Protótipo de Aplicação para Gerenciar Competições de Esportes Eletrônicos (GCEE)

Renato V. dos Santos, André C. Da Silva

Grupo de Pesquisa Mobilidade e Novas Tecnologias de Interação
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP)
Campus Hortolândia – SP – Brasil

renato.v@aluno.ifsp.edu.br, andre.constantino@ifsp.edu.br

Abstract. *With the growth and popularization of e-sports, it has become common to hold events in different places, such as universities, schools and even in small and medium-sized events. For the realization of the championships it is necessary to have a good management of the participants, teams and matches. In this way, we saw the opportunity to automate the processes related to the management of a championship, opening space for the development of a Web application prototype that was developed using the TypeScript programming language and the React open-source framework.*

Resumo. *Com o crescimento e popularização dos esportes eletrônicos, se tornou comum a realização de competições em diversos âmbitos, como universidades, escolas e até mesmo em eventos de pequeno e médio porte. Para a realização dos campeonatos é preciso ter um bom gerenciamento dos participantes, equipes e chaves. Desta forma, se viu a oportunidade de automatizar os processos relacionados à gestão de um campeonato, abrindo espaço para o desenvolvimento de um protótipo de uma aplicação Web que foi desenvolvido utilizando a linguagem de programação TypeScript(TS) juntamente com a biblioteca de código aberto React.*

1. Introdução

Durante o evento “Dia do Nerd”, organizado pelo Instituto Federal de São Paulo campus Hortolândia, ocorreram diversos campeonatos de esportes eletrônicos, onde a maior parte do gerenciamento das competições foi realizada manualmente como o chaveamento realizado pelo método de sorteio, a inscrição e o *check-in* das equipes feito no momento da competição. Essas ações demandam tempo e esforço por parte da equipe. Também foi notado que grande parte das equipes careciam de um ou mais membros para participar da competição, ocasionando assim em um momento de grande confusão onde as equipes saíram em busca de novos membros.

Percebeu-se a necessidade de um aplicativo que possibilitasse a organização automatizada do chaveamento e balanceamento das equipes presentes na competição, bem como o cadastro prévio das equipes, uma lista de competidores disponíveis para preencher as vagas remanescentes das equipes cadastradas e o *check-in* realizado pelo competidor de forma autônoma. O uso de um *software* com estas funcionalidades poderia diminuir o esforço no processo de organização e realização de uma competição relacionada a esportes eletrônicos.

O objetivo deste trabalho é desenvolver um protótipo de aplicação que possibilite o gerenciamento de um torneio de esportes eletrônicos, com o cadastro prévio das equipes, a busca por jogadores para preencher as vagas disponíveis nas equipes incompletas, o chaveamento automático com o balanceamento das partidas e o *check-in* realizado pelos jogadores de forma autônoma.

Este artigo aborda o referencial teórico na Seção 2, trazendo informações sobre engenharia *web* e o Scrum; em seguida aborda na Seção 3 os trabalhos correlatos e a criação da tabela de funcionalidades desejáveis para a aplicação. Na Seção 4 é apresentada a metodologia utilizada para a realização do projeto seguida da Seção 5, onde é relatado o processo de desenvolvimento da aplicação, trazendo o levantamento dos casos de uso e informações de cada *Sprint*. Por fim, o artigo é finalizado na Seção 6 com a conclusão e os trabalhos futuros.

2. Referencial teórico

Nesta Seção é abordado a Engenharia *Web* e a metodologia de desenvolvimento ágil Scrum, pois subsidiam o processo de desenvolvimento seguido para desenvolver a aplicação *Web* proposta.

2.1. Engenharia Web

A Engenharia *Web* pode ser definida como o uso de princípios, conceitos e métodos da Engenharia de *Software*, adaptando estes pontos para as características de uma aplicação *Web* (BEDER, 2012). Apesar de serem semelhantes, a Engenharia *Web* se diferencia da Engenharia de *Software* pela necessidade de implementar novas metodologias e conceitos que são diretamente ligados a uma aplicação *Web*, os quais se diferenciam de uma aplicação convencional em relação a desempenho, disponibilidade, evolução contínua, urgência e segurança, que se tornam características inerentes às aplicações *Web* (BEDER, 2012).

Com o passar do tempo, as aplicações *Web* se tornaram ferramentas de computação sofisticadas, o que acarretou em um levantamento das diversas características e definições de categorias para tais aplicações, diferenciando assim de forma mais clara uma aplicação convencional em relação a uma aplicação *Web* (PRESSMAN, 2016). As principais características que estão diretamente ligadas a uma aplicação *Web* segundo Pressman (2016), de forma resumida, são: o uso intensivo de redes, a concorrência dos usuários, a carga de usuários não previsível, o desempenho da aplicação, a orientação a dados, a sensibilidade no conteúdo disponibilizado, a evolução contínua, o imediatismo para disponibilizar a aplicação, a segurança e a estética.

A partir de tais características pode-se concluir que, para o desenvolvimento de uma aplicação *Web*, a utilização de uma metodologia ágil é a melhor opção. A metodologia ágil traz uma entrega rápida e contínua, possibilitando assim atender o imediatismo da disponibilização da aplicação Web e de incrementos frequentes, se ajustando de acordo com as regras de negócio e especificações do cliente. Ressalta-se que uma aplicação *Web* evolui com o tempo, alterando seus requisitos devido a incertezas sobre funcionalidades e também a novas demandas que surgem com o passar do tempo (BEDER, 2012).

2.2. Scrum

Scrum é um *framework* que possibilita o desenvolvimento ágil de um projeto. O mesmo é fundamentado na teoria de controle de processos empíricos, focando assim na tomada de decisões baseadas na experiência obtida no passado e emprega uma abordagem iterativa e incremental para otimizar a previsibilidade e controlar riscos (SCHWABER; SUTHERLAND, 2020).

Dentro do *framework* Scrum existem três pilares, sendo eles: transparência, inspeção e adaptação. Esses pilares estão diretamente ligados aos valores do Scrum: compromisso, foco,

abertura, respeito e coragem. Os valores do Scrum orientam no processo de tomada de decisões. Eles se referem diretamente ao comportamento de seus participantes, onde os desenvolvedores devem ter o foco no seu trabalho, o compromisso com a entrega, a abertura com o próprio time e o cliente e a coragem para enfrentar os desafios que irão surgir (SCHWABER; SUTHERLAND, 2020). Além dos valores, o *framework* Scrum conta com algumas definições necessárias para sua metodologia, sendo elas o *Scrum Team*, os *Eventos Scrum* e os *Scrum Artifacts.*

O *Scrum Team* é composto pelo *Scrum Master*, pessoa responsável por garantir que o time esteja seguindo os valores, as práticas e as regras do Scrum. O *Product Owner* é a pessoa encarregada do *Product Backlog* (será definido mais a frente) e por garantir o valor entregue pelo trabalho desenvolvido. O *Scrum Team,* ou time, é composto por desenvolvedores responsáveis pelo desenvolvimento das funcionalidades que serão entregues (BEDER, 2012).

Os *Eventos Scrum* são eventos de duração fixa que criam uma regularidade no dia a dia do *Scrum Team*. Entre os eventos presentes no Scrum podemos citar a *Sprint*, sendo esta uma espécie de *container* que encapsula os outros eventos e possui a duração de um mês ou menos, sendo constante ao longo do período. Ao final de cada *Sprin*t, um incremento é entregue.

Dentro de uma *Sprint* existem os eventos: (i) *Sprint Planning*, momento onde é planejado o trabalho a ser desenvolvido na *Sprint* que irá ocorrer e como o desenvolvimento deste trabalho será feito; a (ii) *Daily Scrum*, uma reunião curta que ocorre diariamente onde é inspecionado o progresso em direção a meta da *Sprint* e, caso necessário, será ajustado o conteúdo a ser desenvolvido na *Sprint*; a (iii) *Sprint Review*, na qual possui como finalidade checar o resultado da *Sprint* e determinar as adaptações futuras; e a (iv) *Sprint Retrospective*, onde se analisa a *Sprint* que ocorreu e tenta se absorver o máximo de conhecimento para melhorar a qualidade e a eficiência das *Sprints* futuras (SCHWABER; SUTHERLAND, 2020).

Os *Scrum Artifacts* são responsáveis por representar o trabalho ou valor do projeto, trazendo transparência e estabelecendo metas. Nos *Scrum Artifacts* encontramos o *Product Backlog*, o qual corresponde a uma lista ordenada que possui o que é necessário para atingir a meta do produto ou melhorá-lo. Outro *Scrum Artifacts* é o *Sprint Backlog*, que corresponde aos itens do *Product Backlog* que foram selecionados para serem desenvolvidos na *Sprint* juntamente com a definição de um plano de ação para realizar a entrega dos itens. E o *Incremento*, que correspondente ao item entregue, que será testado para garantir que o mesmo funcione perfeitamente bem com os incrementos posteriores (SCHWABER; SUTHERLAND, 2020).

Estas definições moldam o *framework* Scrum deixando diversos espaços para adaptações, pois este *framework* não foi construído para estabelecer uma receita a ser seguida e sim dar uma base para construir um modelo Scrum, como cita Jeff Sutherland (2020): “O Scrum é construído sobre a inteligência coletiva das pessoas que o utilizam. Em vez de fornecer às pessoas instruções detalhadas, as regras do Guia do Scrum orientam seus relacionamentos e interações”.

3. Trabalhos correlatos

Nesta Seção é abordado aplicações que possuem o propósito de gestão de um campeonato, com o objetivo de levantar funcionalidades para a aplicação em desenvolvimento neste trabalho.

3.1. AKCL

AKCL – Sistema de Gerenciamento e Controle de Alunos (SILVA, 2017) trata-se de uma aplicação na qual os principais objetivos são facilitar o gerenciamento de alunos de uma academia de caratê e a inscrição dos alunos em campeonatos. O sistema possibilita que o aluno realize a auto inscrição nas competições de seu interesse através de um botão "inscreva-se" acompanhado de uma breve descrição dos dados da disputa, como pode ser observado na Figura 1. Ao ser acionado, o botão registra o competidor de forma individual na disputa em questão, facilitando assim para os gestores do torneio o momento de cadastro dos competidores no campeonato.

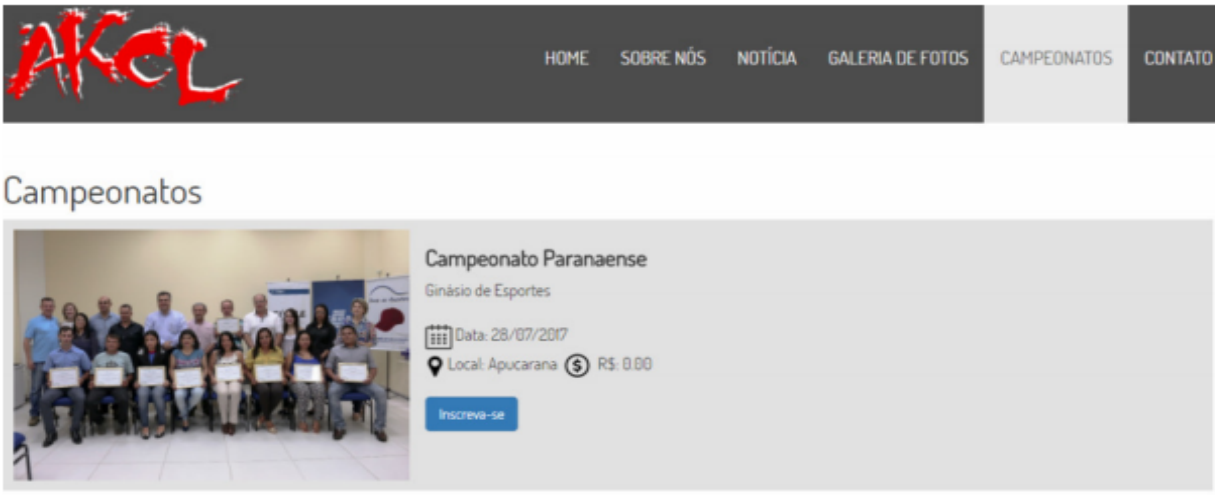

Figura 1. Tela de exibição de campeonatos da aplicação AKCL. Fonte: SILVA (2017).

3.2. Aplicação Web para Gerenciamento de Campeonatos de Futebol

Aplicação *Web* para Gerenciamento de Campeonatos de Futebol (FERREIRA, 2017) é voltada para o gerenciamento de campeonatos de futebol, com o intuito de possibilitar a geração de tabelas, estruturas de rankings do campeonato e exibir os dados para que os usuários possam verificar suas posições nos mesmos.

Na Figura 2, pode-se observar uma tela na qual são listados os campeonatos cadastrados até o momento, com as possibilidades de edição e exclusão dos mesmos. Além disso, a aplicação também possui uma tela na qual é possível visualizar um relatório da partida atual e gerir, de forma manual, os dados relacionados a mesma, como o número de gols de cada equipe, jogadores que marcaram os gols e os cartões amarelos ou vermelhos dos jogadores na partida (Figura 3). A aplicação também exibe chaveamento das partidas de forma textual (Figura 4 e Figura 5).

Na Figura 3, podemos observar todos os campos que o organizador da disputa pode alterar conforme o decorrer da partida. Na Figura 4 possuímos a tela de partidas realizadas com o resultado e as informações da partida como data e horário. Na Figura 5, possuímos os dados das próximas partidas, compondo assim, em conjunto com a tela presente na Figura 4, um chaveamento de forma textual da competição.

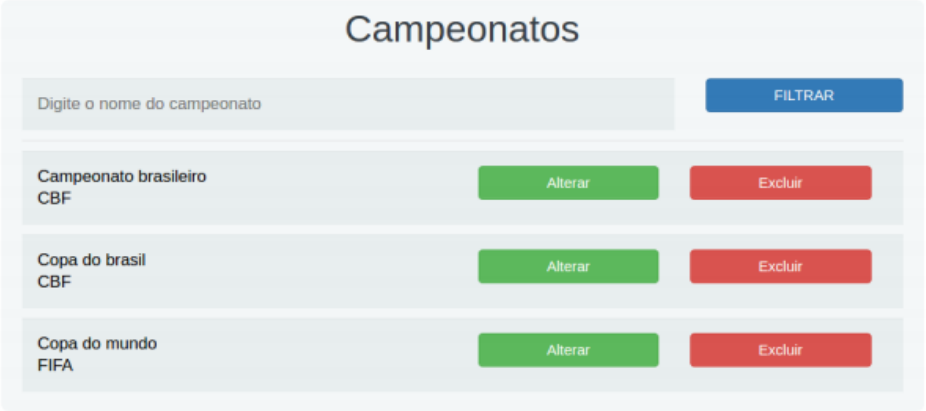

Figura 2. Tela de exibição de campeonatos, Aplicação Web para Gerenciamento de Campeonatos de Futebol. Fonte: FERREIRA (2017).

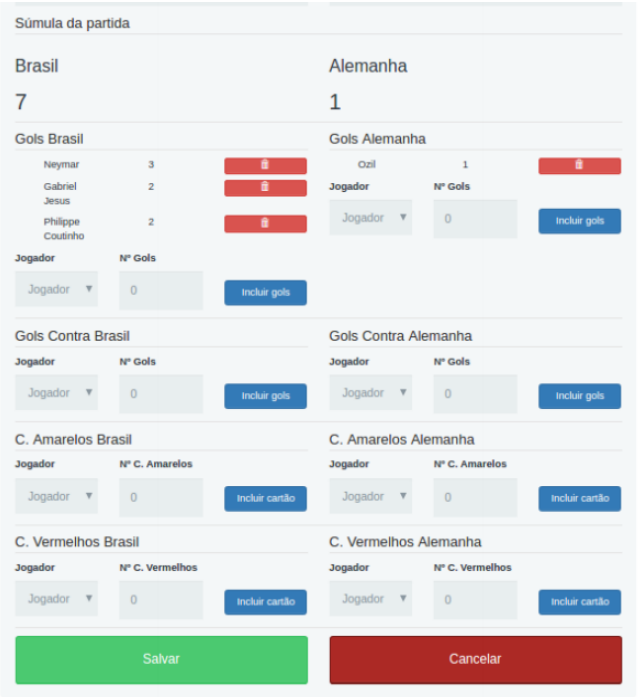

Figura 3. Tela de cadastro da partida, Aplicação Web para Gerenciamento de Campeonatos de Futebol. Fonte: FERREIRA (2017)

Figura 4. Tela de partidas realizadas, Aplicação Web para Gerenciamento de Campeonatos de Futebol. Fonte: FERREIRA (2017)

Figura 5. Tela de próximas partidas, Aplicação Web para Gerenciamento de Campeonatos de Futebol. Fonte: FERREIRA (2017)

3.3. Battlefy

A aplicação *Web* Battlefy fornece um sistema para o gerenciamento de campeonatos de esportes eletrônicos. A ferramenta possui um chaveamento visual (Figura 6), uma lista de equipes participantes e de competidores sortidos (Figura 7), estatísticas, mídias e um resumo do formato do campeonato. No entanto, o chaveamento da disputa não leva em conta nenhum parâmetro para realizar o balanceamento da competição, sendo apenas um sorteio automatizado dos nomes das equipes, sendo distribuídos pela chave de forma visual.

Figura 6. Tabela de chaveamento, aplicação Web Battlefy. Fonte: Battlefy

Da visão de um jogador, a ferramenta também traz a função de inscrição em campeonatos, após acessar os detalhes da competição, tanto como um agente livre para se juntar a uma equipe quanto uma equipe completa, como demonstrado na Figura 8.

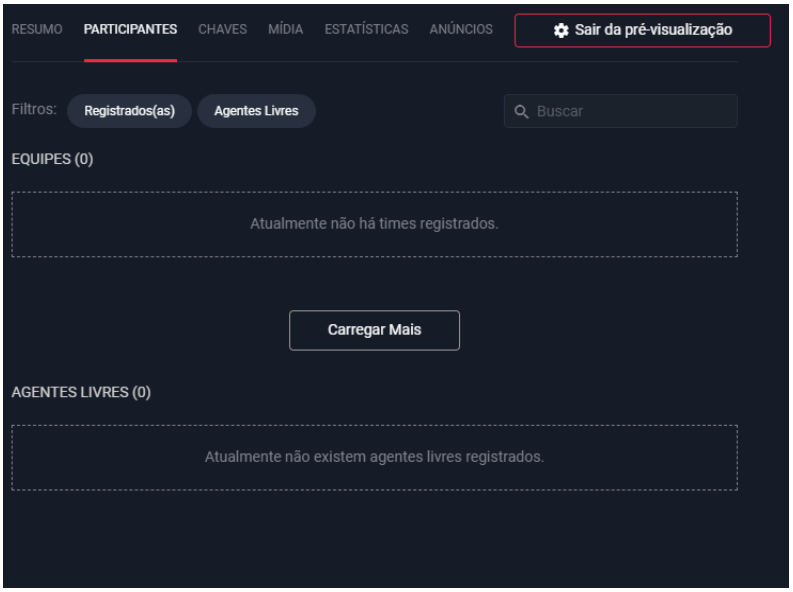

Figura 7. Tela de participantes, aplicação Web Battlefy. Fonte: Battlefy

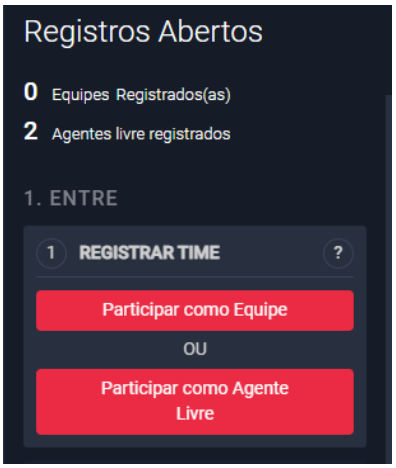

Figura 8. Funcionalidade de inscrição do competidor ou sua equipe, aplicação Web Battlefy. Fonte: Battlefy

3.4. Funcionalidades dos trabalhos correlatos

Após a pesquisa por *softwares* semelhantes, foi realizado o levantamento das principais funcionalidades dos trabalhos correlatos, apresentadas na Tabela 1, para auxiliar no levantamento dos requisitos que estarão presentes na aplicação, sendo todas as funcionalidades citadas desejáveis para o Gerenciador de Competições para Esportes Eletrônicos(GCEE).

4. Metodologia

O projeto empregou a metodologia ágil Scrum e, para a utilização do *framework*, o papel de *Product Owner* e *Scrum Master* foi preenchido pelo professor do Instituto Federal de São Paulo campus Hortolândia, André Constantino da Silva, o qual participou como organizador do evento “Dia do Nerd”. O papel de desenvolvedor foi preenchido pelo aluno do curso de Análise e Desenvolvimento de Sistemas Renato Vinicius dos Santos, desenvolvedor deste TCC.

Tabela 1. Funcionalidades dos trabalhos correlatos

Funcionalidade	AKCL	Aplicação Web para Gerenciamento de Campeonatos de Futebol	Battlefly	GCEE
Permite ao jogador se inscrever na competição	X		X	X
Permite a equipe de organização do campeonato validar as inscrições	X	X	X	X
Possui a opção de check-in da equipe na competição			X	X
Cria o chaveamento das partidas de forma automática			X	X
Possui nivelamento no momento do chaveamento das disputas				X
Exibe graficamente o quadro de partidas			X	X
Fornece estatísticas das partidas		X	X	X
Fornece estatísticas do campeonato		X	X	X
Possui uma lista de jogadores sortidos			X	X
Possibilita que jogadores se inscrevam como agentes livres			X	X

O *Product Backlog* foi definido em conjunto com os organizadores do evento "Dia do Nerd", fazendo uso dos conhecimentos adquiridos por meio dos trabalhos correlatos e as experiências vivenciadas no evento.

Definiu-se o projeto com uma duração correspondente a três *Sprints* de um mês cada, as quais possuíram o *Sprint Backlog* definido de acordo com o andamento do projeto durante uma *Sprint Planning*. Ao final de cada *Sprint* foi feito o *Sprint Review*, com o intuito de validar a entrega juntamente ao *Product Owner*, e uma *Sprint Retrospective* por meio de um relatório sobre a *Sprint,* possuindo os pontos positivos e negativos para que assim a qualidade de desenvolvimento fosse evoluindo durante as *Sprints* e para decidir os passos que foram tomados nas *Sprints* seguintes.

Após as definições da metodologia, foi feita uma pesquisa buscando referenciais teóricos e *softwares* semelhantes com o intuito de realizar o levantamento dos requisitos da aplicação e elaborar as histórias de usuário. Em seguida, foi elaborado o diagrama de caso de uso, permitindo assim que fosse feito o levantamento dos requisitos para o desenvolvimento da aplicação. Essa foi uma fase importante no desenvolvimento de *software*, onde os requisitos levantados possuem o papel de definir as necessidades para o desenvolvimento da aplicação em questão, o que permitiu compreender como o *software* pode atender aos objetivos de negócio e a qualidade (SOMMERVILLE, 2011).

Em seguida, foi definido o *Product Backlog* e então desenvolvido um protótipo do *front-end* por meio da ferramenta Figma, trazendo desta maneira uma ideia das principais telas da aplicação, que foram validadas com um dos organizadores do evento "Dia do Nerd".

Com o protótipo do *front-end* em mãos e o *Product Backlog* definido, a primeira *Sprint Planning* foi realizada, onde definiu-se o *Sprint Backlog* e estabeleceu os itens do *Product Backlog* que foram desenvolvidos no primeiro ciclo, além de trazer as ações para que os objetivos da *Sprint* fossem alcançados. Ao fechamento do ciclo, como citado, foi realizado o *Sprint Review* e *Sprint Retrospective*, o que abriu então o caminho para os demais ciclos, os quais repetiram os processos da primeira *Sprint*.

No desenvolvimento do *front-end* foi utilizada a tecnologia ReactTS, que une no *front-end* as linguagens TypeScript, HTML e CSS, "*react-router-dom*", possibilitando a navegação entre as páginas de forma mais fluida, Axios, auxiliando nas requisições para o *back-end,* e a biblioteca MaterialUI, trazendo um conjunto de ícones e componentes semi-prontos. No *back-end* foi utilizada a biblioteca Express, que é responsável por receber as requisições do *front-end*, Mongoose, que traz o vínculo do back-end com o MongoDB e a biblioteca Bcryptjs, que traz as funções para encriptação, auxiliando assim na segurança do código. Como banco de dados, foi definido o uso do MongoDB, visto que o modelo não relacional traz uma boa eficiência e compatibilidade com aplicações *Web*.

5. Desenvolvimento do trabalho

Nesta Seção são detalhadas as atividades realizadas em relação ao levantamento de requisitos e codificação da aplicação proposta.

5.1. Levantamento de Requisitos

Para início do desenvolvimento do trabalho, foi elaborado um diagrama de caso de uso, baseado nas funcionalidades presentes na tabela de trabalhos correlatos (Tabela 1). Foram identificados quatro atores para o sistema, sendo eles: usuário, jogador, administrador do campeonato e tempo.

O ator usuário, como apresenta a Figura 9, conta com as funcionalidades: acompanhar chave, acompanhar campeonatos, realizar login, cadastrar conta e associar uma conta de jogo que estende o cadastro de conta.

O ator jogador, Figura 10, interage no sistema nos seguintes casos de uso: consultar histórico de partidas, associar uma conta de jogo, criar campeonato ingressar em campeonato, que por sua vez inclui ingressar com equipe, permitindo também ingressar como agente livre, criar equipe, que possibilita enviar *e-mail* de convite e por último a funcionalidade de ingressar em equipe.

O ator administrador de campeonato, Figura 11, interage no sistema nos seguintes casos de uso: manter o campeonato, registrar resultado de partida e manter equipes do campeonato.

O último ator é o tempo, Figura 12, que interage no sistema nos seguintes casos de uso: realizar sorteio das chaves, que inclui enviar e-mails do chaveamento, e sobre partida do dia seguinte.

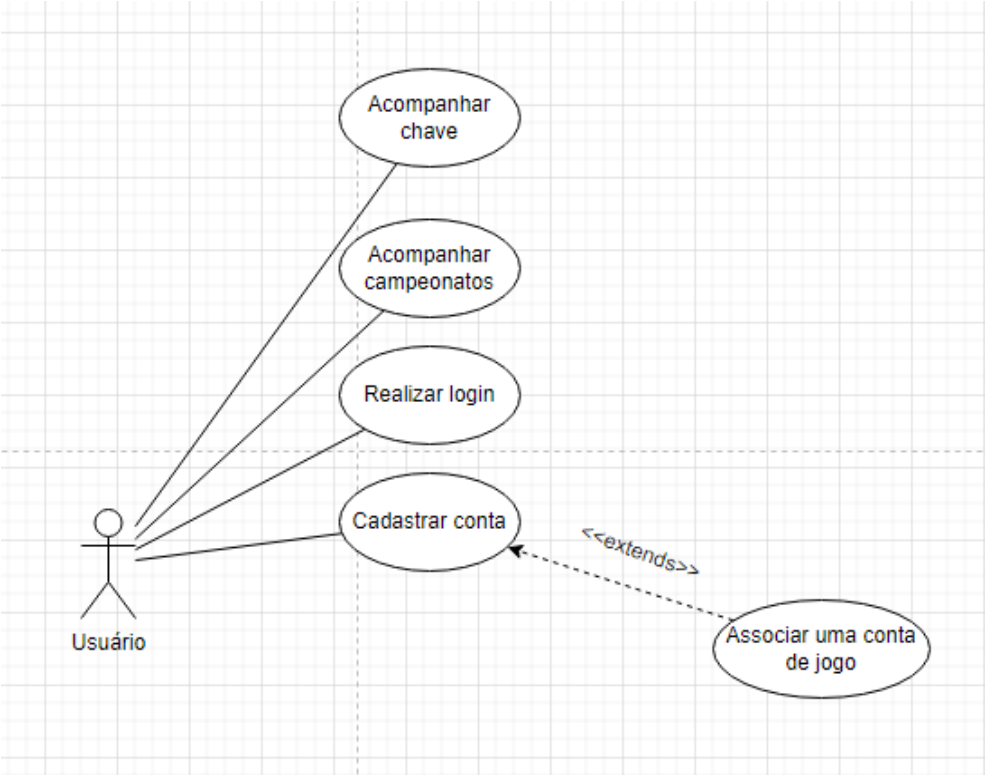

Figura 9. Diagrama de caso de uso na visão do usuário

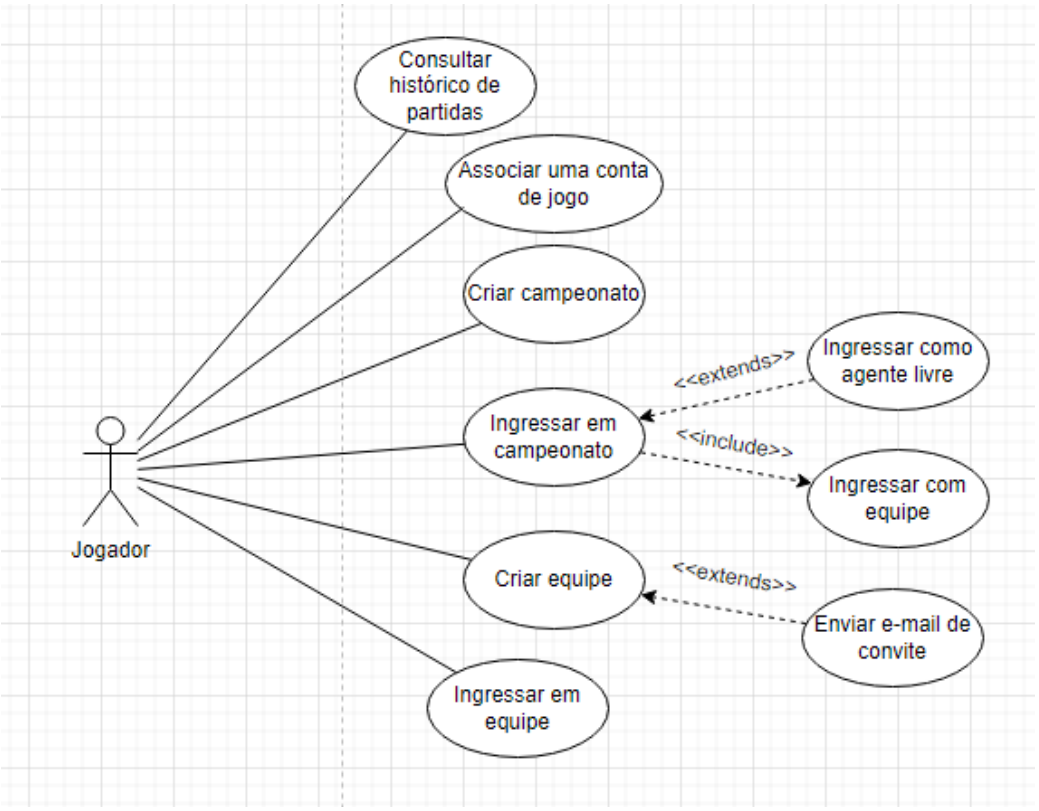

Figura 10. Diagrama de caso de uso do jogador

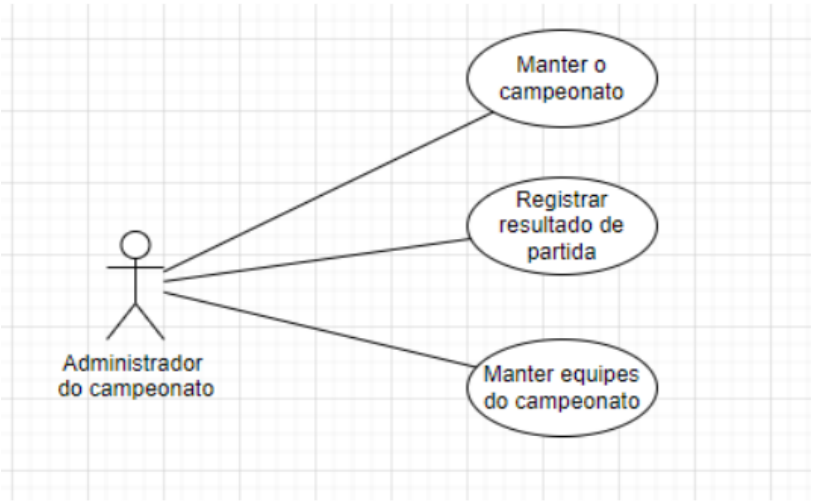

Figura 11. Diagrama de caso de uso do administrador do campeonato

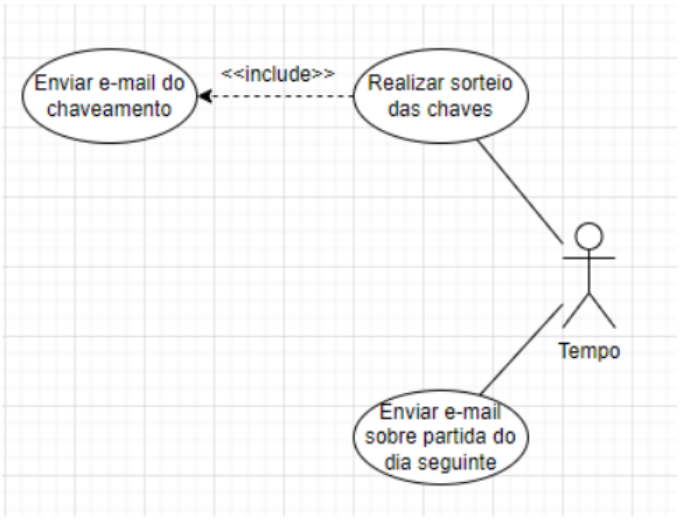

Figura 12. Diagrama de caso de uso do tempo

5.2. Prototipação da interface gráfica

Ao se iniciar o processo de elaboração da interface gráfica, foi realizado esboços no papel das telas principais, como apresenta a Figura 13, que traz o *layout* da tela principal do sistema gerenciador de campeonatos. A mesma possui um logo, um componente com informações do usuário (marcado com o número 1 na Figura 13), um menu (marcado com o número 2 na Figura 13) e uma grade (marcado com o número 3 na Figura 13), esta última irá possuir o conteúdo exibido de acordo com a opção selecionada no menu, começando na tela de competições.

Na Figura 14 são apresentados os elementos que irão compor a página principal, sendo eles o *drop down* (marcado com o número 1 na Figura 14), o menu lateral (marcado com o número 2 na Figura 14) e o *card* de campeonato (marcado com o número 3 na Figura 14). É importante salientar que estas imagens trazem apenas um esboço das ideias elaboradas para a interface, sendo assim podem ocorrer alterações durante a elaboração do protótipo digital e também durante o desenvolvimento da aplicação. A numeração presente na Figura 14

corresponde aos elementos presentes na numeração da Figura 13; o *drop down* corresponde ao elemento de número 1 na Figura 13; o menu corresponde ao elemento de número 2 na Figura 13 e o *card* de campeonato corresponde ao número 3 da Figura 13.

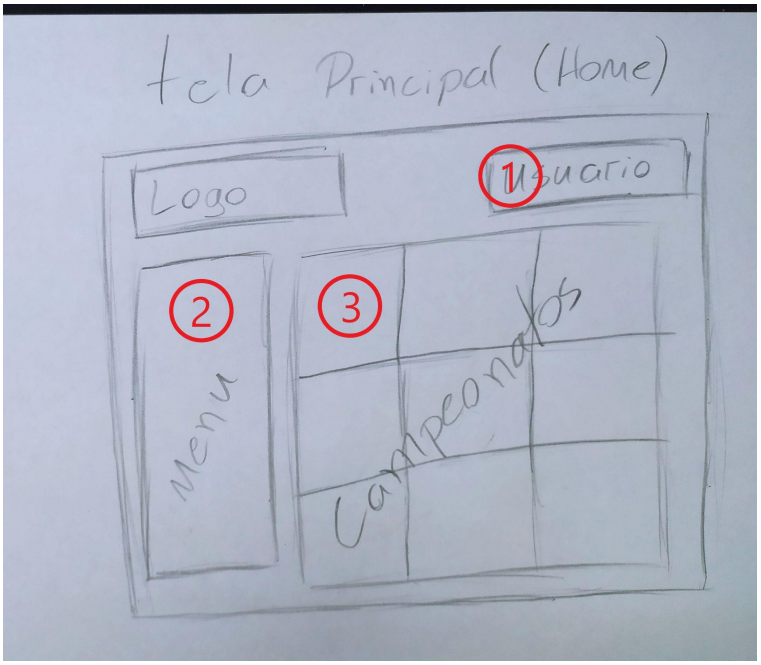

Figura 13. Esboço feito no papel para o *layout* do sistema gerenciador de campeonatos

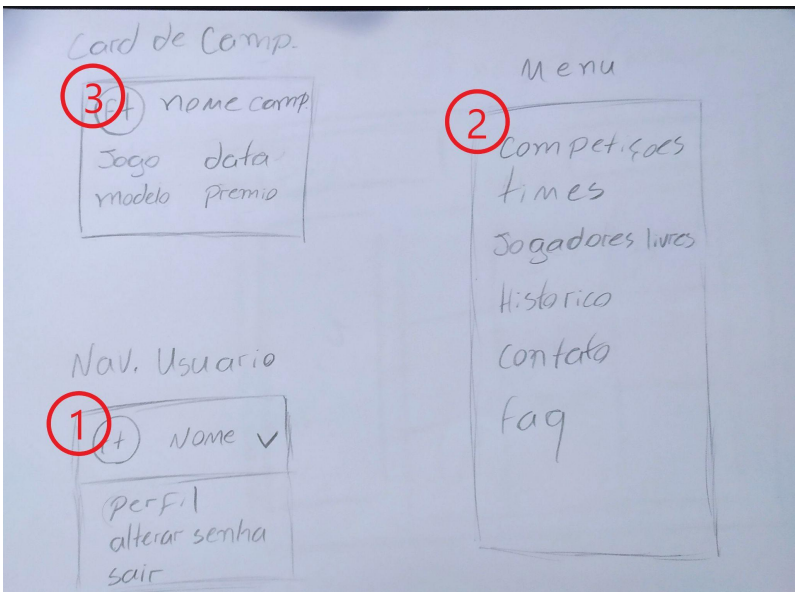

Figura 14. Esboço feito no papel dos componentes que irão estar presentes na tela principal

Após a prototipação das telas em papel, foi realizada uma prototipação digital por meio da ferramenta Figma (Figura 15) a qual traz os principais elementos e também define um *layout* para a aplicação.

Figura 15. Prototipação feita por meio da ferramenta Figma com o *layout* da tela e os principais componentes

5.3. Implementação - *Sprint* 1

Na primeira implementação, foi definido que o *Sprint Backlog* seria o *layout* da tela principal e seus componentes, além do método de consulta de campeonatos.

Com esta definição, foi elaborado o componente *CardCampeonato* (Figura 16), que compõem a grade de campeonatos. Este componente possui alterações de estilo para contemplar a versão do campeonato no qual a equipe está inscrita, atendendo assim os desenhos definidos no protótipo. O *card* é composto pelas informações do campeonato, como o nome, o jogo, o modelo e o prêmio, sendo exibidos por meio das *divs* presentes no componente. Este componente é montado por meio da *div CardActionArea*, que torna toda a região interna do *card* clicável (as linhas 32 a 36 faz a composição de um conjunto de *divs* que encapsulam a variável "campeonato.nome", sendo esta responsável por exibir o nome do campeonato, e este modelo se repete para cada outra variável que será exibida no *card*, tendo como diferencial apenas o estilo presente em cada variável e a alternação entre ícones que ocorre no ternário presente entre as linhas 45 a 51).

Após a elaboração do *card*, foi montado o menu, seguindo o mesmo estilo de codificação do *card* (Figura 17). O menu é encapsulado pelo componente "MenuList", disponibilizado pelo MaterialUI (linha 19 e 39), e dentro desse componente temos a separação dos campos do menu por meio da *tag* "MenuItem", que compõem um item do menu recebendo um estilo e o texto. A *tag* encapsulada é o "*Link*", que é disponibilizado pela biblioteca "react-router-dom", e possibilita a navegação fluida da página ao clicar nas opções do menu, pode-se observar a composição desse conjunto nas linha 20 a 22, que gera a opção "Campeonatos" do menu.

Com os componentes prontos, a tela principal foi montada, exibindo os *cards* de campeonato e o menu de navegação lateral (Figura 18).

Figura 16. Recorte do código referente ao componente *CardCampeonato*

Nessa *Sprint* também foi desenvolvido a primeira interação com o banco de dados MongoDB, sendo esta uma requisição de consulta do *front-end* para o *back-end* por meio da rota respectiva a consulta de campeonatos. Este processo se inicia no serviço presente no *front-end* (Figura 19), que realiza a requisição para a rota no *back-end* por meio de um chamado ao cliente "axios" com o método *get,* passando como parâmetro a URL de requisição, exemplificado nas linhas 7 a 9 (Figura 19), ao receber o chamado. O *back-end* chama a função respectiva a rota, conforme linha 33 da Figura 20.

```jsx
return (
  <div className="Menu">
    <Paper sx={paperStyle} className="menuTeste">
      <MenuList>
        <Link to="/" style={{ textDecoration: "none" }}>
          <MenuItem sx={menuItemStyle}>Campeonatos</MenuItem>
        </Link>
        <hr color="#332C3D" />
        <Link to="/equipes" style={{ textDecoration: "none" }}>
          <MenuItem sx={menuItemStyle}>Equipes Disponiveis</MenuItem>
        </Link>
        <hr color="#332C3D" />
        <Link to="/agentesLivres" style={{ textDecoration: "none" }}>
          <MenuItem sx={menuItemStyle}>Agentes Disponiveis</MenuItem>
        </Link>
        <hr color="#332C3D" />
        <Link to="/faq" style={{ textDecoration: "none" }}>
          <MenuItem sx={menuItemStyle}>FAQ</MenuItem>
        </Link>
        <hr color="#332C3D" />
        <Link to="/contatos" style={{ textDecoration: "none" }}>
          <MenuItem sx={menuItemStyle}>Contato</MenuItem>
        </Link>
      </MenuList>
    </Paper>
  </div>
);
```

Figura 17. Recorte do código referente ao componente *Menu*

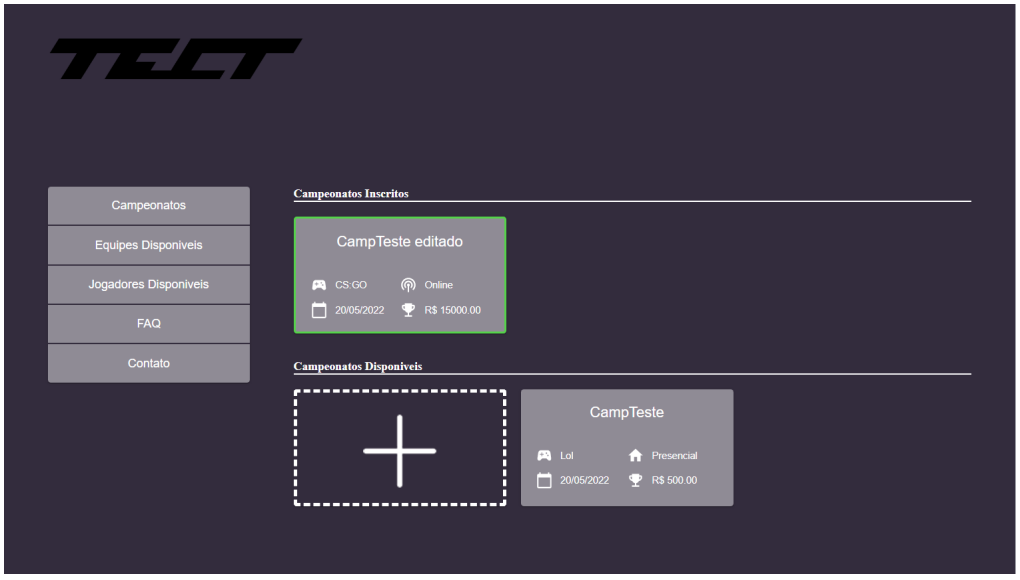

Figura 18. Tela principal da aplicação desenvolvida na *Sprint* 1

```jsx
export const buscaCampeonato = async (
  campeonato: string
): Promise<Campeonato> => {
  const { data } = await axios.get(
    `http://localhost:3333/campeonato/${campeonato}`
  );
  return data;
};
```

Figura 19. Recorte do código referente ao serviço buscaCampeonato

```
31   //Sprint 1
32   routes.get("/", todosCampeonatos);
33   routes.get("/campeonato/:idCampeonato", buscaCampeonato);
```

Figura 20. Recorte do código referente às rotas elaboradas na *Sprint* 1

Ao receber a requisição, a função correspondente a rota que o *front-end* solicitou, (Figura 21), busca pelo parâmetro "idCampeonato" (linha 17) que está presente na URL e executa a busca das informação do campeonato solicitado no banco de dados por meio do comando *findById* da linha 18, retornado os informações (linha 24) ou um erro tratado caso a requisição falhe (linha 21).

Com a finalização da implementação do *back-end*, a *Sprint* 1 chegou ao fim, dando início a um processo de revisão, realizado por meio de uma reunião com o orientador e *product owner*, André Constantino da Silva, onde foi apresentada a tela principal com alguns campeonatos exibidos que já estavam cadastrados no banco de dados. Houve um *feedback* positivo do protótipo e do início das implementações.

```
15  export const buscaCampeonato = async (req: Request, res: Response) => {
16    try {
17      const { idCampeonato } = req.params;
18      const campeonato = await Campeonato.findById(idCampeonato);
19
20      if (!campeonato) {
21        return res.status(404).json();
22      }
23
24      return res.json(campeonato);
25    } catch (err) {
26      console.error(err);
27      return res.status(500).json({ error: "Internal server error." });
28    }
29  };
```

Figura 21. Recorte do código referente a função buscaCampeonato

5.4. Implementação - *Sprint* 2

Para o início do desenvolvimento da *Sprint* 2, foi definido como *Sprint backlog*, as funcionalidades correspondentes a "Cadastrar conta", "Realizar login", "Cadastrar campeonato" e "Manter campeonato".

No primeiro momento, foi elaborado o *card* de criação de campeonato, que possuiu como base o componente *CardCampeonato*, alterando apenas os valores internos, para que o mesmo não seja preenchido pelas cores e possua um ícone centralizado. Após o desenvolvimento do *card*, foi elaborada a tela de cadastro de campeonato (Figura 22), que por sua vez é composta por um formulário com os campos nome, jogo, data, formato, prémio e descrição, além de um botão de cadastro que envia esses dados para um serviço que solicita a rota de cadastro de campeonatos no *back-end*, sendo esta muito parecida com a rota de consulta, trocando apenas o comando utilizado para *create* passando os dados como parâmetro.

Além da tela de cadastro de campeonatos, foi elaborada a tela de cadastro de usuário, *login* e edição de campeonato, todas com suas respectivas rotas e funções para interagirem com o banco de dados.

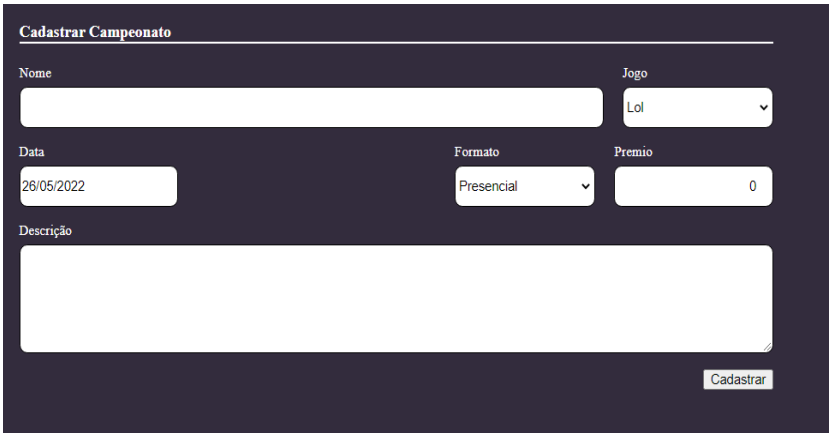

Figura 22. Parte da tela respectiva ao cadastro de um campeonato

Nesta interação, também foi desenvolvido um mecanismo de autenticação, responsável por manter o *login* do usuário e limitar as rotas e telas acessíveis caso o usuário ainda não tenha realizado o *login*. Este mecanismo é composto pelo controle de sessão (Figura 23), que irá consultar o usuário que está tentando realizar o *login* por meio do comando "*findOne*" (linha 9) e, caso encontre esse usuário, vai solicitar a geração de um token na linha 23 por meio da função "*sign*" da biblioteca "jwt" e então retornar essas informações para o *front-end*.

Após realizar o *login* e possuir o *token*, em todas as requisições que o *front-end* realizar, será enviado como *header* para o *back-end* o *token* respectivo ao *login* do usuário, que irá validar e permitir ou negar o acesso a rota por meio da função de autenticação (Figura 24), a qual verifica se o *token* foi enviado (linha 8) e, caso tenha sido enviado, o valida por meio da biblioteca "jwt" com a função "*verify*" (linha 13).

Com a contemplação do requisito não funcional de segurança, a *Sprint* 2 foi finalizada, novamente dando início a um processo de validação com o *Product Owner*. A print foi encerrada com um *feedback* positivo e com destaque para o requisito não funcional de segurança, mas com um chamado de atenção para certos pontos como o tamanho da fonte, o prazo para o encerramento do projeto e a aparência do protótipo.

```
6   export const sessionController = async (req: Request, res: Response) => {
7     const { email, senha } = req.body;
8
9     const user = await Usuario.findOne({ email });
10
11    if (!user) {
12      return res.status(401).json({ error: "Usuario ou senha invalidos" });
13    }
14
15    if (!checkPassword(user, senha)) {
16      return res.status(401).json({ error: "Usuario ou senha invalidos" });
17    }
18
19    const { id } = user;
20
21    return res.json({
22      user: user,
23      token: jwt.sign({ id }, "secret", {
24        expiresIn: "7d",
25      }),
26    });
27  };
```

Figura 23. Código correspondente a função *sessionController*

```
export const auth = async (req: any, res: Response, next: any) => {
  const authHeader = req.headers.authorization;

  if (!authHeader) {
    return res.status(401).json({ error: "Token nao foi fornecido." });
  }
  const [, token] = authHeader.split(" ");

  jwt.verify(token, "secret", (err: any, decoded: any) => {
    if (err) {
      return res.status(401).json({ error: "Token invalido" });
    } else {
      req.userId = decoded.id;
      return next();
    }
  });
};
```

Figura 24. Código correspondente a função *auth*

5.5. Implementação - *Sprint* 3

A terceira e última *Sprint* teve como escopo os casos de uso “Criar equipe”, “Ingressar em equipe”, “Ingressar em campeonato”, “Ingressar como agente livre”, “Realizar sorteio das chaves” e “Acompanhar chave”.

No primeiro momento foi desenvolvido a tela de exibição de campeonatos e exibição de agentes livres, seguindo o mesmo modelo da exibição dos campeonatos, sendo uma grade composta por cards dos respectivos conteúdos. Em seguida, foi elaborada a tela de cadastro de equipe seguindo o modelo da tela de cadastro de campeonato.

Após esta fase inicial, foi desenvolvido a tela que contém os detalhes do campeonato (Figura 25), que é responsável por exibir as informações do campeonato selecionado. Baseado nesta, foi elaborada a tela contendo os detalhes da equipe e a tela dos detalhes do usuário.

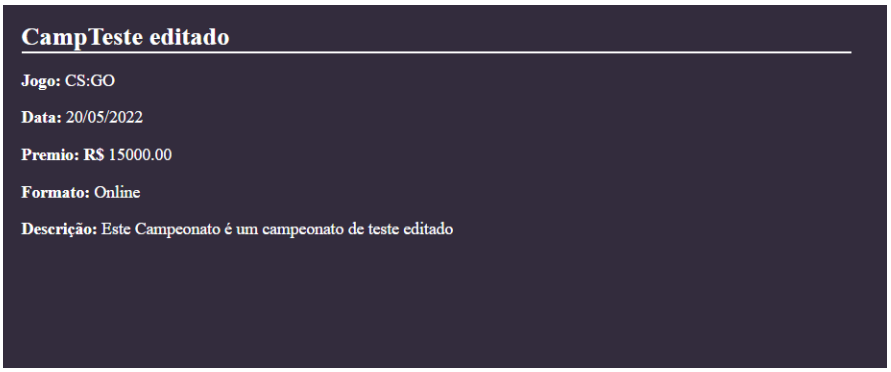

Figura 25. Parte da tela contendo os detalhes de um campeonato

As telas que contém os detalhes dos respectivos conteúdos possuem elementos que variam de acordo com o contexto no qual o usuário se encontra. No caso dos campeonatos temos o botão de inscrever, que será exibido caso o usuário tenha efetuado o *login* em uma conta que possua uma equipe vinculada (botão indicado com o número 1 na Figura 26). Os botões de editar e excluir (indicados com o número 2 na Figura 26), que só serão exibidos caso o usuário seja o criador do campeonato, e são responsáveis por excluir ou possibilitar a edição do campeonato.

Seguindo o molde da tela de detalhes do campeonato, a tela de detalhes da equipe traz o botão de inscrever, que será exibido caso o usuário possua uma conta de jogo e tem a função de cadastrar o usuário em uma lista de jogadores que querem participar da equipe. Os botões aceitar e recusar, que para serem exibidos tem a condição da equipe ter solicitado pelo usuário que está acessando seus detalhes, possibilitando assim a escolha do usuário de aceitar ou não uma solicitação feita por uma equipe.

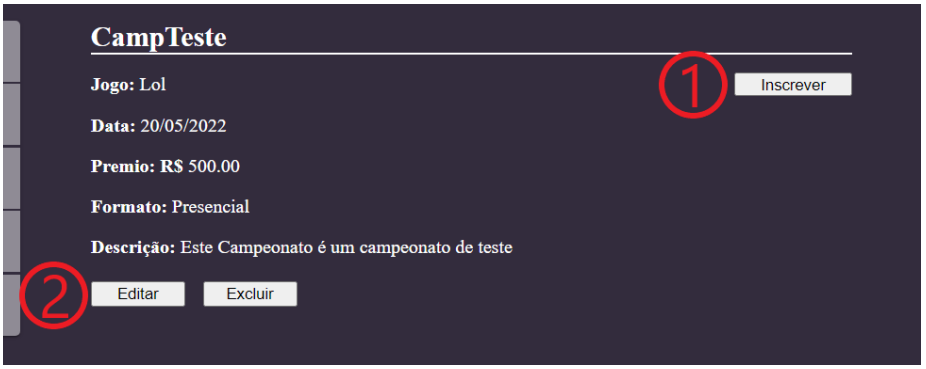

Figura 26. Parte da tela contendo os detalhes de um campeonato

A tela de exibição dos detalhes do usuário possui a mesma ideia da tela de equipes, porém a função do botão de inscrever é adicionar o usuário ao grupo de interesse da equipe a qual está acessando os detalhes deste usuário. Os botões aceitar e recusar são para possibilitar o aceite de uma requisição feita pelo usuário para ingressar na equipe.

O chaveamento é iniciado e exibido na página de detalhes do campeonato (Figura 27), sendo possível realizar o chaveamento apenas se o usuário for o criador do campeonato. O botão de chaveamento corresponde ao botão indicado pelo número 1 na Figura 27 e o chaveamento é representado por *cards* indicados com o número 2 na Figura 27, sendo esses interativos. Quando clicado em um deles, torna a equipe correspondente ao *card* a equipe vencedora da disputa entre os times, alterando o estilo do *card* para possuir uma borda verde.

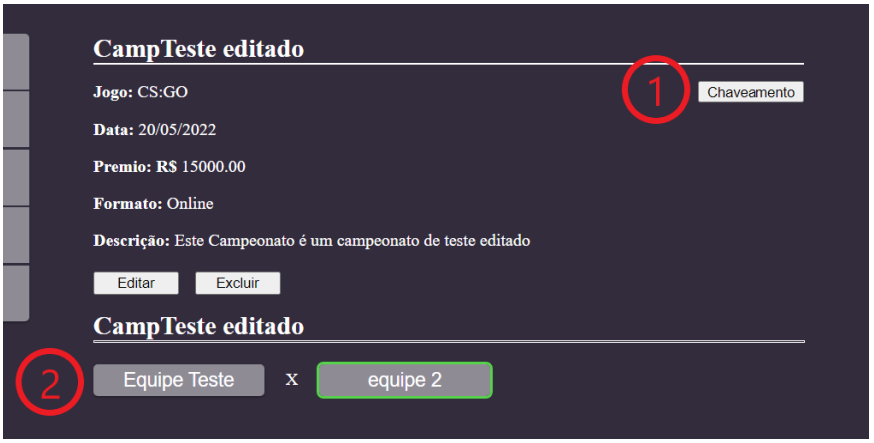

Figura 27. Parte da tela contendo os detalhes de um campeonato e chaveamento das equipes.

Com a finalização do desenvolvimento do chaveamento das disputas, a *Sprint* 3 chegou ao fim, encerrando assim o ciclo de desenvolvimento e realizando a última revisão com o cliente. Novamente houve um *feedback* positivo, porém foi ressaltado a falta das funcionalidades ligadas ao ator tempo e o *design* visual simples do protótipo.

6. Conclusão

Durante o evento “Dia do Nerd”, ocorrido no Instituto Federal de São Paulo em 2019, foi percebida a necessidade de uma aplicação que fosse capaz de auxiliar na gestão dos campeonatos realizados no evento. Dentre as necessidades identificadas, as principais foram a inscrição dos competidores de forma autônoma, o *check-in*, o chaveamento das equipes e a possibilidade de buscar por competidores que ainda não estão vinculados a nenhuma equipe. Identificando-se essas necessidades, este trabalho teve como objetivo desenvolver um protótipo de uma aplicação que possibilite o gerenciamento de campeonatos de esportes eletrônicos, auxiliando tanto o lado da gestão quanto dos jogadores.

Como metodologia de desenvolvimento, optou-se por iniciar com o levantamento do referencial teórico, onde se observou que, para o desenvolvimento de uma aplicação *Web*, um modelo ágil seria a melhor escolha para realizar o projeto, visto que uma aplicação Web requer a possibilidade de incrementos e agilidade nas entregas. Dentro das metodologias ágeis foi definido, para o projeto, o uso da metodologia Scrum, a qual possibilita o desenvolvimento por meio de *Sprints* com suas metas definidas conforme o andamento do projeto. Seguindo a metodologia adotada, foi realizado o levantamento do *Product Backlog* em conjunto com os organizadores do evento “Dia do Nerd”. Também foram analisadas funcionalidades de três trabalhos correlatos e assim levantada uma tabela de funcionalidades desejáveis para a aplicação. Identificados os requisitos, expostos por diagramas de casos de uso, iniciou-se a codificação da aplicação em 3 *Sprints*.

Durante *Sprint* 1 foi desenvolvido a tela principal da aplicação e os componentes que estão dispostos nela, como o *card* de campeonato e o menu lateral, além disso também foi desenvolvido a primeira interação entre o *front-end* e o *back-end*, sendo esta a requisição de consulta aos campeonatos cadastrados na base de dados, concluindo assim a *Sprint* 1 com uma estrutura de *front-end*, *back-end* e com um *feedback* positivo com relação às entregas da *Sprint.*

A *Sprint* 2 teve como *sprint backlog* os casos de uso Cadastrar conta, Realizar login, Cadastrar campeonato e Manter campeonato, todos foram implementados com sucesso e além deles, também foi implementado um requisito não funcional vinculado a segurança. A *Sprint* encerrou com o *feedback* positivo e com destaque para o requisito não funcional de segurança, no entanto houve o levantamento de pontos de melhoria referentes a *design*.

A terceira e última *Sprint* teve como *sprint backlog* os seguintes casos de uso: Criar equipe, Ingressar em equipe, ingressar em campeonato, Ingressar como agente livre, Realizar sorteio das chaves e Acompanhar chave. Todos os casos de uso foram implementados. No entanto, na última revisão com o *Product Owner*, foram levantados pontos que foram definidos como trabalhos futuros.

Desta forma, como trabalhos futuros, temos os casos de uso do ator tempo, as funcionalidades referentes a exibição de estatísticas, chaveamento visual, o nivelamento do chaveamento e o *check in*, além disso temos a melhoria da interface de usuário e do chaveamento das partidas.

No desenvolvimento deste projeto, foram aplicados conhecimentos vinculados às disciplinas: Linguagem de Programação no *back-end*, Banco de Dados no MongoDB, Programação *Web* no *front-end* e Engenharia de software na organização e metodologia do desenvolvimento. Além das disciplinas, também foram articulados conhecimentos adquiridos

sobre ReactTS, Axios, Express e Mongoose, tecnologias essas relacionadas ao desenvolvimento *Web* e banco de dados não relacional.

Referências

BEDER, D. M. **Engenharia Web**: Uma Abordagem Sistemática para o Desenvolvimento de Aplicações Web. 1 ed. São Carlos: EdUFSCar, 2012.

CRUZ, F. **Scrum e Agile em Projetos Guia Completo**. Rio de Janeiro: Editora Brasport, 2018.

FERREIRA, J. **Aplicação Web para Gerenciamento de Campeonatos de Futebol**. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) - Centro Tecnológico, Universidade Federal de Santa Catarina. Florianópolis, p. 167. 2017. Disponível em: <https://repositorio.ufsc.br/xmlui/handle/123456789/177703>. Acesso em: 26 de jul. 2021.

PRESSMAN, R. S.; MAXIM, B. R. **Engenharia de Software**: uma abordagem profissional. 8 ed. Porto Alegre: AMGH, 2016.

SCHWABER, K; SUTHERLAND, J. The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game. 2020. Disponível em: <https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-PortugueseBR-2.0.pdf> Acesso em: 26 de jul. 2021.

SILVA, N. B. AKCL – Sistema de gerenciamento e controle de alunos. **Revista F@pciência**, v.11, n. 1, p. 01-05, 2017. Disponível em: <http://www.fap.com.br/fap-ciencia/11_edicao/001.pdf>. Acesso em: 26 de jul. 2021.

SOMMERVILLE, I. **Software Engineering**. 9. ed. Boston: Pearson Education, 2011.

REACTJS.ORG. Documentação oficial referente ao ReactJS. Disponível em: https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: 18 jul. 2022.

Documento Digitalizado Público

Anexo I - artigo do TCC

Assunto: Anexo I - artigo do TCC
Assinado por: Andre Constantino
Tipo do Documento: Relatório Externo
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Andre Constantino da Silva**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 07/09/2022 20:40:06.

Este documento foi armazenado no SUAP em 07/09/2022. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1093147
Código de Autenticação: a5bb928fc4