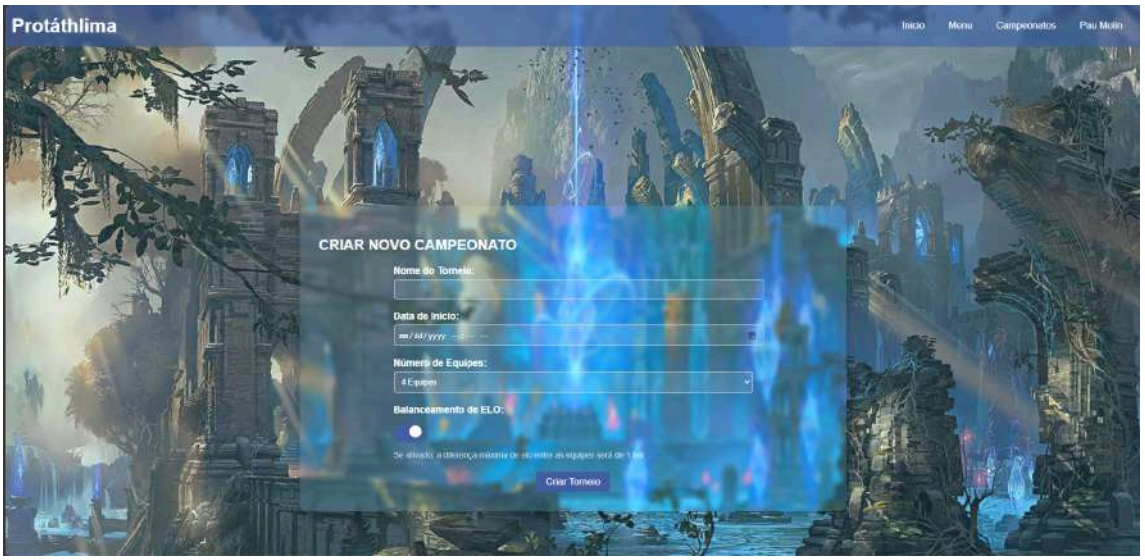


Fonte: Elaborado pelo autor (2024)

5.3.8 TELA DE CRIAÇÃO DE CAMPEONATOS

A tela de criação de campeonatos representada na Figura 35 consiste em um formulário com nome, data de início, número de equipes e com um “flag” para selecionar se o campeonato vai ser balanceado ou seja, se vai validar a diferença de ranque entre as equipes.

Figura 35: Tela de criação de campeonatos

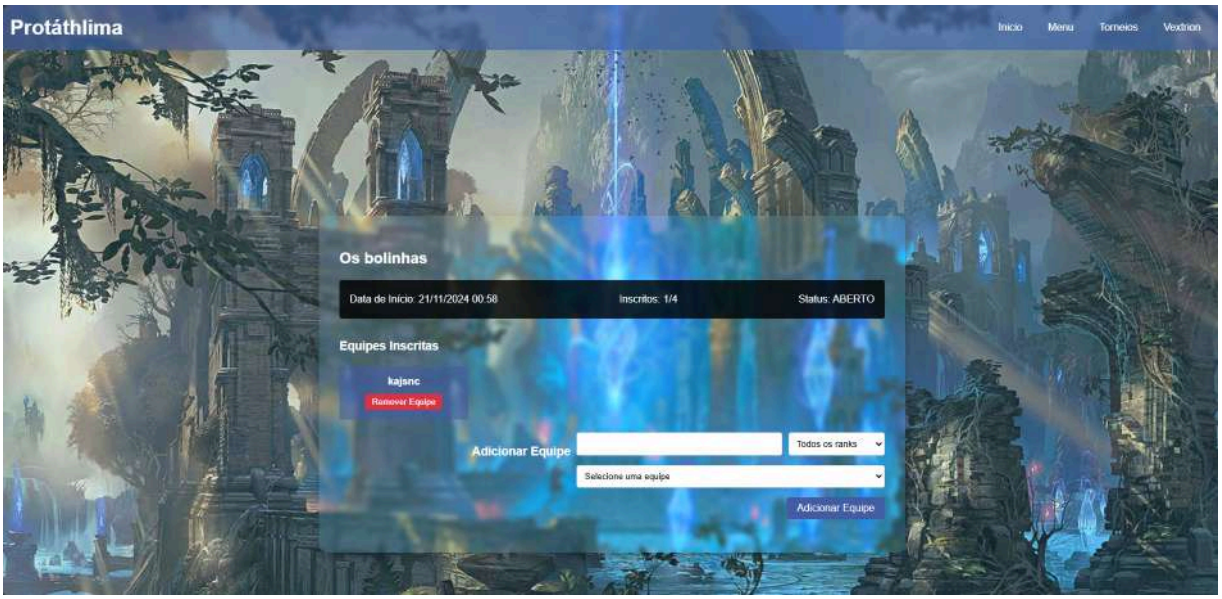


Fonte: Elaborado pelo autor (2024)

5.3.9 GERENCIAMENTO DO CAMPEONATO

A tela de gerenciamento de campeonato representada na Figura 36 é onde o organizador do campeonato deve inscrever as equipes, ela conta com filtro de nome e ranque para selecionar as equipes cadastradas no sistema. Para filtrar pelo nome basta colocar o nome da equipe ou parte dele, já para filtrar pelo ranque é só selecionar o ranque.

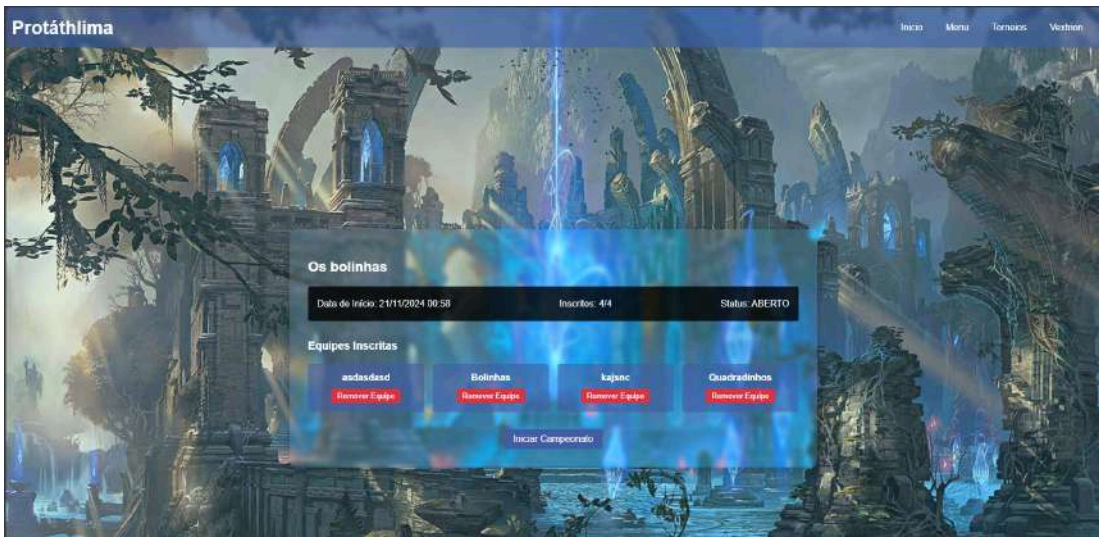
Figura 36: Gerenciamento do campeonato



Fonte: Elaborado pelo autor (2024)

Após todas equipes serem selecionadas irá habilitar o botão para iniciar o campeonato conforme podemos ver na figura 38, iniciando o campeonato as equipes são randomizadas para se enfrentarem conforme figura 39.

Figura 38: Campeonato com todas equipes preenchidas



Fonte: Elaborado pelo autor (2024)

Na Figura 39 é guardado as equipes na variável “equipes” a qual é sorteada por meio da função “shuffle” do php.

Figura 39: Randomização das equipes

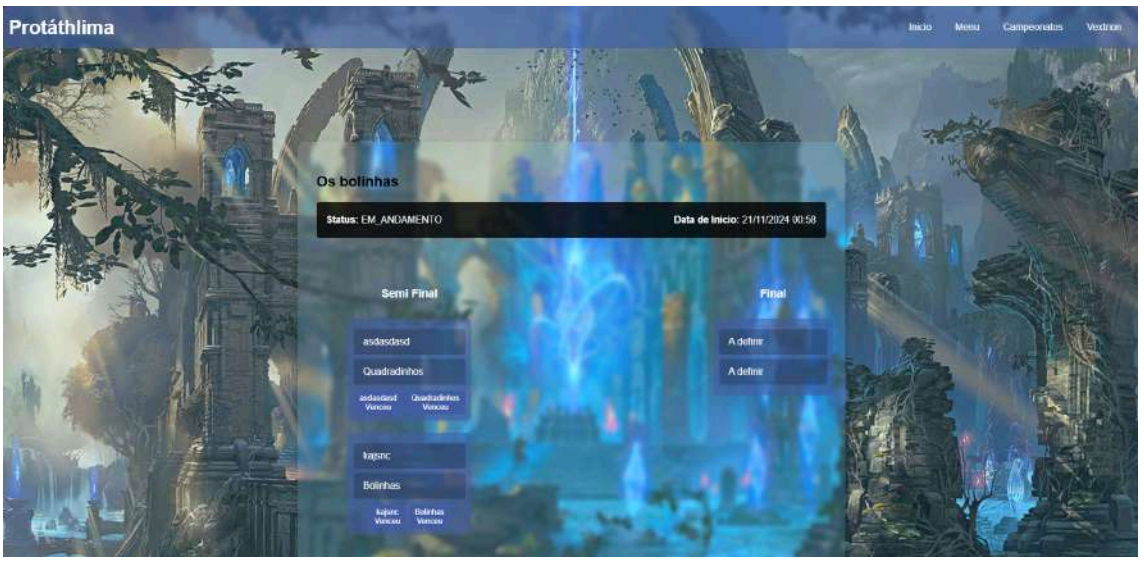
```
$stmt = $pdo->prepare("SELECT * FROM campeonatos WHERE cd_campeonato = ?");
$stmt->execute([$cd_campeonato]);
$campeonato = $stmt->fetch(PDO::FETCH_ASSOC);


$equipes = json_decode($campeonato['equipes_inscritas'], true);


shuffle($equipes);
```

Fonte: Elaborado pelo autor (2024)

5.3.10 TELA DE VISUALIZAÇÃO DE CAMPEONATO

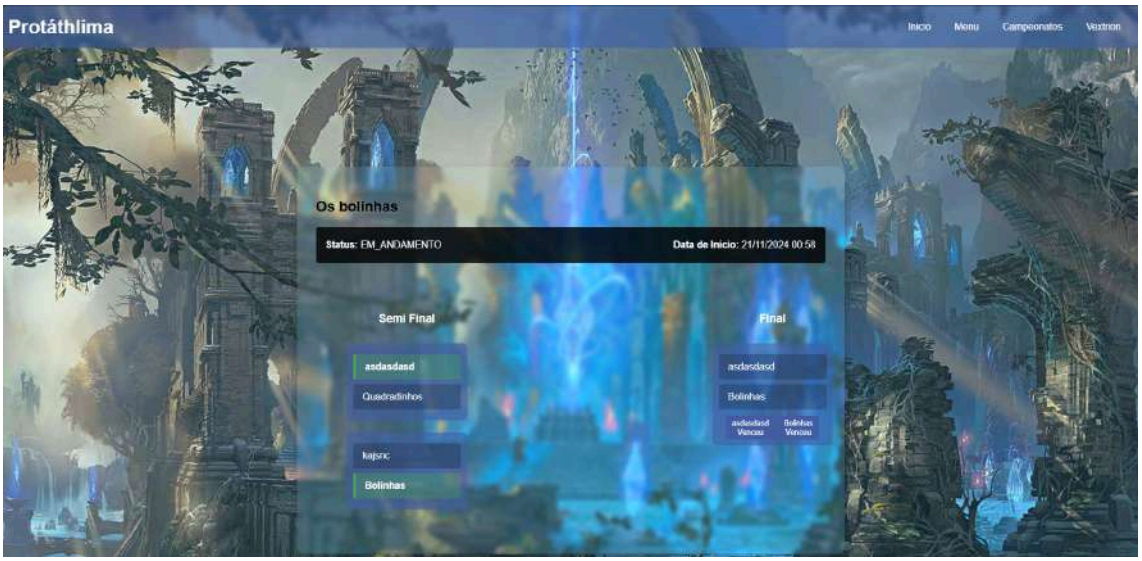
A tela de visualização de campeonato representada na Figura 40 mostra o chaveamento do campeonato. Nesta tela o organizador deve selecionar qual equipe ganhou a partida para ela passar para a próxima etapa do chaveamento conforme indica a Figura 41.

Figura 40: Visualização do campeonato



Fonte: Elaborado pelo autor (2024)

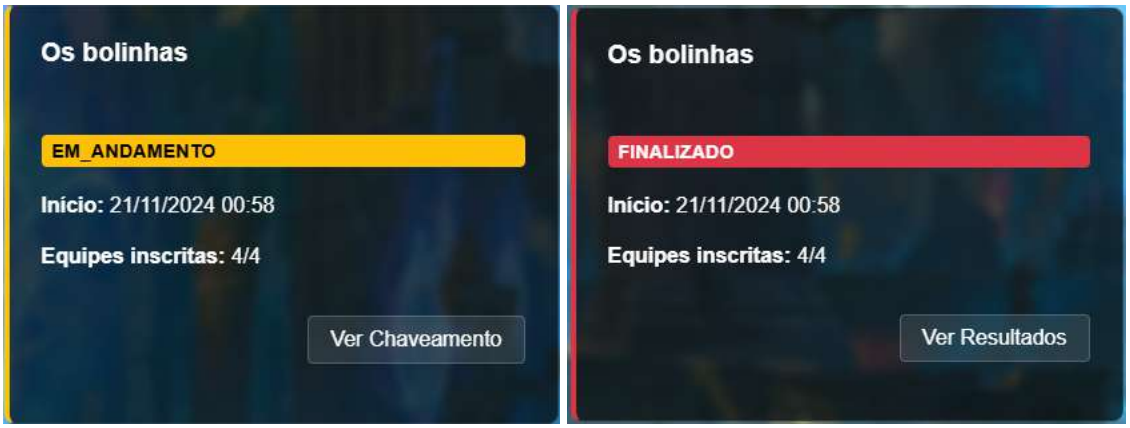
Figura 41: Avanço na chave do campeonato



Fonte: Elaborado pelo autor (2024)

Ao marcar o vencedor da última rodada muda o status do campeonato de “EM_ABERTO” para “FINALIZADO” como podemos observar nas Figuras 42 e 43 e adiciona uma medalha para cada jogador da equipe vencedora caso possuam cadastro na aplicação conforme destacado na Figura 45.

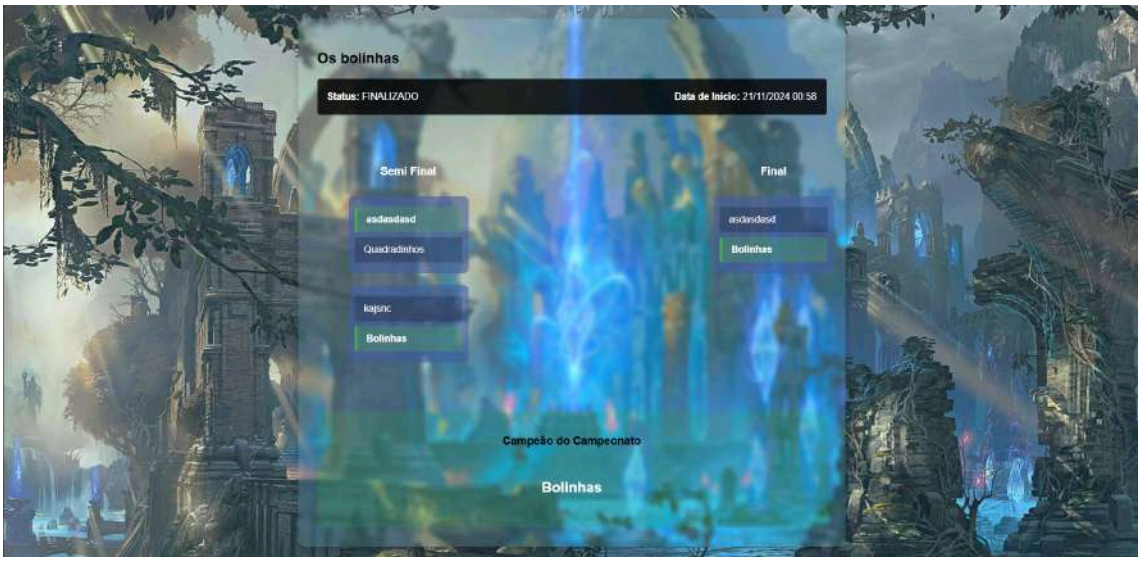
Figura 42 e 43: Status do campeonato



Fonte: Elaborado pelo autor (2024)

Após finalizar o campeonato o botão de ver o chaveamento muda para ver resultados, clicando nele é possível ver quem ganhou cada etapa e qual foi o campeão, a Figura 44 mostra esse resultado.

Figura 44: Visualização do resultado do campeonato



Fonte: Elaborado pelo autor (2024)

Na Figura 45 é feita a distribuição das medalhas para os vencedores que possuem cadastro na aplicação, essa validação é feita a partir de um select no banco de dados validando se o puuid dos jogadores está presente em algum cadastro de usuário.

Figura 45: Distribuição das medalhas para vencedores

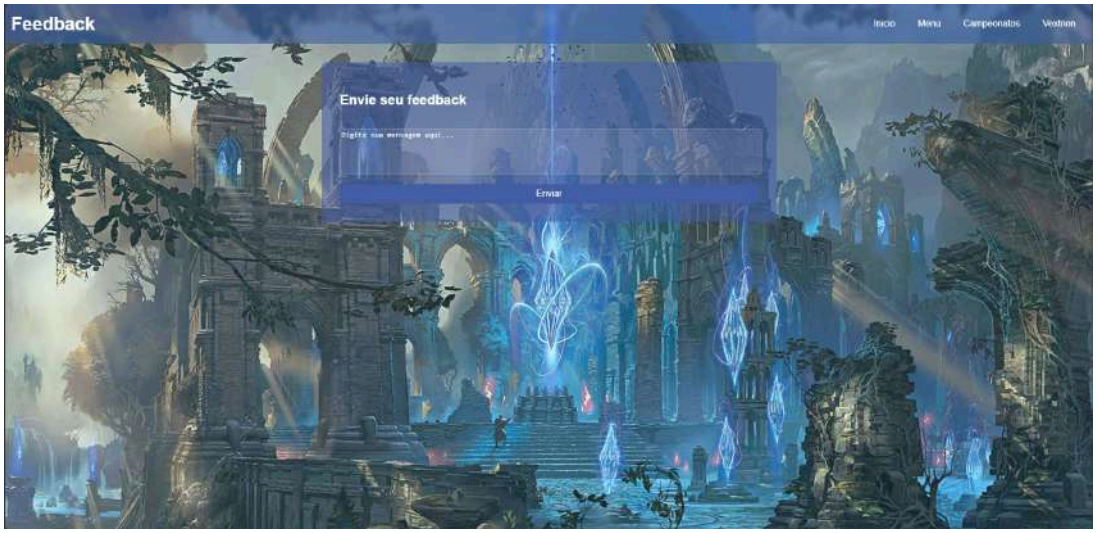
```php
if ($todasPartidasFinalizadas) {

    $stmt = $pdo->prepare("UPDATE campeonatos SET status = 'FINALIZADO' WHERE cd_campeonato = ?");
    $stmt->execute([$cd_campeonato]);


    $stmt = $pdo->prepare("SELECT jogadores FROM equipe WHERE cd_equipe = ?");
    $stmt->execute([$vencedor]);
    $equipe = $stmt->fetch(PDO::FETCH_ASSOC);

    if ($equipe) {
        $jogadores = json_decode($equipe['jogadores'], true);


        $puuids = array_map(function($jogador) {
            return $jogador['puuid'];
        }, $jogadores);


        if (!empty($puuids)) {
            $placeholders = str_repeat('?,', count($puuids) - 1) . '?';
            $stmt = $pdo->prepare("
                UPDATE usuarios
                SET medalhas_usuario = medalhas_usuario + 1
                WHERE puuid IN ($placeholders)
            ");
            $stmt->execute($puuids);
        }
    }
}
```

Fonte: Elaborado pelo autor (2024)

5.3.11 TELA DE FEEDBACK

A tela de feedback (Figura 46) é simples e prática, basta apenas estar logado para habilitá-lá, este é o mecanismo de feedback implementado na aplicação, nessa tela o usuário que desejar contribuir com um feedback deve informá-lo no campo de texto e clicar em enviar, com isso será populado o banco com o ID do usuário, data do feedback e o conteúdo que foi preenchido no campo de texto.

Figura 46: Tela de feedback



Fonte: Elaborado pelo autor (2024)

5.3.12 TESTES

Os testes de unidade são essenciais no processo de desenvolvimento de software, pois permitem validar componentes individuais do sistema. Neste trabalho, foi utilizada a ferramenta PHPUnit para testar as funcionalidades principais do sistema, incluindo o cadastro de usuários, a autenticação de login e as operações de gerenciamento de campeonatos. Os resultados estão presentes na Figura 52.

5.3.12.1 TESTES DO CADASTRO DE USUÁRIOS

Para o cadastro de usuários, foram elaborados cenários de teste que abordam diferentes situações, como:

- **Cadastro bem-sucedido**: Verifica a inserção correta de um novo usuário no banco de dados e garante que os dados sejam armazenados conforme esperado. (Figura 47)
- **Cadastro duplicado**: Simula a tentativa de cadastrar um usuário já existente com o mesmo identificador único (**PUUID**), avaliando a aplicação da restrição de unicidade. (Figura 48)

Figura 47: Teste de usuário válido

```php
public function testCadastroDeUsuarioValido()
{
    $email = 'novo@usuario.com';
    $senha = 'senha123';
    $gameName = 'TestUser';
    $tagLine = 'BR1';
    $puuid = 'abc123';

    $sql = "INSERT INTO usuarios (email_usuario, senha_usuario, gameName_usuario, tagLine_usuario, puuid_usuario)
            VALUES (:email, :senha, :gameName, :tagLine, :puuid)";
    $stmt = $this->pdo->prepare($sql);
    $success = $stmt->execute([
        ':email' => $email,
        ':senha' => $senha,
        ':gameName' => $gameName,
        ':tagLine' => $tagLine,
        ':puuid' => $puuid
    ]);

    $this->assertTrue($success, 'Cadastro de usuário válido falhou.');
}
```

Fonte: Elaborado pelo autor (2024)

Figura 48: Teste de cadastro duplicado

```php
public function testCadastroDeUsuarioDuplicado()
{
    $email = 'usuario@duplicado.com';
    $senha = 'senha123';


    $sql = "INSERT INTO usuarios (email_usuario, senha_usuario) VALUES (:email, :senha)";
    $stmt = $this->pdo->prepare($sql);
    $stmt->execute([':email' => $email, ':senha' => $senha]);


    $this->expectException(PDOException::class);

    $stmt->execute([':email' => $email, ':senha' => $senha]);
}
```

Fonte: Elaborado pelo autor (2024)

5.3.12.2 TESTES DE AUTENTICAÇÃO DE LOGIN

A funcionalidade de login foi testada com o objetivo de validar a segurança e a precisão na autenticação. Os cenários de teste incluem:

- **Login com credenciais válidas**: Simula o login de um usuário existente com email e senha corretos, verificando o início da sessão e o redirecionamento apropriado. (Figura 49)
- **Login com credenciais inválidas**: Avalia a resposta do sistema a tentativas de login com informações incorretas, garantindo a exibição de mensagens de erro adequadas. (Figura 50)

Figura 49: Testes com credenciais válidas

```php
public function testLoginComCredenciaisCorretas()
{

    $email = 'usuario@teste.com';
    $senha = 'senha123';
    $puuid = '5CltpHIxrSCcRlu2ZMc78Ttjs37-QpSJeTJ9cV6vds3HzTiUUbYgjfJWP_Ope-2JtXBO6_Ib9wxrSQ';
    $sql = "INSERT INTO usuarios (email_usuario, senha_usuario, puuid_usuario) VALUES (:email, :senha, :puuid)";
    $stmt = $this->pdo->prepare($sql);
    $stmt->execute([':email' => $email, ':senha' => $senha, ':puuid' => $puuid]);

    $sql = "SELECT * FROM usuarios WHERE email_usuario = :email AND senha_usuario = :senha";
    $stmt = $this->pdo->prepare($sql);
    $stmt->execute([':email' => $email, ':senha' => $senha]);

    $this->assertTrue($stmt->rowCount() > 0, 'Login falhou com credenciais corretas.');
}
```

Fonte: Elaborado pelo autor (2024)

Figura 50: Testes com credenciais inválidas

```php
public function testLoginComCredenciaisIncorretas()
{
    $email = 'usuario@invalido.com';
    $senha = 'senhaerrada';

    $sql = "SELECT * FROM usuarios WHERE email_usuario = :email AND senha_usuario = :senha";
    $stmt = $this->pdo->prepare($sql);
    $stmt->execute([':email' => $email, ':senha' => $senha]);

    $this->assertEquals(0, $stmt->rowCount(), 'Login deveria falhar com credenciais incorretas.');
}
```

Fonte: Elaborado pelo autor (2024)

5.3.12.3 TESTES DE OPERAÇÕES EM CAMPEONATOS

Os testes realizados para o gerenciamento de campeonatos incluem:

- **Criação de campeonatos**: Testa a inserção de novos campeonatos no banco de dados, verificando a consistência e integridade das informações armazenadas. (Figura 51)

Figura 51: Teste de criação de campeonato

```php
public function testCriarCampeonato()
{
    $nome = "Campeonato Teste";
    $dataInicio = date('Y-m-d', strtotime('+1 day'));
    $numeroEquipes = 8;
    $balanceamento_elo = 1;
    $organizador = "TestUser#BR1";
    $sql = "INSERT INTO campeonatos (
        nome,
        data_inicio,
        numero_equipes,
        balanceamento_elo,
        organizador,
        status,
        equipes_inscritas
    ) VALUES (?, ?, ?, ?, ?, 'ABERTO', '[]')";
    $stmt = $this->pdo->prepare($sql);
    $result = $stmt->execute([
        $nome,
        $dataInicio,
        $numeroEquipes,
        $balanceamento_elo,
        $organizador
    ]);
    $this->assertTrue($result);
    $stmt = $this->pdo->query("SELECT * FROM campeonatos WHERE nome = 'Campeonato Teste'");
    $campeonato = $stmt->fetch(PDO::FETCH_ASSOC);
    $this->assertSame($nome, $campeonato['nome']);
    $this->assertSame((string) $numeroEquipes, (string) $campeonato['numero_equipes']);
}
```

Fonte: Elaborado pelo autor (2024)

Figura 52: Resultado dos testes unitários

```
PS C:\xampp\htdocs\TCC> ./vendor/bin/phpunit --testdox
PHPUnit 11.4.4 by Sebastian Bergmann and contributors.

Runtime:       PHP 8.2.12
Configuration: C:\xampp\htdocs\TCC\phpunit.xml

......                                                              6 / 6 (100%)

Time: 00:00.081, Memory: 8.00 MB

Cadastro
 ✓ Cadastro de usuario valido
 ✓ Cadastro de usuario duplicado

Campeonato
 ✓ Criar campeonato

Equipe
 ✓ Criar equipe

Login
 ✓ Login com credenciais corretas
 ✓ Login com credenciais incorretas
```

Fonte: Elaborado pelo autor (2024)

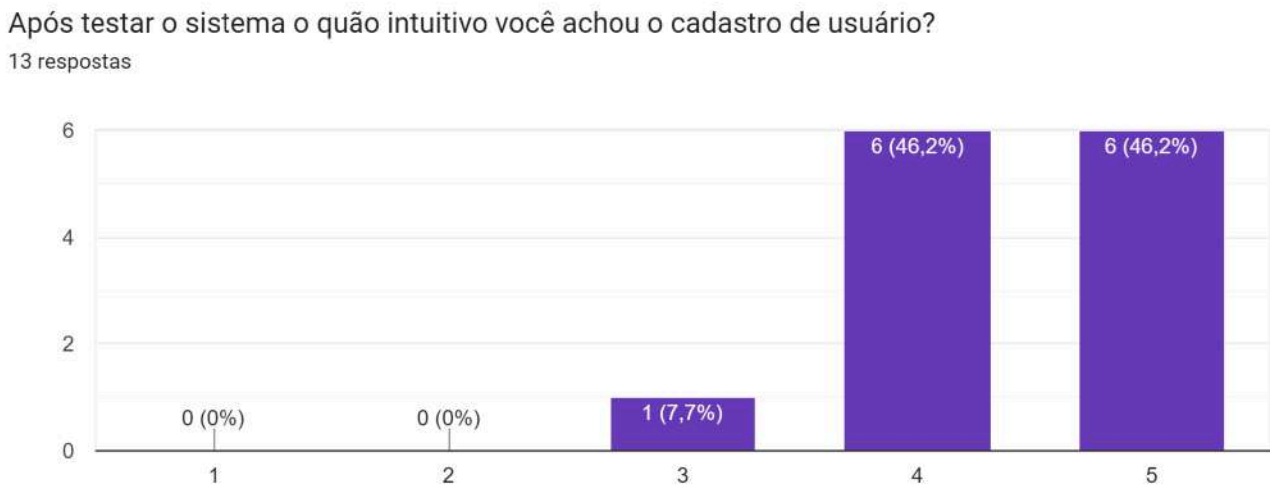
Na Figura 52 podemos ver os resultados da execução dos testes unitários. Como indica o marcador verde, todos retornaram os resultados esperados pré-configurados nos arquivos de teste.

5.3.12.4 TESTES COM USUÁRIOS

Foram realizados testes de usabilidade e facilidade de uso do sistema com voluntários. Esses testes consistiram em criar uma conta na aplicação, utilizar a ferramenta de busca de jogadores, criar uma equipe e, por fim, criar um campeonato. Logo após os testes, os voluntários responderam a um formulário sobre a sua experiência com a aplicação. Conforme descrito por Sommerville (2018), o teste de usuário é uma etapa essencial no processo de desenvolvimento de software, pois permite que o sistema seja avaliado no ambiente real de trabalho do usuário, considerando fatores como confiabilidade, desempenho e usabilidade. Esses aspectos são difíceis de reproduzir nos ambientes artificiais como por exemplo os testes unitários, tornando o feedback dos usuários crucial para validar o sistema no contexto esperado de uso.

Conforme apresentado na Figura 53, os participantes foram questionados sobre o quão intuitivo consideraram o cadastro de usuário no sistema, em uma escala de 1 a 5, onde 1 ‘representa nada intuitivo’ e 5 representa ‘muito intuitivo’. A maioria das respostas foram entre os valores 4 e 5, totalizando 92,4% dos participantes (46,2% para cada valor), indicando uma percepção predominantemente positiva em relação à facilidade do cadastro. Apenas um participante (7,7%) avaliou o cadastro com nota 3, e nenhum participante atribuiu as notas 1 ou 2. Esses dados sugerem que o sistema apresenta um bom nível de usabilidade no processo de cadastro.

Figura 53: Respostas sobre a intuitividade do cadastro.



Fonte: Elaborado pelo autor (2024)

Conforme apresentado na Figura 54, 53,8% dos participantes avaliaram a busca de jogadores como muito intuitiva (nota 5), enquanto 23,1% atribuíram nota 4 e outros 23,1% deram nota 3.

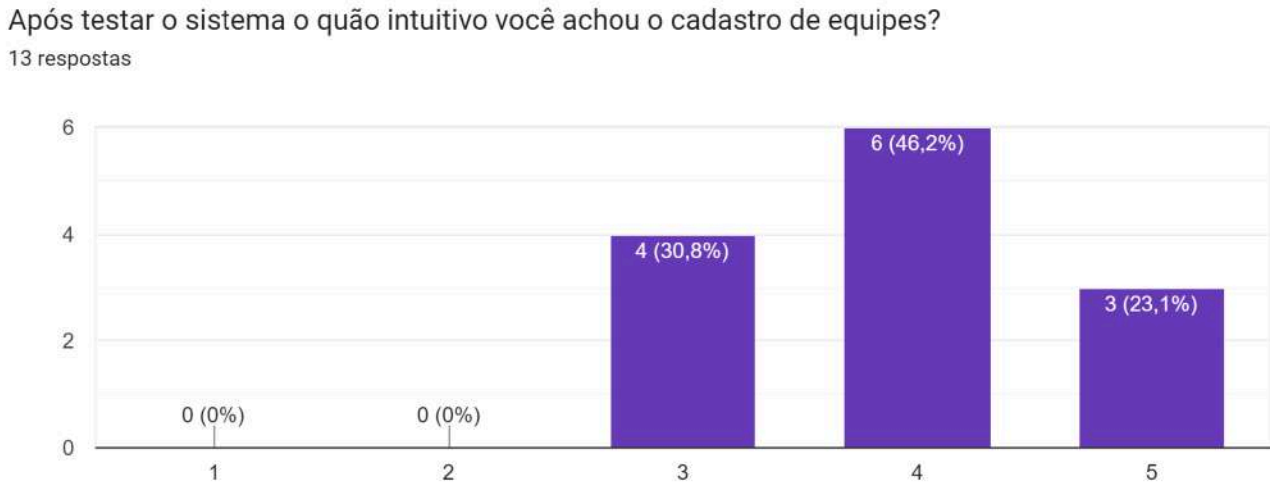
Nenhum participante escolheu as notas 1 ou 2. Esses dados sugerem que a intuitividade é bem avaliada, com a maioria dos usuários indicando uma experiência positiva.

Figura 54: Respostas sobre intuitividade da busca de jogadores.

Fonte: Elaborado pelo autor (2024)

Conforme apresentado na Figura 55, 46,2% dos participantes avaliaram o cadastro de equipes como intuitivo (nota 4), 23,1% atribuíram nota 5, e 30,8% deram nota 3. Nenhum participante selecionou as notas 1 ou 2. Esses resultados indicam que a funcionalidade possui um bom nível de usabilidade, com a maioria das avaliações sendo positivas.

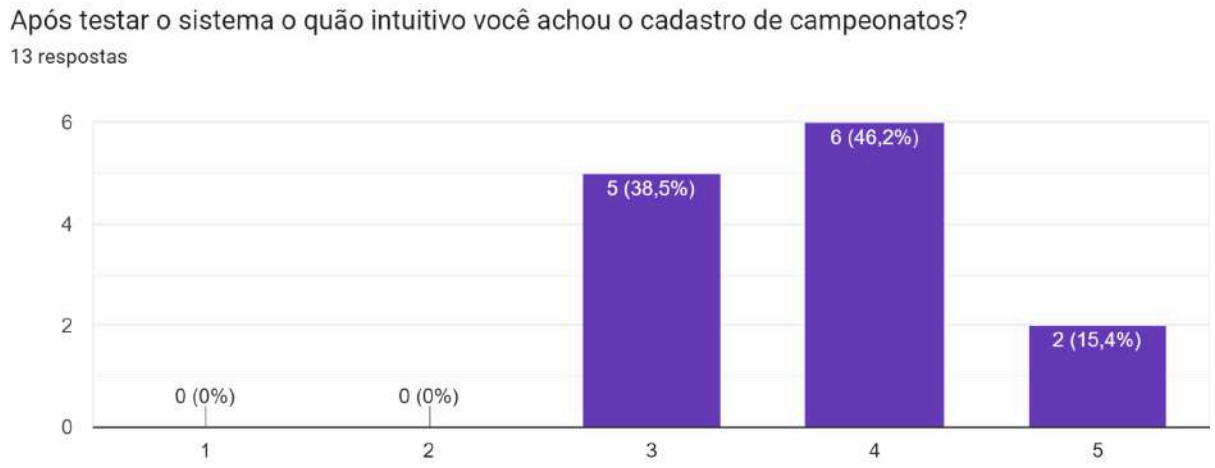
Figura 55: Respostas sobre a intuitividade do cadastro de equipes.



Fonte: Elaborado pelo autor (2024)

Conforme apresentado na Figura 56, 46,2% dos participantes avaliaram o cadastro de campeonatos como intuitivo (nota 4), 15,4% atribuíram nota 5, e 38,5% deram nota 3. Nenhum participante selecionou as notas 1 ou 2. Esses resultados indicam que a funcionalidade apresenta um bom nível de usabilidade, com a maioria das avaliações sendo positivas.

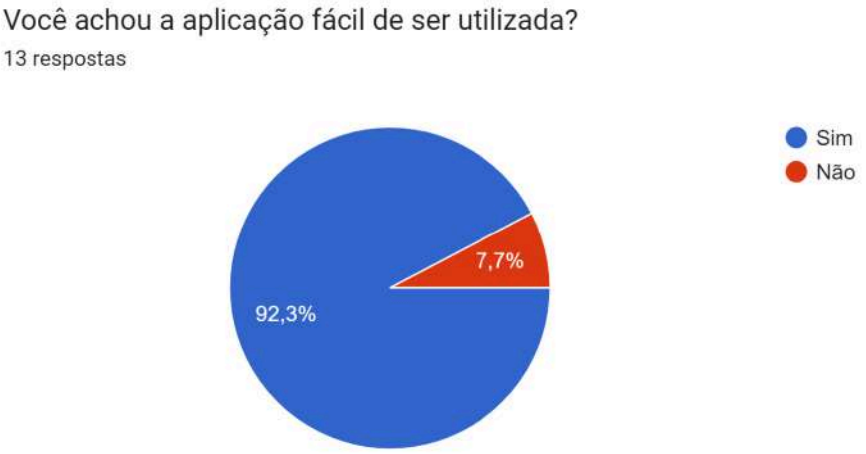
Figura 56: Respostas sobre intuitividade do cadastro de campeonatos.



Fonte: Elaborado pelo autor (2024)

Conforme apresentado na Figura 57, 92,3% dos participantes avaliaram a aplicação como fácil de ser utilizada e 7,7% avaliaram que não é fácil de ser utilizada. Esses resultados indicam que a aplicação é fácil de ser utilizada, com a maioria das avaliações sendo positivas.

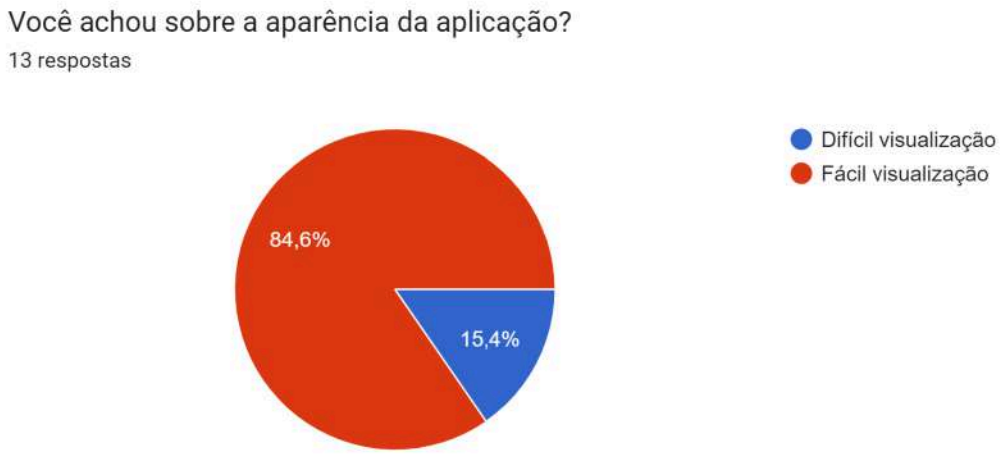
Figura 57: Respostas sobre facilidade de uso da aplicação.



Fonte: Elaborado pelo autor (2024)

Conforme apresentado na Figura 58, 84,6% dos participantes avaliaram a aparência da aplicação como sendo de fácil visualização e 15,4% avaliaram que não é de fácil fácil visualização. Esses resultados indicam que a aplicação é fácil visualização, com a maioria das avaliações sendo positivas.

Figura 58: Respostas sobre a aparência da aplicação.



Fonte: Elaborado pelo autor (2024)

6. CONSIDERAÇÕES FINAIS

Esta monografia teve como propósito principal o desenvolvimento e a documentação da aplicação Protáthlima, uma plataforma web voltada à criação e organização de campeonatos amadores do jogo League of Legends. Desde o levantamento de requisitos até a implementação, o projeto buscou atender às necessidades específicas identificadas junto à comunidade de jogadores, utilizando tecnologias modernas e uma abordagem baseada em boas práticas de engenharia de software.

O desenvolvimento do Protáthlima envolveu desafios técnicos e conceituais que foram superados com o uso de metodologias estruturadas, como entrevistas com usuários e modelagem por meio de diagramas UML. Destacam-se como pontos fortes da plataforma a integração com a API da Riot Games, que permite validações automatizadas de ranque e perfil, e a implementação de funcionalidades como balanceamento de ELO, gerenciamento de equipes e torneios, além do sistema de gamificação que incentiva o engajamento dos usuários.

Ao longo do processo, aspectos como usabilidade, escalabilidade e confiabilidade foram priorizados, garantindo uma experiência intuitiva e eficiente para os usuários. A aplicação demonstrou ser capaz de promover a interação e o espírito competitivo na comunidade de League of Legends, destacando-se como uma solução inovadora para organizar e fomentar competições amistosas.

Os resultados obtidos evidenciam o potencial do Protáthlima em atender ao público-alvo e contribuir para a comunidade gamer. A aplicação oferece uma base sólida para o gerenciamento de torneios, reforçando a importância de soluções tecnológicas que promovam a interação social e a diversão no ambiente virtual.

Como perspectivas futuras, sugere-se a expansão da plataforma para suportar outros jogos além de League of Legends, a inclusão de novas funcionalidades, como integração com redes sociais para compartilhamento de resultados e aumento da visibilidade de torneios, melhorar a usabilidade do sistema para não jogadores de League of Legends adicionando descrição informativa aos campos.

Dessa forma, o Protáthlima não apenas cumpre seu papel como ferramenta funcional, mas também demonstra como a aplicação de tecnologia pode criar impacto positivo em comunidades de nicho, servindo de inspiração para projetos semelhantes.

REFERÊNCIAS

ACTIVEPLAYER.IO. **League of Legends Live Player Count and Statistics.** 2024. Disponível em: <https://activeplayer.io/league-of-legends/> Acesso em: 12 dez.2024.

BRATTI, Giovana. **A relação do ser humano com a competição.** 27 set. 2022. Open Leaders.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **The Unified Modeling Language User Guide.** Addison-Wesley, 1999.

CHALLENGE.PLACE. 2024. Disponível em: <https://challenge.place> Acesso em: 15 dez. 2024.

CSS. **MOZILLA DEVELOPER NETWORK**, CSS. 2022. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Web/CSS > Acesso em: 19 fev. 2024.

DUCKETT, JON. **HTML and CSS: Design and Build Websites.** Wiley. 2011.

DUNIA GAMES. 2024. Disponível em: <https://duniagames.co.id> Acesso em 15 dez. 2024

ELMASRI, R.; NAVATHE, S. B. **Fundamentals of Database Systems.** 7th ed. Boston: Pearson, 2015.

FLANAGAN, DAVID. **JavaScript: The Definitive Guide.** O'Reilly Media. 2020.

FORTA, BEN. **MariaDB Crash Course**. Addison-Wesley. 2011.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures.** Doctoral dissertation, University of California, Irvine, 2000. Disponível em: <https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

GERHARDT, Tatiana Engel; SILVEIRA, Denise Tolfo (Org.). **Métodos de pesquisa**. Porto Alegre: UFRGS, 2009.

HTML: Linguagem de Marcação de Hipertexto. **MOZILLA DEVELOPER NETWORK**. HTML. 2022. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Web/HTML > Acesso em: 19 fev.2024.

JSON. **Introducing JSON**. Disponível em: https://www.json.org/. Acesso em: 12 dez. 2024.

MDN WEB DOCS. **JavaScript basics**. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn/JavaScript>. Acesso em: 12 dez. 2024.

MEYER, ERIC; WEYL, ESTELLE. **CSS: The Definitive Guide: Web Layout and Presentation.** O'Reilly Media. 2023

MySQL AB. **MySQL Reference Manual**. MySQL AB. 2003.

PHP: Manual do PHP. **PHP DOCUMENTATION GROUP**. PHP. 2024. Disponível em: < https://www.php.net/manual/pt_BR/index.php > Acesso em: 19 nov. 2024.

PRÁ, Matheus Dal. **Mercado de eSports: faturamento, audiência e o cenário no Brasil.** 05 ago. 2021. Ge Globo. Disponível em: <https://ge.globo.com/sc/noticia/o-mercado-de-esports-faturamento-audiencia-e-o-cenario-no-brasil.ghtml>. Acesso em: 11 dez. 2024.

SEBASTIAN, M. **PHPUnit Manual.** Disponível em: https://phpunit.de/manual/current/en/. Acesso em: 12 dez. 2024.

SOMMERVILLE, Ian. **Engenharia de software.** 10. ed. São Paulo, SP: Pearson, 2018.