UNIVERSIDADE FEDERAL DE SANTA CATARINA

APLICAÇÃO WEB PARA GERENCIAMENTO DE CAMPEONATOS DE FUTEBOL

JOÃO LUIZ MAFRA FERREIRA

FLORIANÓPOLIS-SC

2017/1

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE SISTEMAS DE INFORMAÇÃO

APLICAÇÃO WEB PARA GERENCIAMENTO DE CAMPEONATOS DE FUTEBOL

JOÃO LUIZ MAFRA FERREIRA

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

FLORIANÓPOLIS-SC

2017/1

JOÃO LUIZ MAFRA FERREIRA

APLICAÇÃO WEB PARA GERENCIAMENTO DE CAMPEONATOS DE FUTEBOL

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador: Leandro José Komosinski

Banca examinadora

Frank Augusto Siqueira

José Eduardo de Lucca

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

UFSC	Universidade Federal de Santa Catarina
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
SGBD	Sistema Gerenciador de Banco de Dados
SQL	Structured Query Language
NoSQL	Not Only SQL
CSS	Cascading Style Sheets
JSON	Javascript Object Notation

Resumo

Para realizar a organização de um campeonato de futebol é necessário muita organização e planejamento. Garantir que o campeonato seja bem organizado e administrado pode ser uma tarefa difícil de se alcançar. Simpatizantes de futebol que desejam pôr em prática a ideia de organizar um evento futebolístico, podem encontrar empecilhos. Perder tempo entre diversas planilhas no Excel e até mesmo desperdícios de folhas de papel montando tabelas, chaves, resultados e rankings são problemas recorrentes de uma pessoa que organiza campeonatos de futebol sem um sistema para tal.

Considerando tal problema, o presente trabalho apresenta uma solução para este problema: uma aplicação web que ajuda o organizador a administrar e gerenciar seus campeonatos de futebol. O sistema busca otimizar o processo de organização e administração de campeonatos através de ferramentas para controle e inserção de dados que simulam a estrutura de um campeonato de futebol. Além disso, o sistema permite que jogadores dos campeonatos e simpatizantes possam acompanhar resultados, tabelas e demais informações dos campeonatos disputados.

Palavras chave: gerenciador de campeonatos, campeonato, futebol, aplicação web.

1 Introdução

Assim como qualquer outro tipo de evento, eventos esportivos necessitam de muita organização e planejamento para acontecer da melhor maneira possível e não ocorrer falhas no seu resultado. Com um campeonato de futebol não é diferente. Tabelas, rankings, partidas, grupos, times e todas as outras informações pertencentes a um campeonato precisam estar bem estruturadas para não haver perda de informações ou informações duplicadas. Muitos organizadores de campeonatos possuem toda essa estrutura apenas em papéis e/ou planilhas no Excel, o que pode não ser o ideal para manter a integridade de informações e também para mostrar as informações do campeonato aos times participantes e demais pessoas que desejam acompanhar a disputa. Manter as disputas apenas em planilhas do Excel pode se tornar muito trabalhoso a medida em que é necessário atualizar os mesmos dados em diferentes planilhas, duplicando as informações.

Além de ser muito mais trabalhoso manter o campeonato apenas em planilhas, o ato de se calcular classificações, saldos de gols e rankings em geral também pode vir a ser um problema, uma vez que esse cálculo pode influenciar no resultado dos campeonatos e precisa ser extremamente correto.

Sendo assim, identificando esta demanda, escolhi desenvolver uma plataforma onde qualquer pessoa ou organização que desejasse planejar um campeonato pudesse fazer de forma mais atual e objetiva, através de uma aplicação web para gerenciamento de campeonatos. Este trabalho portanto, busca desenvolver uma aplicação que possibilitará aos organizadores uma economia de

tempo e folhas de papel, e aos times e jogadores um maior acesso às informações relacionadas ao campeonato, mantendo todas as partes atualizadas e permitindo maior troca de informações entre todas as partes.

Para desenvolver a aplicação foi utilizada uma metodologia baseada no *framework* Scrum, buscando uma maior agilidade no desenvolvimento além de eficácia. Uma adaptação ao Scrum foi necessária, visto que originalmente ao implementá-lo é preciso de diversas pessoas para assumirem os diferentes papéis no desenvolvimento do projeto. O Scrum foi adaptado para uma equipe de um só membro, onde este assume todos os papéis previstos na metodologia ágil proposta.

1.1 Cenário Atual

Hoje em dia muitos organizadores de campeonato perdem muito tempo ao montar planilhas via Excel e muitas vezes de uma maneira atrasada utilizando folhas de papel. Isso causa muita perda de tempo, uma vez que os dados em diversas planilhas e/ou papéis não estão integrados da melhor maneira. As diferentes formas de disputa de um campeonato causam certo empenho para um organizador.

Uma planilha para os grupos, outra para os times, outra para os jogadores, outra para as partidas. Tudo isso sem estar integrado e sem a disponibilidade dos dados para os que desejam acompanhar o desempenho do seu time dentro da disputa.

Folhas e mais folhas de papel são gastas apenas para montar estruturas que seriam facilmente representadas dentro de um sistema específico para a

organização de um torneio de futebol. Desorganização e dados nem sempre tão corretos expostos de maneira desatualizada, colados na parede de um ginásio ou complexo esportivo onde ocorre um campeonato.

A aplicação proposta neste trabalho vem para auxiliar organizadores de campeonatos e otimizar seu tempo e trabalho, tornando o gerenciamento de um torneio algo mais simples de ser realizado.

1.2 Objetivos

Este trabalho tem como objeto desenvolver uma aplicação web que auxilie pessoas ou instituições que desejam organizar um campeonato de futebol. Esta aplicação facilitará a administração e permitirá aos envolvidos no campeonato ter acesso às informações relativas ao evento, tais como classificações, artilharias, resultados dos jogos realizados, partidas agendadas, ranking de defesas menos vazados, times com mais cartões.

1.3 Organização do texto

Este documento está estruturado em seis capítulos organizados da seguinte forma: no primeiro capítulo encontra-se a introdução referente ao projeto, contextualizando e apresentando o atual cenário para organizar-se um campeonato de futebol. No segundo capítulo encontra-se toda a fundamentação teórica utilizada para o desenvolvimento do projeto. O terceiro capítulo mostra as aplicações correlatas que também fazem gerenciamentos de campeonatos. No quarto capítulo está presente a metodologia utilizada, junto com os requisitos das aplicações. O

quinto capítulo mostra o desenvolvimento e implementação da aplicação, explicando sobre as tecnologias utilizadas além dos resultados de cada etapa do processo de desenvolvimento. No sexto e último capítulo são feitas as considerações finais e conclusões decorrentes do trabalho realizado, assim como propostas para futuros trabalhos.

1.1.1 Objetivo geral

Elaborar uma aplicação web que seja capaz de fazer a gerência e atualização de dados referentes a um campeonato de futebol, além de prover um espaço para simpatizantes e jogadores acompanharem as informações e dados do campeonato.

1.1.2 Objetivos específicos

- Desenvolvimento de um módulo cadastral do campeonato, onde será possível personalizá-lo através da forma de disputa, data de início e fim e dados da organização;
- Desenvolvimento de um módulo cadastral onde será possível cadastrar times e escolher os campeonatos que ele participa;
- Desenvolvimento de um módulo cadastral onde será possível cadastrar grupos do campeonato, escolhendo os times presentes nos grupos;
- Desenvolvimento de um módulo cadastral para cadastrar jogadores em seus respectivos times;

- Desenvolvimento de um módulo cadastral para agendar partidas e salvar as informações presentes nas súmulas de uma partida: quem fez os gols e quem foi punido com cartão durante a partida;
- Desenvolvimento de um módulo para acompanhar as informações dos campeonatos: classificação, partidas realizadas, próximas partidas, ranking de artilharia, ranking de cartões amarelo e vermelho e ranking de defesas menos vazadas.

2 Fundamentação teórica

2.1 Aplicação Web

Aplicações Web formam grande parte dos softwares desenvolvidos pelas empresas de tecnologias. Este termo é utilizado para descrever os softwares que estão disponíveis em ambientes Web. De acordo com Pressman, 2002, uma aplicação web pode ser definida como uma simples página, um site completo ou até mesmo sistemas web.

A partir da tecnologia web, que adiciona características de hipermídia aos sistemas de informação, usuário conseguem acessar estas informações disponibilizadas por web sites e que contém os documentos necessários para a visualização da página pelo cliente.

Com o tempo, novos recursos foram sendo adicionados à tecnologia Web, permitindo que usuários interajam com os sistemas por meio de ferramentes específicas para isto, os browsers. Os usuário realizam as requisições ao servidor

web, o qual processa e gera dinamicamente sua resposta ao cliente. Tornando assim, a troca de informações entre cliente e usuário bidirecional.

A partir do protocolo HTTP (*HyperText Transfer Protocol*), o cliente faz a requisição ao servidor permitindo que o usuário visualize os recursos definidos ao se desenvolver uma página web. O protocolo HTTP se baseia em requisições e respostas que possuem elementos característicos. O conteúdo de uma requisição enviada ao servidor é: método HTTP, url acessada e parâmetros do formulário, dependendo do método utilizado. Já o conteúdo da resposta contém: código de status (para representar o sucesso ou não da solicitação), o tipo de conteúdo apresentado na resposta, e o conteúdo em si no formato de HTML, imagens ou documentos.

2.2 Frameworks

Mesmo com as evoluções da tecnologia e das técnicas para o desenvolvimento de software, programar um software ainda é um processo custoso em diversos aspectos: demanda tempo e mão de obra qualificada.

Existem propostas que auxiliam a difícil tarefa de desenvolver um software e aumentar a produtividade das equipes. Uma destas propostas é a técnica de reutilização de software, onde projetistas e desenvolvedores mais experientes sabem que não se deve resolver cada problema a partir do zero, e sim reutilizar códigos e soluções já utilizadas anteriormente que são conhecidas e eficientes. Os principais benefícios desta reutilização são: maior produtividade, visto que não é necessário “reinventar a roda” para solucionar determinado problema; e melhoria na

qualidade, uma vez que como o código será reutilizado, é necessário que ele esteja muito bem testado para não haver falhas graves com riscos mais sérios à aplicação.

Framework pode ser definido como “Uma aplicação semi-completa, reutilizável que pode ser especializada para produzir novas aplicações” (JOHNSON, 1988). Um framework descreve os objetos componentes junto às suas interações e a interface de cada um dos objetos junto aos seus fluxos de controle. Partes em comum de um framework, são as partes estáveis e conhecidas como *cold spots.* Por outro lado, as partes flexíveis, são nomeadas como *hot spot* e tipicamente são incompletas. O desenvolvedor que utilizar um framework, estenderá as funcionalidades já presentes e produzirá sua aplicação a partir das partes flexíveis do framework, botando em prática o conceito de reutilização de software.

Segundo FAYAD (1999), mesmo que os pontos positivos e princípios de projetos sejam livres em relação ao domínio em que serão aplicados, eles são classificados da seguinte forma:

- Frameworks de infra-estrutura de sistema: facilitam a produção de infra-estruturas de sistemas, como frameworks de comunicação entre processos, frameworks para codificação de compiladores, frameworks para interfaces de usuário;
- Frameworks para integração de *middleware*: facilitam a integração de aplicações e componentes distribuídos, apoiando a comunicação e troca de informações entre os componentes do sistema. Por exemplo framework para transação de banco de dados.
- Frameworks de aplicação empresarial: são destinados a aplicações de fim específico, tendo como princípio o conhecimento do domínio e

destinados a usuários finais, como por exemplo engenharia financeira, manufatura e telecomunicações.

2.2.1 Benefícios do uso de Frameworks

Com certeza o maior benefício do uso de framework é o seu reuso, que minimiza o esforço para o desenvolvimento da aplicação. Mas além do reuso, existem outros benefícios citados por SOMMERVILLE (2003):

- A confiança dos usuários no framework, pois visto que é um software amplamente reutilizado, experimentado e testado, torna-se mais confiável pelo bom número de participantes na comunidade.
- Modularidade, já que encapsulam os detalhes de implementações voláteis por meio de interfaces estáveis.
- A inversão do controle, pois o código do framework chama o código do desenvolvedor, fazendo com que o framework controle a estrutura e o fluxo de execução do programa.

2.2.2 Desvantagens do uso de Frameworks

Apesar de possuir diversas vantagens em seu uso, há obstáculos que podem aparecer aos usuários que utilizam frameworks. Segundo FAYAD, 1999, algumas empresas falham ao tentar implementar ou utilizar frameworks se não conseguirem passar por estes problemas: curva de aprendizagem, eficiência na integração, esforço de desenvolvimento, validação e remoção de defeitos e falta de padrões.

Por conta do alto grau de complexidade de desenvolver um framework, fica difícil desenvolver algo que seja de fato reutilizável, com alta qualidade e que ainda seja escalável e eficiente para diversos domínios.

Devido a alta curva de aprendizagem, certas vezes não vale a pena para a instituição investir tempo no aprendizado de um novo framework, a menos que este seja utilizado em muitos projetos, ou em projetos duradouros.

Por conta do uso de tecnologias tidas como antigas e/ou obsoletas, pode haver dificuldade na integração quando há mais de um framework sendo utilizado dentro da empresa.

Além dos fatores já citados anteriormente, é preciso ressaltar que um framework precisa se manter atualizado, ou seja, deve existir a manutenção do mesmo, seja para correção de *bugs*, adição de funcionalidades ou outros fatores. Para haver essa manutenção, os mantenedores precisam ter profundo conhecimento de todos os componentes e suas dependências para não haver problemas na fase de evolução do framework. A validação destas manutenções torna-se difícil visto que um framework pode ser usado em diferentes domínios.

2.3 Scrum

O framework para desenvolvimento ágil de software conhecido como Scrum é ideal para a gestão e planejamento de projetos de software. Segundo o guia "*The Scrum Guide*" (SUTHERLAND; SCHWABER, 2016), Scrum pode ser definido como "Um framework no qual pessoas podem lidar com problemas complexos ajustáveis, enquanto produtiva e criativamente entregam produtos com maior valor possível". O

Scrum possui diversos elementos característicos que serão explicados um a um posteriormente. São eles: time, membros, artefatos e eventos.

O Scrum permite que o desenvolvimento de um software seja dividido em Sprints, que são esforços focados para concluir objetivos previamente definidos, proporcionando um maior controle sob todas as funcionalidades, além de flexibilidade ao definir prioridades nas atividades, permitindo o desenvolvimento de tarefas que agregam maior valor ao produto.

No time os membros ficam responsáveis pela entrega do produto de forma incremental e iterativa, cada um com seu papel. O *Product Owner* fica responsável por motivar e fiscalizar o trabalho do time de desenvolvimento além de maximizar o valor do produto. O *Development Team* responsabiliza-se pelas atividades previstas em cada ciclo. Já o *Scrum Master* supervisiona o resto do time para que o Scrum seja compreendido e seguido utilizando as melhores práticas.

Ao definir um projeto como objetivo, divide-se o projeto em ciclos que são conhecidos como Sprints. O *Sprint* consiste em um conjunto de atividades que devem ser concluídas em um determinado tempo, tipicamente semanal, quinzenal ou mensal. O Scrum torna-se iterativo à medida que os Sprints vão se concluindo um após o outro, até finalizar o escopo do projeto. As atividades previstas em cada Sprint são definidas no *Sprint Planning* e é feita uma reunião diária de curta duração chamada *Daily Scrum* onde o time alinha as tarefas e um plano a ser seguido para o próximo dia. Ao final de cada Sprint é feita um outro tipo de reunião chamada *Sprint Review* para adaptar e avaliar o *Product Backlog* caso haja necessidade. Também ao fim da Sprint é feita mais uma reunião chamada *Sprint Retrospective* para

autoavaliação do time onde são sugeridas melhorias para aprimorar o próximo Sprint.

Com o time, seus membros e os eventos já definidos, tem-se ainda os artefatos. Os artefatos são definidos para haver maior transparência e entendimento do processo de desenvolvimento do Scrum. O *Product Backlog* consiste numa lista ordenada de todas as tarefas que deverão ser implementadas para que o produto seja entregue. O *Sprint Backlog* nada mais é que um subconjunto das tarefas previstas no *Product Backlog* que deverão ser implementadas no Sprint.

Tendo os elementos do Scrum definidos, podemos então demonstrar como fica na prática as relações entre todos estes elementos e o fluxo a ser seguido pelo Scrum.

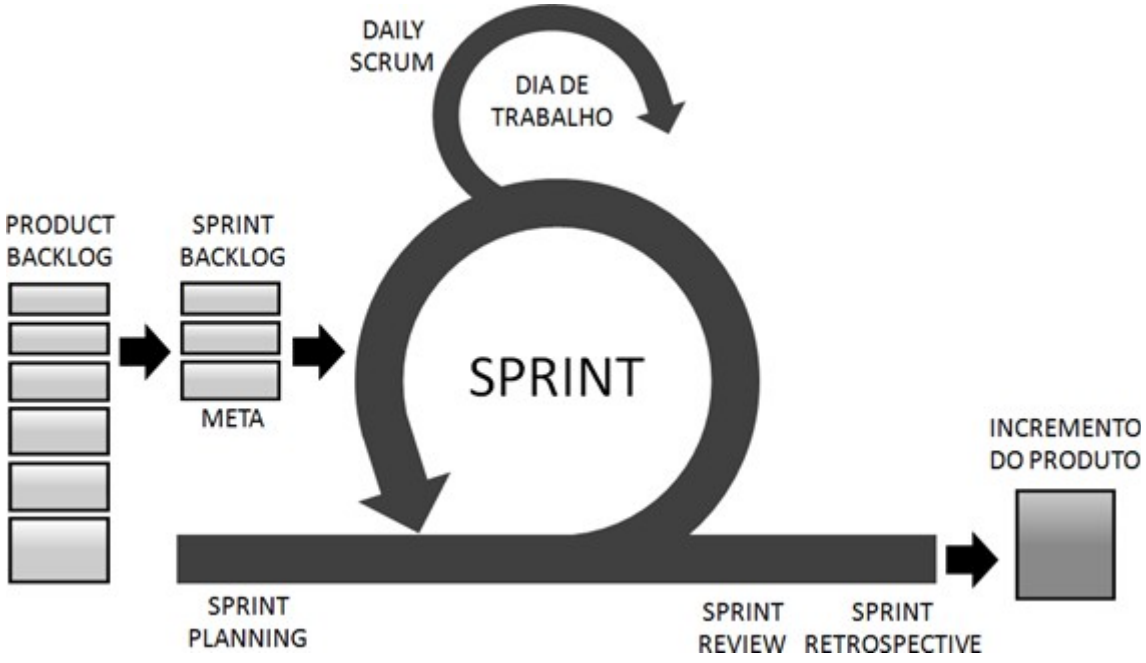


Figura 1 - Ciclo do Scrum. Fonte (DevMedia, 2017)

Primeiramente, o *Product Owner* fica responsável por definir o *Product Backlog*. Com ele definido podes planejar a *Sprint* através do *Sprint Planning* e

assim, definir o *Sprint Backlog*. Com a *Sprint* em andamento, temos a reunião diária *Daily Scrum* e ao final de cada *Sprint* temos as outras duas reuniões: *Sprint Review* e *Sprint Retrospective*. Por fim, itera-se o ciclo para o desenvolvimento de uma nova Sprint.

3 Aplicações correlatas

Ao pesquisar por gerenciadores de campeonato na internet é possível encontrar algumas ferramentas que já existem e têm basicamente o mesmo objetivo deste projeto, inclusive algumas até mais completas que a solução proposta. A grande maioria delas, possuem a possibilidade de administrar diversos tipos de campeonatos, e não somente campeonatos de futebol. Além disso grande parte das soluções não são totalmente gratuitas, sendo grátis somente até certo ponto, com limites de número de campeonatos gerenciáveis e número de jogadores presentes nas disputas. As diferentes soluções encontradas foram testadas e avaliadas pelo autor, buscando aperfeiçoar a solução proposta neste trabalho.

Foram selecionadas duas destas aplicações para fazer a correlação junto à este projeto. Optou-se por aplicações que possuíam algum contraponto a este projeto. Seja por falhas na responsividade da aplicação, ou por falta de usabilidade e não utilização de interfaces limpas e intuitivas. Também foi dada preferência para aplicações web e que fossem específicas para campeonatos de futebol, além de terem uma opção gratuita. Sendo assim, as duas aplicações escolhidas foram Ritmo do Esporte e Campeonato de Verão.

3.1 Ritmo do Esporte

Esta aplicação está presente no seguinte link www.ritmodoesporte.com.br. O principal diferencial desta aplicação é a automatização das partidas realizadas e na formação dos grupos de disputa, funcionalidades que não foram escolhidas para estarem presentes nesta primeira parte do projeto para o TCC, mas que poderão ser implementadas em trabalhos futuros. Através de conversas com um organizador de campeonatos aqui da grande Florianópolis, foi informado que o sorteio de grupos normalmente é realizado na presença de todos os capitães dos times participantes para que não haja possibilidade de fraudes.

Esta aplicação possui diferentes versões para planos gratuitos e pagos. O plano gratuito limita-se o número de competições ativas por ano além de não possuir itens para a comunicação entre organizador e usuários que desejam acompanhar a disputa, como notícias e fotos. É uma aplicação bastante completa, mesmo em sua versão gratuita, com muitas opções de personalização, o que acaba deixando o processo de criação do campeonato um pouco confuso, além de tornar a interface muito carregada de informações e pecando também na usabilidade. Sua visão não torna a experiência do usuário simples e intuitiva, o que foi visto como ponto negativo da aplicação. Quando aberta em celulares, a versão conta com ainda mais erros de usabilidade, visto que a falta de responsividade faz o usuário ter q ir rolando a tela até o ponto desejado para que faça o uso da maneira correta.

Figura 2 - Interface carregada e erros na responsividade. Fonte: (Ritmo do Esporte, 2017)

3.2 Campeonato de verão

Esta aplicação encontra-se presente no seguinte endereço: www.campeonatodeverao.com.br. Diferente da outra solução, esta possui uma interface mais intuitiva e com suas opções mais claras. Também possui gratuidade até certo ponto, sendo limitada no número de campeonatos ativos, número de

equipes participantes e duração do campeonato. Esta solução se mostrou muito parecida com a solução proposta neste trabalho, apesar de conter algumas informações consideradas irrelevantes pelo autor para a criação de um campeonato, como por exemplo a opção de cadastrar árbitros e possibilidade de cadastrar fotos.

O principal ponto negativo desta aplicação foi a limitação quanto à sua responsividade. Apesar da interface mais intuitiva e clara em sua versão web, a administração do campeonato torna-se muito complexa se feita pelo celular. Hoje em dia a usabilidade de uma aplicação web em celulares tablets é algo extremamente importante, visto que com a expansão do uso de smartphones e tablets o acesso à internet pode ser feito de qualquer lugar. Assim, para atualizar os dados de um partida ou até mesmo para um usuário acessar as informações sobre o campeonato, o celular não seria a melhor escolha. Além disso, a aplicação conta com alguns erros que deveriam ser tratados da maneira correta, pois isto pode acabar confundindo o usuário leigo.

figura 3 - Falhas na responsividade e erros não tratados. Fonte: (Campeonato de Verão)

4 Solução proposta

4.1 Metodologia utilizada

Para desenvolver este trabalho, a metodologia escolhida para execução será a metodologia ágil Scrum explicada anteriormente na seção 2.3, sendo adaptada para equipe de apenas uma pessoa. O Scrum é um framework utilizado para gerenciamento de projetos e é ideal para trabalhos complexos onde não é possível ao início do projeto prever tudo que irá ocorrer ao longo do desenvolvimento. Apesar de ser um framework com uma proposta de trabalho para equipe, foi adaptado para apenas uma pessoa cumprir todos os papéis do projeto: Product Owner, Scrum Master e o time de desenvolvimento. Por conta desta adaptação alguns itens tornam-se desnecessários, como por exemplo o Daily Scrum, que é uma avaliação diária do que foi feito no projeto e priorização de tarefas para o dia seguinte.

Para o controle das tarefas e sprints definidos para o trabalho utilizou-se o software Trello, que permitiu um grande gerenciamento e facilidade no manuseio das tarefas previstas no Product Backlog e no Sprint Backlog, elementos no Scrum.

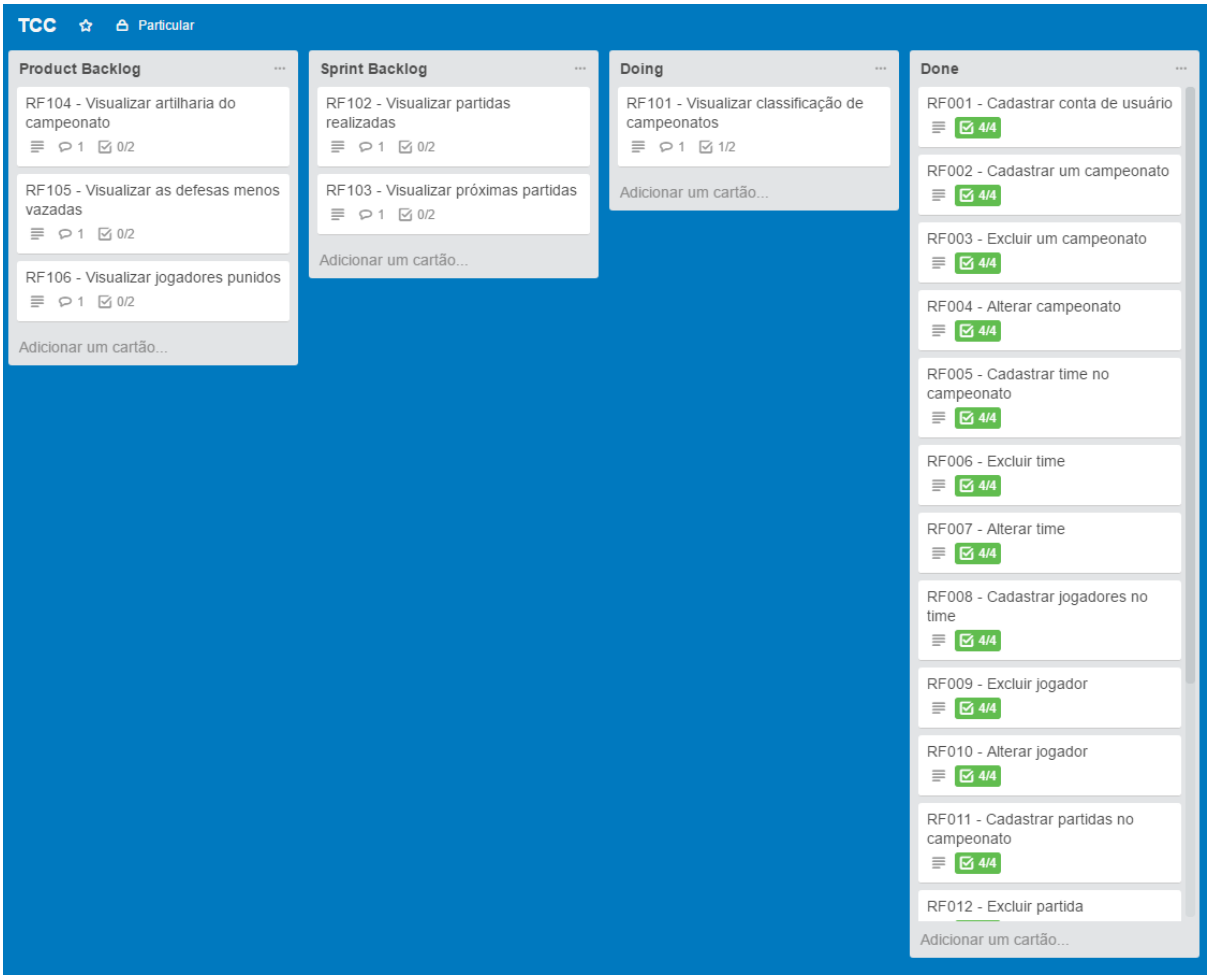


Figura 4 - Gerenciamento de tarefas no Trello. Fonte: (Trello)

Conforme especificação do Scrum, foi desenvolvido o Product Backlog baseando-se no requisitos necessários e previstos pelo autor para o desenvolvimento da solução proposta. A imagem a seguir mostra o Product Backlog.

Nome	Importância	Estimativa (hrs)
Cadastrar conta de usuário	100	8
Cadastrar campeonato	100	10
Cadastrar time no campeonato	100	10
Cadastrar jogadores no time	100	15
Cadastrar partidas no campeonato	100	20
Cadastrar grupos no campeonato	100	10
Alterar campeonato	75	8
Alterar time	75	8
Alterar jogadores	75	8
Alterar partidas	75	8
Alterar grupos	75	8
Excluir campeonato	75	2
Excluir time	75	2
Excluir jogadores	75	2
Excluir partidas	75	2
Excluir grupos	75	2
Visualizar classificação do campeonato	75	8
Visualizar partidas realizadas	75	8
Visualizar próximas partidas	50	8
Visualizar artilharia do campeonato	50	6
Visualizar as melhores defesas	50	6
Visualizar jogadores punidos	40	6

Figura 5 - Product Backlog. Fonte: (Autor)

Ainda conforme o Scrum, foram desenvolvidos os Sprints que deveriam ser seguidos ao decorrer do andamento do projeto. Estes Sprints levaram em consideração os requisitos do sistema. A seguir é possível ver a imagem que retrata os Sprints planejados.

Sprint	Descrição	Release	Estimativa
Cadastrar conta de usuário	O usuário poderá criar uma conta com login e senha para acessar o sistema	1	8
Cadastrar campeonato	O usuário cadastrado poderá criar um campeonato	2	10
Alterar campeonato	O usuário cadastrado poderá alterar dados do campeonato	2	8
Excluir campeonato	O usuário cadastrado poderá excluir um campeonato	2	2
Cadastrar time no campeonato	O usuário cadastrado poderá cadastrar um time e escolher o campeonato que ele participa	3	10
Alterar time	O usuário cadastrado poderá alterar dados do time	3	8
Excluir time	O usuário cadastrado poderá excluir um time	3	2
Cadastrar jogadores no time	O usuário cadastrado poderá adicionar jogadores ao seus times criados	4	15
Alterar jogadores	O usuário cadastrado poderá alterar dados do jogador	4	8
Excluir jogador	O usuário cadastrado poderá excluir um jogador	4	2
Cadastrar partidas no campeonato	O usuário cadastrado poderá adicionar partidas ao campeonato e editar os eventos desta partida, tais como gols e cartões	5	20
Alterar partida	O usuário cadastrado poderá alterar a partida e seus eventos	5	8
Excluir partidas	O usuário cadastrado poderá excluir uma partida	5	2
Cadastrar grupos no campeonato	O usuário cadastrado poderá adicionar grupos aos campeonatos com formato de "Copa"	6	10
Alterar grupo	O usuário cadastrado poderá alterar os grupos de campeonatos com formato "Copa"	6	8
Excluir grupos	O usuário cadastrado poderá excluir um grupo	6	8
Visualizar classificação do campeonato	Qualquer usuário poderá visualizar a classificação de um campeonato	7	8
Visualizar partidas realizadas	Qualquer usuário poderá visualizar as partidas realizadas de um campeonato	7	8
Visualizar próximas partidas	Qualquer usuário poderá visualizar as próximas partidas de um campeonato	7	6
Visualizar artilharia do campeonato	Qualquer usuário poderá visualizar a artilharia de um campeonato	8	6
Visualizar as melhores defesas	Qualquer usuário poderá visualizar as melhores defesas de um campeonato	8	6
Visualizar jogadores punidos	Qualquer usuário poderá visualizar a lista de jogadores punidos de um campeonato	8	6

Figura 6 - Sprints do projeto. Fonte: (Autor)

Para a produção da aplicação será utilizado o framework Meteor com o armazenamento do banco de dados utilizando MongoDB. Ambas tecnologias foram estudadas através de tutoriais e fontes da internet e serão explicadas posteriormente no capítulo 5 de tecnologias utilizadas.

4.2 Requisitos do sistema

Para realizar o desenvolvimento de uma aplicação é necessário iniciar o processo com o levantamento dos requisitos do sistema. Segundo Sommervile

(2003) o processo de levantamento de requisitos segue um processo genérico e iterativo contendo as seguintes etapas:

- Compreensão do domínio: analistas desenvolvem sua compreensão quanto ao domínio da solução;
- Coleta de requisitos: processo de interação com os *stakeholders* do sistema, onde são extraídos os requisitos e compreendido ainda mais o domínio da solução;
- Classificação: organiza os requisitos coletados de maneira que fiquem em grupos coerentes;
- Resolução de conflitos: se houverem múltiplos *stakeholders* do sistema, há grandes chances dos requisitos sofrerem conflitos. Esta etapa serve para resolvê-los;
- Definição de propriedades: de todos os requisitos levantados, sempre haverá alguns mais importantes que outros. Esta etapa serve para priorizar aqueles requisitos de maior importância;
- Verificação de requisitos: os requisitos são verificados para ver se estão completos e condizentes com aquilo que se espera do sistema.

As principais dificuldades encontradas para se realizar um bom levantamento de requisitos são a falta de conhecimento do usuário principal, que não consegue expressar-se ao analista da melhor maneira e muitas vezes até não sabe o que quer que o sistema faça. Outro ponto que pode gerar um requisito problemático é o analista não descrever de modo claro, sem ambiguidades e consistente quanto aos aspectos significativos do sistema. Além disso também é necessário utilizar uma

técnica adequada para a extração dos requisitos. Todos estes problemas mencionados contribuem para alguns softwares terem um baixo grau de satisfação dos usuários finais.

Existem diversas técnicas para realizar o levantamento de requisitos. Estas técnicas buscam minimizar os problemas relativos à essa etapa do desenvolvimento de um software. Cada técnica possui sua vantagem e desvantagem, e elas podem ser combinadas para um melhor aproveitamento do analista.

Nas próximas seções serão explicadas as duas técnicas utilizadas neste trabalho para realizar o levantamento de requisitos, são elas o *Brainstorming* e a Prototipagem. Porém é necessário salientar que existem outras técnicas conhecidas para esta etapa do processo: Levantamento orientado a pontos de vista, Etnografia, Workshops, Entrevistas, Questionários e JAD (*Joint Application Design)*.

Após aplicar as técnicas, foi possível ter um conjunto de requisitos que buscassem atender às solicitações dos usuários finais.

4.2.1 Brainstorming

Esta é uma técnica utilizada para geração de ideias. São realizadas uma ou mais reuniões que permitem aos participantes explorarem suas ideias.

Os participantes são selecionados em função das suas possíveis contribuições para o andamento da sessão, dando preferência às pessoas que possuem algum conhecimento sobre o domínio da solução. Os participantes escolhidos para este *Brainstorming* foram amigos do autor que participam de campeonatos e que gostariam de acompanhar os resultados dos mesmo em um

sistema, além de outro amigo do autor que possui experiência com organização de campeonatos de futebol, com foco principalmente em campeonatos disputados dentro do ambiente acadêmico. As sessões foram realizadas algumas vezes, sempre encorajando os participantes a terem ideias. Estas sessões de conversa foram realizadas de maneira informal, e por isto, não houve um documento que registrasse as ideias geradas.

No *Brainstorming,* todas as ideias são bem-vindas. Mesmo que inicialmente essa ideia possa parecer não convencional, estimula-se os participantes para que haja cada vez mais ideias criativas, inclusive combinando as diferentes ideias propostas para alcançar um objetivo em comum.

Ao finalizar este processo as ideias são avaliadas e as mais valiosas são mantidas e classificadas quanto a sua prioridade. No caso deste trabalho, optou-se por ideias que fossem dinâmicas o suficiente e que não possuíam alto grau de complexidade, uma vez que o sistema deveria ser simples o suficiente para que a tarefa de organizar um campeonato não se tornasse um processo maçante.

4.2.2 Prototipagem

Esta técnica é um processo iterativo que consiste em explorar aspectos de determinada funcionalidade. É indicada para questões de interface de usuário e viabilidade de atendimento dos requisitos.

Ao fim de cada uma das Sprints, o protótipo desenvolvido era disponibilizado aos mesmo colaboradores que auxiliaram no processo de *Brainstorming* em sessões individuais, e também como no Brainstorming, foram realizadas de maneira

informal, sem a presença de um documento para registro de tal. O objetivo desta técnica é identificar principalmente problemas na usabilidade da aplicação e identificar se os requisitos levantados estavam cumprindo com seus objetivos. No fim de cada sessão era solicitado a cada um dos auxiliares os aspectos positivos e negativos da funcionalidade testada, sendo possível identificar os problemas para atualização dos requisitos caso fosse necessário.

4.3 Requisitos funcionais

Nome do requisito	**Descrição**	**Entrada e pré-condições**	**Saídas e pós condição**	**Prioridade**
RF001 - Cadastrar conta de usuário	O usuário poderá criar uma conta com login e senha para acessar o sistema	Login e senha	Uma conta de usuário é cadastrada no sistema	Essencial
RF002 - Cadastrar um campeonato	O usuário cadastrado poderá criar um campeonato	Nome do campeonato, nome do organizador, email do organizador, cidade do campeonato, data de início, data de fim e formato (Copa, Liga, Mata-Mata)	O campeonato é cadastrado no sistema	Essencial
RF003 - Excluir um campeonato	O usuário cadastrado poderá excluir um campeonato	O campeonato que se deseja excluir	O campeonato é excluído do sistema	Importante

RF004 - Alterar campeonato	O usuário cadastrado poderá alterar dados do campeonato	O campeonato e os atributos que se deseja alterar	O campeonato é alterado	Importante
RF005 - Cadastrar time no campeonato	O usuário cadastrado poderá cadastrar um time e escolher o campeonato que ele participará	Nome do time, responsável, email do responsável e campeonatos que o time participará	O time é cadastrado no(s) campeonato(s)	Essencial
RF006 - Excluir time	O usuário cadastrado poderá excluir um time	O time que se deseja excluir	O time é excluído do sistema	Importante
RF007 - Alterar time	O usuário cadastrado poderá alterar dados do campeonato	O time e os atributos que se deseja alterar	O time é alterado	Importante
RF008 - Cadastrar jogadores no time	O usuário cadastrado poderá adicionar jogadores ao seus times criados	Nome do jogador, número do documento, time que ele jogará	O jogador é adicionado no time	Essencial
RF009 - Excluir jogador	O usuário cadastrado poderá excluir um jogador	O jogador que se deseja excluir	O jogador é excluído	Importante
RF010 - Alterar jogador	O usuário cadastrado poderá alterar dados do jogador	O jogador e os atributos que se deseja alterar	O jogador é alterado	Importante

RF011 - Cadastrar partidas no campeonato	O usuário cadastrado poderá adicionar partidas ao campeonato e editar os eventos desta partida, tais como gols e cartões	Campeonato da partida, fase, turno, grupo, local, data, horario, adversários, nome do arbitro, status da partida, resultado final, time vencedor, eventos da partida (gols e cartões)	A partida é adicionada ao campeonato	Essencial
RF012 - Excluir partida	O usuário cadastrado poderá excluir uma partida	A partida que se deseja excluir	A partida é excluída	Importante
RF013 - Alterar partida	O usuário cadastrado poderá alterar a partida e seus eventos	A partida e os atributos que se deseja alterar	A partida é alterada	Importante
RF014 - Cadastrar grupos no campeonato	O usuário cadastrado poderá adicionar grupos aos campeonatos com formato de "Copa"	Campeonato do grupo, nome do grupo, times pertencentes ao grupo	O grupo é adicionado ao campeonato	Essencial
RF015 - Excluir grupo	O usuário cadastrado poderá excluir um grupo	O grupo que se deseja excluir	O grupo é excluído	Importante
RF016 - Alterar grupo	O usuário cadastrado poderá alterar os grupos de campeonatos com formato "Copa"	O grupo e os atributos que se deseja alterar	O grupo é alterado	Importante

RF101 - Visualizar classificação de campeonatos	Qualquer usuário poderá visualizar a classificação de um campeonato	O campeonato que se deseja ver a classificação	A classificação do campeonato escolhido	Essencial
RF102 - Visualizar partidas realizadas	Qualquer usuário poderá visualizar as partidas realizadas de um campeonato	O campeonato que se deseja ver as partidas realizadas	As partidas realizadas do campeonato escolhido	Importante
RF103 - Visualizar próximas partidas	Qualquer usuário poderá visualizar as próximas partidas de um campeonato	O campeonato que se deseja ver as próximas partidas	As próximas partidas do campeonato escolhido	Importante
RF104 - Visualizar artilharia do campeonato	Qualquer usuário poderá visualizar a artilharia de um campeonato	O campeonato que se deseja ver a artilharia	A tabela de artilheiros do campeonato	Importante
RF105 - Visualizar as defesas menos vazadas	Qualquer usuário poderá visualizar as melhores defesas de um campeonato	O campeonato que se deseja ver as defesas menos vazadas	A tabela com defesas menos vazadas do campeonato	Importante
RF106 - Visualizar jogadores punidos	Qualquer usuário poderá visualizar a lista de jogadores punidos de um campeonato	O campeonato que se deseja ver os jogadores punidos	A tabela de jogadores punidos do campeonato	Desejável

Tabela 1 - Requisitos funcionais

4.3.1 Diagrama de casos de uso

Um diagrama de casos de uso serve para documentar e representar quais as funcionalidades do software e como os diferentes tipos de usuários, chamados de ator, fazem a interação com estas funcionalidades. Para o trabalho proposto, identificou-se dois tipos de usuários: o usuário que deseja acompanhar o campeonato visualizando suas informações, identificado como “Ator usuário” no diagrama; e o usuário que representa o organizador de um campeonato - aquele que fará a configuração e seleção de opções do campeonato, chamado de “Ator organizador” no diagrama.

A imagem a seguir representa o diagrama de caso de uso da aplicação proposta.

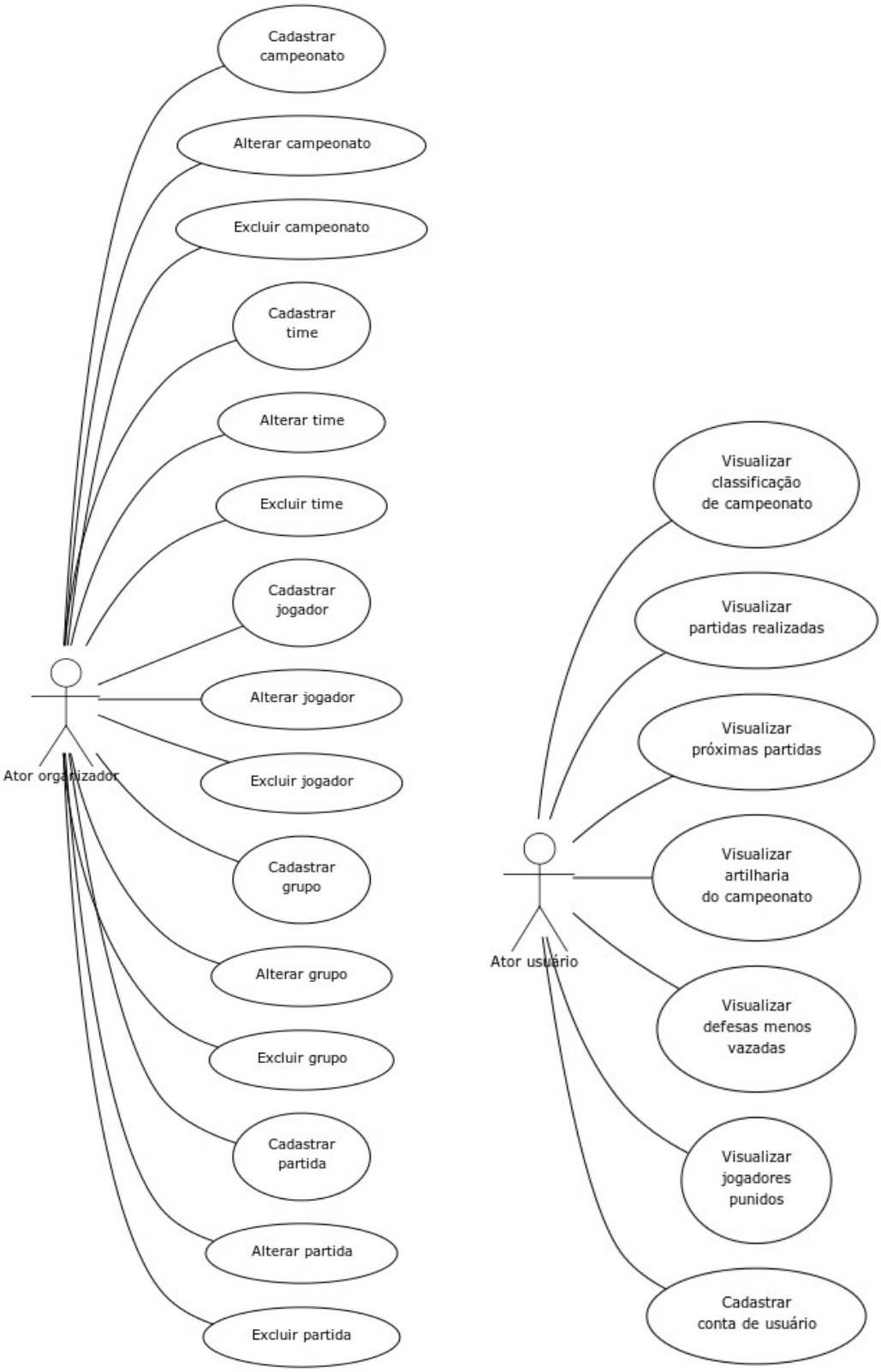


Figura 7 - Casos de uso. Fonte: (Autor)

5 Desenvolvimento

5.1 Tecnologias utilizadas

5.1.1 MeteorJS

Meteor, ou MeteorJS, é um projeto open-source de um framework escrito em Node.js. É um framework relativamente novo, sendo lançado oficialmente com o nome de Skybreak em dezembro de 2011. A ideia foi vista com tamanho potencial que a empresa responsável pelo Meteor recebeu um aporte de 11,2 milhões de dólares para que pudessem focar no aprimoramento do framework. Após isso foi adquirido pelo Meteor Development Group no ano de 2014 com o objetivo de expandir o seu suporte.

O principal lema de Meteor é: simplicidade é igual a produtividade. Ou seja, quanto mais simples for a tecnologia utilizada, mais produtividade no desenvolvimento do seu projeto você conseguirá. A simplicidade vêm na abstração de arquivos de configuração, onde o próprio Meteor consegue verificar as extensões dos arquivos HTML, CSS e JS, e compilá-los automaticamente sem que você precise fazer diversas configurações e importações. Além disso Meteor se torna ainda mais simples por ser uma plataforma *fullstack*, que significa ter o banco de dados, o *back-end* e o *front-end* integrados facilmente. E tudo isso com apenas uma linguagem: javascript. A ferramenta é escrita em Node.js e utiliza MongoDB para o armazenamento de dados, e no lugar do AngularJS tem uma view engine própria

chamada de Blaze, mas que não impede que você utilize AngularJS ou até mesmo React.

Quando se fala em programação para web, logo se pensa em milhares de linguagens e conceitos que precisam ser pensados para que tudo se encaminhe corretamente, desde o *front-end*, passando pelo *back-end* e até o banco de dados. Isto acaba fazendo você gastar muito tempo e esforço que poderia estar sendo investido em uma maneira mais dinâmica de desenvolver. Meteor tem a capacidade de você lidar com o *front-end*, *back-end* e banco de dados apenas com javascript. O maior benefício de escrever um código em uma mesma linguagem é que a mesma linha de código pode rodar tanto no servidor como no cliente, fazendo coisas diferentes nos dois ambientes. Por exemplo, ao executar o comando de criar uma coleção no lado do servidor, ela cria a coleção. Porém ao executar o mesmo comando no lado de cliente, Meteor permite a criação de uma coleção local no browser do usuário. Então o usuário pode atualizar e tratar o dado daquela coleção, permitindo que o dado se atualize em sua tela instantaneamente, enquanto Meteor sincroniza as alterações locais do browser com o servidor, por baixo dos panos.

A medida que a tecnologia e a web evoluem, usuários esperam cada vez mais por aplicações interagindo quase que instantaneamente. Meteor utiliza o conceito de reatividade para auxiliar nesta tarefa. A reatividade tem a ver com o fluxo de dados e a propagação das mudanças de estado dos objetos. Uma variável que for alterada em qualquer lugar da internet, em um de seus clientes, será automaticamente atualizada no HTML em todas as abas abertas sem necessidade de nenhuma programação adicional para isto. O Meteor consegue até mesmo persistir estes dados e gravá-los na camada do banco de dados, o MongoDB.

Aplicações multi-client se tornam muito simples de se desenvolver graças à esta reatividade.

Quanto a hospedagem de uma aplicação, em Meteor fica muito simples de se fazer. Isto porque a própria plataforma disponibiliza gratuitamente um serviço de hospedagem para aplicações com baixo tráfego e até 200 emails por dia. Tudo isso em uma nuvem pública e gratuita do próprio Meteor.

Meteor se baseia em alguns princípios que serão discutidos a seguir.

- Dados na rede - Meteor não transmite HTML pela rede, mas sim dados que serão renderizados em cada um dos clientes. Isso aumenta sua performance pois diminui o tráfego ao transmitir menos informação, diferentes de Web Services em rest e JSON.
- Apenas uma linguagem - como já citado anteriormente, você utiliza javascript para escrever em ambos os lados, cliente e servidor. No cliente isso é padrão por conta dos browsers atuais, e no lado do servidor o Node.js quem faz este trabalho e permite que se escreva em Javascript no servidor.
- Banco de dados em todo lugar - o banco de dados fica disponível tanto no lado do cliente como no lado do servidor, evitando requisições HTTP desnecessárias para conectar-se com o banco.
- Compensação de latência - No lado do cliente, Meteor faz um pré carregamento de alguns dados e simula os modelos de dados presentes no servidor, fazendo com que as requisições ao servidor pareçam instantâneas. É mais ou menos algo parecido com cache presentes em outros frameworks.

- Update automático em todas camadas - tempo real é essencial para o Meteor. Desde o banco de dados, até o templates HTML, são atualizados quando necessário. Tudo que muda no cliente, muda no servidor também. Isso torna a aplicação bastante responsiva, por não é necessário baixar a página inteira novamente para renderizar o conteúdo.

A maioria das linguagens de programação contam com gerenciamento de pacotes. Ruby tem as Gems, Python o Pip e o Node.js tem o NPM. Como Meteor é escrito em Node.js, também é possível utilizar módulos NPM nele. Porém Meteor possui seu próprio repositório de pacotes, chamado Atmosphere. A diferença é que os pacotes Atmosphere fazem o uso da reatividade presente no framework e também podem conter segmentos de código HTML dinâmico, facilitando o processo de criação de templates no HTML. Existem diversos pacotes para diversos problemas com suas respectivas soluções, onde você pode utilizar um pacote desenvolvido por terceiros para agilizar seu processo de desenvolvimento. Um pacote utilizado na aplicação proposta por este trabalho foi para realizar o controle de usuários da aplicação. Normalmente o desenvolvedor precisaria criar as tabelas no banco de dados para então programar as requisições dos usuários. Utilizando o pacote, obtém-se muito mais agilidade para o desenvolvimento do software, economizando tempo.

5.1.2 Mongo DB

Com a atual crescente da manipulação de informações, grandes concentrações de dados vêm se formando ao decorrer do tempo. Este conceito é denominado de Big Data e trouxe novos desafios para o armazenamento e manipulação de grandes volumes de dados. O modelo mais tradicional e mais difundido atualmente entre os desenvolvedores é o modelo relacional. O surgimento do modelo não relacional, trouxe novas classes de banco de dados que são identificadas como NoSQL, explicado na seção anterior. Entre elas, está o Banco de dados orientado à documentos. Grandes empresas que possuem muita informação e são conhecidas mundialmente, utilizam estes conceito do NoSQL para manipular seus dados. Entre elas estão o Facebook, Netflix, Yahoo e algumas outras. Isto acaba fazendo crescer a comunidade de NoSQL. O MongoDB é um banco não relacional pertencente à esta classe de bancos. A seguir será mostrado o lado deste tipo de bases de dados, os não relacionais orientados a documentos, mais precisamente o MongoDb, cujo as características podem agilizar o desenvolvimento de um software.

A agilidade no desenvolvimento de uma base de dados utilizando MongoDB, se dá pelo fato deste não possuir um esquema fixo, permitindo inserção de registros sem seguir um padrão, o que facilita a paralelização do trabalho, contribuindo para o desenvolvimento em módulos por diferentes equipes. Além disto, o MongoDB não possui tabelas, joins, chaves estrangeiras e outras funções já conhecidas por modelos relacionais. Isto pode acabar complicando um pouco a cabeça do desenvolvedor que está acostumado com modelos relacionais, porém com o tempo isso muda e o MongoDB passa a ser seu amigo.

A familiaridade dos desenvolvedores com o SQL, ainda faz com que hoje em dia os bancos de dados relacionais sejam mais conhecidos e utilizados. Porém MongoDB, com sua alta escalabilidade traz cada vez mais adeptos em busca de maior flexibilidade no desenvolvimento. Por ser orientado a documentos e não possuir tabelas, MongoDB troca o conceito de linhas por documentos, onde grandes informações com alto grau de hierarquia possam ser representadas em apenas um registro. Pela sua flexibilidade e escalabilidade, além de outras características oriundas de bancos de dados relacionais, o foco do desenvolvedor prioriza a programação do software em si, deixando a base de dados em segundo plano.

A estrutura dos documentos de MongoDB é baseada em JSON. O JSON permite a representação de dados em formato de textos, facilitando ao desenvolvedor enxergar os dados e extrair as informações necessárias. O JSON permite armazenar e trocar dados de maneira simples e objetiva. Diferente de bases de dados relacionais, ondes os dados são estruturados individualmente em tabelas, o MongoDB estrutura seus dados em documentos utilizando o BSON, que é um formato de dados próprio da empresa MongoDB e baseado em JSON, tornando-o eficaz por permitir a adição separada de dados no documento.

Queries são conhecidas pelos desenvolvedores que utilizam o modelo relacional, porém nem todas as bases de dados são relacionais e, portanto, estão aptas a utilizá-las. MongoDB permite a utilização de queries para trazer o poder de consulta conhecidos em bancos relacionais.

Os índices secundários em MongoDB são desenvolvidos em árvores-B, permitindo declarar até 64 índices por documento. Um dos grandes benefícios ao implementar os índices, é tornar a consulta mais rápida, trazendo agilidade no

acesso à dados, sem ter que percorrer toda a base de dados na busca de determinadas informações.

Como tudo dentro da área de tecnologia, MongoDB também possui limitações. A principal limitação para ele é a necessidade de sistemas com arquitetura em 64 bits. Isso porque MongoDB necessita de 2GB de RAM para mapear arquivos de dados e sistemas 32 bits não reconhecem mais que 4GB, sendo que parte da memória é para o sistema operacional. Como a forma de operação do MongoDB se resume à mapeamentos de arquivos em memória, caso o *journaling* estiver inativo, uma falta de luz, por exemplo, poderia causar corrompimento de dados.

Como já citado anteriormente, por não possuir tabelas, colunas e tipagem, o desenvolvimento usando MongoDB se mostra muito flexível. O modelo da base de dados do MongoDB constitui em *collections*, que são como se fosse tabelas em bases relacionais, e documentos que são semelhantes às tuplas em cada tabela de um banco relacional. Um exemplo de documento de uma base de dados qualquer vem a seguir:

Figura 8 - Exemplo de documento em MongoDB. Fonte: (Autor)

Na figura 8 podemos visualizar o uso do JSON para representar *collections* nos documentos. Nota-se a semelhança com as classes de outras linguagens orientadas a objetos. Além disso, é importante enfatizar que caso fosse necessário adicionar mais uma informação à um documento que representa a mesma *collection* do exemplo acima, não afetaria a estrutura da *collection*, podendo inclusive conter diferentes atributos que um documento da mesma *collection*. Isso enfatiza a flexibilidade e a escalabilidade do MongoDB.

Apesar de ser não relacional, MongoDB também possibilita a relação entre documentos, porém faz isso através de *embedding* ou referencing. *Embedding* é como adicionar arrays dentro do documentos, com suas informações específicas, permitindo adicionar vários itens dentro deste array. *Referencing* é bem parecido com referências externas em bases relacionais, com referências à outros documentos. O uso de *embedding*, segundo autores, aumenta a velocidade de acesso às informações pois os dados já estão nos discos. Porém é necessário analisar o contexto para adequar quais das práticas são recomendadas para cada caso. Segundo Banker (2012, p. 249), a diferença é “Usa-se *embedding* quando os objetos filhos sempre surgirem dentro do contexto dos objetos pais. Caso contrário armazene os objetos filhos em *collections* separadas.”.

O MongoDB é o banco de dados padrão utilizado no framework MeteorJS, por esse motivo foi utilizado na aplicação e ainda contou com este capítulo em especial para ser explicado devidamente antes de mostrar como foi o desenvolvimento da solução proposta

5.2 Implementação

Seguindo os conceitos do Scrum e os adaptando para equipes de um só membro, este capítulo mostra todo o desenvolvimento do projeto seguindo as Sprints definidas e mostradas na figura 6 da seção 4.1. Cada uma das próximas seções mostra o que foi feito para chegar no resultado esperado pela Sprint. Cada Sprint planejado pelo autor teve duração de aproximadamente 20 horas, sendo este tempo utilizado entre pesquisas em tutoriais, adequação de layout e programação. O Sprint com menor tempo de duração foi o da primeira release, com aproximadamente 8 horas de desenvolvimento. Já o mais longo foi o da quinta Release, sendo gastos 30 horas de desenvolvimento. O tempo total de horas estimadas para os Sprints foi de 169 horas, tendo uma soma ao decorrer do desenvolvimento de aproximadamente 9 semanas, com trabalhos de 20 horas semanais.

O layout da aplicação foi desenvolvido utilizando as classes do Bootstrap, um framework para desenvolvimento *front-end* considerado responsivo e ideal para aplicações web que desejam a versatilidade de uso em desktops, tablets e smartphones. O Bootstrap conta com um conjunto de folhas de estilo, que nada mais são que classes pré estilizadas com CSS, HTML e até mesmo JS. Isso facilitou muito o desenvolvimento, por conter classes CSS pré definidas, além de classes que auxiliam na responsividade do projeto.

Como já citado, pelo fato de MongoDB ser um banco “schemaless”, isto é, sem esquema definido previamente, não há um consenso sobre como representar visualmente os dados de um banco Mongo, assim como é feito a modelagem

entidade relacional em um banco de dados relacional. Por este motivo, optou-se por representar o esquema do mongo utilizando algo parecido com o esquema relacional já conhecido, porém sem cardinalidades, apenas referências. A figura 9 representa os dados presentes nos documentos de cada coleção e como eles estão dispostos no schema.

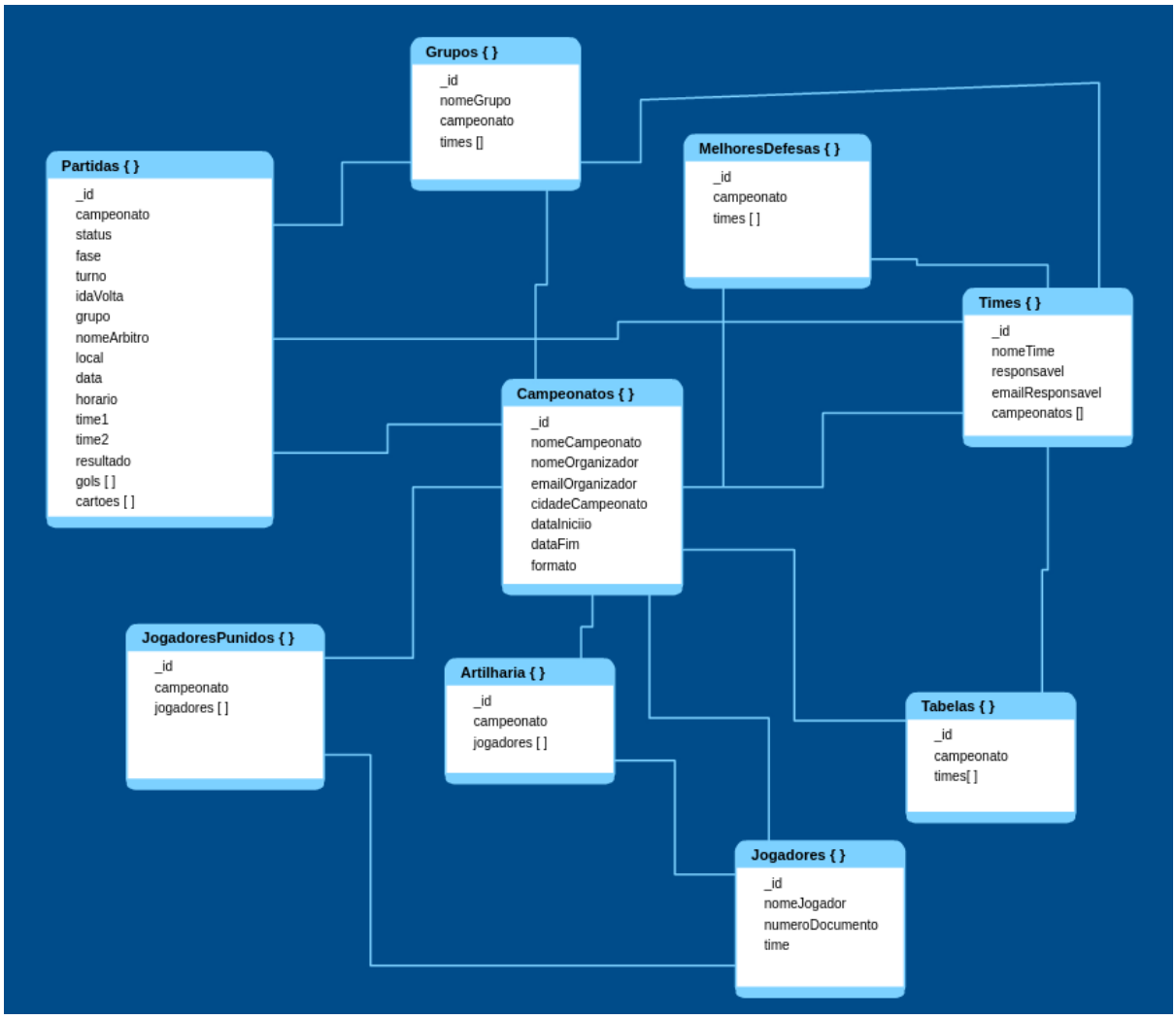


Figura 9 - Schema de dados MongoDB. Fonte (Autor).

5.2.1 Sprint 1 - Usuário

Ao planejar este primeiro Sprint do projeto, foi determinado que seria realizada o desenvolvimento da funcionalidade para autenticação de usuário do sistema. Cada usuário autenticado possui um login e uma senha, e tem a possibilidade de cadastrar campeonatos, times, jogadores, grupos e partidas ao sistema. Usuários não autenticados apenas têm acesso à página inicial da aplicação, que conta com as informações de cada campeonato, como classificação, artilharia, melhores defesas, jogadores punidos, partidas realizadas e próximas partidas. O desenvolvimento desta Sprint levou aproximadamente 8 horas, sendo a mais curta do projeto.

Como citado anteriormente na seção 5.1.1, o framework utilizado Meteor possui um vasto repositório de pacotes. Estes pacotes têm diferentes funções. Um dos pacotes utilizado foi o pacote_accounts-password. Este pacote tem como função facilitar e dar agilidade para desenvolvedores que necessitam de um sistema com autenticação de usuário. Ou seja, a função de realizar a autenticação do usuário fica por conta do pacote, sem o desenvolvedor precisar modelar as coleções no banco de dados e nem programar as funções de autenticação. O principal trabalho desenvolvido nesta Sprint, foi chamar a função de autenticação fornecida pelo pacote e adequá-la no layout da aplicação.

Como resultado da Sprint obteve-se a seguinte tela de autenticação do usuário mostrada na figura 10.

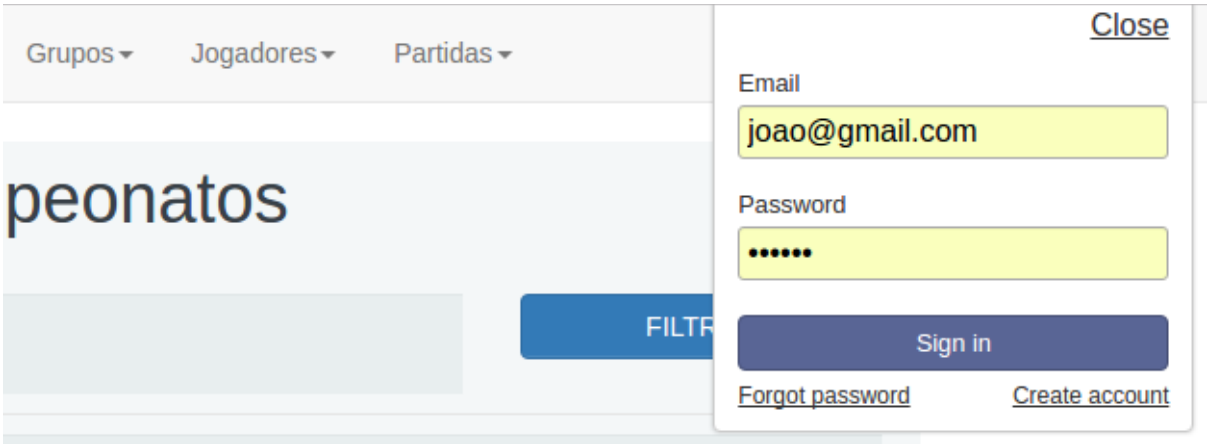


Figura 10 - Autenticação de usuário. Fonte (Autor).

Por fim, a primeira Sprint foi considerada um sucesso e cumpriu com seus requisitos, que eram o de permitir que usuários se cadastrem, além de realizar a autenticação com suas informações de login e senha. As páginas restritas aos usuários autenticados também puderam ser configuradas, evitando que usuários não autenticados realizassem operações restritas por aqueles autenticados.

5.2.2 Sprint 2 - Campeonato

A segunda Sprint foi destinada ao cadastro de Campeonato e teve duração de 20 horas. No Sprint Backlog foram adicionados três itens: Cadastrar campeonato, Alterar campeonato e Excluir campeonato.

Foram desenvolvidas duas telas neste Sprint. A primeira tela, na figura 11, é a tela onde o usuário consegue ver os seus campeonatos já cadastrados com botões para alterá-los e excluí-los. Ao clicar no botão de Excluir, o usuário exclui permanentemente o campeonato selecionado. Ao clicar no botão de Alterar, uma variável de sessão é setada com o id do campeonato, e o usuário é redirecionado à

mesma tela onde é cadastrado um novo campeonato, porém com os campos preenchidos com as informações do campeonato escolhido para se alterar, obtidos através da variável de sessão com o id do campeonato. A segunda tela, representada na figura 12, é a tela onde o usuário cadastra um novo campeonato, ou altera um campeonato já existente, baseado na opção escolhida pelo usuário. Se o usuário clicou no botão de alterar presente na listagem de campeonatos, ele vai alterar o torneio escolhido. Se o usuário clicou na opção do menu para um novo campeonato, ele irá cadastrar um novo torneio.

É importante explicar os formatos disponíveis para o campeonato. Existem 3 formatos diferentes: Copa, Liga e Mata-mata. No formato de Copa, o usuário deverá cadastrar grupos de times que disputarão entre si a fase classificatória. Os primeiros colocados, garantem vaga nas próximas fases eliminatórias, que podem ser oitavas, quartas ou qualquer outra fase escolhida pelo organizador do campeonato. No formato de Liga, os times se enfrentam todos contra todos, com possibilidade de turno e returno. O campeão é aquele que obtiver mais pontos ao decorrer da disputa. Já no último formato, o de mata-mata, o campeonato possui somente as fases eliminatórias, com opções de jogo de ida e volta, até chegar a final do campeonato, onde o ganhador consagra-se campeão da disputa.

Figura 11 - Listagem de campeonatos. Fonte (Autor).

Figura 12 - Cadastro de campeonato. Fonte (Autor).

Por fim, esta Sprint foi considerada um sucesso levando em consideração seus requisitos. Após este Sprint, os usuários que testaram o protótipo puderam

criar campeonatos, alterá-los e excluí-los. Tudo isso através de uma interface dinâmica e de forma rápida e simples, sem muitas complicações.

5.2.3 Sprint 3 - Time

A terceira Sprint foi destinada ao cadastro de Time e teve duração de 20 horas. No Sprint Backlog foram adicionados três itens: Cadastrar time, Alterar time e Excluir time.

Seguindo o padrão proposto nas Sprints anteriores, foram desenvolvidas duas telas. A primeira tela, na figura 13, é a tela onde o usuário consegue ver os seus times já cadastrados, com botões para visualizar os jogadores deste time, alterá-los e excluí-los. Ao clicar no botão de Jogadores, uma variável de sessão é setada com o id do time e o usuário é redirecionado para a tela de listagem de jogadores, onde são filtrados apenas aqueles jogadores pertencentes ao time selecionado, obtido através da variável de sessão com o id do time. Ao clicar no botão de Excluir, o usuário exclui permanentemente o time selecionado. Ao clicar no botão de Alterar, uma variável de sessão é setada com o id do time, e o usuário é redirecionado à mesma tela onde é cadastrado um novo time, porém com os campos preenchidos com as informações do time escolhido para se alterar, obtidos através da variável de sessão com o id do time. A segunda tela, representada na figura 14, é a tela onde o usuário cadastra um novo time, ou altera um time já existente, baseado na opção escolhida pelo usuário. Se o usuário clicou no botão de alterar presente na listagem de times, ele vai alterar o time escolhido. Se o usuário clicou na opção do menu para um novo time, ele irá cadastrar um novo time. Em

ambas opções o usuário pode preencher as informações do time e escolher os campeonatos que o time participará, sendo possível mais de um campeonato por time.

Figura 13 - Listagem de times. Fonte (Autor).

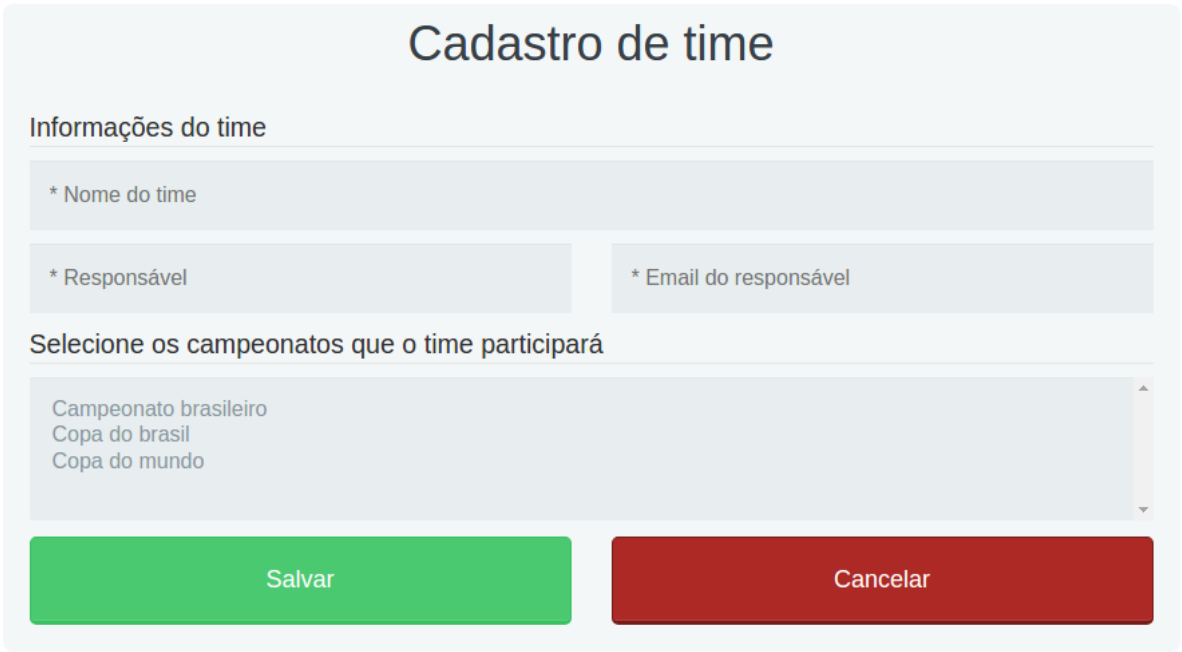


Figura 14 - Cadastro de time. Fonte (Autor)

Por fim, a Sprint foi considerada um sucesso levando em consideração seus requisitos. Após este Sprint, os usuários que testaram o protótipo puderam criar times, alterá-los e excluí-los, cumprindo com as especificações levantadas na análise de requisitos.

5.2.4 Sprint 4 - Jogador

A quarta Sprint foi destinada ao cadastro de Jogador e durou cerca de 25 horas. Assim como nas Sprints anteriores, foram adicionados três itens no Sprint Backlog: Cadastrar jogador, Alterar jogador e Excluir jogador.

No mesmo formato das duas Sprints anteriores, foram desenvolvidas duas telas. A primeira tela, na figura 15, é a tela onde o usuário consegue ver os seus jogadores já cadastrados com botões para alterá-los e excluí-los. Ao clicar no botão

de Excluir, o usuário exclui permanentemente o jogador selecionado. Ao clicar no botão de Alterar, uma variável de sessão é setada com o id do jogador, e o usuário é redirecionado à mesma tela onde é cadastrado um novo jogador, porém com os campos preenchidos com as informações do jogador escolhido para se alterar, obtidos através da variável de sessão com o id do jogador. A segunda tela, representada na figura 16, é a tela onde o usuário cadastra um novo jogador, ou altera um jogador já existente, baseado na opção escolhida pelo usuário. Se o usuário clicou no botão de alterar presente na listagem de jogadores, ele vai alterar o jogador escolhido. Se o usuário clicou na opção do menu para um novo jogador, ele irá cadastrar um novo jogador. Em ambas opções o usuário pode preencher as informações do jogador e escolher o time que o jogador atuará, sendo possível apenas um time por jogador.

Figura 15 - Listagem de jogadores. Fonte (Autor).

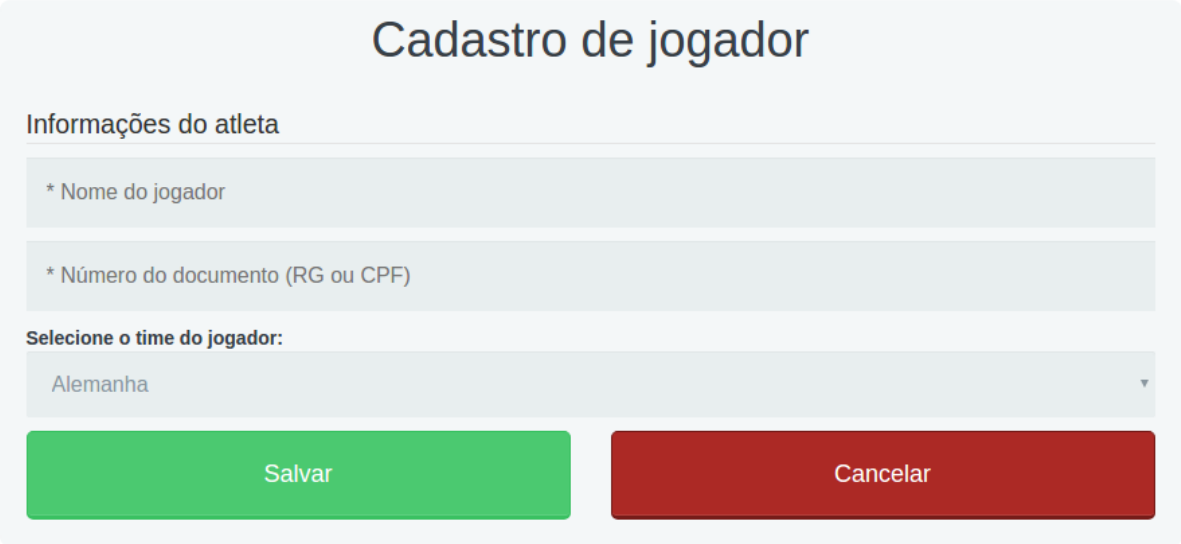


Figura 16 - Cadastro de jogador. Fonte (Autor).

Por fim, a Sprint foi considerada um sucesso levando em consideração seus requisitos. Após este Sprint, os usuários que testaram o protótipo puderam criar jogadores, alterá-los e excluí-los, cumprindo com as especificações levantadas na análise de requisitos.

5.2.5 Sprint 5 - Grupo

A quinta Sprint foi destinada ao cadastro de Grupo e durou cerca de 25 horas. Assim como nas Sprints anteriores, foram adicionados três itens no Sprint Backlog: Cadastrar grupo, Alterar grupo e Excluir grupo.

Como já definido, a Sprint também seguiu o padrão proposto no projeto, com duas telas desenvolvidas. A primeira tela, na figura 17, é a tela onde o usuário consegue ver os seus jogadores já cadastrados com botões para alterá-los e excluí-los. Ao clicar no botão de Excluir, o usuário exclui permanentemente o grupo

selecionado. Ao clicar no botão de Alterar, uma variável de sessão é setada com o id do grupo, e o usuário é redirecionado à mesma tela onde é cadastrado um novo grupo, porém com os campos preenchidos com as informações do grupo escolhido para se alterar, obtidos através da variável de sessão com o id do grupo. A segunda tela, representada na figura 18, é a tela onde o usuário cadastra um novo grupo, ou altera um grupo já existente, baseado na opção escolhida pelo usuário. Se o usuário clicou no botão de alterar presente na listagem de grupos, ele vai alterar o grupo escolhido. Se o usuário clicou na opção do menu para um novo grupo, ele irá cadastrar um novo grupo. Em ambas opções o usuário pode preencher as informações do grupo e escolher o campeonato a qual pertence, além de selecionar os times que estarão presentes no grupo. Um grupo só poderá pertencer a um campeonato, se este campeonato tiver o formato de Copa, já explicado anteriormente na seção 5.2.2.

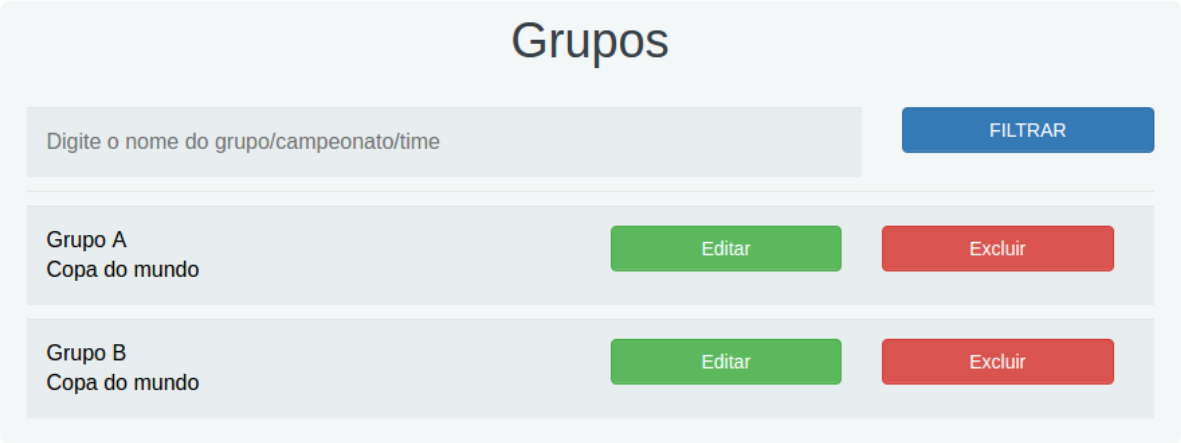


Figura 17 - Listagem de grupos. Fonte (Autor).

Figura 18 - Cadastro de grupo. Fonte (Autor).

Por fim, a Sprint foi considerada um sucesso levando em consideração seus requisitos. Após este Sprint, os usuários que testaram o protótipo puderam criar grupos, alterá-los e excluí-los, cumprindo com as especificações levantadas na análise de requisitos.

5.2.6 Sprint 6 - Partida

O planejamento feito para o sexto Sprint foi destinado a realização do cadastro de partidas. Este Sprint foi o que demandou mais tempo de desenvolvimento, com cerca de 30 horas gastas. O Sprint Backlog contou com três itens: cadastrar partida, alterar partida e excluir partida.

Assim como todas as outras Sprints, as telas desenvolvidas foram duas. A primeira tela, representada na figura 19, conta com a listagem de partidas já cadastradas pelo usuário e com os botões de edição e exclusão. Ao clicar no botão de Excluir, o usuário exclui permanentemente a partida selecionada. Ao clicar no botão de Alterar, uma variável de sessão é setada com o id da partida, e o usuário é redirecionado à mesma tela onde é cadastrado uma nova partida, porém com os campos preenchidos com as informações da partida escolhido para se alterar, obtidos através da variável de sessão com o id da partida.

A segunda tela obtida como resultado desta Sprint é destinada para o cadastro ou alteração de uma partida e pode ser vista na figura 20. Uma partida pertence a um campeonato, a um grupo (caso o campeonato seja em formato de Copa), tem uma fase correspondente, que pode ser: Classificatória, 16º de final, 8ª de final, 4ª de final, semifinal. Também possui a informação se é uma partida do turno ou returno, caso a fase seja classificatória, além da informação se é a partida de ida, volta ou jogo único, caso seja das fases pós classificatórias. Informações dos times adversários e local, horário e data também são salvos. As informações dos eventos da partida, contam com opções para gravar os jogadores que fizeram os gols de cada um dos times, os gols contra, e os cartões amarelos e vermelhos recebidos por cada jogador dos times que se enfrentam. A partir das informações da partida, são atualizadas as tabelas de classificação, o ranking da artilharia, as defesas menos vazadas e os jogadores punidos com cartões. Esta Sprint foi a mais longa em virtude de todas as atualizações citadas anteriormente e também pelo desenvolvimento da parte onde se cadastra os eventos das partidas.

Figura 19 - Listagem de partidas. Fonte (Autor).

Cadastro de partida

Informações da partida

Copa do mundo

Final | Jogo único

Maracana | 10/05/2017 | 16:00

Brasil | Alemanha

Súmula da partida

Brasil

7

Alemanha

1

Gols Brasil

Neymar	3	
Gabriel Jesus	2	
Philippe Coutinho	2	

Jogador | **Nº Gols**

Jogador | 0 | Incluir gols

Gols Alemanha

Ozil	1	

Jogador | **Nº Gols**

Jogador | 0 | Incluir gols

Gols Contra Brasil

Jogador | **Nº Gols**

Jogador | 0 | Incluir gols

Gols Contra Alemanha

Jogador | **Nº Gols**

Jogador | 0 | Incluir gols

C. Amarelos Brasil

Jogador | **Nº C. Amarelos**

Jogador | 0 | Incluir cartão

C. Amarelos Alemanha

Jogador | **Nº C. Amarelos**

Jogador | 0 | Incluir cartão

C. Vermelhos Brasil

Jogador | **Nº C. Vermelhos**

Jogador | 0 | Incluir cartão

C. Vermelhos Alemanha

Jogador | **Nº C. Vermelhos**

Jogador | 0 | Incluir cartão

Salvar | Cancelar

Figura 20 - Cadastro de partida. Fonte (Autor).

Sendo assim, ao finalizar a Sprint, obteve-se resultados satisfatórios quanto ao cadastro de partida, cumprindo com o pré-requisitos levantados na análise de requisitos além de atingir um bom grau de responsividade e usabilidade conforme análise por usuários que testaram o protótipo. Porém, foram levantados alguns pontos pelos usuários que testaram a funcionalidade. É necessário adicionar um status à partida, com as opções de Agendada e Finalizada, assim é possível ter o controle correto das partidas que estão finalizadas e agendadas. Outra ponto proposto foi o de adicionar o nome do árbitro da partida, para melhor controle dos organizadores do campeonato. E o último ponto levantado por um dos usuários foi, na listagem das partidas aparecer a data da partida ao lado do campeonato ao qual ela pertence, assim o usuário tem melhor ideia de qual partida editar/excluir. Estes três itens foram adicionados após a finalização deste Sprint.

5.2.7 Sprint 7 - Classificação, partidas realizadas e próximas

Ao planejar o sétimo Sprint, destinado a adicionar as informações aos que desejam acompanhar os campeonatos, adicionou-se três itens ao Sprint Backlog: visualizar a classificação do campeonato ou grupo, ver as informações das partidas já realizadas de determinado campeonato e ver as informações das próximas partidas. Esta Sprint teve duração de aproximadamente 22 horas.

Esta Sprint resultou na adição de 3 colunas na tela inicial da aplicação, a única tela disponível aos usuários não autenticados. Nesta tela o usuário consegue

filtrar um campeonato para que sejam exibidas suas informações divididas em três colunas com duas linhas cada.

Na primeira coluna da primeira linha, é possível ver a informação da classificação do campeonato, conforme figura 21, podendo ser filtrada por grupo, caso seja um campeonato em forma de copa. Caso o campeonato seja um mata-mata, esta coluna ficará vazia, uma vez que não há classificação a ser exibida. A classificação conta com os times pertencentes ao campeonato ou grupo e os dados referentes à pontuação (3 pontos em caso de vitória, 1 ponto em caso de empate e nenhum ponto em caso de derrota), número de partidas realizadas, número de vitórias, número de empates, número de derrotas, número de gols feitos, número de gols sofridos e saldo de gols. A ordem da classificação segue os seguintes critérios: maior número de pontos, maior número de vitórias, saldo de gols, gols pró e gols contra.

Na segunda coluna da primeira linha, é possível ver as informações dos jogos já realizados, conforme figura 22, podendo ser filtrado por grupo em caso de copas. As informações que aparecem são o resultado da partida, os times adversários, a data do jogo, o local, e qual fase/grupo a partida pertence.

Na terceira coluna da primeira linha, é possível ver as informações dos jogos agendados para aquele campeonato ou grupo em caso de copas, conforme figura 23. As informações que aparecem são os times adversários, a data do jogo, o local, e qual fase/grupo a partida pertence.

Classificação

Grupo Grupo A

Nome do time	Pontos	J	V	E	D	GP	GS	SG
Avai	9	3	3	0	0	9	3	6
Criciúma	6	3	2	0	1	4	3	1
Chapecoense	3	3	1	0	2	3	6	-3
Figueirense	0	3	0	0	3	2	6	-4

Figura 21 - Tabela de classificação. Fonte (Autor).

Figura 22. Lista das partidas realizadas. Fonte (Autor).

Figura 23 - Lista das próximas partidas. Fonte (Autor).

Ao fim da Sprint, foi possível concluir que os requisitos levantados na fase de análise cumpriram com o seu objetivo principal, que era prover aos simpatizantes, jogadores ou qualquer outro tipo de pessoa a acompanhar a classificação, os resultados dos jogos, e os jogos a serem realizados.

5.2.8 Sprint 8 - Artilharia, melhores defesas e punições

O oitavo e último Sprint previsto para o projeto, teve como principal objetivo disponibilizar os rankings e estatísticas do campeonato. O planejamento contou com a adição de três itens no Sprint Backlog: visualizar o ranking de artilharia, visualizar as melhores defesas do campeonato, visualizar os jogadores punidos com cartão amarelo. O Sprint levou cerca de 18 horas para ser concluído.

Completando a Sprint anterior, esta resultou na adição de mais 3 colunas na tela inicial da aplicação, onde já existia as colunas adicionadas pela Sprint anterior.

Na primeira coluna da segunda linha da tela, foi disponibilizada as informações referentes à artilharia do campeonato, conforme figura 24. A artilharia do campeonato é composta pelo nome do jogador e número de gols dele, e nada mais é que um ranking em ordem decrescente dos jogadores com mais gols marcados nas partidas daquele campeonato. O número de gols de um jogador é a soma de todos os gols dele no campeonato.

Na segunda coluna da segunda linha, aparecem as informações das melhores defesas do campeonato, conforme figura 25. As melhores defesas é um ranking em ordem crescente composto dos nomes das equipes e o número de gols que elas sofreram. A equipe que possui menos gols sofridos nas partidas disputadas daquele torneio aparece em primeiro, e assim sucessivamente, formando o ranking de melhores defesas.

Na terceira coluna da segunda linha, aparecem as informações dos jogadores punidos com cartões amarelos e vermelhos, conforme figura 26. O jogador que houver recebido mais cartões amarelos e vermelhos no campeonato aparece no topo da lista.

Artilharia

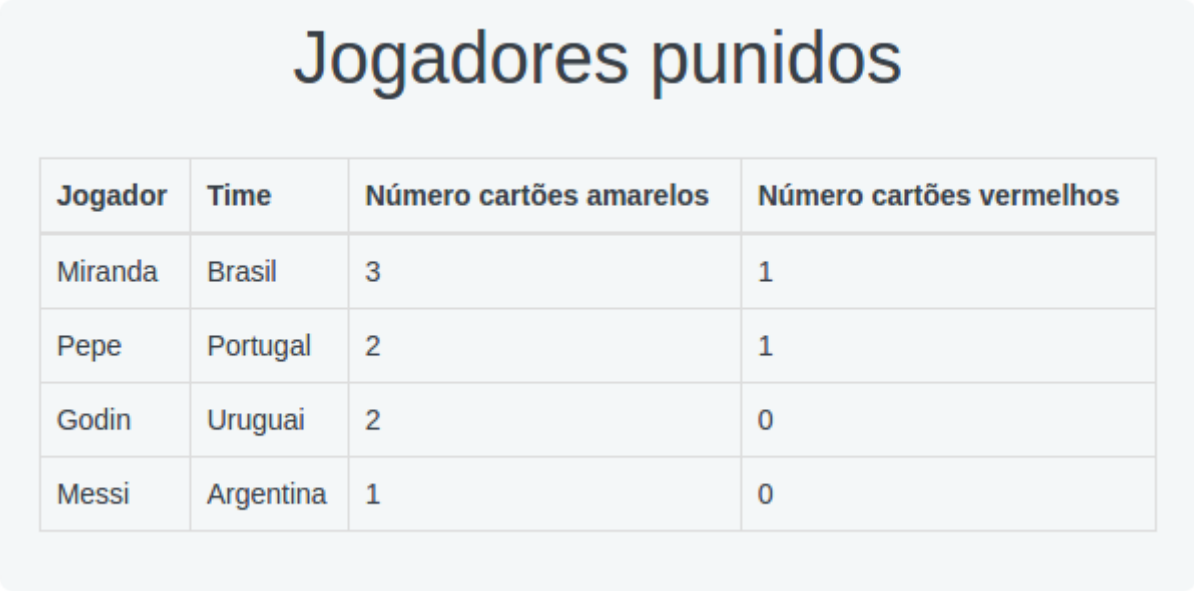
Jogador	Time	Número de gols
Neymar	Brasil	8
Gabriel Jesus	Brasil	6
Messi	Argentina	5
Di Maria	Argentina	3
Philippe Coutinho	Brasil	3
Kedhira	Alemanha	1
Ozil	Alemanha	1

Figura 24 - Tabela de artilharia. Fonte (Autor).

Melhores defesas

Nome do Time	Número de gols sofrido
Brasil	3
Argentina	5
Alemanha	6
Portugal	9
Chile	13
Espanha	13

Tabela 25 - Tabela de melhores defesas. Fonte (Autor).



Jogadores punidos

Jogador	Time	Número cartões amarelos	Número cartões vermelhos
Miranda	Brasil	3	1
Pepe	Portugal	2	1
Godin	Uruguai	2	0
Messi	Argentina	1	0

Figura 26 - Tabela de jogadores punidos. Fonte (Autor).

No fim desta Sprint, o resultado foi tido como satisfatório, pois cumpriu com os pré-requisitos levantados previamente. Com isto, qualquer usuário que deseja ver as estatísticas de artilharia, gols sofridos e cartões, pode ver esta informação de forma clara e dinâmica.

6 Conclusão

Com o desenvolvimento deste trabalho, concluiu-se que uma aplicação para gerenciamento de campeonatos de futebol torna-se um grande auxílio para organizadores deste tipo de evento. A organização de um campeonato realizada via planilhas de Excel e folhas de papel demanda muito mais tempo que utilizando uma aplicação destinada à isso. Apesar de já possuir alguns softwares no mercado com esta finalidade, este trabalho pôde agregar principalmente no quesito simplicidade e

dinamismo, além de alto grau de usabilidade observada pelos usuários teste. A responsividade também foi um ponto positivo para aplicação, algo que hoje em dia é essencial.

Os desafios encontrados para desenvolver a aplicação, foram principalmente os desefios quanto à linguagem javascript, pois o autor do projeto não havia muita experiência com a linguagem. Apesar disso o apredizado se mostrou bastante rápido muito em função do framework utilizado. Os conceitos de um banco de dados não relacional também se mostraram como um grande desafio ultrapassado, uma vez que o autor estava acostumado com o tipo relacional.

A partir da metodologia Scrum, foi possível organizar suficientemente o trabalho e atingir todos os requisitos levantados junto ao usuário com experiência em organização de campeonatos. O Scrum permitiu ao autor atingir grande agilidade no desenvolvimento da aplicação, graças a sua forma de organizar os Sprints e ao seu processo cíclico. Permitiu também que a cada etapa do desenvolvimento, o protótipo fosse se aproximando cada vez mais da forma final, facilitando os testes feitos tanto pelo próprio desenvolvedor, como pelos usuários escolhidos para testar o protótipo.

Concluiu-se também que o uso do framework MeteorJS foi de grande benefício ao autor. Mesmo sem ter amplo conhecimento na linguagem javascript, foi possível obter um bom resultado ao desenvolver usando este framework. Tudo isso graças a uma série de vantagens que inclusive são levantadas pelos autores do framework. Ótima curva de aprendizado muito por causa de ser um framework fullstack. Simplicidade, pois não é necessário realizar importações e configurações. É tudo desenvolvido em uma só linguagem, do banco ao back-end. Têm uma ótima

comunidade na internet, desde forums até blogs especializados. Grande número de pacotes com diferentes funcionalidades. Todos estes pontos foram essenciais para o aprendizado do framework e desenvolvimento da aplicação proposta.

Por fim, através de conversa com os usuários que testaram a aplicação, avaliou-se que a aplicação atinge o objetivo proposto, que é o de gerenciar um campeonato de futebol e permitir que usuários acompanhem os campeonatos que desejam. Há alguns pontos que podem melhorar e que serão citados na seção de Trabalhos Futuros, mas de maneira geral o trabalho atingiu seus objetivos traçados no início do desenvolvimento.

6.1 Trabalhos futuros

Para trabalhos futuros, existem uma série de requisitos que podem ser implementados e funcionalidades melhoradas. A automatização na geração de partidas pode ser o ponto principal que fará com que a aplicação se torne completa e possa bater de frente com outras aplicações do mercado.

O uso de ícones pode ser um item que pode ser implementado, baseando-se nas opiniões dos usuários que realizaram o teste da aplicação. Além disso, imagens para identificar os times e campeonatos também são pontos que podem ser implementados. A personalização do critério de desempate também foi levantada como opção de melhoria por um dos usuários que testaram a aplicação. Além disso, seria bastante eficiente uma otimização no código fonte para evitar redundâncias e trazer mais agilidade ao protótipo e correção de pequenos bugs.

Ou seja, mesmo cumprindo com seu objetivo principal proposto no início do trabalho, a aplicação possui alguns pontos que podem melhorar ainda mais a usabilidade do usuário e facilitar a vida dele na hora de gerar os confrontos e, consequentemente, gerenciar o campeonato.

Referências

MongoDB. MONGODB for Giant ideas. Disponível em: <https://www.mongodb.com/mongodb-architecture>. Acesso em: 30 de Junho de 2016.

Meteor. METEOR, the fatest way to build JS APPS. Disponível em: <https://www.meteor.com/>. Acesso em: 30 de Junho de 2016.

Meteor. Meteor Guide. Disponível em: <https://guide.meteor.com>. Acesso em: 30 de outubro de 2016.

SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide.** Disponível em: <http://www.scrumguides.org/scrum-guide.html>. Acesso em 15 dez. 2016.

FAYAD, Mohamed; SCHMIT, Douglas; JOHNSON, Ralph. **Object-Oriented Application Frameworks.** Disponível em: <http://www.cs.wustl.edu/~schmidt/CACM-frameworks.html>. Acesso em 15 jan. 2017.

PRESSMAN, R. *Engenharia de Software*, Rio de Janeiro: McGraw Hill, 2002.

SOMMERVILLE, Ian. **Engenharia de software.** Tradução por Selma Shin Shimizu Melnikoff; Reginaldo Arakaki; Edilson de Andrade Barbosa. 8. ed. São Paulo.

Ritmo do esporte. Disponível em: <http://www.ritmodoesporte.com.br>. Acesso em 20 mar. 2017.

Campeonato de Verão. Disponível em: <http://www.campeonatodeverao.com.br>. Acesso em 20 mar. 2017.

ANEXO A - Código Fonte

/artilharia.js

```
Artilharias = new Mongo.Collection("artilharias");
```

/campeonatos.js

```
Campeonatos = new Mongo.Collection("campeonatos");
```

/grupos.js

```
Grupos = new Mongo.Collection("grupos");
```

/jogadores.js

```
Jogadores = new Mongo.Collection("jogadores");
```

/jogadoresPunidos.js

```
JogadoresPunidos = new Mongo.Collection("jogadoresPunidos");
```

/melhoresDefesas.js

```
MelhoresDefesas = new Mongo.Collection("melhoresDefesas");
```

/partidas.js

```
Partidas = new Mongo.Collection("partidas");
```

/tabelas.js

```
Tabelas = new Mongo.Collection("tabelas");
```

/times.js

```
Times = new Mongo.Collection("times");
```

/router.js

```
Router.configure({

});

Router.map(function() {
    this.route('home', {
        path: '/',
    });

    this.route('novoCampeonato');
    this.route('listaCampeonatos');

    this.route('novoTime')  ;
    this.route('listaTimes');

    this.route('novoJogador');
    this.route('listaJogadores');

    this.route('novaPartida');
    this.route('listaPartidas');

    this.route('cadastroGrupo');
    this.route('listaGrupos');

});
```

```
{
  "name": "meuCampeonato",
  "private": true,
  "scripts": {
```

```
    "start": "meteor run"
  },
  "dependencies": {
    "babel-runtime": "^6.20.0",
    "bcrypt": "^1.0.2",
    "meteor-node-stubs": "~0.2.4"
  }
}
```

```
<template name="pageNotFound">

        {{> nav}}

    <div class="container">
      <h3>404 - Página não encontrada</h3>
      <h5>A págna que você procura não foi encontrada</h5>
    </div>

    {{> footer}}

</template>
```

```
<template name="pageNotFound">

        {{> nav}}

    <div class="container">
      <h3>404 - Página não encontrada</h3>
      <h5>A págna que você procura não foi encontrada</h5>
    </div>

    {{> footer}}

</template>
```

```
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1,
user-scalable=no">
  <title>Gerenciador de Campeonatos</title>
</head>

<body>
```

```
</body>
<template name="footer">
  <div class="container">
    <hr>
    <p>
      Desenvolvido por João Mafra.
    </p>
  </div>
</template>
```

```
Template.novoTime.onDestroyed(function(){
        Session.set('idTime', null)
});

Template.novoTime.helpers({
    campeonatos: function(){
        return Campeonatos.find({}, { sort: {nomeCampeonato: 1} });
    },

        times: function(){
                var idTime = Session.get('idTime');
                var listaTimes = Times.findOne({_id: idTime});

                return listaTimes;
        },

        campeonatoSelecionado(idCampeonato){
                if(Session.get('idTime')){
                        var selecionado = '';
                        var idTime = Session.get('idTime');
                        var listaTimes = Times.findOne({_id: idTime});

                        for (var i = 0; i < listaTimes.campeonatos.length; i++) {
                            if(idCampeonato == listaTimes.campeonatos[i].idCampeonato){
                                        return 'selected';
                                }
                        }

                }
        },
});

Template.novoTime.events({

        'submit .novo-time': function(event){
```

```
                        if(event.target.nomeTime.value)
                                var nomeTime = event.target.nomeTime.value;
                        if(event.target.responsavel.value)
                                var responsavel = event.target.responsavel.value;
                        if(event.target.emailResponsavel.value)
                                var emailResponsavel = event.target.emailResponsavel.value;

                        var campeonatos = [];

                        $('#campeonato-select option:selected').each(function() {
                           campeonatos.push({ "idCampeonato" : this.value,
"nomeCampeonato" : this.textContent });
                        });

                        if(campeonatos.length < 1){
                                campeonatos = undefined;
                        }

                        if(Session.get('idTime')){
                                Times.update(
                                  { _id: Session.get('idTime') },
                                  {
                                        nomeTime : nomeTime,
                                                nomeResponsavel : responsavel,
                                                emailResponsavel : emailResponsavel,
                                                campeonatos
                                  }
                                )
                        }
                        else{
                                Times.insert({
                                        nomeTime : nomeTime,
                                        nomeResponsavel : responsavel,
                                        emailResponsavel : emailResponsavel,
                                        campeonatos
                                });
                        }

                        alert("Time salvo!");
                        document.getElementById(".novo-time").reset();

                        return false;
                },

           "change #campeonato-select": function (event, template) {
```

```
        var campeonato = $(event.currentTarget).val();
        console.log("campeonato : " + campeonato);
    },

          'click .cancela': function(){
                    event.preventDefault();
          Router.go('listaTimes');
          }
});
```

```
<template name="novoTime">
{{> nav}}
    <form class="novo-time">
        <h1>Cadastro de time</h1>

        <legend>Informações do time</legend>

        <input type="text" name="nomeTime" placeholder="* Nome do time"
value={{times.nomeTime}} required>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
            <input type="text" name="responsavel" placeholder="* Responsável"
value={{times.nomeResponsavel}} required>
          </div>
          <div class="col-sm-6 col-xs-12">
            <input type="text" name="emailResponsavel" placeholder="* Email do responsável"
value={{times.emailResponsavel}} required>
          </div>
        </div>

        <legend>Selecione os campeonatos que o time participará</legend>

        <select id="campeonato-select" multiple>
          {{#each campeonatos}}
            <option name="idCampeonatos" value={{_id}} {{campeonatoSelecionado
_id}}>{{nomeCampeonato}}</option>
          {{/each}}
        </select>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
            <button type="submit" name="cadastrar" >Salvar</button>
          </div>
          <div class="col-sm-6 col-xs-12">
```

```
          <button class="cancela" formnovalidate>Cancelar</button>
        </div>
      </div>
  </form>

{{> footer}}
</template>
```

```
Template.listaTimes.onDestroyed(function(){
        Session.set('filtroTime', undefined)
});

Template.listaTimes.helpers({

        times : function() {
                var valorFiltro = Session.get('filtroTime');
                return Times.find({
                        $or: [
                                { nomeTime: {$regex : new RegExp(valorFiltro, "i") } },
                                { "campeonatos.nomeCampeonato": {$regex : new
RegExp(valorFiltro, "i") } }
                        ]
                },
                {sort: {nomeTime: 1}});
        }
});

Template.listaTimes.events({

        'click .remove': function(){
                var timeID = this._id;
                var a = confirm("Você tem certeza que deseja excluir?");
                if (a == true) {
                    Times.remove({ _id: timeID });
                }
    },

    'click .filtra': function(event, template){
        var valorFiltro = $('#filtro').val();
        Session.set('filtroTime', valorFiltro);
    },

    'click .edita': function(event, template) {
        event.preventDefault();
        Session.set('idTime', this._id);
```

```
        Router.go('novoTime');

        },

    'click .jogadores': function(event, template) {
        event.preventDefault();
        Session.set('idTime', this._id);
        Router.go('listaJogadores');

        }
});
```

```
<template name="listaTimes">
        {{> nav}}

        <div class="fundo">
                <h1>Times</h1>

                <div class="row">
                        <div class="col-xs-9">
                                <input type="text" id="filtro" name="filtroTime"
placeholder="Digite o nome do time/campeonato">
                        </div>
                        <div class="col-xs-3">
                                <button class="btn btn-primary filtra">FILTRAR</button>
                        </div>
                </div>

                <div class="separador"></div>

                {{#each times}}
                        <div class="row fundo-listagem">
                                <div class="col-xs-3">
                                        <div class="row">
                                                {{nomeTime}}
                                        </div>
                                        <div class="row">
                                                {{nomeResponsavel}}
                                        </div>
                                </div>
                                <div class="col-xs-3">
                                        <button class="btn btn-info
jogadores">Jogadores</button>
                                </div>
                                <div class="col-xs-3">
```

```
                                        <button class="btn btn-success edita">Editar</button>
                                    </div>
                                    <div class="col-xs-3">
                                        <button class="btn btn-danger
remove">Excluir</button>
                                    </div>
                                </div>
                        {{/each}}
                </div>

                {{> footer}}
</template>
```

```
Template.novoJogador.onDestroyed(function(){
        Session.set('idJogador', null)
});

Template.novoJogador.helpers({
   times: function(){
      return Times.find({}, { sort: {nomeTime: 1} });

   },

   jogadores: function(){
                var idJogador = Session.get('idJogador');
                var listaJogadores = Jogadores.findOne({_id: idJogador});

                return listaJogadores;
        },

        timeSelecionado(idTime){
                if(Session.get('idJogador')){
                        var idJogador = Session.get('idJogador');
                        var listaJogadores = Jogadores.findOne({_id: idJogador});
                        if(idTime == listaJogadores.idTime){
                                return 'selected';
                        }
                        else{
                                return '';
                        }
                }
        }
});

Template.novoJogador.events({
```

```
    'submit .novo-jogador': function(event){

                        var nomeJogador = event.target.nomeJogador.value;
                        var numeroDocumento = event.target.numeroDocumento.value;
                        var idTime = $("#time-select option:selected").val();
                        var nomeTime = $("#time-select option:selected").text();

                        if(Session.get('idJogador')){
                                Jogadores.update(
                                        { _id: Session.get('idJogador') },
                                        {
                                        nomeJogador : nomeJogador,
                                        numeroDocumento : numeroDocumento,
                                        idTime: idTime,
                                });
                        }
                        else{
                                Jogadores.insert({
                                        nomeJogador : nomeJogador,
                                        numeroDocumento : numeroDocumento,
                                        idTime: idTime
                                });
                        }

                        alert("Jogador salvo!");
                        document.getElementById(".novo-jogador").reset();

                        return false;
        },

    "change #time-select": function (event, template) {
        var time = $(event.currentTarget).val();
    },

        'click .cancela': function(){
                event.preventDefault();
        Router.go('listaJogadores');
        }

});
```

```
<template name="novoJogador">
{{> nav}}
   <form class="novo-jogador">

      <h1>Cadastro de jogador</h1>

        <legend>Informações do atleta</legend>

        <input type="text" name="nomeJogador" placeholder="* Nome do jogador"
value={{jogadores.nomeJogador}} required>

        <input type="text" name="numeroDocumento" placeholder="* Número do documento
(RG ou CPF)" value={{jogadores.numeroDocumento}} required>

        <b>Selecione o time do jogador:</b>
        <select id="time-select" required>
                    {{#each times}}
                       <option name="idTime" value="{{_id}}" {{timeSelecionado
_id}}>{{nomeTime}}</option>
                    {{/each}}
                  </select>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
            <button type="submit" name="cadastrar" >Salvar</button>
          </div>
          <div class="col-sm-6 col-xs-12">
            <button class="cancela" formnovalidate>Cancelar</button>
          </div>
        </div>

   </form>
{{> footer}}
</template>
```

```
Template.listaJogadores.onDestroyed(function(){
        Session.set('filtroJogador', undefined)
        Session.set('idTime', undefined)
});

Template.listaJogadores.helpers({

        jogadores : function() {
                if(Session.get('idTime')){
```

```
                return Jogadores.find(
                        { idTime: Session.get('idTime')},
                        {sort: {nomeJogador: 1}}
                    );
            }
            var valorFiltro = Session.get('filtroJogador');
            return Jogadores.find(
                        { nomeJogador: {$regex : new
RegExp(valorFiltro, "i") } },
                        {sort: {nomeJogador: 1}}
                    );


    },

    nomeTime : function(idTime){
            var time = Times.findOne({_id: idTime});
            return time.nomeTime;
    }
});

Template.listaJogadores.events({

    'click .remove': function(){
            var jogadorID = this._id;
            var a = confirm("Você tem certeza que deseja excluir?");
            if (a == true) {
               Jogadores.remove({ _id: jogadorID });
            }
  },

  'click .filtra': function(event, template){
        var valorFiltro = $('#filtro').val();
        Session.set('filtroJogador', valorFiltro);
  },

  'click .edita': function(event, template) {
        event.preventDefault();
        Session.set('idJogador', this._id);
        Router.go('novoJogador');

        }
});
```

```
<template name="listaJogadores">
```

```
	{{> nav}}

	<div class="fundo">
		<h1>Jogadores</h1>

		<div class="row">
			<div class="col-xs-9">
				<input type="text" id="filtro" name="filtroJogador"
placeholder="Digite o nome do jogador">
			</div>
			<div class="col-xs-3">
				<button class="btn btn-primary filtra">FILTRAR</button>
			</div>
		</div>

		<div class="separador"></div>

		{{#each jogadores}}
			<div class="row fundo-listagem">
				<div class="col-xs-6">
					<div class="row">
						{{nomeJogador}}
					</div>
					<div class="row">
						{{nomeTime idTime}}
					</div>
				</div>
				<div class="col-xs-3">
					<button class="btn btn-success edita">Alterar</button>
				</div>
				<div class="col-xs-3">
					<button class="btn btn-danger
remove">Excluir</button>
				</div>
			</div>
		{{/each}}
	</div>

	{{> footer}}
</template>

---

Template.listaGrupos.onDestroyed(function(){
	Session.set('filtroGrupo', undefined)
});
```

```
Template.listaGrupos.helpers({

	grupos : function() {
		var valorFiltro = Session.get('filtroGrupo');
		return Grupos.find(
							{ nomeGrupo: {$regex : new RegExp(valorFiltro,
"i") } },
							{sort: { nomeGrupo: 1}}
					);
	},

	nomeCampeonato : function(idCampeonato){
		var campeonato = Campeonatos.findOne({_id: idCampeonato});
		return campeonato.nomeCampeonato;
	}
});

Template.listaGrupos.events({

	'click .remove': function(){
		var idGrupo = this._id;
		var a = confirm("Você tem certeza que deseja excluir?");
		if (a == true) {
		    Grupos.remove({ _id: idGrupo });
		}
   },

   'click .filtra': function(event, template){
	var valorFiltro = $('#filtro').val();
	Session.set('filtroGrupo', valorFiltro);
   },

   'click .edita': function(event, template) {
	event.preventDefault();
	Session.set('idGrupo', this._id);
	Router.go('cadastroGrupo');

	}
});
```

```
<template name="listaGrupos">
	{{> nav}}

	<div class="fundo">
		<h1>Grupos</h1>
```

```
                <div class="row">
                        <div class="col-xs-9">
                                <input type="text" id="filtro" name="filtroTime"
placeholder="Digite o nome do grupo">
                        </div>
                        <div class="col-xs-3">
                                <button class="btn btn-primary filtra">FILTRAR</button>
                        </div>
                </div>

                <div class="separador"></div>

                {{#each grupos}}
                        <div class="row fundo-listagem">
                                <div class="col-xs-6">
                                        <div class="row">
                                                {{nomeGrupo}}
                                        </div>
                                        <div class="row">
                                                {{nomeCampeonato idCampeonato}}
                                        </div>
                                </div>
                                <div class="col-xs-3">
                                        <button class="btn btn-success edita">Editar</button>
                                </div>
                                <div class="col-xs-3">
                                        <button class="btn btn-danger
remove">Excluir</button>
                                </div>
                        </div>
                {{/each}}
        </div>

        {{> footer}}
</template>
```

```
Template.cadastroGrupo.onCreated(function(){

        times = [];
        tabelaClassificacao = [];
        this.listaTimesGrupo = new ReactiveVar([]);
        this.listatabelaClassificacao = new ReactiveVar([]);

        if(Session.get('idGrupo')){
```

```
                var idGrupo = Session.get('idGrupo');
                var listaGrupos = Grupos.findOne({_id: idGrupo});

                if(listaGrupos.times){
                        for (var i = 0; i < listaGrupos.times.length; i++) {
                                times.push({
                                        "idTime" : listaGrupos.times[i].idTime,
                                        "nomeTime" : listaGrupos.times[i].nomeTime
                                });

                                tabelaClassificacao.push({
                                        "idTime" : listaGrupos.times[i].idTime,
                                        "nomeTime" : listaGrupos.times[i].nomeTime,
                                        "pontos" : 0,
                                        "jogos"  : 0,
                                        "vitorias": 0,
                                        "empates": 0,
                                        "derrotas": 0,
                                        "golsPro": 0,
                                        "golsContra": 0,
                                        "saldo": 0
                                });
                        }
                        this.listaTimesGrupo.set(times);
                        this.listatabelaClassificacao.set(tabelaClassificacao);
                }
        }
});

Template.cadastroGrupo.onDestroyed(function(){
        Session.set('idGrupo', null);
        Session.set('campeonatoSelect', null)
});

Template.cadastroGrupo.helpers({
   times: function(){
        if(Session.get('campeonatoSelect')){
                return Times.find({"campeonatos.idCampeonato":
Session.get('campeonatoSelect')}, { sort: {nomeTime: 1} });;
        }
        else if(Session.get('idGrupo')){
                var listaGrupos  = Grupos.findOne({_id: Session.get('idGrupo')});
                return Times.find({"campeonatos.idCampeonato":
listaGrupos.idCampeonato}, { sort: {nomeTime: 1} })
        }
   },
```

```
    campeonatos: function(){
        return Campeonatos.find({ formato: "1" }, { sort: {nomeCampeonato: 1} });
    },

    listaTimesGrupo(){
            return Template.instance().listaTimesGrupo.get();
    },

            grupos: function(){
                    var idGrupo = Session.get('idGrupo');
                    var listaGrupos = Grupos.findOne({_id: idGrupo});

                    return listaGrupos;
            },

            campeonatoSelecionado(idCampeonato){
                    if(Session.get('idGrupo')){
                            var idGrupo = Session.get('idGrupo');
                            var listaGrupos = Grupos.findOne({_id: idGrupo});
                            if(idCampeonato == listaGrupos.idCampeonato){
                                    return 'selected';
                            }
                            else{
                                    return '';
                            }
                    }
            }
});

Template.cadastroGrupo.events({
    'submit .novo-grupo': function(event){

                            var nomeGrupo = event.target.nomeGrupo.value;
                            var idCampeonato = $("#campeonato-select option:selected").val();
                            var nomeCampeonato = $("#campeonato-select
option:selected").text();

                            if(times.length < 1){
                                    times = undefined;
                                    tabelaClassificacao = undefined;
                            }

                            if(Session.get('idGrupo')){
                                    var idGrupo = Session.get('idGrupo');
```

```
                    var listaTabelas = Tabelas.findOne({"grupo.idGrupo":
idGrupo});

                    Grupos.update(
                        { _id: Session.get('idGrupo') },
                        {
                            nomeGrupo : nomeGrupo,
                            idCampeonato: idCampeonato,
                            times
                        });

                    if(listaTabelas){
                        if(tabelaClassificacao){
                        Tabelas.update(
                            {_id: listaTabelas._id},
                            {
                                idGrupo: idGrupo,
                                idCampeonato: idCampeonato,
                                tabelaClassificacao
                            }
                            );
                        }
                        else{
                            Tabelas.remove(
                                {_id: listaTabelas._id},
                                { justOne: true}
                            );

                        }
                    }
                    else if(tabelaClassificacao){
                        Tabelas.insert(
                            {
                                idGrupo: idGrupo,
                                idCampeonato: idCampeonato,
                                tabelaClassificacao
                            }
                        );
                    }
                }
                else{
                    var idGrupo = Grupos.insert({
                        idGrupo: idGrupo,
                        idCampeonato: idCampeonato,
                        nomeGrupo : nomeGrupo,
                        times
```

```
                        });

                        Tabelas.insert({
                                idGrupo: idGrupo,
                                idCampeonato: idCampeonato,
                                tabelaClassificacao
                        });
                }

                alert("Grupo salvo!");
                document.getElementById(".novo-grupo").reset();

                return false;
},

'click .adiciona': function(e, template){
        event.preventDefault();

        var idTime = $("#time-select option:selected").val();
        var nomeTime = $("#time-select option:selected").text();
        var idCampeonato = $("#campeonato-select option:selected").val();

        if(idTime == -1){
                return false;
        }

        if(idCampeonato == -1){
                alert("Primeiro selecione um campeonato!");
                return false;
        }

        for (var i = 0; i < times.length; i++) {
           if(idTime == times[i].idTime){
                        return false;
                }
        }

        times.push({
                "idTime" : idTime,
                "nomeTime" : nomeTime
        });

        tabelaClassificacao.push({
                "idTime" : idTime,
                "nomeTime" : nomeTime,
                "pontos" : 0,
```

```
                        "jogos"  : 0,
                        "vitorias": 0,
                        "empates": 0,
                        "derrotas": 0,
                        "golsPro": 0,
                        "golsContra": 0,
                        "saldo": 0
                });
                template.listaTimesGrupo.set(times);
                template.listatabelaClassificacao.set(tabelaClassificacao);
        },

        'click .cancela': function(){
                event.preventDefault();
        Router.go('listaGrupos');
        },

        'click .remove': function(e, template){
                event.preventDefault();
                for (var i = 0; i < times.length; i++) {
                   if(this.idTime == times[i].idTime){
                                times.splice(i, 1);
                                tabelaClassificacao.splice(i, 1);
                        }
                }
                template.listaTimesGrupo.set(times)
                template.listatabelaClassificacao.set(tabelaClassificacao)
        },

        "change #campeonato-select": function(e, t){
        return Session.set("campeonatoSelect", $("#campeonato-select
option:selected").val());
   }

});
```

```
<template name="cadastroGrupo">
{{> nav}}
   <form class="novo-grupo">

      <h1>Cadastro de grupo</h1>

        <legend>Informações do grupo</legend>

        <div class="row">
```

```
          <div class="col-sm-6 col-xs-12">
            <b>Selecione o campeonato do grupo</b>
            <select id="campeonato-select" required>
            <option disabled="disabled" value="-1" selected>Selecionar campeonato</option>
              {{#each campeonatos}}
                  <option name="idCampeonato" value="{{_id}}" {{campeonatoSelecionado
_id}}>{{nomeCampeonato}}</option>
              {{/each}}
            </select>
          </div>
          <div class="col-sm-6 col-xs-12">
          <br>
            <input type="text" name="nomeGrupo" placeholder="* Nome do grupo"
value={{grupos.nomeGrupo}} required>
          </div>
        </div>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
          <b>Adicionar time ao grupo</b>
            <select id="time-select">
            <option disabled="disabled" value="-1" selected>Primeiro selecione um
campeonato</option>
              {{#each times}}
                  <option name="idTime" value="{{_id}}">{{nomeTime}}</option>
              {{/each}}
            </select>
          </div>
          <div class="col-sm-6 col-xs-12">
          <br>
            <button class="btn btn-info adiciona">Adicionar ao grupo</button>
          </div>
        </div>
        <br>
        <legend>Times adicionados</legend>

        {{#each listaTimesGrupo}}
          <div class="row fundo-listagem">
            <div class="col-xs-6">
              <div class="row">
                {{nomeTime}}
              </div>
            </div>
            <div class="col-xs-6">
              <button class="btn btn-danger remove" formnovalidate>Remover do
grupo</button>
```

```
          </div>
        </div>
      {{/each}}


      <div class="row">
        <div class="col-sm-6 col-xs-12">
          <button type="submit" name="cadastrar" >Salvar</button>
        </div>
        <div class="col-sm-6 col-xs-12">
          <button class="cancela" formnovalidate>Cancelar</button>
        </div>
      </div>
  </form>
{{> footer}}
</template>
```

```
<template name="listaCampeonatos">
        {{> nav}}

        <div class="fundo">
                <h1>Campeonatos</h1>

                <div class="row">
                        <div class="col-xs-9">
                                <input type="text" id="filtro" name="filtroCampeonato"
placeholder="Digite o nome do campeonato">
                        </div>
                        <div class="col-xs-3">
                                <button class="btn btn-primary filtra">FILTRAR</button>
                        </div>
                </div>

                <div class="separador"></div>

                {{#each campeonatos}}
                        <div class="row fundo-listagem">
                                <div class="col-xs-6">
                                        <div class="row">
                                                {{nomeCampeonato}}
                                        </div>
                                        <div class="row">
                                                {{nomeOrganizador}}
                                        </div>
                                </div>
```

```
                                <div class="col-xs-3">
                                        <button class="btn btn-success edita"><span class="fa
fa-times"></span>Alterar</button>
                                </div>
                                <div class="col-xs-3">
                                        <button class="btn btn-danger
remove">Excluir</button>
                                </div>
                        </div>
                {{/each}}
        </div>

        {{> footer}}
</template>
```

```
Template.listaCampeonatos.onDestroyed(function(){
        Session.set('filtroCampeonato', undefined)
});

Template.listaCampeonatos.helpers({

        campeonatos : function() {
                var valorFiltro = Session.get('filtroCampeonato');
                return Campeonatos.find({nomeCampeonato: {$regex : new
RegExp(valorFiltro, "i") }},
                        { sort: {nomeCampeonato: 1} });

        }
});

Template.listaCampeonatos.events({

        'click .remove': function(){
                var campeonatoID = this._id;
                var a = confirm("Você tem certeza que deseja excluir?");
                if (a == true) {
                    Campeonatos.remove({ _id: campeonatoID });
                }
   },

   'click .filtra': function(event, template){
        var valorFiltro = $('#filtro').val();
        Session.set('filtroCampeonato', valorFiltro);
   },
```

```
    'click .edita': function(event, template) {
          event.preventDefault();
          Session.set('idCampeonato', this._id);
          Router.go('novoCampeonato');

          }
});
```

```
<template name="novoCampeonato">
  {{> nav}}

    <form class="novo-campeonato">

        <h1>Cadastro de campeonato</h1>

          <legend>Informações do campeonato</legend>

          <input type="text" name="nomeCampeonato" placeholder="* Nome do campeonato"
value={{campeonatos.nomeCampeonato}} required>

          <div class="row">
            <div class="col-sm-6 col-xs-12">
              <input type="text" name="nomeOrganizador" placeholder="* Nome do organizador"
value={{campeonatos.nomeOrganizador}} required>
            </div>
            <div class="col-sm-6 col-xs-12">
              <input type="text" name="emailOrganizador" placeholder="* Email do organizador"
value={{campeonatos.emailOrganizador}} required>
            </div>
          </div>

          <input type="text" name="cidadeCampeonato" placeholder="Cidade do campeonato"
value={{campeonatos.cidadeCampeonato}}>

          <div class="row">
            <div class="col-sm-6 col-xs-12">
              <b>Início</b>
              <input type="date" name="dataInicio" placeholder="Data de início"
value={{campeonatos.dataInicio}}>
            </div>
            <div class="col-sm-6 col-xs-12">
              <b>Fim</b>
              <input type="date" name="dataFim" placeholder="Data do fim"
value={{campeonatos.dataFim}}>
            </div>
```

```
        </div>

        <br>
        <label class="radio-inline formatoCopa">
          <input required type="radio" name="formato" value="1" {{copa
campeonatos.formato}}>Copa
        </label>

        <label class="radio-inline formatoLiga">
          <input type="radio" name="formato" value="2" {{liga campeonatos.formato}}>Liga
        </label>

        <label class="radio-inline formato">
          <input type="radio" name="formato" value="3" {{matamata
campeonatos.formato}}>Mata-Mata
        </label>
        <br>
        <br>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
            <button type="submit" name="cadastrar" >Salvar</button>
          </div>
          <div class="col-sm-6 col-xs-12">
            <button class="cancela" formnovalidate>Cancelar</button>
          </div>
        </div>


    </form>

  {{> footer}}
</template>

---

Template.novoCampeonato.onDestroyed(function(){
          Session.set('idCampeonato', null)
});

Template.novoCampeonato.helpers({
    times: function(){
        return Campeonatos.find({}, { sort: {nome: 1} });
    },

    campeonatos: function(){
                    var idCampeonato = Session.get('idCampeonato');
```

```
            return Campeonatos.findOne({_id: idCampeonato});
        },

        copa: function(formato) {
        return formato == "1" ?'checked':'';
        },

        liga: function(formato) {
           return formato == "2" ?'checked':'';
        },

        matamata: function(formato) {
           return formato == "3" ?'checked':'';
        }
});

Template.novoCampeonato.events({
        'submit .novo-campeonato': function(event){
                var nomeCampeonato = event.target.nomeCampeonato.value;
                var nomeOrganizador = event.target.nomeOrganizador.value;
                var emailOrganizador = event.target.emailOrganizador.value;
                var formato =
document.querySelector('input[name="formato"]:checked').value;
                if(event.target.cidadeCampeonato.value)
                        var cidadeCampeonato = event.target.cidadeCampeonato.value;
                if(event.target.dataInicio.value)
                        var dataInicio = event.target.dataInicio.value;
                if(event.target.dataFim.value)
                        var dataFim = event.target.dataFim.value;

                if(Session.get('idCampeonato')){
                        Campeonatos.update(
                                        { _id: Session.get('idCampeonato') },
                                        {
                                                nomeCampeonato: nomeCampeonato,
                                                cidadeCampeonato: cidadeCampeonato,
                                                nomeOrganizador : nomeOrganizador,
                                                emailOrganizador: emailOrganizador,
                                                dataInicio: dataInicio,
                                                dataFim: dataFim,
                                                formato: formato
                                        }
                                );

                }
                else{
```

```
                        var doc = Campeonatos.insert({
                                nomeCampeonato: nomeCampeonato,
                                cidadeCampeonato: cidadeCampeonato,
                                nomeOrganizador : nomeOrganizador,
                                emailOrganizador: emailOrganizador,
                                dataInicio: dataInicio,
                                dataFim: dataFim,
                                formato: formato
                        });

                        Artilharias.insert({
                                idCampeonato: doc
                        })

                        JogadoresPunidos.insert({
                                idCampeonato: doc
                        })

                        MelhoresDefesas.insert({
                                idCampeonato: doc
                        })
                }

                alert("Campeonato salvo!");
                document.getElementById(".novo-campeonato").reset();

                return false;
        },

        'click .cancela': function(){
                event.preventDefault();
        Router.go('listaCampeonatos');
        }
});
```

```
<template name="listaPartidas">
        {{> nav}}

        <div class="fundo">
                <h1>Partidas</h1>

                <div class="row">
                        <div class="col-xs-9">
                                <input type="text" id="filtro" name="filtroPartidas"
placeholder="Digite o nome de um dos times ou do campeonato">
```

```
                </div>
                <div class="col-xs-3">
                        <button class="btn btn-primary filtra">FILTRAR</button>
                </div>
        </div>

        <div class="separador"></div>

        {{#each partidas}}
                <div class="row fundo-listagem">
                        <div class="col-xs-6">
                                <div class="row">
                                        {{nomeTime time1.idTime}} {{placarTime1}} x
{{placarTime2}} {{nomeTime time2.idTime}}
                                </div>
                                <div class="row">
                                        {{nomeCampeonato
campeonato.idCampeonato}}
                                </div>
                        </div>
                        <div class="col-xs-3">
                                <button class="btn btn-success edita">Alterar</button>
                        </div>
                        <div class="col-xs-3">
                                <button class="btn btn-danger
remove">Excluir</button>
                        </div>
                </div>
        {{/each}}
    </div>

    {{> footer}}
</template>
```

```
Template.listaPartidas.onCreated(function(){
   tabelaClassificacao = [];
});

Template.listaPartidas.onDestroyed(function(){
    Session.set('filtroPartida', undefined)
});

Template.listaPartidas.helpers({

    partidas : function() {
```

```
                var valorFiltro = Session.get('filtroPartida');
                return Partidas.find({
                        $or: [
                                        { "time1.nomeTime": {$regex : new RegExp(valorFiltro,
"i") } },
                                        { "time2.nomeTime": {$regex : new RegExp(valorFiltro,
"i") } },
                                        { "campeonato.nomeCampeonato": {$regex : new
RegExp(valorFiltro, "i") } }
                                ]
                },
                {sort: {"campeonato.nomeCampeonato": 1, data: 1, "time1.nomeTime": 1}});
        },

   formatDate(data){
        return moment(data).format('DD/MM/YYYY');
   },

   nomeCampeonato(idCampeonato){
        var campeonato = Campeonatos.findOne({_id: idCampeonato});
        return campeonato.nomeCampeonato;
   },

   nomeTime(idTime){
        var time = Times.findOne({_id: idTime});
      if (time)
          return time.nomeTime;
      else
        return false;
   }

});

Template.listaPartidas.events({

        'click .remove': function(){
                var partidaID = this._id;
                var a = confirm("Você tem certeza que deseja excluir?");
                if (a == true) {
        var partida = Partidas.findOne({_id: this._id});
        var tabela;

        if(partida.idGrupo){
           tabela = Tabelas.findOne({idGrupo: partida.idGrupo});
        }
        else if(partida.idCampeonato){
```

```
    tabela = Tabelas.findOne({idCampeonato: partida.idCampeonato});
}
var idVencedor = partida.vencedor;
var idTime1 = partida.time1.idTime;
var idTime2 = partida.time2.idTime;
var placarTime1 = partida.placarTime1;
var placarTime2 = partida.placarTime2;


if(tabela.tabelaClassificacao){
for (var i = 0; i < tabela.tabelaClassificacao.length; i++) {

        tabelaClassificacao.push({
            "idTime" : tabela.tabelaClassificacao[i].idTime,
            "nomeTime" : tabela.tabelaClassificacao[i].nomeTime,
            "pontos" : tabela.tabelaClassificacao[i].pontos,
            "jogos" : tabela.tabelaClassificacao[i].jogos,
            "vitorias" : tabela.tabelaClassificacao[i].vitorias,
            "empates" : tabela.tabelaClassificacao[i].empates,
            "derrotas" : tabela.tabelaClassificacao[i].derrotas,
            "golsPro" : tabela.tabelaClassificacao[i].golsPro,
            "golsContra" : tabela.tabelaClassificacao[i].golsContra,
            "saldo" : tabela.tabelaClassificacao[i].saldo
        });
    }
}

if(placarTime1 > placarTime2){
    var pontos1 = 3;
    var vitorias1 = 1;
    var empates1 = 0;
    var derrotas1 = 0;
    var golsPro1 = placarTime1;
    var golsContra1 = placarTime2;
    var saldo1 = placarTime1 - placarTime2;

    var pontos2 = 0;
    var vitorias2 = 0;
    var empates2 = 0;
    var derrotas2 = 1;
    var golsPro2 = placarTime2;
    var golsContra2 = placarTime1;
    var saldo2 = placarTime2 - placarTime1;

}
else if(placarTime2 > placarTime1){
```

```
    var pontos2 = 3;
    var vitorias2 = 1;
    var empates2 = 0;
    var derrotas2 = 0;
    var golsPro2 = placarTime2;
    var golsContra2 = placarTime1;
    var saldo2 = placarTime2 - placarTime1;

    var pontos1 = 0;
    var vitorias1 = 0;
    var empates1 = 0;
    var derrotas1 = 1;
    var golsPro1 = placarTime1;
    var golsContra1 = placarTime2;
    var saldo1 = placarTime1 - placarTime2;
}
else{
    var pontos1 = 1;
    var vitorias1 = 0;
    var empates1 = 1;
    var derrotas1 = 0;
    var golsPro1 = placarTime1;
    var golsContra1 = placarTime2;
    var saldo1 = placarTime1 - placarTime2;

    var pontos2 = 1;
    var vitorias2 = 0;
    var empates2 = 1;
    var derrotas2 = 0;
    var golsPro2 = placarTime2;
    var golsContra2 = placarTime1;
    var saldo2 = placarTime2 - placarTime1;
}

if(partida.idGrupo && partida.fase == 1 && partida.status == 3){
    for (var i = 0; i < tabelaClassificacao.length; i++) {
        if(tabelaClassificacao[i].idTime == idTime1){
            tabelaClassificacao[i].pontos -= pontos1;
            tabelaClassificacao[i].jogos -= 1;
            tabelaClassificacao[i].vitorias -= vitorias1;
            tabelaClassificacao[i].empates -= empates1;
            tabelaClassificacao[i].derrotas -= derrotas1;
            tabelaClassificacao[i].golsPro -= golsPro1;
            tabelaClassificacao[i].golsContra -= golsContra1;
            tabelaClassificacao[i].saldo -= saldo1;
        }
```

```
        else if(tabelaClassificacao[i].idTime == idTime2){
            tabelaClassificacao[i].pontos -= pontos2;
            tabelaClassificacao[i].jogos -= 1;
            tabelaClassificacao[i].vitorias -= vitorias2;
            tabelaClassificacao[i].empates -= empates2;
            tabelaClassificacao[i].derrotas -= derrotas2;
            tabelaClassificacao[i].golsPro -= golsPro2;
            tabelaClassificacao[i].golsContra -= golsContra2;
            tabelaClassificacao[i].saldo -= saldo2;
        }
    }
}
else if(partida.idCampeonato && partida.fase == 1 && partida.status == 3){
    for (var i = 0; i < tabelaClassificacao.length; i++) {
        if(tabelaClassificacao[i].idTime == idTime1){
            tabelaClassificacao[i].pontos -= pontos1;
            tabelaClassificacao[i].jogos -= 1;
            tabelaClassificacao[i].vitorias -= vitorias1;
            tabelaClassificacao[i].empates -= empates1;
            tabelaClassificacao[i].derrotas -= derrotas1;
            tabelaClassificacao[i].golsPro -= golsPro1;
            tabelaClassificacao[i].golsContra -= golsContra1;
            tabelaClassificacao[i].saldo -= saldo1;
        }
        else if(tabelaClassificacao[i].idTime == idTime2){
            tabelaClassificacao[i].pontos -= pontos2;
            tabelaClassificacao[i].jogos -= 1;
            tabelaClassificacao[i].vitorias -= vitorias2;
            tabelaClassificacao[i].empates -= empates2;
            tabelaClassificacao[i].derrotas -= derrotas2;
            tabelaClassificacao[i].golsPro -= golsPro2;
            tabelaClassificacao[i].golsContra -= golsContra2;
            tabelaClassificacao[i].saldo -= saldo2;
        }
    }
}

if(partida.idGrupo && partida.fase == 1 && partida.status == 3){
    var tabelas = Tabelas.findOne({ idGrupo: partida.idGrupo });
    Tabelas.update({ _id: tabelas._id },
    {
        idGrupo: partida.idGrupo,
        idCampeonato: partida.idCampeonato,
        tabelaClassificacao
    });
}
```

```
        else if(partida.idCampeonato && partida.fase == 1 && partida.status == 3){
           var tabelas = Tabelas.findOne({ idCampeonato: partida.idCampeonato });
           if(tabelas){
              Tabelas.update({ _id: tabelas._id },
              {
                 idCampeonato: partida.idCampeonato,
                 tabelaClassificacao
              });
           }
        }
        Partidas.remove({ _id: partidaID });
        }
   },

   'click .filtra': function(event, template){
        var valorFiltro = $('#filtro').val();
        Session.set('filtroPartida', valorFiltro);
   },

   'click .edita': function(event, template) {
        event.preventDefault();
        Session.set('idPartida', this._id);
        Router.go('novaPartida');
        }
});
```

```
Template.novaPartida.onCreated(function(){

        jogadoresTime1 = [];
        jogadoresTime2 = [];
        gols1 = [];
        gols2 = [];
        golsC1 = [];
        golsC2 = [];
        cartaoA1 = [];
        cartaoA2 = [];
        cartaoV1 = [];
        cartaoV2 = [];
        tabelaClassificacao = [];
        artilharia = [];
        artilharia1 = [];
        artilharia2 = [];
        melhoresDefesas = [];
        jogadoresPunidos = [];
        this.listaJogadoresTime1 = new ReactiveVar([]);
```

```
        this.listaJogadoresTime2 = new ReactiveVar([]);
        this.listaJogadoresTime1Gol = new ReactiveVar([]);
        this.listaJogadoresTime2Gol = new ReactiveVar([]);
        this.listaJogadoresTime1GolC = new ReactiveVar([]);
        this.listaJogadoresTime2GolC = new ReactiveVar([]);
        this.listaJogadoresTime1CA = new ReactiveVar([]);
        this.listaJogadoresTime2CA = new ReactiveVar([]);
        this.listaJogadoresTime1CV = new ReactiveVar([]);
        this.listaJogadoresTime2CV = new ReactiveVar([]);
        numeroGolsTime1 = 0;
        numeroGolsTime2 = 0;
        this.numeroGolsTime1 = new ReactiveVar(0);
        this.numeroGolsTime2 = new ReactiveVar(0);

        if(Session.get('idPartida')){

                var idPartida = Session.get('idPartida');
                var listaPartidas = Partidas.findOne({_id: idPartida});
                var idCampeonato = listaPartidas.campeonato.idCampeonato;
                var idGrupo = listaPartidas.idGrupo;
                var idFase = listaPartidas.fase;
                var idTime1 = listaPartidas.time1.idTime;
                var idTime2 = listaPartidas.time2.idTime;
                var tabela;
                var tabelaArtilharia = Artilharias.findOne({"idCampeonato": idCampeonato});
                var tabelaMelhoresDefesas = MelhoresDefesas.findOne({"idCampeonato":
idCampeonato});
                var tabelaJogadoresPunidos = JogadoresPunidos.findOne({"idCampeonato":
idCampeonato});

                if(idGrupo && idFase == 1){
                        tabela = Tabelas.findOne({"idGrupo": idGrupo});
                }
                else if(idCampeonato && idFase == 1){
                        tabela = Tabelas.findOne({"idCampeonato": idCampeonato});
                }

                if(tabela.tabelaClassificacao){
                        for (var i = 0; i < tabela.tabelaClassificacao.length; i++) {
                                tabelaClassificacao.push({
                                        "idTime" : tabela.tabelaClassificacao[i].idTime,
                                        "nomeTime" :
tabela.tabelaClassificacao[i].nomeTime,
                                        "pontos" : tabela.tabelaClassificacao[i].pontos,
                                        "jogos" : tabela.tabelaClassificacao[i].jogos,
                                        "vitorias" : tabela.tabelaClassificacao[i].vitorias,
```

```
                                                "empates" :
tabela.tabelaClassificacao[i].empates,
                                                "derrotas" :
tabela.tabelaClassificacao[i].derrotas,
                                                "golsPro" : tabela.tabelaClassificacao[i].golsPro,
                                                "golsContra" :
tabela.tabelaClassificacao[i].golsContra,
                                                "saldo" : tabela.tabelaClassificacao[i].saldo
                                        });
                        }
                }

                if(tabelaArtilharia.artilharia){
                        for(var i = 0; i < tabelaArtilharia.artilharia.length; i++){
                                artilharia.push({
                                        "idJogador": tabelaArtilharia.artilharia[i].idJogador,
                                        "nomeJogador":
tabelaArtilharia.artilharia[i].nomeJogador,
                                        "idTime": tabelaArtilharia.artilharia[i].idTime,
                                        "nomeTime": tabelaArtilharia.artilharia[i].nomeTime,
                                        "numeroGols": tabelaArtilharia.artilharia[i].numeroGols
                                })
                        }
                }

                var listaJogadores1 = Jogadores.find({"idTime": idTime1}, {sort:
{nomeJogador: 1}});
                this.listaJogadoresTime1.set(listaJogadores1);

                var listaJogadores2 = Jogadores.find({"idTime": idTime2}, {sort:
{nomeJogador: 1}});
                this.listaJogadoresTime2.set(listaJogadores2);

                if(listaPartidas.gols){
                        for (var i = 0; i < listaPartidas.gols.length; i++) {
                                if(listaPartidas.gols[i].idTime == idTime1){
                                        gols1.push({ "idTime" : listaPartidas.gols[i].idTime,
"idJogador" : listaPartidas.gols[i].idJogador, "nomeJogador" :
listaPartidas.gols[i].nomeJogador, "numeroGols" : listaPartidas.gols[i].numeroGols });
                                        numeroGolsTime1 =
(parseInt(numeroGolsTime1)+parseInt(listaPartidas.gols[i].numeroGols));
                                }
                                else{
                                        gols2.push({ "idTime" : listaPartidas.gols[i].idTime,
"idJogador" : listaPartidas.gols[i].idJogador, "nomeJogador" :
listaPartidas.gols[i].nomeJogador, "numeroGols" : listaPartidas.gols[i].numeroGols });
```

```
                                        numeroGolsTime2 =
(parseInt(numeroGolsTime2)+parseInt(listaPartidas.gols[i].numeroGols));
                                }
                        }
                        this.listaJogadoresTime1Gol.set(gols1);
                        this.listaJogadoresTime2Gol.set(gols2);
                }

                if(listaPartidas.golsContra){
                        for (var i = 0; i < listaPartidas.golsContra.length; i++) {
                                if(listaPartidas.golsContra[i].idTime == idTime1){
                                        golsC1.push({ "idTime" :
listaPartidas.golsContra[i].idTime, "idJogador" : listaPartidas.golsContra[i].idJogador,
"nomeJogador" : listaPartidas.golsContra[i].nomeJogador, "numeroGols" :
listaPartidas.golsContra[i].numeroGols });
                                        numeroGolsTime2 =
(parseInt(numeroGolsTime2)+parseInt(listaPartidas.golsContra[i].numeroGols));
                                }
                                else{
                                        golsC2.push({ "idTime" :
listaPartidas.golsContra[i].idTime, "idJogador" : listaPartidas.golsContra[i].idJogador,
"nomeJogador" : listaPartidas.golsContra[i].nomeJogador, "numeroGols" :
listaPartidas.golsContra[i].numeroGols });
                                        numeroGolsTime1 =
(parseInt(numeroGolsTime1)+parseInt(listaPartidas.golsContra[i].numeroGols));
                                }
                        }
                        this.listaJogadoresTime1GolC.set(golsC1);
                        this.listaJogadoresTime2GolC.set(golsC2);
                }
                ;
                this.numeroGolsTime1.set(numeroGolsTime1);
                this.numeroGolsTime2.set(numeroGolsTime2);

                if(listaPartidas.cartaoAmarelo){
                        for (var i = 0; i < listaPartidas.cartaoAmarelo.length; i++) {
                                if(listaPartidas.cartaoAmarelo[i].idTime == idTime1){
                                        cartaoA1.push({ "idTime" :
listaPartidas.cartaoAmarelo[i].idTime, "idJogador" : listaPartidas.cartaoAmarelo[i].idJogador,
"nomeJogador" : listaPartidas.cartaoAmarelo[i].nomeJogador, "numeroCartoes" :
listaPartidas.cartaoAmarelo[i].numeroCartoes });
                                }
                                else{
                                        cartaoA2.push({ "idTime" :
listaPartidas.cartaoAmarelo[i].idTime, "idJogador" : listaPartidas.cartaoAmarelo[i].idJogador,
```

```
"nomeJogador" : listaPartidas.cartaoAmarelo[i].nomeJogador, "numeroCartoes" :
listaPartidas.cartaoAmarelo[i].numeroCartoes });
                                        }
                                }
                                this.listaJogadoresTime1CA.set(cartaoA1);
                                this.listaJogadoresTime2CA.set(cartaoA2);
                        }

                        if(listaPartidas.cartaoVermelho){
                                for (var i = 0; i < listaPartidas.cartaoVermelho.length; i++) {
                                        if(listaPartidas.cartaoVermelho[i].idTime == idTime1){
                                                cartaoV1.push({ "idTime" :
listaPartidas.cartaoVermelho[i].idTime, "idJogador" :
listaPartidas.cartaoVermelho[i].idJogador, "nomeJogador" :
listaPartidas.cartaoVermelho[i].nomeJogador, "numeroCartoes" :
listaPartidas.cartaoVermelho[i].numeroCartoes });
                                        }
                                        else{
                                                cartaoV2.push({ "idTime" :
listaPartidas.cartaoVermelho[i].idTime, "idJogador" :
listaPartidas.cartaoVermelho[i].idJogador, "nomeJogador" :
listaPartidas.cartaoVermelho[i].nomeJogador, "numeroCartoes" :
listaPartidas.cartaoVermelho[i].numeroCartoes });
                                        }
                                }
                                this.listaJogadoresTime1CV.set(cartaoV1);
                                this.listaJogadoresTime2CV.set(cartaoV2);
                        }
                }
}
);

Template.novaPartida.onRendered(function(){
        if(Session.get('idPartida'))
        {
                var listaPartidas = Partidas.findOne({_id: Session.get('idPartida')});
                var listaCampeonato = Campeonatos.findOne({_id:
listaPartidas.campeonato.idCampeonato});
                var idFase = listaPartidas.fase;
                if(listaPartidas.status == 3){
                        $(".nova-partida :input").prop("disabled", true);
                        $(".cancela").prop("disabled", false);
                }
                if(listaCampeonato.formato == 1){
                        if(idFase == 1){
                                document.getElementById('turno-select').style.display ='block';
```

```
                                document.getElementById('idavolta-select').style.display
='none';
                                document.getElementById('grupo-select').style.display ='block';
                        }
                        else{
                                document.getElementById('turno-select').style.display ='none';
                                document.getElementById('idavolta-select').style.display
='block';
                                document.getElementById('grupo-select').style.display ='none';
                        }
                }
                else if(listaCampeonato.formato == 2){
                        if(idFase == 1){
                                document.getElementById('turno-select').style.display ='block';
                                document.getElementById('idavolta-select').style.display
='none';
                                document.getElementById('grupo-select').style.display ='none';
                        }
                        else{
                                document.getElementById('turno-select').style.display ='none';
                                document.getElementById('idavolta-select').style.display
='block';
                                document.getElementById('grupo-select').style.display ='none';
                        }
                }
                else{
                        document.getElementById('turno-select').style.display ='none';
                        document.getElementById('idavolta-select').style.display ='block';
                        document.getElementById('grupo-select').style.display ='none';
                }
        }
        else{
                document.getElementById('grupo-select').style.display ='none';
                document.getElementById('turno-select').style.display ='none';
                document.getElementById('idavolta-select').style.display ='none';
        }
});

Template.novaPartida.onDestroyed(function(){
        Session.set('idPartida', null);
        Session.set('campeonatoSelect', null);
        Session.set('grupoSelect', null);
        Session.set('idTime1', null);
        Session.set('idTime2', null);
});
```

```
Template.novaPartida.helpers({
   times: function(){

         if(Session.get('grupoSelect')){
                  var listaGrupo = Grupos.findOne({_id: Session.get('grupoSelect')}, {times: 1,
_id: 0});
                  if(listaGrupo.times){
                           return listaGrupo.times;

                  }
         }
         else if(Session.get('idPartida')){
                  var listaPartidas = Partidas.findOne({_id: Session.get('idPartida')});
                  if(listaPartidas.idGrupo){
                           var listaGrupo = Grupos.findOne({_id: listaPartidas.idGrupo}, {times:
1, _id: 0});

                           if(listaGrupo.times)
                                    return listaGrupo.times;
                  }
                  else{
                           var listaPartidas = Partidas.findOne({_id: Session.get('idPartida')});
                           var listaTime = Times.find({"campeonatos.idCampeonato":
listaPartidas.campeonato.idCampeonato}, { sort: {nomeTime: 1} })
                           return listaTime;
                  }
         }
         else if(Session.get('campeonatoSelect')){
                  var listaTime = Times.find({"campeonatos.idCampeonato":
Session.get('campeonatoSelect')}, { sort: {nomeTime: 1} })
                  return listaTime;
         }
         else{
                  return false;
         }
   },

   isEditGrupo: function(){
         if(Session.get('idPartida')){
                  var listaPartidas = Partidas.findOne({_id: Session.get('idPartida')});
                           if(listaPartidas.idGrupo){
                                    return true;
                           }
                  }
                  else{
                           return true;
```

```
                }
    },

    campeonatos: function(){
        return Campeonatos.find({}, { sort: {nomeCampeonato: 1} });
    },

    partidas: function(){
                var idPartida = Session.get('idPartida');
                var listaPartidas = Partidas.findOne({_id: idPartida});
                return listaPartidas;
        },

        grupos: function(){

        if(Session.get('campeonatoSelect')){
                        var listaCampeonato = Campeonatos.findOne({_id:
Session.get('campeonatoSelect')});
                if(listaCampeonato.formato == 1){
                        return Grupos.find({"idCampeonato":
Session.get('campeonatoSelect')}, { sort: {nomeGrupo: 1} });
                }
        }
        else if(Session.get('idPartida')){
                        var listaPartidas = Partidas.findOne({_id: Session.get('idPartida')});

                        var listaCampeonato = Campeonatos.findOne({_id:
listaPartidas.campeonato.idCampeonato});
                        if(listaCampeonato.formato == 1){
                        return Grupos.find({"idCampeonato":
listaPartidas.campeonato.idCampeonato}, { sort: {nomeGrupo: 1} });
                }
        }
    },

    statusSelecionado(idStatus){
        if(Session.get('idPartida')){
                        var idPartida = Session.get('idPartida');
                        var listaPartidas = Partidas.findOne({_id: idPartida});
                        if(idStatus == listaPartidas.status){
                                return 'selected';
                        }
                        else{
                                return '';
                        }
                }
```

```
},

    campeonatoSelecionado(idCampeonato){
        if(Session.get('idPartida')){
            var idPartida = Session.get('idPartida');
            var listaPartidas = Partidas.findOne({_id: idPartida});
            if(idCampeonato == listaPartidas.campeonato.idCampeonato){
                return 'selected';
            }
            else{
                return '';
            }
        }
    },

    time1Selecionado(idTime){
        if(Session.get('idPartida')){
            var idPartida = Session.get('idPartida');
            var listaPartidas = Partidas.findOne({_id: idPartida});
            if(idTime == listaPartidas.time1.idTime){
                return 'selected';
            }
            else{
                return '';
            }
        }
    },

    time2Selecionado(idTime){
        if(Session.get('idPartida')){
            var idPartida = Session.get('idPartida');
            var listaPartidas = Partidas.findOne({_id: idPartida});
            if(idTime == listaPartidas.time2.idTime){
                return 'selected';
            }
            else{
                return '';
            }
        }
    },

    faseSelecionada(idFase){
        if(Session.get('idPartida')){
            var idPartida = Session.get('idPartida');
            var listaPartidas = Partidas.findOne({_id: idPartida});
            if(idFase == listaPartidas.fase){
```

```
                return 'selected';
            }
            else{
                return '';
            }
        }
},

turnoSelecionado(idTurno){
        if(Session.get('idPartida')){
            var idPartida = Session.get('idPartida');
            var listaPartidas = Partidas.findOne({_id: idPartida});
            if(idTurno == listaPartidas.turno){
                return 'selected';
            }
            else{
                return '';
            }
        }
},

grupoSelecionado(idGrupo){
        if(Session.get('idPartida')){
            var idPartida = Session.get('idPartida');
            var listaPartidas = Partidas.findOne({_id: idPartida});
            if(idGrupo == listaPartidas.idGrupo){
                return 'selected';
            }
            else{
                return '';
            }
        }
},

idavoltaSelecionado(idavolta){
        if(Session.get('idPartida')){
            var idPartida = Session.get('idPartida');
            var listaPartidas = Partidas.findOne({_id: idPartida});
            if(idavolta == listaPartidas.idaVolta){
                return 'selected';
            }
            else{
                return '';
            }
        }
},
```

```
	nomeTime1(){
		if(Session.get('idPartida')){
			var listaPartidas = Partidas.findOne({_id: Session.get('idPartida')});
			return listaPartidas.time1.nomeTime;
		}
		else if(Session.get('idTime1')){
			return $("#time1-select option:selected").text();
		}
	},

	nomeTime2(){
		if(Session.get('idPartida')){
			var listaPartidas = Partidas.findOne({_id: Session.get('idPartida')});
			return listaPartidas.time2.nomeTime;
		}
		else if(Session.get('idTime2')){
			return $("#time2-select option:selected").text();
		}
	},

	golsTime1(){
		return Template.instance().numeroGolsTime1.get();
	},

	golsTime2(){
		return Template.instance().numeroGolsTime2.get();
	},

	listaJogadoresTime1(){
	return Template.instance().listaJogadoresTime1.get();
},

	listaJogadoresTime2(){
	return Template.instance().listaJogadoresTime2.get();
},

	listaJogadoresTime1Gol(){
	return Template.instance().listaJogadoresTime1Gol.get();
},

	listaJogadoresTime2Gol(){
	return Template.instance().listaJogadoresTime2Gol.get();
},

	listaJogadoresTime1GolC(){
```

```
        return Template.instance().listaJogadoresTime1GolC.get();
    },

        listaJogadoresTime2GolC(){
        return Template.instance().listaJogadoresTime2GolC.get();
    },

        listaJogadoresTime1CA(){
        return Template.instance().listaJogadoresTime1CA.get();
    },

        listaJogadoresTime2CA(){
        return Template.instance().listaJogadoresTime2CA.get();
    },

        listaJogadoresTime1CV(){
        return Template.instance().listaJogadoresTime1CV.get();
    },

        listaJogadoresTime2CV(){
        return Template.instance().listaJogadoresTime2CV.get();
    }



});

Template.novaPartida.events({
    'submit .nova-partida': function(event){

                            var local = event.target.local.value;
                            var data = event.target.data.value;
                            var horario = event.target.horario.value;
                            var nomeArbitro = event.target.nomeArbitro.value;
                            var idCampeonato = $("#campeonato-select option:selected").val();
                            var nomeCampeonato = $("#campeonato-select
option:selected").text();
                            var idStatus = $("#status-select option:selected").val();
                            var idFase = $("#fase-select option:selected").val();
                            var idavolta = $("#idavolta-select option:selected").val();
                            var idTurno = $("#turno-select option:selected").val();
                            var idGrupo = $("#grupo-select option:selected").val();
                            var nomeGrupo = $("#grupo-select option:selected").text();
                            var idTime1 = $("#time1-select option:selected").val();
                            var nomeTime1 = $("#time1-select option:selected").text();
                            var idTime2 = $("#time2-select option:selected").val();
```

```
                        var nomeTime2 = $("#time2-select option:selected").text();
                        var idVencedor = '';
                        var gols = gols1.concat(gols2);
                        var golsContra = golsC1.concat(golsC2);
                        var cartaoAmarelo = cartaoA1.concat(cartaoA2);
                        var cartaoVermelho = cartaoV1.concat(cartaoV2);
                        var listaArtilharia = Artilharias.findOne({idCampeonato:
idCampeonato});
                        var tabela;
                        var artilharia = artilharia1.concat(artilharia2);

                if(idGrupo && idFase == 1){
                        tabela = Tabelas.findOne({"idGrupo": idGrupo});
                }
                else if(idCampeonato && idFase == 1){
                        tabela = Tabelas.findOne({"idCampeonato": idCampeonato});
                }

                if(tabela.tabelaClassificacao){
                        for (var i = 0; i < tabela.tabelaClassificacao.length; i++) {
                                tabelaClassificacao.push({
                                        "idTime" : tabela.tabelaClassificacao[i].idTime,
                                        "nomeTime" :
tabela.tabelaClassificacao[i].nomeTime,
                                        "pontos" : tabela.tabelaClassificacao[i].pontos,
                                        "jogos" : tabela.tabelaClassificacao[i].jogos,
                                        "vitorias" : tabela.tabelaClassificacao[i].vitorias,
                                        "empates" :
tabela.tabelaClassificacao[i].empates,
                                        "derrotas" :
tabela.tabelaClassificacao[i].derrotas,
                                        "golsPro" : tabela.tabelaClassificacao[i].golsPro,
                                        "golsContra" :
tabela.tabelaClassificacao[i].golsContra,
                                        "saldo" : tabela.tabelaClassificacao[i].saldo
                                });
                        }
                }

                        if(idStatus == 3){
                                var a = confirm("Uma vez finalizada a partida, não será mais
possível alterá-la. Confirmar?");
                                if (a == false) {
                                    return false;
                                }
```

```
if(numeroGolsTime1 > numeroGolsTime2){
        idVencedor = idTime1;
        var pontos1 = 3;
        var vitorias1 = 1;
        var empates1 = 0;
        var derrotas1 = 0;
        var golsPro1 = numeroGolsTime1;
        var golsContra1 = numeroGolsTime2;
        var saldo1 = numeroGolsTime1 - numeroGolsTime2;

        var pontos2 = 0;
        var vitorias2 = 0;
        var empates2 = 0;
        var derrotas2 = 1;
        var golsPro2 = numeroGolsTime2;
        var golsContra2 = numeroGolsTime1;
        var saldo2 = numeroGolsTime2 - numeroGolsTime1;

}
else if(numeroGolsTime2 > numeroGolsTime1){
        idVencedor = idTime2
        var pontos2 = 3;
        var vitorias2 = 1;
        var empates2 = 0;
        var derrotas2 = 0;
        var golsPro2 = numeroGolsTime2;
        var golsContra2 = numeroGolsTime1;
        var saldo2 = numeroGolsTime2 - numeroGolsTime1;

        var pontos1 = 0;
        var vitorias1 = 0;
        var empates1 = 0;
        var derrotas1 = 1;
        var golsPro1 = numeroGolsTime1;
        var golsContra1 = numeroGolsTime2;
        var saldo1 = numeroGolsTime1 - numeroGolsTime2;
}
else{
        idVencedor = 'empate';
        var pontos1 = 1;
        var vitorias1 = 0;
        var empates1 = 1;
        var derrotas1 = 0;
        var golsPro1 = numeroGolsTime1;
        var golsContra1 = numeroGolsTime2;
        var saldo1 = numeroGolsTime1 - numeroGolsTime2;
```

```
                var pontos2 = 1;
                var vitorias2 = 0;
                var empates2 = 1;
                var derrotas2 = 0;
                var golsPro2 = numeroGolsTime2;
                var golsContra2 = numeroGolsTime1;
                var saldo2 = numeroGolsTime2 - numeroGolsTime1;
            }

            if(idFase == 1){
                if(tabelaClassificacao.length < 1){
                    tabelaClassificacao.push({
                        "idTime" : idTime1,
                        "nomeTime" : nomeTime1,
                        "pontos" : 0,
                        "jogos"  : 0,
                        "vitorias": 0,
                        "empates": 0,
                        "derrotas": 0,
                        "golsPro": 0,
                        "golsContra": 0,
                        "saldo": 0
                    });

                    tabelaClassificacao.push({
                        "idTime" : idTime2,
                        "nomeTime" : nomeTime2,
                        "pontos" : 0,
                        "jogos"  : 0,
                        "vitorias": 0,
                        "empates": 0,
                        "derrotas": 0,
                        "golsPro": 0,
                        "golsContra": 0,
                        "saldo": 0
                    });
                }

                for (var i = 0; i < tabelaClassificacao.length; i++) {
                    if(tabelaClassificacao[i].idTime == idTime1){
                        tabelaClassificacao[i].pontos +=
pontos1;

                        tabelaClassificacao[i].jogos += 1;
                        tabelaClassificacao[i].vitorias +=
vitorias1;
```

```
                                                tabelaClassificacao[i].empates +=
empates1;
                                                tabelaClassificacao[i].derrotas +=
derrotas1;
                                                tabelaClassificacao[i].golsPro +=
golsPro1;
                                                tabelaClassificacao[i].golsContra +=
golsContra1;
                                                tabelaClassificacao[i].saldo += saldo1;
                                        }
                                        else if(tabelaClassificacao[i].idTime == idTime2)
{
                                                tabelaClassificacao[i].pontos +=
pontos2;
                                                tabelaClassificacao[i].jogos += 1;
                                                tabelaClassificacao[i].vitorias +=
vitorias2;
                                                tabelaClassificacao[i].empates +=
empates2;
                                                tabelaClassificacao[i].derrotas +=
derrotas2;
                                                tabelaClassificacao[i].golsPro +=
golsPro2;
                                                tabelaClassificacao[i].golsContra +=
golsContra2;
                                                tabelaClassificacao[i].saldo += saldo2;
                                        }
                                }
                        }
                }

                if(idTime1 == idTime2){
                        alert("Escolha times diferentes!");
                        return false;
                }

                if(gols.length < 1){
                        gols = undefined;
                }

                if(golsContra.length < 1){
                        golsContra = undefined;
                }

                if(cartaoAmarelo.length < 1){
                        cartaoAmarelo = undefined;
```

```
}

if(cartaoVermelho.length < 1){
	cartaoVermelho = undefined;
}

if(Session.get('idPartida')){
	Partidas.update(
		{ _id: Session.get('idPartida') },{
		status : idStatus,
		local : local,
		data : data,
		horario : horario,
		nomeArbitro : nomeArbitro,
		fase: idFase,
		idaVolta: idavolta,
		turno: idTurno,
		idGrupo: idGrupo,
		vencedor: idVencedor,
		placarTime1: numeroGolsTime1,
		placarTime2: numeroGolsTime2,
		campeonato : {
			idCampeonato: idCampeonato,
			nomeCampeonato: nomeCampeonato
		},
		time1 : {
			idTime: idTime1,
			nomeTime: nomeTime1
		},
		time2 : {
			idTime: idTime2,
			nomeTime: nomeTime2
		},
		gols,
		golsContra,
		cartaoAmarelo,
		cartaoVermelho
	});

	if(idGrupo && idFase == 1 && idStatus == 3){
		var tabelas = Tabelas.findOne({ idGrupo: idGrupo });

		Tabelas.update({ _id: tabelas._id },
		 {
			idGrupo: idGrupo,
			idCampeonato: idCampeonato,
```

```
                                        tabelaClassificacao
                                    });
                                }
                                else if(idCampeonato && idFase == 1 && idStatus == 3){
                                    var tabelas = Tabelas.findOne({ idCampeonato:
idCampeonato });
                                    if(tabelas){
                                        Tabelas.update({ _id: tabelas._id },
                                        {
                                            idCampeonato: idCampeonato,
                                            tabelaClassificacao
                                        });
                                    }
                                    else{
                                        Tabelas.insert({ _id: tabelas._id },
                                        {
                                            idCampeonato: idCampeonato,
                                            tabelaClassificacao
                                        });
                                    }
                                }


                                if(idStatus==3){
                                    if(artilharia.length > 0){
                                        Artilharias.update({_id: listaArtilharia._id},
                                        {idCampeonato: idCampeonato,
                                        artilharia})
                                    }
                                }

                            }
                            else{
                                Partidas.insert({
                                    status : idStatus,
                                    local : local,
                                    data : data,
                                    horario : horario,
                                    nomeArbitro : nomeArbitro,
                                    fase: idFase,
                                    idaVolta: idavolta,
                                    turno: idTurno,
                                    idGrupo: idGrupo,
                                    vencedor: idVencedor,
                                    placarTime1: numeroGolsTime1,
                                    placarTime2: numeroGolsTime2,
```

```
                campeonato : {
                        idCampeonato: idCampeonato,
                        nomeCampeonato: nomeCampeonato
                },
                time1 : {
                        idTime: idTime1,
                        nomeTime: nomeTime1
                },
                time2 : {
                        idTime: idTime2,
                        nomeTime: nomeTime2
                }
                ,
                gols,
                golsContra,
                cartaoAmarelo,
                cartaoVermelho
        });

        if(idGrupo && idFase == 1 && idStatus == 3){
                var tabelas = Tabelas.findOne({ idGrupo: idGrupo });
                if(tabelas){
                        Tabelas.update({ _id: tabelas._id },
                        {
                                idGrupo: idGrupo,
                                idCampeonato: idCampeonato,
                                tabelaClassificacao
                        });
                }
        }
        else if(idCampeonato && idFase == 1 && idStatus == 3){
                var tabelas = Tabelas.findOne({ idCampeonato:
idCampeonato });
                if(tabelas){
                        Tabelas.update({ _id: tabelas._id },
                        {
                                idCampeonato: idCampeonato,
                                tabelaClassificacao
                        });
                }
        }


        if(idStatus==3){
                if(artilharia.length > 0){
```

```
                                        Artilharias.update({_id: listaArtilharia._id},
                                        {idCampeonato: idCampeonato,
                                        artilharia})
                                }


                        }
                }

                alert("Partida salva!");
                document.getElementById(".nova-partida").reset();

                return false;
        },

   "change #time1-select": function (event, template) {
      var idTime1 = $("#time1-select option:selected").val();
                Session.set('idTime1',  idTime1);
                var listaJogadores = Jogadores.find({idTime: idTime1}, {sort: {nomeJogador:
1}});
                template.listaJogadoresTime1.set(listaJogadores);

                return false;
   },

   "change #time2-select": function (event, template) {
      var idTime2 = $("#time2-select option:selected").val();
                Session.set('idTime2',  idTime2);
                var listaJogadores = Jogadores.find({idTime: idTime2}, {sort: {nomeJogador:
1}});
                template.listaJogadoresTime2.set(listaJogadores);

                return false;
   },

        'click .cancela': function(){
                event.preventDefault();

        Router.go('listaPartidas');
        },

        "change #campeonato-select": function(e, t){
                var idCampeonato = $("#campeonato-select option:selected").val();
                Session.set("campeonatoSelect", idCampeonato);

                document.getElementById('turno-select').style.display ='block';
```

```
                document.getElementById('idavolta-select').style.display ='none';
                document.getElementById('grupo-select').style.display ='none';

        return false;
    },

    "change #grupo-select": function(e, t){
                var idGrupo = $("#grupo-select option:selected").val();
                Session.set("grupoSelect", idGrupo);

        return false;
    },

        "change #fase-select": function(e, t){
                var idFase = $("#fase-select option:selected").val();
                var listaCampeonato = Campeonatos.findOne({_id:
Session.get('campeonatoSelect')});
                if(listaCampeonato.formato == 1){
                        if(idFase == 1){
                                document.getElementById('turno-select').style.display ='block';
                                document.getElementById('idavolta-select').style.display
='none';
                                document.getElementById('grupo-select').style.display ='block';
                        }
                        else{
                                document.getElementById('turno-select').style.display ='none';
                                document.getElementById('idavolta-select').style.display
='block';
                                document.getElementById('grupo-select').style.display ='none';
                        }
                }
                else if(listaCampeonato.formato == 2){
                        if(idFase == 1){
                                document.getElementById('turno-select').style.display ='block';
                                document.getElementById('idavolta-select').style.display
='none';
                                document.getElementById('grupo-select').style.display ='none';
                        }
                        else{
                                document.getElementById('turno-select').style.display ='none';
                                document.getElementById('idavolta-select').style.display
='block';
                                document.getElementById('grupo-select').style.display ='none';
                        }
                }
                else{
```

```
                document.getElementById('turno-select').style.display ='none';
                document.getElementById('idavolta-select').style.display ='block';
                document.getElementById('grupo-select').style.display ='none';
            }
    },

    'click .golTime1': function(e, template){
            event.preventDefault();

            var idJogador = $("#jogador1gol-select option:selected").val();
            var nomeJogador = $("#jogador1gol-select option:selected").text();
            var numeroGols = $("#golTime1").val();
            var idTime = $("#time1-select option:selected").val();
            var nomeTime = $("#time1-select option:selected").text();
            var idCampeonato = $("#campeonato-select option:selected").val();
            var tabelaArtilharia = Artilharias.findOne({idCampeonato: idCampeonato});

            if(idJogador == -1){
                return false;
            }

            if(numeroGols < 1){
                return false;
            }

            for (var i = 0; i < gols1.length; i++) {
               if(idJogador == gols1[i].idJogador){
                        return false;
                }
            }
            if(tabelaArtilharia.artilharia){
                for(var i = 0; i < tabelaArtilharia.artilharia.length; i++){
                    artilharia1.push({ "idTime" : tabelaArtilharia.artilharia[i].idTime,
                            "nomeTime" : tabelaArtilharia.artilharia[i].nomeTime,
                            "idJogador" : tabelaArtilharia.artilharia[i].idJogador,
                            "nomeJogador" :
tabelaArtilharia.artilharia[i].nomeJogador,
                            "numeroGols" :
tabelaArtilharia.artilharia[i].numeroGols });
                }
            }



            if(artilharia1.length > 0){
                for(var i = 0; i < artilharia1.length; i++){
```

```
                                        if(artilharia1[i].idJogador == idJogador){
                                                artilharia1[i].numeroGols =
parseInt(artilharia1[i].numeroGols)+parseInt(numeroGols);
                                                break;
                                        }
                                        else if(parseInt(i) == parseInt(artilharia1.length)-parseInt(1)){
                                                artilharia1.push({ "idTime" : idTime, "nomeTime" :
nomeTime, "idJogador" : idJogador, "nomeJogador" : nomeJogador, "numeroGols" :
numeroGols });
                                        }
                                }
                        }
                        else{
                                artilharia1.push({ "idTime" : idTime, "nomeTime" : nomeTime,
"idJogador" : idJogador, "nomeJogador" : nomeJogador, "numeroGols" : numeroGols });
                        }

                        gols1.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroGols" : numeroGols });
                        numeroGolsTime1 = (parseInt(numeroGolsTime1)+parseInt(numeroGols));

                        template.listaJogadoresTime1Gol.set(gols1);
                        template.numeroGolsTime1.set(numeroGolsTime1);
                },

                'click .removeGol1': function(e, template){
                        event.preventDefault();
                        for (var i = 0; i < gols1.length; i++) {
                           if(this.idJogador == gols1[i].idJogador){
                                        numeroGolsTime1 = numeroGolsTime1-gols1[i].numeroGols;
                                        gols1.splice(i, 1);
                                }
                        }
                        template.numeroGolsTime1.set(numeroGolsTime1);
                        template.listaJogadoresTime1Gol.set(gols1);
                },

           'click .golTime2': function(e, template){
                        event.preventDefault();

                        var idJogador = $("#jogador2gol-select option:selected").val();
                        var nomeJogador = $("#jogador2gol-select option:selected").text();
                        var numeroGols = $("#golTime2").val();
                        var idCampeonato = $("#campeonato-select option:selected").val();
                        var idTime = $("#time2-select option:selected").val();
                        var nomeTime = $("#time2-select option:selected").text();
```

```
		var tabelaArtilharia = Artilharias.findOne({idCampeonato: idCampeonato});

		if(idJogador == -1){
			return false;
		}

		if(numeroGols < 1){
			return false;
		}

		for (var i = 0; i < gols2.length; i++) {
		   if(idJogador == gols2[i].idJogador){
				return false;
			}
		}

		if(tabelaArtilharia.artilharia){
			for(var i = 0; i < tabelaArtilharia.artilharia.length; i++){
				artilharia2.push({ "idTime" : tabelaArtilharia.artilharia[i].idTime,
					"nomeTime" : tabelaArtilharia.artilharia[i].nomeTime,
					"idJogador" : tabelaArtilharia.artilharia[i].idJogador,
					"nomeJogador" :
tabelaArtilharia.artilharia[i].nomeJogador,
					"numeroGols" :
tabelaArtilharia.artilharia[i].numeroGols });
			}
		}



		if(artilharia2.length > 0){
			for(var i = 0; i < artilharia2.length; i++){
				if(artilharia2[i].idJogador == idJogador){
					artilharia2[i].numeroGols =
parseInt(artilharia2[i].numeroGols)+parseInt(numeroGols);
					break;
				}
				else if(parseInt(i) == parseInt(artilharia2.length)-parseInt(1)){
					artilharia2.push({ "idTime" : idTime, "nomeTime" :
nomeTime, "idJogador" : idJogador, "nomeJogador" : nomeJogador, "numeroGols" :
numeroGols });
				}
			}
		}
		else{
```

```
                        artilharia2.push({ "idTime" : idTime, "nomeTime" : nomeTime,
"idJogador" : idJogador, "nomeJogador" : nomeJogador, "numeroGols" : numeroGols });
                }

                numeroGolsTime2 = (parseInt(numeroGolsTime2)+parseInt(numeroGols));
                gols2.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroGols" : numeroGols });
                template.numeroGolsTime2.set(numeroGolsTime2);
                template.listaJogadoresTime2Gol.set(gols2);
        },

        'click .removeGol2': function(e, template){
                event.preventDefault();
                for (var i = 0; i < gols2.length; i++) {
                   if(this.idJogador == gols2[i].idJogador){
                        numeroGolsTime2 = numeroGolsTime2-gols2[i].numeroGols;
                                gols2.splice(i, 1)
                        }
                }
                template.numeroGolsTime2.set(numeroGolsTime2);
                template.listaJogadoresTime2Gol.set(gols2)
        },

   'click .golCTime1': function(e, template){
                event.preventDefault();

                var idJogador = $("#jogador1golC-select option:selected").val();
                var nomeJogador = $("#jogador1golC-select option:selected").text();
                var numeroGols = $("#golCTime1").val();
                var idTime = $("#time1-select option:selected").val();

                if(idJogador == -1){
                        return false;
                }

                if(numeroGols < 1){
                        return false;
                }

                for (var i = 0; i < golsC1.length; i++) {
                   if(idJogador == golsC1[i].idJogador){
                                return false;
                        }
                }
```

```
                numeroGolsTime2 = (parseInt(numeroGolsTime2)+parseInt(numeroGols));
                golsC1.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroGols" : numeroGols });
                template.numeroGolsTime2.set(numeroGolsTime2);
                template.listaJogadoresTime1GolC.set(golsC1);
        },

        'click .removeGolC1': function(e, template){
                event.preventDefault();
                for (var i = 0; i < golsC1.length; i++) {
                   if(this.idJogador == golsC1[i].idJogador){
                        numeroGolsTime2 = numeroGolsTime2-golsC1[i].numeroGols;
                                golsC1.splice(i, 1)
                        }
                }
                template.numeroGolsTime2.set(numeroGolsTime2);
                template.listaJogadoresTime1GolC.set(golsC1)
        },

   'click .golCTime2': function(e, template){
                event.preventDefault();

                var idJogador = $("#jogador2golC-select option:selected").val();
                var nomeJogador = $("#jogador2golC-select option:selected").text();
                var numeroGols = $("#golCTime2").val();
                var idTime = $("#time2-select option:selected").val();

                if(idJogador == -1){
                        return false;
                }

                if(numeroGols < 1){
                        return false;
                }

                for (var i = 0; i < golsC2.length; i++) {
                   if(idJogador == golsC2[i].idJogador){
                                return false;
                        }
                }



                numeroGolsTime1 = (parseInt(numeroGolsTime1)+parseInt(numeroGols));
                golsC2.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroGols" : numeroGols });
```

```
            template.listaJogadoresTime2GolC.set(golsC2);
            template.numeroGolsTime1.set(numeroGolsTime1);
        },

        'click .removeGolC2': function(e, template){
            event.preventDefault();
            for (var i = 0; i < golsC2.length; i++) {
               if(this.idJogador == golsC2[i].idJogador){
                    numeroGolsTime1 = numeroGolsTime1-golsC2[i].numeroGols;
                        golsC2.splice(i, 1)
                    }
            }
            template.numeroGolsTime1.set(numeroGolsTime1);
            template.listaJogadoresTime2GolC.set(golsC2)
        },

   'click .caTime1': function(e, template){
            event.preventDefault();

            var idJogador = $("#jogador1CA-select option:selected").val();
            var nomeJogador = $("#jogador1CA-select option:selected").text();
            var numeroCartoes = $("#caTime1").val();
            var idTime = $("#time1-select option:selected").val();

            if(idJogador == -1){
                    return false;
            }

            if(numeroCartoes < 1){
                    return false;
            }

            for (var i = 0; i < cartaoA1.length; i++) {
               if(idJogador == cartaoA1[i].idJogador){
                        return false;
                    }
            }


            cartaoA1.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroCartoes" : numeroCartoes });
            template.listaJogadoresTime1CA.set(cartaoA1);
        },

        'click .removeCA1': function(e, template){
            event.preventDefault();
```

```
		for (var i = 0; i < cartaoA1.length; i++) {
		   if(this.idJogador == cartaoA1[i].idJogador){
				cartaoA1.splice(i, 1)
			}
		}
		template.listaJogadoresTime1CA.set(cartaoA1)
	},

  'click .caTime2': function(e, template){
		event.preventDefault();

		var idJogador = $("#jogador2CA-select option:selected").val();
		var nomeJogador = $("#jogador2CA-select option:selected").text();
		var numeroCartoes = $("#caTime2").val();
		var idTime = $("#time2-select option:selected").val();

		if(idJogador == -1){
			return false;
		}

		if(numeroCartoes < 1){
			return false;
		}

		for (var i = 0; i < cartaoA2.length; i++) {
		   if(idJogador == cartaoA2[i].idJogador){
				return false;
			}
		}


		cartaoA2.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroCartoes" : numeroCartoes });
		template.listaJogadoresTime2CA.set(cartaoA2);
	},

	'click .removeCA2': function(e, template){
		event.preventDefault();
		for (var i = 0; i < cartaoA2.length; i++) {
		   if(this.idJogador == cartaoA2[i].idJogador){
				cartaoA2.splice(i, 1)
			}
		}
		template.listaJogadoresTime2CA.set(cartaoA2)
	},
```

```
    'click .cvTime1': function(e, template){
                event.preventDefault();

                var idJogador = $("#jogador1CV-select option:selected").val();
                var nomeJogador = $("#jogador1CV-select option:selected").text();
                var numeroCartoes = $("#cvTime1").val();
                var idTime = $("#time1-select option:selected").val();

                if(idJogador == -1){
                        return false;
                }

                if(numeroCartoes < 1){
                        return false;
                }

                for (var i = 0; i < cartaoV1.length; i++) {
                   if(idJogador == cartaoV1[i].idJogador){
                                return false;
                        }
                }


                cartaoV1.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroCartoes" : numeroCartoes });
                template.listaJogadoresTime1CV.set(cartaoV1);
        },

        'click .removeCV1': function(e, template){
                event.preventDefault();
                for (var i = 0; i < cartaoV1.length; i++) {
                   if(this.idJogador == cartaoV1[i].idJogador){
                                cartaoV1.splice(i, 1)
                        }
                }
                template.listaJogadoresTime1CV.set(cartaoV1)
        },

    'click .cvTime2': function(e, template){
                event.preventDefault();

                var idJogador = $("#jogador2CV-select option:selected").val();
                var nomeJogador = $("#jogador2CV-select option:selected").text();
                var numeroCartoes = $("#cvTime2").val();
                var idTime = $("#time2-select option:selected").val();
```

```
                if(idJogador == -1){
                        return false;
                }

                if(numeroCartoes < 1){
                        return false;
                }

                for (var i = 0; i < cartaoV2.length; i++) {
                   if(idJogador == cartaoV2[i].idJogador){
                                return false;
                        }
                }


                cartaoV2.push({ "idTime" : idTime, "idJogador" : idJogador, "nomeJogador" :
nomeJogador, "numeroCartoes" : numeroCartoes });
                template.listaJogadoresTime2CV.set(cartaoV2);
        },

        'click .removeCV2': function(e, template){
                event.preventDefault();
                for (var i = 0; i < cartaoV2.length; i++) {
                   if(this.idJogador == cartaoV2[i].idJogador){
                                cartaoV2.splice(i, 1)
                        }
                }
                template.listaJogadoresTime2CV.set(cartaoV2)
        },

});
```

```
<template name="novaPartida">
{{> nav}}
   <form class="nova-partida">

      <h1>Cadastro de partida</h1>

        <legend>Informações da partida</legend>

        <div class="row">
          <div class="col-xs-6">
            <select id="campeonato-select" name="campeonato-select" required>
            <option disabled="disabled" selected="selected" value="">Selecionar
campeonato</option>
```

```
            {{#each campeonatos}}
               <option name="idCampeonato" value="{{_id}}" {{campeonatoSelecionado
_id}}>{{nomeCampeonato}}</option>
            {{/each}}
           </select>
          </div>
          <div class="col-xs-6">
           <select id="status-select" name="status-select" required>
            <option disabled="disabled" selected="selected" value="">Selecionar
status</option>
            <option value="1" {{statusSelecionado 1}}>Agendada</option>
            <option value="2" {{statusSelecionado 2}}>Incompleta</option>
            <option value="3" {{statusSelecionado 3}}>Finalizada</option>
           </select>
          </div>
         </div>

         <div class="row">
          <div class="col-sm-4 col-xs-12">
           <select id="fase-select" name="fase-select" required>
            <option disabled="disabled" selected="selected" value="">Selecionar
fase</option>
            <option value="1" {{faseSelecionada 1}}>Classificatória</option>
            <option value="2" {{faseSelecionada 2}}>16-final</option>
            <option value="3" {{faseSelecionada 3}}>8-final</option>
            <option value="4" {{faseSelecionada 4}}>4-final</option>
            <option value="5" {{faseSelecionada 5}}>Semi-final</option>
            <option value="6" {{faseSelecionada 6}}>Final</option>
            <option value="6" {{faseSelecionada 7}}>Repescagem</option>
           </select>
          </div>
          <div class="col-sm-4 col-xs-12">
           <select id="turno-select" name="turno-select">
            <option disabled="disabled" selected="selected" value="">Selecionar
turno</option>
            <option value="1" {{turnoSelecionado 1}}>Turno</option>
            <option value="2" {{turnoSelecionado 2}}>Returno</option>
           </select>
           <select id="idavolta-select" name="idavolta-select">
            <option disabled="disabled" selected="selected" value="">Selecionar
ordem</option>
            <option value="1" {{idavoltaSelecionado 1}}>Partida de ida</option>
            <option value="2" {{idavoltaSelecionado 2}}>Partida de volta</option>
            <option value="3" {{idavoltaSelecionado 3}}>Jogo único</option>
           </select>
          </div>
```

```
          <div class="col-sm-4 col-xs-12">
            <select id="grupo-select" name="grupo-select">
            <option disabled="disabled" selected="selected" value="">Selecionar
grupo</option>
              {{#each grupos}}
                 <option name="idGrupo" value="{{_id}}" {{grupoSelecionado
_id}}>{{nomeGrupo}}</option>
              {{/each}}
            </select>
          </div>
        </div>

        <div class="row">
          <div class="col-sm-3 col-xs-12">
            <input type="text" name="nomeArbitro" placeholder="Nome do árbitro"
value={{partidas.nomeArbitro}}>
          </div>
          <div class="col-sm-3 col-xs-12">
            <input type="text" name="local" placeholder="Local" value={{partidas.local}}>
          </div>
          <div class="col-sm-3 col-xs-12">
            <input type="date" name="data" placeholder="Data" value={{partidas.data}}>
          </div>
          <div class="col-sm-3 col-xs-12">
            <input type="text" name="horario" placeholder="Horário"
value={{partidas.horario}}>
          </div>
        </div>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
          {{#if isEditGrupo}}
            <select id="time1-select" name="time1-select" required>
            <option disabled="disabled" selected="selected" value="">Selecionar time
1</option>
              {{#each times}}
                <option name="idTime1" value="{{idTime}}" {{time1Selecionado
idTime}}>{{nomeTime}}</option>

              {{/each}}
            </select>
          {{else}}
            <select id="time1-select" name="time1-select" required>
            <option disabled="disabled" selected="selected" value="">Selecionar time
1</option>
              {{#each times}}
```

```
                <option name="idTime1" value="{{_id}}" {{time1Selecionado
_id}}>{{nomeTime}}</option>
              {{/each}}
            </select>
          {{/if}}
          </div>
          <div class="col-sm-6 col-xs-12">
          {{#if isEditGrupo}}
            <select id="time2-select" name="time2-select" required>
            <option disabled="disabled" selected="selected" value="">Selecionar time
2</option>
              {{#each times}}
                <option name="idTime2" value="{{idTime}}" {{time2Selecionado
idTime}}>{{nomeTime}}</option>
              {{/each}}
            </select>
          {{else}}
            <select id="time2-select" name="time2-select" required>
            <option disabled="disabled" selected="selected" value="">Selecionar time
2</option>
              {{#each times}}
                <option name="idTime2" value="{{_id}}" {{time2Selecionado
_id}}>{{nomeTime}}</option>
              {{/each}}
            </select>
          {{/if}}
          </div>
        </div>

        <legend>Súmula da partida</legend>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
            <h3>{{nomeTime1}}</h3>
            <h2>{{golsTime1}}</h2>
          </div>
          <div class="col-sm-6 col-xs-12">
            <h3>{{nomeTime2}}</h3>
            <h2>{{golsTime2}}</h2>
          </div>
        </div>

        <div class="separador"></div>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
```

```
 <legend>Gols {{nomeTime1}}</legend>


 {{#each listaJogadoresTime1Gol}}
 <div class="row">
  <div class="col-xs-4">
     <ul>{{nomeJogador}}</ul>
  </div>
  <div class="col-xs-4">
     <ul>{{numeroGols}}</ul>
  </div>
  <div class="col-xs-4">
    <button type="button" class="btn btn-xs btn-danger removeGol1">
     <span class="glyphicon glyphicon-trash"></span> 
    </button>
  </div>
 </div>
 {{/each}}

 <div class="row">
  <div class="col-sm-4 col-xs-12">
     <label>Jogador</label>
     <select id="jogador1gol-select" name="jogador1gol-select">
     <option disabled="disabled" selected="selected" value="-1">Jogador</option>
        {{#each listaJogadoresTime1}}
          <option name="idJogador1gol" value="{{_id}}">{{nomeJogador}}</option>
        {{/each}}
     </select>
  </div>
  <div class="col-sm-4 col-xs-12">
     <label>Nº Gols</label>
     <input id="golTime1" type="number" name="golTime1" value="0">
  </div>
  <div class="col-sm-4 col-xs-12">
     <br>
     <br>
     <button class="btn btn-primary golTime1">Incluir gols</button>
  </div>
 </div>

</div>
<div class="col-sm-6 col-xs-12">
 <legend>Gols {{nomeTime2}}</legend>

 {{#each listaJogadoresTime2Gol}}
 <div class="row">
```

```
            <div class="col-xs-4">
              <ul>{{nomeJogador}}</ul>
            </div>
            <div class="col-xs-4">
              <ul>{{numeroGols}}</ul>
            </div>
            <div class="col-xs-4">
              <button type="button" class="btn btn-xs btn-danger removeGol2"
formnovalidate>
                <span class="glyphicon glyphicon-trash"></span> 
              </button>
            </div>
          </div>
          {{/each}}

          <div class="row">
            <div class="col-sm-4 col-xs-12">
              <label>Jogador</label>
              <select id="jogador2gol-select" name="jogador2gol-select">
              <option disabled="disabled" selected="selected" value="-1">Jogador</option>
                {{#each listaJogadoresTime2}}
                  <option name="idJogador2gol" value="{{_id}}">{{nomeJogador}}</option>
                {{/each}}
              </select>
            </div>
            <div class="col-sm-4 col-xs-12">
              <label>Nº Gols</label>
              <input id="golTime2" type="number" name="golTime2" value="0">
            </div>
            <div class="col-sm-4 col-xs-12">
              <br>
              <br>
              <button class="btn btn-primary golTime2">Incluir gols</button>
            </div>
          </div>

        </div>
      </div>

      <div class="separador"></div>

      <div class="row">
        <div class="col-sm-6 col-xs-12">
          <legend>Gols Contra {{nomeTime1}}</legend>

          {{#each listaJogadoresTime1GolC}}
```

```
 <div class="row">
  <div class="col-xs-4">
     <ul>{{nomeJogador}}</ul>
  </div>
  <div class="col-xs-4">
     <ul>{{numeroGols}}</ul>
  </div>
  <div class="col-xs-4">
    <button type="button" class="btn btn-xs btn-danger removeGolC1">
     <span class="glyphicon glyphicon-trash"></span> 
    </button>
  </div>
 </div>
 {{/each}}

 <div class="row">
  <div class="col-sm-4 col-xs-12">
     <label>Jogador</label>
     <select id="jogador1golC-select" name="jogador1golC-select">
     <option disabled="disabled" selected="selected" value="-1">Jogador</option>
        {{#each listaJogadoresTime1}}
          <option name="idJogador1golC" value="{{_id}}">{{nomeJogador}}</option>
        {{/each}}
     </select>
  </div>
  <div class="col-sm-4 col-xs-12">
     <label>Nº Gols</label>
     <input id="golCTime1" type="number" name="golCTime1" value="0">
  </div>
  <div class="col-sm-4 col-xs-12">
     <br>
     <br>
     <button class="btn btn-primary golCTime1">Incluir gols</button>
  </div>
 </div>

</div>
<div class="col-sm-6 col-xs-12">
 <legend>Gols Contra {{nomeTime2}}</legend>

 {{#each listaJogadoresTime2GolC}}
 <div class="row">
  <div class="col-xs-4">
     <ul>{{nomeJogador}}</ul>
  </div>
  <div class="col-xs-4">
```

```
                <ul>{{numeroGols}}</ul>
              </div>
              <div class="col-xs-4">
                <button type="button" class="btn btn-xs btn-danger removeGolC2"
formnovalidate>
                  <span class="glyphicon glyphicon-trash"></span> 
                </button>
              </div>
            </div>
            {{/each}}

            <div class="row">
              <div class="col-sm-4 col-xs-12">
                <label>Jogador</label>
                <select id="jogador2golC-select" name="jogador2golC-select">
                <option disabled="disabled" selected="selected" value="-1">Jogador</option>
                  {{#each listaJogadoresTime2}}
                    <option name="idJogador2golC" value="{{_id}}">{{nomeJogador}}</option>
                  {{/each}}
                </select>
              </div>
              <div class="col-sm-4 col-xs-12">
                <label>Nº Gols</label>
                <input id="golCTime2" type="number" name="golCTime2" value="0">
              </div>
              <div class="col-sm-4 col-xs-12">
                <br>
                <br>
                <button class="btn btn-primary golCTime2">Incluir gols</button>
              </div>
            </div>

          </div>
        </div>

        <div class="separador"></div>

        <div class="row">
          <div class="col-sm-6 col-xs-12">
            <legend>C. Amarelos {{nomeTime1}}</legend>

            {{#each listaJogadoresTime1CA}}
            <div class="row">
              <div class="col-xs-4">
                <ul>{{nomeJogador}}</ul>
              </div>
```

```
  <div class="col-xs-4">
    <ul>{{numeroCartoes}}</ul>
  </div>
  <div class="col-xs-4">
    <button type="button" class="btn btn-xs btn-danger removeCA1">
      <span class="glyphicon glyphicon-trash"></span> 
    </button>
  </div>
 </div>
 {{/each}}

 <div class="row">
  <div class="col-sm-4 col-xs-12">
    <label>Jogador</label>
    <select id="jogador1CA-select" name="jogador1CA-select">
    <option disabled="disabled" selected="selected" value="-1">Jogador</option>
      {{#each listaJogadoresTime1}}
        <option name="idJogador1CA" value="{{_id}}">{{nomeJogador}}</option>
      {{/each}}
    </select>
  </div>
  <div class="col-sm-4 col-xs-12">
    <label>Nº C. Amarelos</label>
    <input id="caTime1" type="number" name="caTime1" value="0">
  </div>
  <div class="col-sm-4 col-xs-12">
    <br>
    <br>
    <button class="btn btn-primary caTime1">Incluir cartão</button>
  </div>
 </div>

</div>
<div class="col-sm-6 col-xs-12">
 <legend>C. Amarelos {{nomeTime2}}</legend>

 {{#each listaJogadoresTime2CA}}
 <div class="row">
  <div class="col-xs-4">
    <ul>{{nomeJogador}}</ul>
  </div>
  <div class="col-xs-4">
    <ul>{{numeroCartoes}}</ul>
  </div>
  <div class="col-xs-4">
    <button type="button" class="btn btn-xs btn-danger removeCA2" formnovalidate>
```

```
        <span class="glyphicon glyphicon-trash"></span> 
       </button>
      </div>
     </div>
     {{/each}}

     <div class="row">
      <div class="col-sm-4 col-xs-12">
        <label>Jogador</label>
        <select id="jogador2CA-select" name="jogador2CA-select">
        <option disabled="disabled" selected="selected" value="-1">Jogador</option>
          {{#each listaJogadoresTime2}}
            <option name="idJogador2CA" value="{{_id}}">{{nomeJogador}}</option>
          {{/each}}
        </select>
      </div>
      <div class="col-sm-4 col-xs-12">
        <label>Nº C. Amarelos</label>
        <input id="caTime2" type="number" name="caTime2" value="0">
      </div>
      <div class="col-sm-4 col-xs-12">
        <br>
        <br>
        <button class="btn btn-primary caTime2">Incluir cartão</button>
      </div>
     </div>

   </div>
</div>

<div class="separador"></div>


<div class="row">
  <div class="col-sm-6 col-xs-12">
    <legend>C. Vermelhos {{nomeTime1}}</legend>
    {{#each listaJogadoresTime1CV}}
    <div class="row">
      <div class="col-xs-4">
        <ul>{{nomeJogador}}</ul>
      </div>
      <div class="col-xs-4">
        <ul>{{numeroCartoes}}</ul>
      </div>
      <div class="col-xs-4">
       <button type="button" class="btn btn-xs btn-danger removeCV1">
```

```
        <span class="glyphicon glyphicon-trash"></span> 
      </button>
    </div>
  </div>
  {{/each}}

  <div class="row">
    <div class="col-sm-4 col-xs-12">
      <label>Jogador</label>
      <select id="jogador1CV-select" name="jogador1CV-select">
      <option disabled="disabled" selected="selected" value="-1">Jogador</option>
        {{#each listaJogadoresTime1}}
          <option name="idJogador1CV" value="{{_id}}">{{nomeJogador}}</option>
        {{/each}}
      </select>
    </div>
    <div class="col-sm-4 col-xs-12">
      <label>Nº C. Vermelhos</label>
      <input id="cvTime1" type="number" name="cvTime1" value="0">
    </div>
    <div class="col-sm-4 col-xs-12">
      <br>
      <br>
      <button class="btn btn-primary cvTime1">Incluir cartão</button>
    </div>
  </div>

</div>
<div class="col-sm-6 col-xs-12">
  <legend>C. Vermelhos {{nomeTime2}}</legend>

  {{#each listaJogadoresTime2CV}}
  <div class="row">
    <div class="col-xs-4">
      <ul>{{nomeJogador}}</ul>
    </div>
    <div class="col-xs-4">
      <ul>{{numeroCartoes}}</ul>
    </div>
    <div class="col-xs-4">
      <button type="button" class="btn btn-xs btn-danger removeCV2" formnovalidate>
        <span class="glyphicon glyphicon-trash"></span> 
      </button>
    </div>
  </div>
  {{/each}}
```

```
          <div class="row">
            <div class="col-sm-4 col-xs-12">
              <label>Jogador</label>
              <select id="jogador2CV-select" name="jogador2CV-select">
              <option disabled="disabled" selected="selected" value="-1">Jogador</option>
                {{#each listaJogadoresTime2}}
                  <option name="idJogador2CV" value="{{_id}}">{{nomeJogador}}</option>
                {{/each}}
              </select>
            </div>
            <div class="col-sm-4 col-xs-12">
              <label>Nº C. Vermelhos</label>
              <input id="cvTime2" type="number" name="cvTime2" value="0">
            </div>
            <div class="col-sm-4 col-xs-12">
              <br>
              <br>
              <button class="btn btn-primary cvTime2">Incluir cartão</button>
            </div>
          </div>

        </div>
      </div>

      <div class="separador"></div>

      <div class="row">
        <div class="col-sm-6 col-xs-12">
          <button type="submit" name="cadastrar" >Salvar</button>
        </div>
        <div class="col-sm-6 col-xs-12">
          <button class="cancela" formnovalidate>Cancelar</button>
        </div>
      </div>
    </form>
{{> footer}}
</template>
```

```
<template name="home">
  {{> nav}}

        <div class="row">

              <div class="col-xs-2">
```

```
                        <h2>Campeonato</h2>
                    </div>
                    <div class="col-xs-6">
                    <select id="campeonato-select" name="campeonato-select">
                        <option name="idCampeonato2" value="">Selecione o
campeonato</option>
                        {{#each campeonatos}}
                  <option name="idCampeonato" value="{{_id}}" >{{nomeCampeonato}}</option>
                        {{/each}}
                    </select>
                    </div>

        </div>

    <div class="row">
        <div class="col-md-4 col-sm-6 col-xs-12">
                        {{> tabelaClassificacao}}
                    </div>
                    <div class="col-md-4 col-sm-6 col-xs-12">
                        {{> partidasRealizadas}}
                    </div>
                    <div class="col-md-4 col-sm-6 col-xs-12">
                        {{> proximasPartidas}}
                    </div>
        <div class="col-md-4 col-sm-6 col-xs-12">
                        {{> tabelaArtilharia}}
                    </div>
                    <div class="col-md-4 col-sm-6 col-xs-12">
                        {{> tabelaMelhoresDefesas}}
                    </div>
                    <div class="col-md-4 col-sm-6 col-xs-12">
                        {{> tabelaPunicoes}}
                    </div>
    </div>

  {{> footer}}
</template>
```

```
Template.home.helpers({

        campeonatos : function() {

                return Campeonatos.find({},
                        { sort: {nomeCampeonato: 1} });
```

```
        }
});

Template.home.events({

        "change #campeonato-select": function(e, t){
                var idCampeonato = $("#campeonato-select option:selected").val();
                Session.set("filtroCampeonato", idCampeonato);

        return false;
   },
});
```

```
<template name="partidasRealizadas">
        <div class="fundo">
                <h1>Partidas realizadas</h1>


        <div class="row">

                <div class="col-xs-2">
                        <h2>Grupo</h2>
                </div>
                <div class="col-xs-10">
                <select id="grupo-select" name="grupo-select">
                        <option name="idGrupo2" value="" selected>Selecione o
grupo</option>
                {{#each grupos}}
        <option name="idGrupo" value="{{_id}}">{{nomeGrupo}}</option>
                {{/each}}
        </select>
        </div>

        </div>

        <div class="separador"></div>

                <div class="pre-scrollable">
        {{#each partidas}}
                        <div class="row fundo-listagem">
                                        <div class="row">
                                                <div class="col-xs-12" style="text-align: center">
                                                        <h5>{{fase}} <b>{{data}}</b>  {{local}}
<b>{{horario}}</b> </h5>
```

```
                                        </div>
                                    </div>
                                    <div class="row">
                                        <div class="col-xs-12" style="text-align: center">
                                            <h2>{{time1.nomeTime}} {{placarTime1}}
x {{placarTime2}} {{time2.nomeTime}}</h2>
                                        </div>
                                    </div>

                                </div>
                        {{/each}}
                        </div>
                </div>
</template>
```

```
Template.partidasRealizadas.helpers({

        partidas : function() {

                var filtroGrupo = Session.get('grupoSelect');

                return partidas.find({"campeonato.idGrupo": filtroGrupo, status : 3},
                        { sort: {data: 1} });

        },

        grupos : function() {
                var filtroCampeonato = Session.get('filtroCampeonato');

                return Grupos.find({"idCampeonato": filtroCampeonato}, { sort: {nomeGrupo:
1} });
        }
});

Template.partidasRealizadas.events({

        "change #grupo-select": function(e, t){
                var idGrupo = $("#grupo-select option:selected").val();
                Session.set("grupoSelect", idGrupo);

        return false;
    },
});
```

```
<template name="proximasPartidas">
	<div class="fundo">
		<h1>Próximas partidas</h1>


	<div class="row">

		<div class="col-xs-2">
			<h2>Grupo</h2>
		</div>
		<div class="col-xs-10">
		<select id="grupo-select" name="grupo-select">
			<option name="idGrupo2" value="" selected>Selecione o
grupo</option>
		{{#each grupos}}
	<option name="idGrupo" value="{{_id}}">{{nomeGrupo}}</option>
		{{/each}}
	</select>
	</div>

	</div>

	<div class="separador"></div>

		<div class="pre-scrollable">
	{{#each partidas}}
			<div class="row fundo-listagem">
				<div class="row">
					<div class="col-xs-12" style="text-align: center">
						<h5>{{fase}} <b>{{data}}</b>  {{local}}
<b>{{horario}}</b> </h5>
					</div>
				</div>
				<div class="row">
					<div class="col-xs-12" style="text-align: center">
						<h2>{{time1.nomeTime}} x
{{time2.nomeTime}}</h2>
					</div>
				</div>

			</div>
		{{/each}}
		</div>
	</div>
</template>
```

```
Template.proximasPartidas.helpers({

	partidas : function() {

		var filtroGrupo = Session.get('grupoSelect');

		return partidas.find({"campeonato.idGrupo": filtroGrupo, status : 3},
			{ sort: {data: 1} });

	},

	grupos : function() {
		var filtroCampeonato = Session.get('filtroCampeonato');

		return Grupos.find({"idCampeonato": filtroCampeonato}, { sort: {nomeGrupo:
1} });
	}
});

Template.proximasPartidas.events({

	"change #grupo-select": function(e, t){
		var idGrupo = $("#grupo-select option:selected").val();
		Session.set("grupoSelect", idGrupo);

	return false;
   },
});
```

```
<template name="tabelaArtilharia">
	<div class="fundo">
		<h1>Artilharia</h1>


	<table class="table table-bordered">
		  <thead>
		    <tr>
		      <th>Jogador</th>
		      <th>Time</th>
		      <th>Número de gols</th>
		    </tr>
		  </thead>
		  <tbody>
```

```
                {{#each artilharia}}
                  <tr>
                    <td>{{nomeJogador}}</td>
                    <td>{{nomeTime}}</td>
                    <td>{{numeroGols}}</td>
                  </tr>
                {{/each}}
                </tbody>
              </table>

        </div>
</template>
```

```
Template.tabelaArtilharia.helpers({

        artilharia : function() {
                var valorFiltro = Session.get('filtroCampeonato');

                return Artilharias.find({idCampeonato: valorFiltro },
                        { sort: {numeroGols: -1} });

        }
});
```

```
<template name="tabelaClassificacao">
        <div class="fundo">
                <h1>Classificação</h1>

                <div class="row">

                <div class="col-xs-2">
                        <h2>Grupo</h2>
                </div>
                <div class="col-xs-10">
                <select id="grupo-select" name="grupo-select">
                        <option name="idGrupo2" value="" selected>Selecione o
grupo</option>
                {{#each grupos}}
        <option name="idGrupo" value="{{_id}}" >{{nomeGrupo}}</option>
                {{/each}}
        </select>
        </div>
```

```
        </div>

        <table class="table table-bordered">
                <thead>
                  <tr>
                    <th>Nome do time</th>
                    <th>Pontos</th>
                    <th>J</th>
                    <th>V</th>
                    <th>E</th>
                    <th>D</th>
                    <th>GP</th>
                    <th>GS</th>
                    <th>SG</th>
                  </tr>
                </thead>
                <tbody>
                {{#each timeClassificacao}}
                  <tr>
                    <td>{{nomeTime}}</td>
                    <td>{{pontos}}</td>
                    <td>{{jogos}}</td>
                    <td>{{vitorias}}</td>
                    <td>{{empates}}</td>
                    <td>{{derrotas}}</td>
                    <td>{{golsPro}}</td>
                    <td>{{golsContra}}</td>
                    <td>{{saldoGol}}</td>
                  </tr>
                      {{/each}}
                </tbody>
              </table>

        </div>
</template>
```

```
Template.tabelaClassificacao.helpers({

grupos : function() {
                var filtroCampeonato = Session.get('filtroCampeonato');

                return Grupos.find({"idCampeonato": filtroCampeonato}, { sort: {nomeGrupo:
1} });
        },
```

```
timeClassificacao : function() {
		var filtroGrupo = Session.get('grupoSelect');

		return Tabelas.find({"idGrupo": filtroGrupo}, { sort: {pontos: -1} });
	}

});

Template.tabelaClassificacao.events({

	"change #grupo-select": function(e, t){
		var idGrupo = $("#grupo-select option:selected").val();
		Session.set("grupoSelect", idGrupo);

	return false;
   },
});
```

```
<template name="tabelaMelhoresDefesas">
	<div class="fundo">
		<h1>Melhores defesas</h1>


	<table class="table table-bordered">
		  <thead>
		    <tr>
		      <th>Nome do Time</th>
		      <th>Número de gols sofridos</th>
		    </tr>
		  </thead>
		  <tbody>
		  {{#each melhoresDefesas}}
		    <tr>
		      <td>{{nomeTime}}</td>
		      <td>{{numeroGols}}</td>
		    </tr>
		   {{/each}}
		  </tbody>
		</table>

	</div>
</template>
```

```
Template.tabelaMelhoresDefesas.helpers({

	melhoresDefesas : function() {
		var valorFiltro = Session.get('filtroCampeonato');

		return MelhoresDefesas.find({idCampeonato: valorFiltro },
			{ sort: {numeroGols: 1} });

	}
});
```

```
<template name="tabelaPunicoes">
	<div class="fundo">
		<h1>Jogadores punidos</h1>


	<table class="table table-bordered">
		  <thead>
		    <tr>
		      <th>Jogador</th>
		      <th>Time</th>
		      <th>Número cartões amarelos</th>
		      <th>Número cartões vermelhos</th>
		    </tr>
		  </thead>
		  <tbody>
		  {{#each jogadoresPunidos}}
		    <tr>
		        <td>{{nomeJogador}}</td>
		      <td>{{nomeTime}}</td>
		      <td>{{numeroCartaoAmarelo}}</td>
		      <td>{{numeroCartaoVermelho}}</td>
		    </tr>
		  {{/each}}
		  </tbody>
		</table>

	</div>
</template>
```

```
Template.tabelaPunicoes.helpers({
```

```
	jogadoresPunidos : function() {
		var valorFiltro = Session.get('filtroCampeonato');

		return jogadoresPunidos.find({idCampeonato: valorFiltro },
			{ sort: {numeroCartaoVermelhos: -1} });

	}
});
```

APÊNDICE A

Aplicação Web para gerenciamento de campeonatos de futebol

João Luiz Mafra Ferreira[1], Leandro José Komosinski[1]

[1]Departamento de Informática e estatística
Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC - Brasil

joaomafra21@gmail.com, leandro.komosinski@ufsc.br

Abstract. *This meta-paper describes the development process of a web application to manage soccer tournaments. The article show the steps to achieve the main objective, which is to produce a tool to facilitate the tournament administration besides an area to players and sympathizer follow the progress of the championship.*

Resumo. *Este artigo descreve o processo de desenvolvimento de uma aplicação web para realizar a gerência de um campeonato de futebol. O artigo mostra os passos realizados para atingir o objetivo principal, que é a produção de uma ferramenta que facilite a administração de um campeonato além de uma área para jogadores e simpatizantes acompanharem o andamento do campeonato.*

1. Introdução

Eventos esportivos necessitam de uma boa organização e planejamento para conseguirem manter a qualidade da competição e ocorrer tudo do melhor jeito. Para organizar um campeonato de futebol também não é diferente, é necessário manter tudo organizado e mapeado para não haver erros no resultado da competição. Ter as informações precisas pode ser complicado, e ter uma ferramenta que possibilite ter as informações reunidas em um lugar pode ser algo que facilite muito a administração do campeonato.

Além de facilitar o gerenciamento do campeonato, uma ferramenta própria para isso traz a oportunidade de criar um ambiente onde jogadores e pessoas interessadas acompanhem os resultados dos jogos, rankings de artilharia, cartões, melhores defesas assim como próximos jogos a serem realizados e tabela de classificação.

A metodologia utilizada para realizar o desenvolvimento da aplicação foi baseada no Scrum para se obter um desenvolvimento ágil e eficaz. Apesar de ser previsto para equipes, o Scrum foi adaptado para um só membro, onde este teve a responsabilidade por todos os papéis previstos.

O protótipo desenvolvimento não está completo e possui algumas funcionalidades que poderão sem implementadas em trabalhos futuros, para garantir ainda mais um bom gerenciamento de um campeonato de futebol.

2. Metodologia utilizada

A metodologia para o desenvolvimento, como ja citado anteriormente, foi uma adaptação do Scrum. O Scrum é um framework de desenvolvimento ágil que se torna iterativo e incremental a medida que as etapas e ciclos do desenvolvimento vão sendo entregues. Originalmente, o Scrum é previsto para equipes onde cada membro tem o seu papel, porém com a adaptação realizada para equipe de um só membro, este foi responsável por todos os papéis previstos.

Os papéis que o membro único desenvolveu foram os de Product Owner, Scrum Master e Development Team. O item também previsto no Scrum conhecido como Daily Scrum, é desnecessário neste contexto de um único membro, porém todos os outros eventos e artefatos são mantidos.

Na figura 1 é apresentado o gerenciamento das atividades previstas. Cada um dos requisitos foi implementado dentro dos Sprints definidos para o desenvolvimento.

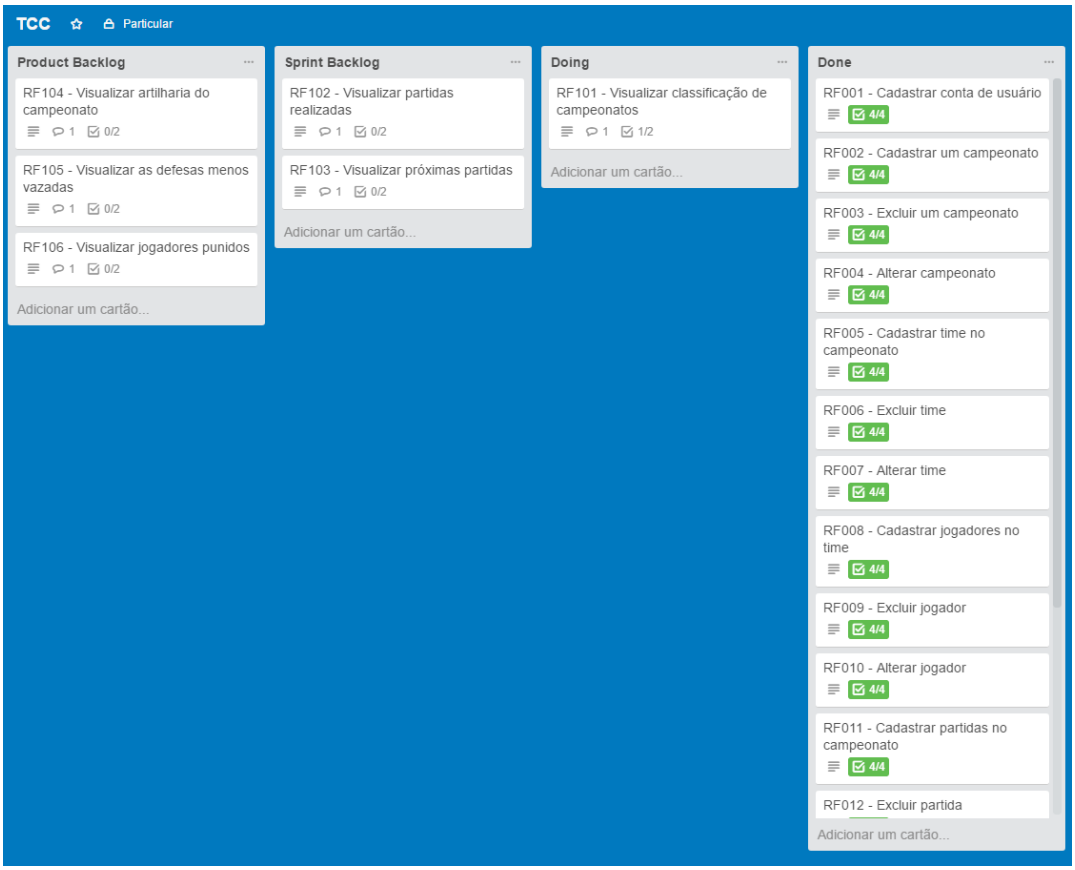


Figura 1. Gerenciamento de tarefas no Trello

3. Requisitos do sistema

O levantamento de requisitos do sistema é o processo que busca definir informações e funcionalidades necessárias para o sistema. Para realizar este processo existem algumas

técnicas conhecidas que auxiliam na formação dos requisitos. As técnicas utilizadas no desenvolvimento da aplicação foram Brainstorm e Prototipagem.

A técnica de Brainstorming tem como finalidade a geração de ideias pertinentes ao sistema. Os participantes são selecionados de acordo com as possíveis contribuições ao sistema e é dado preferência aqueles indivíduos que possuem conhecimento sobre o domínio do sistema. A técnica foi utilizada de um modo informal com amigos do autor que já participaram ou tiverem interesse de acompanhar algum campeonato além de um organizador de campeonatos de nível amador.

A segunda técnica utilizada pelo autor foi a de prototipagem. A prototipagem busca explorar aspectos de determinadas funcionalidades com a finalidade de testar principalmente o cumprimento dos requisitos levantados previamente. Além disso também é indicada para questões de interface gráfica e usabilidade do sistema. Ao fim do desenvolvimentos das funcionalidades previstas para o Sprint , o protótipo era testado pelos mesmos indivíduos que auxiliaram na etapa de Brainstorming e os pontos que eram levantados por eles eram levados em consideração na Sprint seguinte.

Na figura 2 é possível visualizar os requisitos levantados no sistema, junto com uma descrição e estimativa de tempo em horas para desenvolver tal funcionalidade.

Sprint	Descrição	Release	Estimativa
Cadastrar conta de usuário	O usuário poderá criar uma conta com login e senha para acessar o sistema	1	8
Cadastrar campeonato	O usuário cadastrado poderá criar um campeonato	2	10
Alterar campeonato	O usuário cadastrado poderá alterar dados do campeonato	2	8
Excluir campeonato	O usuário cadastrado poderá excluir um campeonato	2	2
Cadastrar time no campeonato	O usuário cadastrado poderá cadastrar um time e escolher o campeonato que ele participa	3	10
Alterar time	O usuário cadastrado poderá alterar dados do time	3	8
Excluir time	O usuário cadastrado poderá excluir um time	3	2
Cadastrar jogadores no time	O usuário cadastrado poderá adicionar jogadores ao seus times criados	4	15
Alterar jogadores	O usuário cadastrado poderá alterar dados do jogador	4	8
Excluir jogador	O usuário cadastrado poderá excluir um jogador	4	2
Cadastrar partidas no campeonato	O usuário cadastrado poderá adicionar partidas ao campeonato e editar os eventos desta partida, tais como gols e cartões	5	20
Alterar partida	O usuário cadastrado poderá alterar a partida e seus eventos	5	8
Excluir partidas	O usuário cadastrado poderá excluir uma partida	5	2
Cadastrar grupos no campeonato	O usuário cadastrado poderá adicionar grupos aos campeonatos com formato de "Copa"	6	10
Alterar grupo	O usuário cadastrado poderá alterar os grupos de campeonatos com formato "Copa"	6	8
Excluir grupos	O usuário cadastrado poderá excluir um grupo	6	8
Visualizar classificação do campeonato	Qualquer usuário poderá visualizar a classificação de um campeonato	7	8
Visualizar partidas realizadas	Qualquer usuário poderá visualizar as partidas realizadas de um campeonato	7	8
Visualizar próximas partidas	Qualquer usuário poderá visualizar as próximas partidas de um campeonato	7	6
Visualizar artilharia do campeonato	Qualquer usuário poderá visualizar a artilharia de um campeonato	8	6
Visualizar as melhores defesas	Qualquer usuário poderá visualizar as melhores defesas de um campeonato	8	6
Visualizar jogadores punidos	Qualquer usuário poderá visualizar a lista de jogadores punidos de um campeonato	8	6

Figura 2. Requisitos do sistema

4. Tecnologias utilizadas

Para o desenvolvimento da aplicação foi utilizado um framework open-source Javascript chamado MeteorJS. O framework foi escolhido pois segundo a própria empresa que disponibiliza o framework, é muito fácil e ágil de utilizar. Meteor possui uma série de vantagens que facilitam muito o desenvolvimento de uma aplicação, como por exemplo a abstração de arquivos de configuração e extensões, onde consegue identificar os arquivos html, css e js sem precisar importá-los nas páginas que desejarem utilizar algum código. Meteor também se torna muito simples pelo fato de ser fullstack, tendo seu front-end, back-end e banco de dados integrados facilmente e desenvolvidos em uma só linguagem: javascript. Além disso o framework utiliza o conceito de reatividade, que tem a ver com o fluxo de dados e a propagação das mudanças de estado dos objetos.

Como banco de dados, por ser o padrão do framework Meteor, MongoDB foi utilizado. Mongo é um banco de dados não-relacional orientado a documentos. Este tipo de banco de dados armazena os dados em documentos que são onde os dados são armazenados em formato JSON. Isso facilita ao desenvolvedor enxergar os dados e extrair as informações necessárias.

5. Implementação

O desenvolvimento ocorreu seguindo as Sprints definidas para cada release. Foram previstas um total de 8 Sprints que tinham como finalidade cumprir os requisitos do sistema.

Nas figuras 3 e 4 podemos visualizar os resultados alcançados em duas sprints definidas pelo autor.

Nome do time	Pontos	J	V	E	D	GP	GS	SG
Avai	9	3	3	0	0	9	3	6
Criciúma	6	3	2	0	1	4	3	1
Chapecoense	3	3	1	0	2	3	6	-3
Figueirense	0	3	0	0	3	2	6	-4

Figure 3. Tabela de classificação disponibilizada

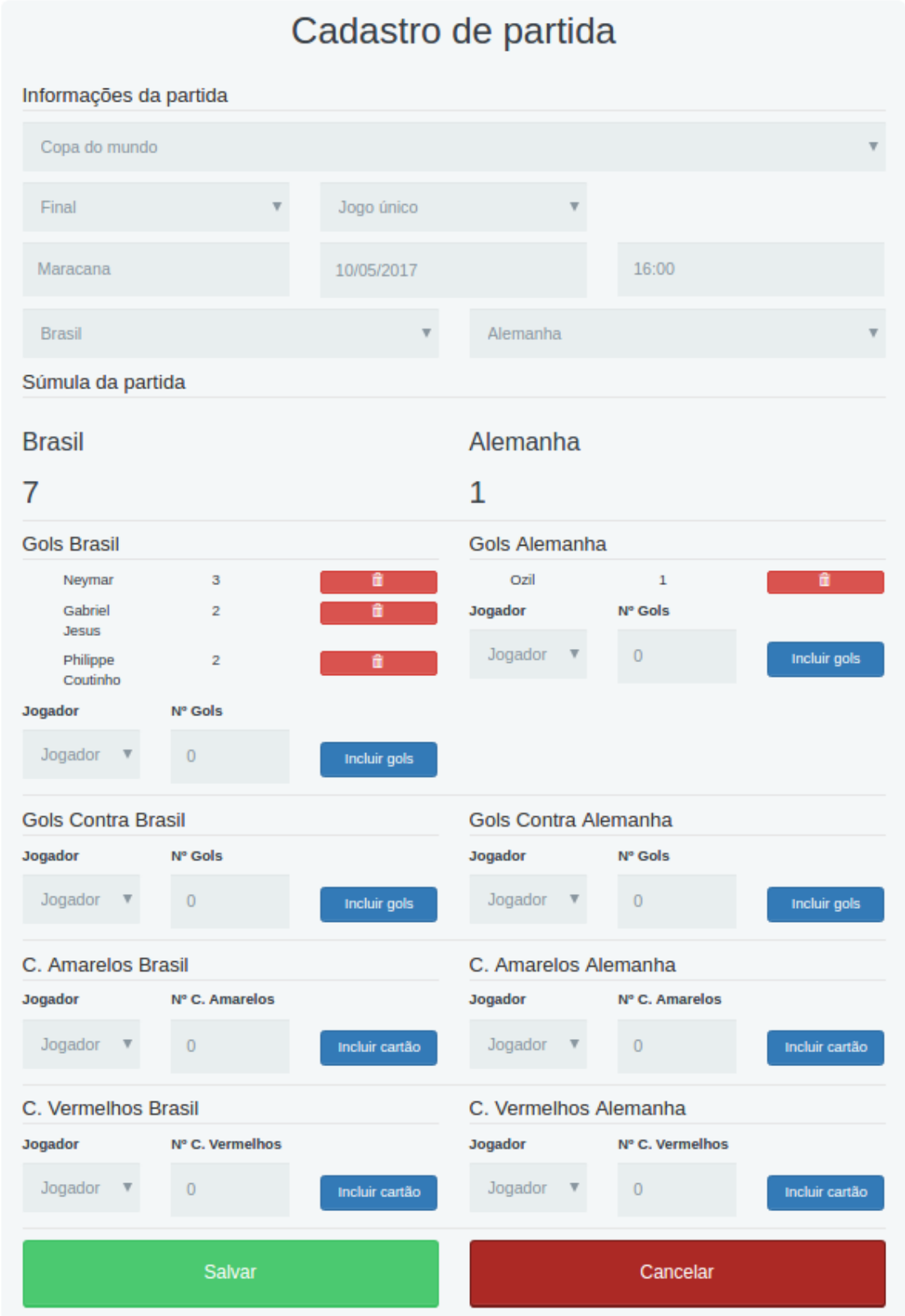


Figure 4. Cadastro de partida

6. Conclusão

Como conclusão do projeto, observou-se o cumprimento dos requisitos levantados, além de atingir os objetivos gerais e específicos definidos pelo autor. O framework utilizado mostrou-se muito útil e com uma ótimo curva de aprendizado, considerando que o autor não possuía muito conhecimento da linguagem javascript.

O Scrum também foi uma ótima escolha pois permitiu uma organização alta, além de agilidade no desenvolvimento, muito pelo sua iteratividade atingidas com os ciclos do Sprint.

Apesar de atingir os objetivos específicos e geral, ainda há algumas funcionalidades que podem ajudar ainda mais o sistema ser uma boa ferramenta para organizadores de um campeonato.

Referências

MongoDB. MONGODB for Giant ideas. Disponível em: <https://www.mongodb.com/mongodb-architecture>. Acesso em: 07 de Julho de 2017.

Meteor. METEOR, the fatest way to build JS APPS. Disponível em: <https://www.meteor.com/>. Acesso em: 07 de Julho de 2017.

SCHWABER, Ken; SUTHERLAND, Jeff. The Scrum Guide. Disponível em: <http://www.scrumguides.org/scrum-guide.html>. Acesso em 07 julho 2017.

SOMMERVILLE, Ian. Engenharia de software. Tradução por Selma Shin Shimizu Melnikoff; Reginaldo Arakaki; Edilson de Andrade Barbosa. 8. ed. São Paulo.