UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAPHAEL DE LEON FERREIRA LUPCHINSKI

Desenvolvimento de uma Aplicação de Página-Única e Banco de Dados Não-Relacional para Organização e Controle de Eventos Esportivos

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Leandro Krug Wives

Porto Alegre
2015

AGRADECIMENTOS

Os mais profundos agradecimento à família acolhedora, carinhosa e incentivadora que tenho e que muito me auxiliou para a conclusão deste trabalho. À minha namorada, que cuidou de mim em nossa nova casa e me deu total apoio durante este projeto. À minha querida dinda Maria Lúcia (*In Memoriam*), que infelizmente não estará conosco para presenciar a conclusão desta importante etapa. Aos demais amigos e familiares pelo carinho durante todos esses anos. Aos colegas da ADP, pelas ideias e apoio para concluir este trabalho e finalmente ao Prof. Leandro Wives pela receptividade, incentivo, ter acreditado nesta ideia e por ter me guiado durante estes meses.

RESUMO

O esporte é uma peça fundamental na sociedade pelo seu caráter transformador e inclusivo na vida das pessoas. A prática esportiva promove ganhos físicos e mentais notáveis, melhorando a saúde e aumentando as capacidades motoras. É comum encontrar uma abundância de aplicações que visam auxiliar a prática de esportes, em especial gerenciando grupos de pessoas com interesses esportivos similares. Este trabalho descreve os passos tomados para a implementação de uma aplicação que cobre funcionalidades de gerenciamento de grupos e eventos esportivos, além de apresentar uma pesquisa realizada com diversos participantes acerca das qualidades do produto final. Trata-se de uma aplicação de página única (SPA) desenvolvida em AngularJS, utilizando um banco de dados não-relacional e sistema de autenticação fornecidos pelo Firebase.

Palavras-chave: Angularjs. SPA. firebase. NoSQL. bootstrap. eventos esportivos.

Developing a Single-Page Application with NoSQL Database for Sports Events Management

ABSTRACT

Sports are an essential component in our society judging by its shifter and inclusive character in peoples' lives. Practicing sports grants notable physical and mental gains, increasing our health and motor skills. It's usual to find an abundance of applications that aims to support sports practice, particularly managing groups of people with similar sports interests. This work describes the steps taken to implement such application that covers these groups and events management features and also shows the results of a survey answered by several participants on final product's qualities. This application follows a SPA model, developed in AngularJS using a non-relational database and authentication system provided by Firebase.

Keywords: angularjs, SPA, firebase, NoSQL, bootstrap, sports events.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BAAS	Back-end As A Service
CSS	Cascading Style Sheets
DOM	Document Object Model
HTML	HyperText Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
MVW	Model-View-Whatever
NoSQL	Not Only Structured Query Language
RDBMS	Relational Database Management System
VCS	Version Control Systems
WYSIWYM	What You See Is What You Mean
XHR	XML HTTP Request
XML	eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

O esporte é uma atividade metódica que demanda esforço físico e mental do seu praticante, podendo ser classificado de diversas maneiras. Normalmente, leva-se em consideração o número de participantes da atividade, o grau de colaboração e o tipo de terreno da prática. No Brasil, segundo a pesquisa Diagnóstico Nacional do Esporte (ESPORTE, 2013) realizada pelo Ministério do Esporte, é estimado que 54,1% da população entre 15 e 74 anos pratique algum esporte ou atividade física. A intenção do governo é incentivar que esses números melhorem mediante a alarmante quantidade de sedentários no país, incentivos da ONU e a aproximação dos Jogos Olímpicos do Rio em 2016. O Diesporte é uma iniciativa inédita no país, não sendo possível avaliar com precisão uma tendência de melhoria ou não desses números com relação aos censos anteriores.

Atualmente existem diversas aplicações oferecidas gratuitamente para auxiliar na organização de eventos esportivos, mas poucas delas para o público no Brasil. Em especial, ferramentas como o Joga +1, da Google, e o Peladeiro.com atuam como os principais sites onde o usuário consegue organizar e gerenciar uma atividade esportiva. Outras ferramentas de amplo alcance entre os praticantes de esporte são utilizadas na tentativa de organizar eventos, como o Facebook e o WhatsApp. Nestas há uma limitação de funcionalidades, abordadas nos capítulos futuros, que efetivamente colaborem para a realização da prática do esporte.

Afim de oferecer uma ferramenta que cubra as lacunas presentes nesta área, o autor deste trabalho se propõe e desenvolver uma ferramenta de organização e gerência de grupos para a prática de esportes variados, lançando mão de tecnologias modernas de desenvolvimento web e banco de dados não-relacional.

1.1 Objetivo

O objetivo deste trabalho é o projeto e o desenvolvimento de uma aplicação Web para a gestão e controle de grupos e eventos esportivos. Em especial, oferecendo uma interface intuitiva, responsiva e adaptativa para o usuário, possibilitando a prática de múltiplos esportes e facilitando a escolha entre tantas aplicações para organizar um evento esportivo.

Adicionalmente este trabalho tem como objetivo apresentar o uso de tecnologias modernas de desenvolvimento Web, como o AngularJS, para desenvolver uma aplica-

ção puramente Front-End, onde toda a execução da lógica da aplicação se dá no lado do cliente. Para tanto foi utilizada uma tecnologia BaaS (*Back-End as a Service*) chamada Firebase que oferece hospedagem, sistema de autenticação e banco de dados não-relacional para manipulação de documentos JSON.

1.2 Estrutura do texto

Este trabalho está dividido em 7 capítulos. O capítulo 1 abordada a contextualização e a motivação do trabalho, apresentando em seguida os objetivos e estrutura do texto. O capítulo 2 se desenvolve em torno das tecnologias utilizadas neste trabalho e aplicações e trabalhos relacionados, descrevendo os diferencias desta aplicação para as demais e fundamentando a sua necessidade. O capítulo 3 explica o projeto da aplicação, descrevendo suas funcionalidades e tecnologias. O capítulo 4 fala sobre o desenvolvimento da aplicação, detalhando as abordagens tomadas para a manipulação dos dados em um banco de dados não-relacional. O capítulo 5 apresenta o uso da aplicação e suas funcionalidades. O capítulo 6 descreve a avaliação por usuários do aplicativo disponível online. Finalmente, o capítulo 7 apresenta as conclusões, retrospectivas e resultados obtidos além de relacionar outras funcionalidades que podem ser adicionadas em versões futuras do projeto.

2 FUNDAMENTOS, APLICAÇÕES E TRABALHOS RELACIONADOS

Neste capítulo são abordados dois temas: o primeiro é uma base teórica e conjunto de fundamentos e tecnologias utilizados para o desenvolvimento deste trabalho. O segundo consiste numa descrição das aplicações e trabalhos existentes que oferecem soluções (similares) para a organização de eventos esportivos. Essas serão descritas abordando suas funcionalidades, escopo e estatísticas e tipos de uso. É importante frisar que nem todas aplicações fazem uma divulgação detalhada de seus números, não sendo possível avaliar com clareza seu crescimento e abrangência.

2.1 Fundamentos e Tecnologias

Dado que o foco deste trabalho é uma aplicação Web, é fundamental que seja feita uma descrição detalhada das tecnologias utilizadas. Para tanto, serão demonstrados aqui o básico de alguns conceitos e tecnologias empregadas nesta aplicação, como a separação entre Front-End e Back-End, Javascript, AngularJS e Firebase.

2.1.1 Front-End / Back-End

O Front-End de uma aplicação é o conjunto de módulos que fazem a captura de dados inseridos por um usuário para serem processados e módulos que transmitem uma informação processada ou armazenada de volta para o usuário. O Back-End é o conjunto de módulos que recebem dados enviados por um usuário, os processa, armazena ou não, e envia para o usuário feedback de suas ações ou respostas aos seus pedidos.

Esta separação também é conhecida como Client-Side / Server-Side (TABLELESS, 2012), onde se define que uma aplicação é Client-Side se sua execução e processamentos ocorrem no navegador do usuário, ou Server-Side caso haja um servidor responsável por executar os pedidos de um usuário para posteriormente lhes enviar a resposta desejada.

Para este trabalho é importante fazer uma separação dos conceitos dado que o foco é proporcionar uma aplicação web sem a necessidade de codificar nada no lado do servidor, movendo toda a regra de negócios para o lado do cliente e deixando o armazenamento dos dados em um banco não-relacional a cargo do Firebase (apresentado na seção 2.1.5).

2.1.2 HTML, CSS e DOM

HTML e CSS são as ferramentas básicas para gerar páginas Web estáticas. A linguagem de marcação HTML é reconhecida por todos os navegadores modernos utilizando o paradigma WYSIWYM, mostrando um resultado compilado e possivelmente diferente da forma como foi escrito. CSS entra no conjunto das ferramentas suportadas pelos navegadores modernos agindo como um conjunto de regras de estilo para certos componentes. Com tais regras é possível alterar totalmente a aparência de um documento, tornando a aplicação mais agradável para o usuário final.

O código HTML, agregado aos estilos do CSS, é compilado e transformado no DOM, que fica armazenado na memória do navegador de internet (FRANKLIN, 2011; COYIER, 2013). O armazenamento do DOM se dá pela estrutura de árvore e este é acessado por meio do DOM API, que oferece métodos para acessar um nodo específico desta árvore para manipular seu conteúdo, estilo ou eventos relacionados. É comum que a manipulação do DOM seja feita com o auxílio de JavaScript e bibliotecas como p. ex. JQuery.

Neste trabalho serão utilizados HTML e CSS para a geração das páginas da aplicação. Adicionalmente, AngularJS (seção 2.1.4) e Bootstrap (seção 2.1.6) farão uma extensa manipulação destas ferramentas para uma melhor experiência do usuário, garantindo uma aplicação responsiva e adaptativa sem duplicação de código ou manipulação direta no DOM.

2.1.3 JavaScript

(MOZILLA, 2015a) define a linguagem como "[..] leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecido como uma linguagem de script para páginas Web [..]"e é suportada atualmente por todos os browsers modernos a partir de sua versão 5.1. JavaScript é também interpretado por ambientes sem browser, como o node.js (NODEJS, 2015) que roda em servidores e desktops utilizando um motor de interpretação criado pela Google, batizado de *V8*.

JavaScript sofre de um problema bem conhecido entre seus programadores: Executar todos os seus processos em uma única *thread*. Internamente, JS possui apenas uma fila de execução de tarefas, e cada interação que o usuário tem com a aplicação gera um evento, que é adicionado à esta fila de tarefas para ser executada em um momento futuro.

Uma complicação desse modelo é que uma rotina que ocupe a thread principal por muito tempo interfira no processamento da Interface do Usuário, acarretando em travamento na aplicação e possivelmente garantindo que o usuário afetado jamais volte a usar o produto em questão (ROUSSET, 2015).

JavaScript é o âmago desta aplicação, servindo de base para o framework AngularJS (seção 2.1.4) e como principal motor de execução da lógica de negócios do sistema.

2.1.4 AngularJS

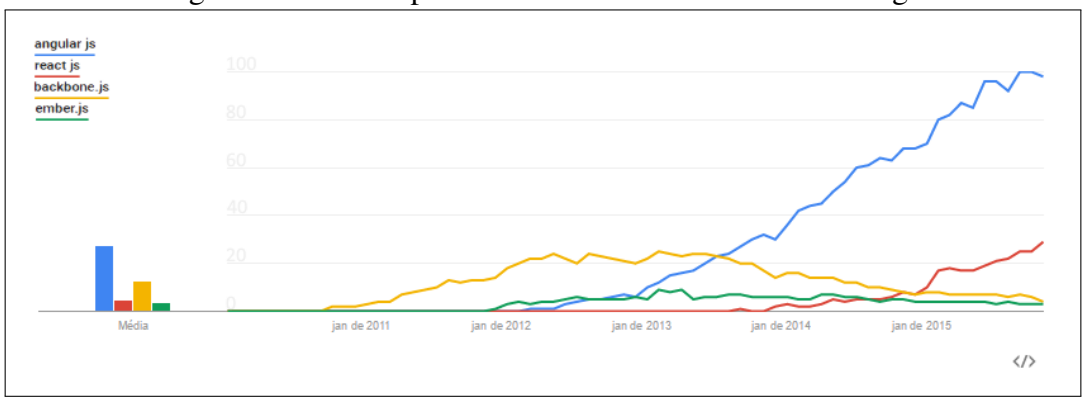
Criado pela Google em 2009 e auto-entitulado "Um Framework JavaScript MVW Superheróico", AngularJS (ANGULARJS, 2010a; GREEN; SESHADRI, 2013) entra no conjunto dos mais utilizados frameworks Front-End de desenvolvimento Web para JavaScript. Segundo o site Developer Economics (CLASSEN, 2015), AngularJS aparece, de acordo com estatísticas coletadas de (GITHUB, 2014), como o framework com mais estrelas e contribuidores, seguido dos frameworks Backbone.js, Ember.js e ReactJS (desenvolvido e mantido pelo facebook). De acordo com o Google Trends (figura 2.1), AngularJS segue isolado em primeiro lugar como o termo mais buscado dentre os quatro maiores frameworks Front-End de JS desde Janeiro de 2010.

AngularJS é categorizado como um framework que roda no lado do cliente e emprega uma mistura das arquiteturas MVC e MVVM (batizada de MVW (ou MV*)), dadas suas características e habilidades de manipulação direta de elementos no DOM sem a necessidade de utilizar um Controlador para tal. Esta funcionalidade, conhecida como *Two-Way Data Binding* (ou *Ligação de Dados Bidirecional*), dá ao usuário a capacidade de transmitir dados entre elementos do DOM sem que um código JS explicitamente escrito pelo usuário seja executado, também evitando manipulações diretas do DOM com JQuery e outras bibliotecas de acesso ao DOM API (ANGULARJSa).

O framework possui uma série de serviços nativos para lidar com problemas comuns do Javascript, como por exemplo lidar com chamadas XHR ($http), manipular rotas de aplicações SPA ($routeProvider), gerenciar *Promises* (MOZILLA, 2015b) para processamento assíncrono ($q), etc.

AngularJS possui uma curva de aprendizado muito suave e foi uma ferramenta chave para a construção desta aplicação por permitir que todo o processamento e regras de negócios pudessem ser calculadas do lado do cliente.

Figura 2.1: Buscas por frameworks Front-End JS no Google



Fonte: (GOOGLE, 2015)

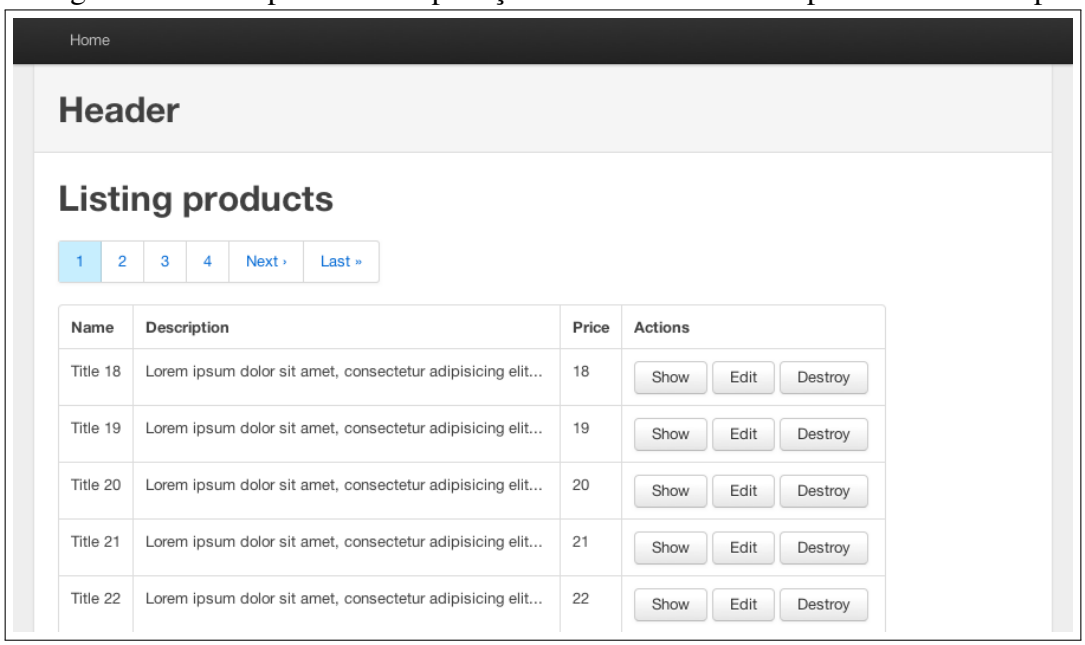
2.1.5 Firebase

Firebase é um serviço criado em 2009 para auxiliar desenvolvedores de aplicações web e móveis a se preocuparem menos com a arquitetura da aplicação, sistemas de autenticação e hospedagem para dar ênfase ao produto final e gerar uma aplicação de qualidade rapidamente (FIREBASE, 2009a). Como serviçoes oferecidos gratuitamente aos usuários cadastrados encontram-se a hospedagem gratuita de conteúdo estático (p. ex. páginas HTML e arquivos JS), sistema de autenticação (anônimo, por e-mail ou por redes sociais (Facebook, Google+, Twitter, GitHub)) e armazenamento de dados em formato JSON em um banco de dados não-relacional.

No começo de 2014 o Firebase anunciou o suporte ao AngularJS, oferecendo um módulo (*AngularFire*) para o framework com alguns componentes e serviços que permitiriam integrar aplicações web ao Back-End oferecido. Deste modo as funcionalidades de tempo-real oferecidas pelo Firebase (sincronia de dados por meio de websockets (MOZILLA, 2015c)) poderiam ser incorporadas aos ciclos de processamento de dados do AngularJS de modo transparente ao desenvolvedor. No final de 2014 a equipe do Firebase foi integrada à Google.

Para este trabalho, o Firebase serviu como um *Back-end As A Service* (BAASBOX, 2015). Isso garante que seja escrito muito pouco ou quase nada de código no lado do servidor pois o serviço já faz a gerência de sistemas que esta aplicação utiliza, como a autenticação por e-mail, hospedagem de conteúdo estático e armazenamento de dados em forma de documentos JSON.

Figura 2.2: Exemplo de uma aplicação utilizando os estilos puros do Bootstrap



Fonte: http://www.opinionatedprogrammer.com/2011/11/twitter-bootstrap-on-rails/

2.1.6 Bootstrap

Bootstrap é um framework de desenvolvimento web Front-End criado no Twitter em 2010 que oferece uma série de componentes e comportamentos desenvolvidos com JQuery (BOOTSTRAP, 2015). Seu foco é *Mobile First* e sendo assim, o framework oferece diversas classes CSS para manipular o conteúdo estático da página de modo que esta se adapte a qualquer dispositivo de navegação, seja o celular, tablet ou desktop.

Este framework contribuiu para o desenvolvimento da aplicação por oferecer uma série de novas diretivas HTML para diversos componentes como barras de navegação ou seletores de datas e também oferecer aparência pré-fabricadas para basicamente todas as diretivas padrão do HTML.

2.1.7 Controle de versionamento

O controle de versionamento é uma prática muito comumente empregada por desenvolvedores de software ou designers para manter um histórico de manipulação de arquivos em um projeto (GIT, 2015). São utilizadas nessa prática sistemas de controle de versão (*VCS*) que mantém um histórico detalhado de cada atualização informando o autor, a data e o que foi modificado em relação à versão anterior. Utilizar um VCS é uma boa prática de backup caso seja necessário identificar quando um problema foi introduzido no

sistema ou arquivos tenham sido apagados por acidente.

Para este trabalho, foi utilizado o sistema de versionamento GIT e o trabalho está disponível no BitBucket (URL: <https://bitbucket.org/27raphalupi/game-finder>), um serviço da Atlassian para armazenamento de controle de versões privado e gratuito.

2.2 Aplicações e Trabalhos Relacionados

Nesta seção serão apresentados alguns programas e aplicativos que oferecem aos seus usuários diferentes maneiras de organizar eventos esportivos. Será feita uma breve descrição de cada aplicação além de mostrar o que é oferecido pelo sistema sendo avaliado e como isso ajuda o usuário a organizar seus eventos esportivos.

2.2.1 Facebook

Fundado em 2004, o Facebook é atualmente maior rede social do mundo, contando com cerca de 1,5 bilhão de usuários mensalmente ativos (incluir a fonte). A aplicação oferece em suas versões Web e Mobile suporte a sistemas de grupos e eventos.

Grupos são aglomerações de usuários no sistema. Com um grupo é possível reunir diversos conhecidos e amigos para organizar as atividades esportivas. Seu gerenciamento fica a cargo dos administradores do grupo. O sistema de grupos engloba as seguintes funcionalidades:

- **Controle de Privacidade**: Um grupo está sujeito a critérios de visibilidade na rede, limitando os usuários que podem visualizar o conteúdo ou participar de suas atividades.
- **Manutenção e Listagem de Participantes**: Membros de um grupo são capazes de listar os participantes, convidar novas pessoas para o grupo. Administradores ainda são capazes de remover membros de um grupo.
- **Gerência de Eventos**: É possível acompanhar os eventos que o grupo promove. Usuários que acessam eventos de um grupo específico só podem confirmar sua participação se fizerem parte do grupo anfitrião.

Evento é uma maneira de de juntar membros da aplicação para realizar uma determinada atividade. Um evento pode ser público na rede ou restrito a membros de um

Figura 2.3: Página de um evento no Facebook

Fonte: Facebook - Novembro de 2015

grupo. O sistema de eventos conta com as seguintes funcionalidades:

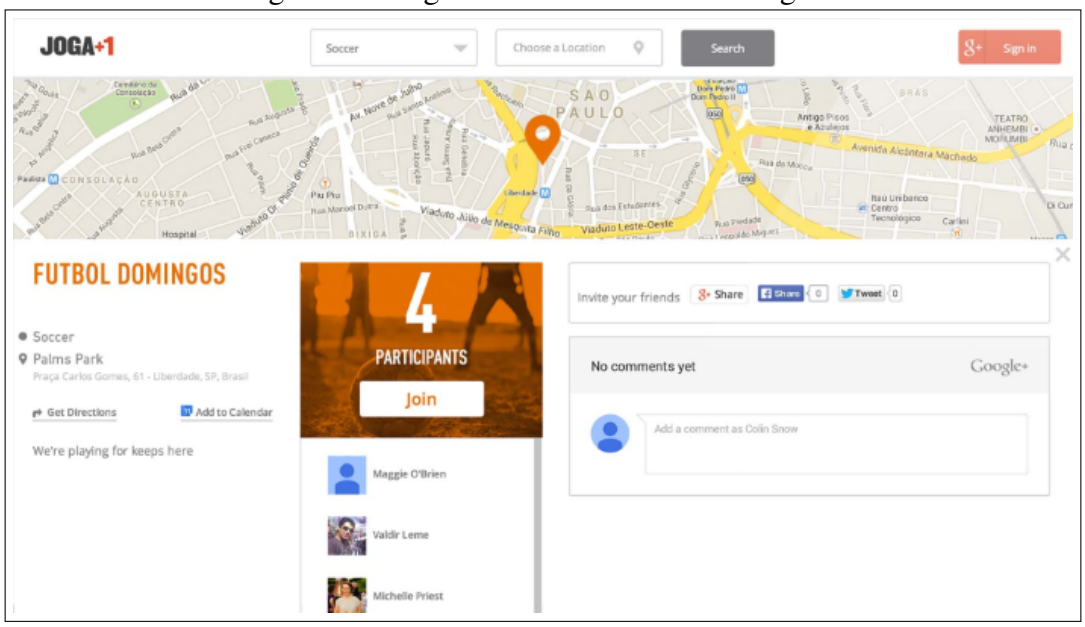
- **Convidar Pessoas**: Durante a criação de um evento é possível convidar pessoas de dentro dos círculos de amizade do usuário ou membros de um grupo que você faça parte.
- **Descrever a Atividade**: O usuário categorizado como anfitrião de um evento pode informar diversas características do evento, como sua localidade, data, hora e previsão do tempo. Demais tipos de informação só são possíveis de descrever em um campo de texto livre.

O Facebook não oferece ferramentas que tematizem os eventos como atividades esportivas. Funcionalidades importantes como controlar a ordem de confirmação de participantes, definir o preço do evento ou qual esporte será praticado não se encontram presentes, como mostra a figura 2.3. Um grupo que deseja gerenciar as atividades de seu grupo esportivo precisa lançar mão de outras aplicações que cubram estas lacunas.

2.2.2 Joga +1

O Joga +1 é uma aplicação web criada pela Google em 2014 em parceria com as ONGs Atletas pelo Brasil, Fundação Gol de Letra, Cufa, Instituto Bola para Frente e Liga Solidária e com o site Catraca Livre (Incluir fonte). Seu objetivo era incentivar a prática de esporte no Brasil, permitindo que usuários participantes da rede Goo-

Figura 2.4: Página de uma atividade no Joga +1



Fonte: Use All Five (<useallfive.com>) - Novembro de 2015

gle+ pudessem acessar e buscar por atividades de um esporte específico próximo de uma localidade. O Joga +1 foi descontinuado no ano de 2015 e se encontrava no domínio <https://jogamaisum.withgoogle.com>.

A aplicação, ainda que minimalista, focava com sucesso na realização da prática esportiva. O usuário não podia criar um grupo nem adicionar outros membros como amigos (deixando essas funcionalidades a cargo do Google+), mas podia facilmente criar uma atividade em um endereço específico e esta ficava disponível em um mapa para que outros usuários pudessem se unir ao participantes e aproveitarem a atividade.

Assim como no Facebook, certas funcionalidades importantes ficaram de fora, como: Gerenciar um grupo, controlar o número de participantes da atividade, informar o preço da atividade e controlar e gerenciar pagamentos dos participantes. A figura 2.4 mostra a interface da aplicação, onde era possível buscar por atividades por esporte e localização, e então o usuário confirmava ou não sua presença.

2.2.3 Peladeiro.com

O *Peladeiro.com*, criado em 2000, é atualmente a aplicação voltada a criação e organização de grupos esportivos com mais usuários no Brasil. Contando com cerca de 50 mil grupos e 800 mil usuários cadastrados em diversos países como Estados Unidos, Portugal, Espanha e Itália, a aplicação segue ganhando força no nicho da organização de

Figura 2.5: Página de uma grupo no Peladeiro.com

Fonte: Peladeiro.com - Novembro de 2015

partidas de futebol amadoras e semi-profissionais, batizadas de *peladas*.

A aplicação entra no quadro das ferramentas que auxiliam praticantes de esportes a organizarem suas atividades com mais comodidade possuindo basicamente todos os itens importantes para um organização pré e pós jogo, indo desde a organização de grupos com amigos até o controle detalhado de pagamento dos participantes de uma partida.

Como funcionalidades importantes da aplicação, tanto no sistema de grupos quanto no sistema de eventos, podemos citar:

- **Perfis e Sistema de Amigos**: Na aplicação é possível dar uma descrição detalhada dos próprios dados, incluindo a localização do usuário e telefones para contato. Ainda é possível ser
- **Gerenciamento de Eventos**: O usuário que confirma ou não a sua presença em um evento é alocado em uma listagem específica dentro da página do evento, informando se este participará ou não da partida. Usuários podem ser categorizados nos grupos para usufruírem de ordens de preferência para as vagas disponíveis.
- **Controle de Pagamentos**: Também presente no sistema se encontra um sistema de gerenciamento de pagamentos das partidas. O administrados de uma partida pode determinar o preço por jogador e liberar que cada usuário que deseje pagar a sua parte com cartão de crédito o faça.

2.2.4 Aplicações em dispositivos móveis

Foram encontradas também algumas aplicações para dispositivos Android que realizam ou auxiliam na organização de eventos esportivos. Em sua grande maioria, estas aplicações são tematizadas com futebol, possuindo em seus nomes referência ao termo *pelada* (e sem possuir vínculo algum com o Peladeiro.com).

Os aplicativos e uma breve descrição de suas funcionalidades se encontram a seguir:

- **Peladeiro App**: Oferece um controle limitado do gerenciamento dos grupos, permitindo que o usuário possa somente controlar os participantes dos times, tempo de partida e controle momentâneo do pagamento dos participantes. O controle das informações é centralizado, dando somente ao usuário com o aplicativo instalado o controle das informações.
- **Peladeiros**: Oferece um controle completo do grupo de modo centralizado (todas as informações do grupo ficam salvas em um celular somente), oferecendo sistemas de criação de grupo, controle de tempo de partida e pagamentos, organização do times, rankings e estatísticas.
- **Futeba**: Faz o simples controle da distribuição dos jogadores em times. Também tematizado para o Futebol.
- **Minha Pelada**: Organiza os jogadores em grupos de modo centralizado, oferecendo um sistema para a distribuição dos times e controle de tempo das partidas.

2.2.5 Outras aplicações

Existem também outras aplicações menos expressivas no Brasil para o controle dos eventos esportivos, possivelmente por se tratarem de ferramentas estrangeiras. Uma listagem das mais populares e breve descrição de suas funcionalidades se encontram a seguir:

- **Do League**: Oferece sistemas de grupos e eventos para os usuários onde o controle e gerenciamento dos grupos é feito distribuidamente, ou seja, todos usuários possuem uma conta particular e podem participar de diversos grupos e eventos. É possível também fazer amigos na aplicação. O Do League é focado para a organização de competições, oferecendo ferramentas para montar tabelas de campeonatos

com múltiplos times.

- **Timpik**: Trata-se de uma aplicação muito semelhante ao Peladeiro.com em relação às funcionalidades, mas com temática multi esportiva já integrada, onde é possível marcar eventos esportivos de outros esportes que não seja futebol, por exemplo. A aplicação também conta com sistema integrado de pagamentos onde o usuário pode previamente quitar suas dívidas com o grupo com pagamentos online.

2.3 Comparativo de Funcionalidades

Extraindo desta listagem de aplicações algumas funcionalidades chave, é possível gerar um quadro comparativo de quais aplicações possuem determinados sistemas. Na tabela 2.1 é mostrado um comparativo por aplicação de cada funcionalidade.

As funcionalidades foram categorizadas em quatro grupos:

- *Distribuição*: Agrupa funcionalidades referentes ao modo como a aplicação é distribuída na rede, seja possuindo versões gratuitas ou pagas ou possuindo versões web ou mobile.
- *Social*: Agrupa funcionalidades de socialização entre os usuários, como poder fazer login na sua conta e ter um perfil, definir esportes favoritos, possuir amigos ou montar grupos, etc.
- *Gerência*: Agrupa funcionalidades de gerência dos eventos esportivos, como por exemplo controlar o pagamento dos participantes, montar times e controlar o tempo da partida.
- *Esportes*: Agrupa informações sobre os tipos de esportes que a aplicação dá suporte, direta ou indiretamente, baseado na sistema de classificação de esportes em função de cooperação e oposição (GONZALEZ, 2004).

Tabela 2.1: Comparativo de funcionalidades das aplicações semelhantes

	Funcionalidades / Aplicativo	*Facebook*	*Joga +1*	*Peladeiro.com*	*Peladeiro App*	*Peladeiros*	*Futeba*	*Minha Pelada*	*Do League*	*Timpik*
Distrib.	Versão gratuita	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Versão paga			✓	✓	✓			✓	
	Versão web	✓	✓	✓					✓	✓
	Versão móvel	✓	✓	✓	✓	✓	✓	✓	✓	✓
Social	Login com redes sociais	-	✓	✓			✓			✓
	Criação de perfil	✓	✓	✓		✓	✓	✓	✓	✓
	Esportes favoritos		✓							✓
	Sistema de amigos	✓		✓					✓	✓
	Grupos/Times	✓		✓	✓	✓	✓	✓	✓	✓
	Trocar mensagens	✓		✓					✓	✓
Gerência	Criar Partidas	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ordenar participantes			✓	✓	✓		✓	✓	
	Distribuição de times			✓	✓	✓	✓			
	Controle de pagamento			✓	✓	✓				✓
	Tempo da partida				✓	✓		✓		
Esportes	Esportes com interação	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Esportes sem interação	✓	✓						✓	✓
	Esportes coletivos	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Esportes individuais	✓	✓						✓	✓

3 PROJETO DA APLICAÇÃO

Neste capítulo será feita uma apresentação completa do projeto deste trabalho. Serão apresentadas listas das funcionalidades desejadas para a aplicação além de diagramas que mostram detalhadamente sua arquitetura. Adicionalmente será feita uma explicação mais detalhada das tecnologias que foram utilizadas neste trabalho, como o AngularJS e o Firebase, explicando detalhes internos de suas arquiteturas e como isso ajuda ou não no desenvolvimento desta aplicação.

3.1 User Stories

Baseado em uma avaliação de o que as demais aplicações do mercado ofereciam e num estudo das necessidades do autor deste trabalho foi gerada uma lista de funcionalidades que a aplicação deveria possuir para conseguir um *time-to-market* baixo. Dividiremos aqui as funcionalidades em cinco grupos lógicos de acordo com o escopo onde cada uma se aplica.

Os escopos são os níveis de *usuário*, *grupo*, *evento*, *dashboard* e *busca*. A descrição de cada funcionalidade é apresentada como uma estória de usuário (*user story* ou *US*). Neste formato os requisitos possuem uma estrutura tal qual "Como um <ator>, quero <algo> para que <benefício>.", onde a terceira oração é facultativa segundo (COHN, 2008; COHN, 2014; AMBLER, 2009).

3.1.1 Usuário

As estórias de usuário focam nas capacidades sociais do sistema, onde é possível fazer um registro na aplicação, manter um perfil público, editar suas informações, visualizar outros membros da rede e configurar as preferências esportivas.

- *Como um usuário não cadastrado, quero poder me cadastrar no sistema*: Envolve fazer um registro na aplicação utilizando o sistema de autenticação oferecido pelo Firebase.
- *Como um usuário cadastrado, quero poder acessar o sistema*: Utilizar o sistema de autenticação do Firebase para autenticar o usuário com e-mail e senha.
- *Como um usuário cadastrado, quero poder fazer logout do sistema*

- *Como um usuário, quero que minhas informações pessoais sejam mostradas em meu perfil público*
- *Como um usuário, quero poder editar minhas informações pessoais*
- *Como um usuário, quero poder acessar o perfil de outro usuário*
- *Como um usuário, quero poder editar meus esportes favoritos*

3.1.2 Grupo

As estórias de grupo são definidor por dois tipos diferentes de atores. O primeiro tipo, o usuário, deverá ser capaz de criar um novo grupo na aplicação, requisitar acesso a um grupo existente ou visualizar seu conteúdo e membros. O segundo tipo, o administrador, é responsável pela gerência e organização do grupo. No momento em que um usuário cria um grupo, este se torna administrador do mesmo. É possível também que um administrador promova outros membros do grupo ao *status* de administrador ou aceitar/rejeitar que outros membros da rede participem.

- *Como um usuário, quero poder criar um grupo*
- *Como um usuário, quero poder solicitar participação em um grupo privado*
- *Como um administrador de grupo, quero poder editar as informações do grupo*
- *Como um administrador de grupo, quero poder aceitar/rejeitar a solicitação participação de um usuário*: Usuários que acessarem a página de um evento podem solicitar participação para o administrador do mesmo, ficando pendente uma confirmação do administrador se o usuário ingressará ou não.
- *Como um administrador de grupo, quero poder expulsar um membro*
- *Como um membro de grupo, quero poder visualizar os membros do grupo*
- *Como um membro de grupo, quero poder visualizar a lista de eventos deste grupo*
- *Como um administrador de grupo, quero poder excluir o grupo*

3.1.3 Evento

Para que a aplicação consiga ajudar membros e administradores de grupos a organizar eventos esportivos é necessário que uma série de informações sejam dadas ao sistema e que estas fiquem disponíveis de forma clara. Tais informações como o esporte

que será praticado, data e local do evento, preço por participante, etc. são fundamentais neste ponto.

Ao tratarmos eventos como atividades exclusivas de um grupo, somente usuário administradores de algum grupo dentro da aplicação podem criar novos eventos para estes grupos. Adicionalmente é importante que os participantes vejam quem, dentre os membros do grupo que está organizando o evento, participarão da atividade, além de ver se há vagas sobrando ou se sua confirmação no evento garantirá uma vaga. Um usuário do sistema só pode acessar um evento se este fizer parte do grupo que o hospeda. Adicionalmente, eventos não possuem recorrência, sendo necessário recriar uma atividade sempre que necessário.

- *Como administrador de um grupo, quero poder criar um evento para este grupo*: O usuário só pode criar novos eventos se fizer parte de um grupo na condição de administrador. Caso não seja satisfeita esta condição, o usuário deve ser enviado para outra página da aplicação e ser informado que não pode criar um evento.
- *Como um membro de grupo, quero poder (des)confirmar participação nos eventos do grupo*
- *Como um administrador de um grupo, quero poder editar as informações dos eventos*
- *Como um administrador de um grupo, quero poder cancelar eventos*
- *Como um membro de grupo, quero poder ver as informações o evento*
- *Como um membro de grupo, quero poder ver os participantes do evento*

3.1.4 Dashboard

O Dashboard (ou *painel*) da aplicação foi pensado como uma área onde informações relevantes seriam mostradas para os usuários, servindo como a página inicial da aplicação. Assim como será apresentado na subseção 5.2.2, o painel foi dividido em três áreas para agrupar melhor a informação.

Na seção principal (ou corpo da aplicação) é esperado que fiquem informações críticas aos usuários, como os eventos mais próximos ou pessoas que estão esperando para fazer parte de algum grupo que o usuário administra. No menu lateral de navegação, mostrar a lista de grupos que o usuário faz parte e a lista dos eventos mais próximos, como em um calendário.

- *Como um usuário, quero que os eventos futuros dos meus grupos sejam mostrados ordenados pela data*
- *Como um usuário, quero poder visualizar os grupos que faço parte*
- *Como um administrador de um grupo, quero ver quantos membros estão esperando por aprovação neste grupo*

3.1.5 Busca

O sistema de busca, ainda que projetado de modo simples para esta versão da aplicação (sem opções de busca avançada como a busca de eventos por proximidade geográfica ou com filtros mais complexos), deverá fornecer meios para que um usuário seja capaz de encontrar outros usuários do sistema além dos grupos cadastrados, para que o usuário possa no futuro se integrar a estes para a realização das atividades esportivas.

- *Como um usuário, quero poder buscar por outros usuários*
- *Como um usuário, quero poder buscar por outros grupos*

3.2 Entidades

Como descrito na seção 3.1, a aplicação foi projetada contendo alguns tipos de entidades definidas previamente em suas estórias. Podemos, a partir disso, gerar uma definição mais clara de o que cada entidade precisa ter, e qual o seu relacionamento no sistema com os demais componentes.

A figura 3.1 explicita o modelo de entidades e relacionamentos projetados para este sistema. Embora a aplicação esteja sendo projetada para um banco de dados não-relacional, um diagrama de entidade e relacionamento pode não ser o mais adequado, mas certamente colabora para um entendimento global da estruturação do projeto. Surge então a necessidade de reestruturar o diagrama afim de se adequar melhor ao modelo *NoSQL*. A seguir serão explicados detalhes de cada componente do diagrama Entidade-Relacionamento e sua função no projeto.

Primeiramente, definiremos que uma relação entre a entidade de autenticação e o usuário do sistema possuem um relacionamento *1 para 1*, chamado de registro. Adicionalmente um usuário pode ter diversos esportes na sua lista de favoritos, então definiremos uma entidade *Esporte* que abriga as informações das atividades. Um usuário possui um

relacionamento *N para N* com os esportes, pois um usuário pratica vários esportes e um esporte é praticado por vários usuários.

Um usuário do sistema e um grupo possuem um relacionamento *N para N*, pois um usuário pode fazer parte de diversos grupos e cada grupo pode ter diversos usuários. É neste relacionamento que se informa qual o tipo de acesso que o usuário tem neste grupo, podendo ser um membro e até mesmo um administrador.

Finalmente temos os eventos, que descrevem as atividades que serão executadas por diversos membros pertencentes a um grupo. Eventos possuem um relacionamento *N para N* com os usuários e relacionamentos *N para 1* com grupos (pois um evento só pode ter um grupo que o hospede) e *N para 1* com eventos, pois uma atividade esportiva só abriga um evento.

3.2.1 Autenticação

Esta entidade tem o papel de armazenar informações chave sobre como o usuário fez a autenticação no sistema e qual usuário fez a autenticação. A entidade *Autenticação* possui um relacionamento *1 para 1* com a entidade *Usuário* e comporta os seguintes atributos:

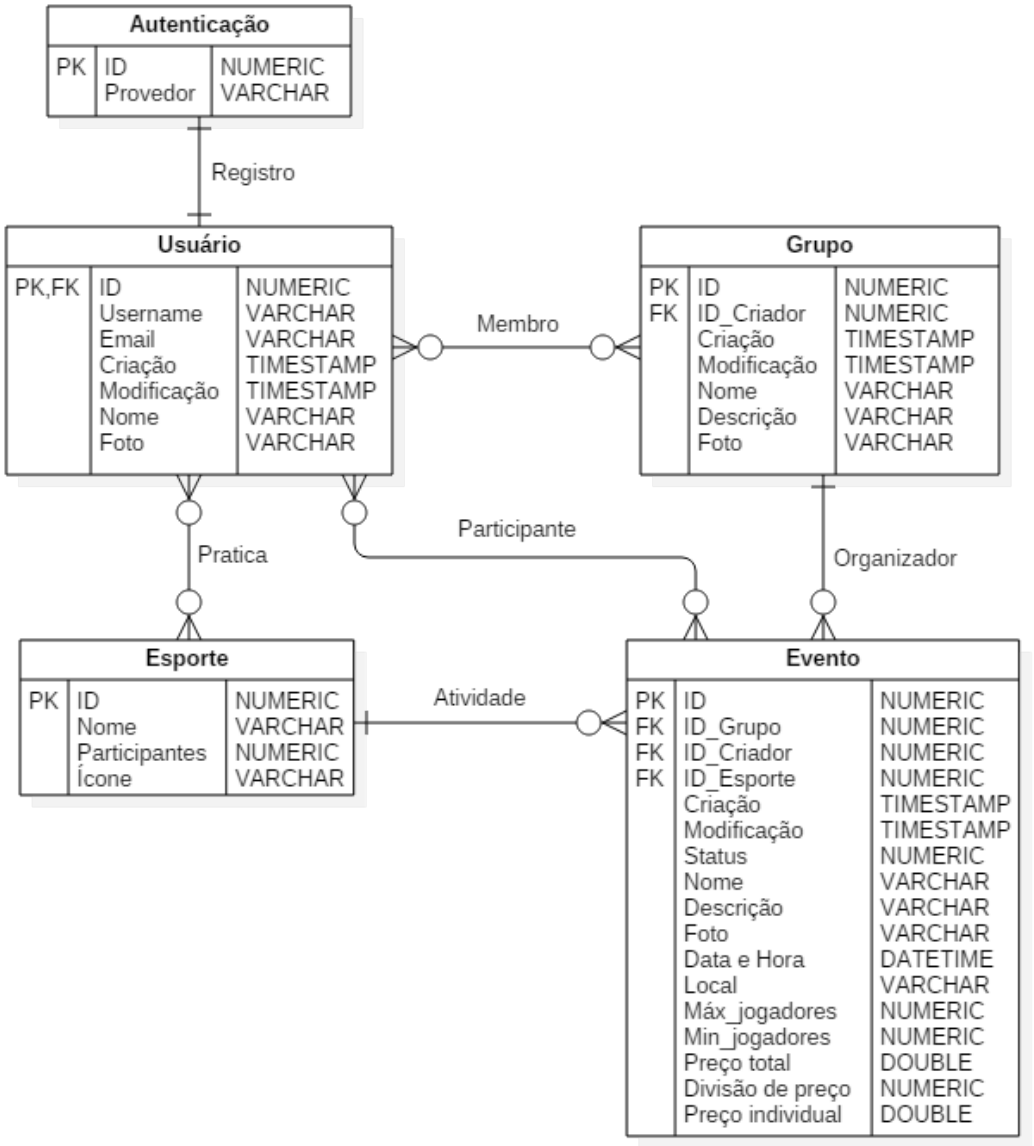
- **ID**: Identificador único de usuário no sistema de autenticação.
- **Provedor**: Qual método foi utilizado para realizar a autenticação (e-mail ou alguma rede social).

3.2.2 Usuário

Esta entidade guarda todas as informações de um usuário do sistema. São utilizadas aqui informações que identificam o usuário unicamente, aproveitando o relacionamento *1 para 1* com a entidade *Autenticação* e utilizando o seu identificador o ID vindo desta entidade. Esta entidade possui os seguintes atributos:

- **ID**: Identificador único de usuário no sistema, importado da entidade de autenticação.
- **Username**: Nome de usuário que será utilizado para montar as rotas de navegação da aplicação. O nome de usuário deve ser único no sistema.

Figura 3.1: Diagrama Entidade-Relacionamento do Game Finder



- **Email**: O e-mail que o usuário utiliza para fazer seu registro no sistema.
- **Criação**: A data em que o usuário criou ser perfil.
- **Modificação**: A data da última vez em que o usuário alterou o seu perfil.
- **Nome**: O nome do usuário
- **Foto**: Uma URL contendo a foto que o usuário quer usar como perfil.

Um usuário do sistema ainda possui relacionamentos *N para N* com outros grupo, outros eventos e outros esportes.

3.2.3 Esporte

A entidade *Esporte* guarda informações acerca das atividades que os usuários podem realizar dentro de um evento esportivo. Para fins de usabilidade, é importante que um esporte especifique o seu nome, o número mínimo de participantes da atividade e um ícone para auxílio visual. Um esporte não pode ser cadastrado por usuários do sistema, nesta etapa da aplicação, sendo possível somente que administradores do sistema os adicionem no banco de dados. Um esporte possui relacionamentos *N para N* com usuários e *1 para N* com outros eventos, pois um evento só realiza uma atividade esportiva por vez. A seguir a lista de atributos que a entidade *Esporte* comporta:

- **ID**: Identificador único de esporte.
- **Nome**: Nome do esporte
- **Participantes**: Número mínimo de participantes para realizar a atividade (pode ser alterado).
- **Ícone**: Diretório onde o ícone do esporte se encontra.

3.2.4 Grupo

A entidade *Grupo* guarda as informações sobre um grupo que realiza atividades esportivas. Grupos, dentro do Game Finder, são aglomerações de usuário que praticam atividades esportivas organizadas pelo sistema. Um grupo é comporto dos seguintes atributos:

- **ID**: Identificador único do grupo. Também utilizado para montar as rotas de navegação da aplicação.

- **ID Criador**: Identificador único do usuário que criou o grupo, referenciando um objeto da entidade *Usuário*.
- **Criação**: A data em que o grupo foi criado.
- **Modificação**: A última data em que o grupo foi modificado
- **Nome**: O nome do grupo
- **Descrição**: A descrição do grupo
- **Foto**: Uma URL contendo a foto que o grupo usa para se identificar.

Um grupo possui relacionamentos *N para N* com outros usuários, onde sua relação informa se um usuário é membro e/ou administrador do grupo, e *1 para N* com eventos, onde seu relacionamento informa se o evento está ativo para aquele grupo ou se foi cancelado.

3.2.5 Evento

A entidade dos eventos guarda as informações sobre as atividades esportivas que serão feitas entre os membros de um grupo. Uma atividade esportiva necessita de informações que auxiliem os membros do grupo a completarem a atividade. São necessárias diversas informações para que a organização do evento seja feita corretamente, de acordo com dados levantados pelo autor deste trabalho durante as organizações de seus eventos. A seguir uma lista dos atributos que caracterizam um evento:

- **ID**: Identificador único da atividade esportiva, necessário para gerar as rotas da aplicação que levam até ele.
- **ID Grupo**: Identificador único do grupo que está hospedando o evento.
- **ID Criador**: Identificador único do usuário do sistema que criou o evento.
- **ID Esporte**: Identificador único do esporte que será praticado no evento.
- **Criação**: A data em que o evento foi criado.
- **Modificação**: A última data em que o evento foi modificado.
- **Status**: Informa o estado atual do evento, estando ele ativo, expirado ou cancelado.
- **Nome**: O nome do evento.
- **Descrição**: A descrição do evento (se necessária).
- **Foto**: Uma URL contendo a foto do evento
- **Data e Hora**: A data e hora em que o evento será realizado.

- **Local**: O local onde o evento será realizado (textual ou um JSON contendo dados do Google Maps).
- **Max. jogadores**: Número mínimo de jogadores que podem participar da atividade.
- **Min. jogadores**: Número máximo de jogadores que podem participar da atividade.
- **Preço total**: Preço total do evento (preço da quadra esportiva, por exemplo).
- **Divisão de preço**: Como o valor será dividido entre os participantes (dividir o valor total igualmente ou cada membro paga um taxa específica).
- **Preço indiv.**: Preço que cada participante deverá pagar para o organizador do evento para que possa participar do evento.

3.3 Frameworks e Tecnologias

Para esta seção serão comentadas características das tecnologias utilizadas. AngularJS e Firebase são tecnologias relativamente novas e possuem curvas de aprendizado bem diferentes entre si, sendo necessário abordar diversos conceitos antes de utilizá-las. A seguir serão abordados tais conceitos e será feita uma explicação de quais práticas devem ser seguidas para uma correta utilização das tecnologias.

3.3.1 AngularJS

Como descrito na subseção 2.1.4, Angular JS é um framework MV* e neste trabalho será utilizado como um framework MVC. Deste modo é necessário definir por meio da listagem de entidades quais componentes devem ser desenvolvidos afim de tratar as regras de negócio corretamente do lado do usuário do sistema. Listaremos a seguir alguns conceitos e como o AngularJS aborda cada um deles.

3.3.1.1 Arquitetura em Camadas

AngularJS define como *Model* todo objeto em memória que é passível de interação do usuário e é apresentado na *View* (ANGULARJS, 2010b). A partir desta definição, toda a manipulação em memória de objetos, arrays e variáveis da aplicação podem ser tratados como manipulações no modelo.

A *View* é aquilo que o usuário vê e interage com. Esta camada é definida na

aplicação como o HTML estendido por uma série de diretivas e expressões que recebem um tratamento de *parsing* do AngularJS no período de inicialização.

Os *Controllers* (ANGULARJS, 2010c) são componentes opcionais que fazem a ligação dos dados vindos do e para o *Model* para e pela *View*. São os controladores da aplicação os responsáveis por uma melhor definição do escopo de variáveis e por auxiliarem na elaboração de funções complexas para a manipulação e tratamento de dados do sistema. É importante frisar que controladores são dependentes da *View* em que estão ligados, não podendo ser reutilizados no sistema.

Como uma estratégia de facilitar o reuso de porções de código que manipulem dados afim de cobrir regras de negócio do sistema, AngularJS oferece também o conceito de Serviços (ANGULARJS, 2010e), os tratando como componentes que podem ser injetados em outras partes da aplicação. Um serviço tem a finalidade de, sendo instanciado uma única vez durante a execução do programa (como um *Singleton*), garantir que um conjunto específico de variáveis e métodos possam ser reutilizados por todas as partes do código.

3.3.1.2 Organização de Arquivos

AngularJS possui, entre seus desenvolvedores, dois modos principais de organização de projetos: Organização por componente ou Organização por funcionalidade. O primeiro modo descreve que os diretórios do projeto devem ser estruturados para guardar todos os componentes semelhantes na mesma pasta, deixando por exemplo todos os controladores na pasta *Controladores* e todas as diretivas na pasta *Diretivas*. O segundo modo descreve que funcionalidades semelhantes devem ser armazenadas na mesma pasta, nomeadas pelo nome da funcionalidade (KUKIC, 2014).

Para este trabalho foi feita uma abordagem mista, de modo que fosse possível explicar vantagens dos dois modelos. Assim sendo, existem diretórios que abrigam as funcionalidades específicas de uma entidade e há diretórios para armazenar uma coleção de serviços do sistema, dadas suas capilaridades na aplicação.

3.3.1.3 Ligação de Dados

Uma das marcas deste framework é a capacidade de agir quase que instantaneamente na manipulação da *View* quando um *Model* é modificado, e vice-versa. O conceito atrelado à esta funcionalidade é a *Ligação de Dados Bidirecional* (ou *Two-Way Data Bin-*

ding) (ANGULARJS, 2010d) e seu funcionamento se dá pela capacidade do AngularJS realizar uma compilação do HTML estendido (o *template*) e adicionar comportamentos próprios nos elementos do DOM. Assim, quando um componente específico é manipulado, disparam-se eventos que capturam as mudanças e se atualizam os modelos em memória. De forma análoga, uma mudando nos dados do modelo impactam em mudanças quase que instantâneas na tela.

3.3.2 Firebase

A seção 2.1.5 comenta sobre alguns serviços oferecidos pelo Firebase ainda em sua versão gratuita. Entre elas estão o sistema de autenticação, o armazenamento de arquivos estáticos e o banco de dados não-relacional (FIREBASE, 2009b). A seguir serão explicados com mais detalhes cada uma destas funcionalidades. O banco de dados será explicado em uma subseção a parte (3.4).

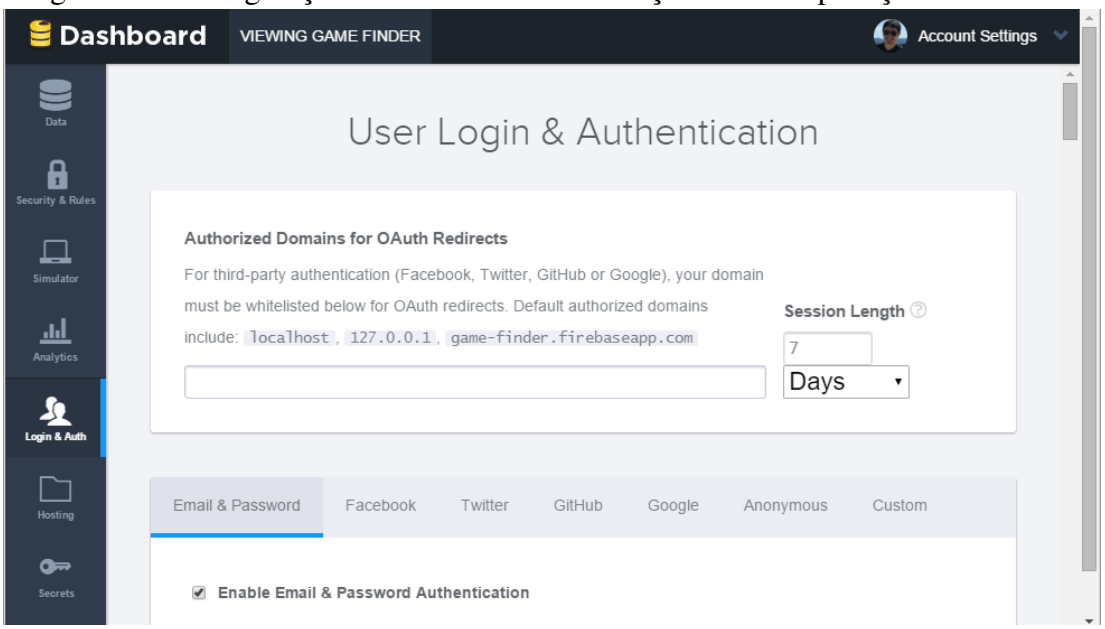
3.3.2.1 Autenticação

Utilizando o módulo *AngularFire*, disponível para aplicações AngularJS que desejam se conectar aos serviços do Firebase com maior facilidade, é possível utilizar uma série de métodos que facilitam o processo de autenticação de um usuário na aplicação. Primeiro o administrador da aplicação deve informar no painel de configurações quais são os métodos de autenticação que a aplicação dará suporte, como por exemplo e-mail ou Facebook. A figura 3.2 mostra a interface que o usuário tem na hora de configurar este sistema, listando em abas cada provedor de autenticação que o Firebase da suporte.

3.3.2.2 Hospedagem

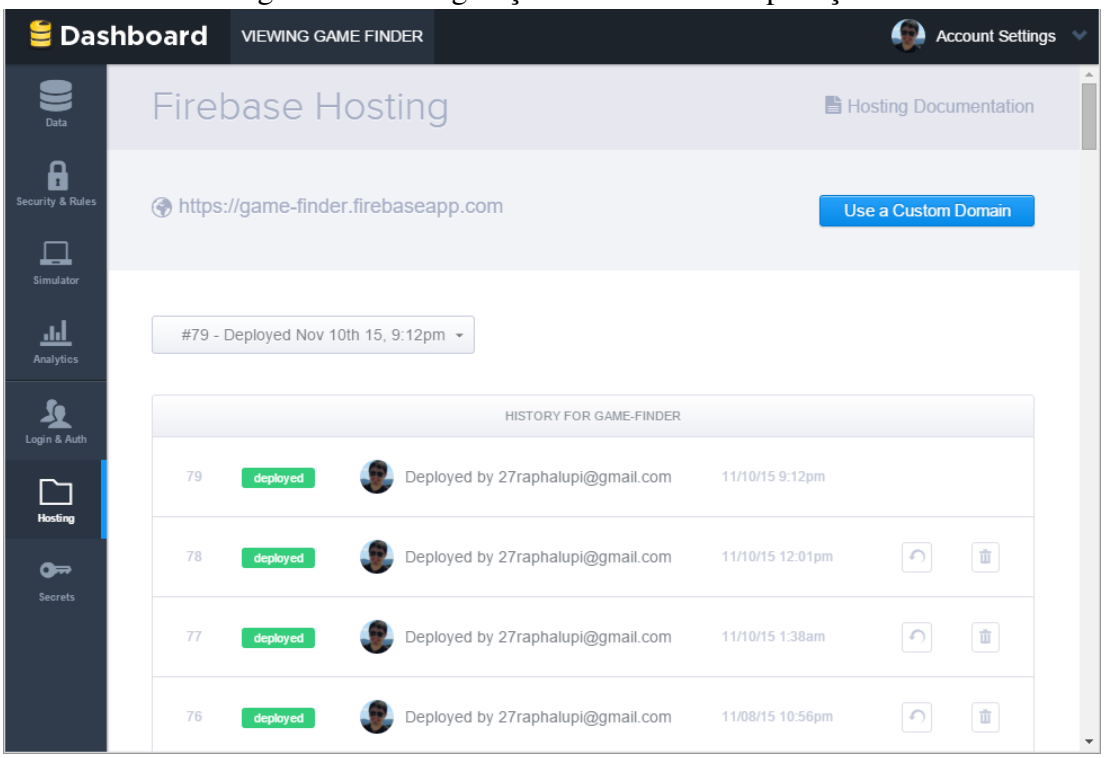
Após uma configuração do kit de ferramentas do Firebase no computador, pelo terminal, o usuário pode fazer o envio de uma versão da aplicação para o servidor (*deploy*). As versões geradas pelo sistema podem ser manipuladas, permitindo que o usuário faça rollbacks de forma simples tanto pela interface quanto pela linha de comando. A figura 3.3 mostra a interface que a aplicação oferece para que o usuário consiga controlar seu versionamento.

Figura 3.2: Configuração do sistema de autenticação de uma aplicação no Firebase



Fonte: Painel de configuração do Firebase

Figura 3.3: Configuração das versões da aplicação



Fonte: Painel de configuração do Firebase

3.4 Banco de Dados

Ao utilizar o serviço de banco de dados em tempo real que o Firebase oferece, o desenvolvedor do projeto precisa entender um pouco mais sobre bancos de dados não-relacionais e a estruturação de dados orientada a documentos. (MONGODB, 2015) fala sobre três grandes mudanças de paradigma que ocorreram nos últimos anos desde o surgimento dos bancos de dados SQL: A emergência pela computação na nuvem - onde dados necessitam ser replicados em múltiplos servidores e a realização de *JOINS* entre tabelas de diferentes fontes é um desafio enorme -, a necessidade de utilizar dados não-estruturados aumenta constantemente e o contante uso de metodologias ágeis no desenvolvimento de projetos acarreta em mudanças de design conforme a demanda evolui - baseando-se no fato de que operações de alteração de estrutura do banco (*ALTER TABLES*) são muito custosas.

Afim de remediar estas dificuldades, sistemas alternativos de armazenamento de dados surgiram para tentar estruturar a informação de um sistema de forma mais lógica e dinâmica. Entre os meios mais populares (DB-ENGINES, 2014) se encontra os sistemas *NoSQL*. O termo infere que nem todas as manipulações nos dados ocorrem de forma padronizada, dando liberdade para que diferentes estruturas de armazenamento sejam criadas dentro do mesmo conjunto semântico de dados e que a leitura possa ser feita de modos diferentes, dependendo da situação (COUCHBASE, 2011).

Para este trabalho em específico, a vantagem oferecida por um sistema NoSQL como o Firebase foi cobrir as constantes mudanças e experimentações feitas na modelagem dos dados afim de tentar cobrir da melhor forma possível os critérios *ACID* (Atomicidade, Consistência, Isolamento e Durabilidade). Serão levantados no capítulo 4 diversas questões onde o Firebase auxilia e atrapalha o desenvolvimento, em especial por ser uma tecnologia nova e em constante mudança.

É necessário neste ponto entender que a estrutura do banco de dados em um sistema não-relacional é ditada pelas consultas que o desenvolvedor deseja realizar. Deste modo, consultas como os membros de um grupo ou os grupos de um membro requerem ou um replicação total dos dados (e consequente gerenciamento de modificações para manter consistência) ou incluir apontadores para outros objetos do documento afim de realizar uma série de consultas recursivas até a completa obtenção dos dados desejados.

O sistema em tempo-real oferecido pelo Firebase funciona a partir de uma série de *listeners* por porções específicas de uma coleção ou documento no banco. É pos-

Figura 3.4: Configuração de um *listener* em um documento no Firebase

```
documentReference.on('child_added', function(dataSnapshot, previousChildKey) {
    // code to handle new child
})
```

Fonte: Documentação Firebase (On) - <https://www.firebase.com/docs/web/api/query/on.html>

sível neste modelo definir que a aplicação "gostaria de ser informada"toda vez que um evento específico aconteça dentro do diretório dos dados, como por exemplo a inclusão ou remoção de um dado, a alteração do conteúdo do documento, etc. Na figura 3.4 o `documentReference` é um objeto Firebase que aponta para um documento específico, `on` é o método que cria um *listener* para este documento, `'child_added'` é o tipo de evento que será vinculado, `dataSnapshot` é o estado do objeto no documento no momento em que foi capturado e `previsouChildKey` é chave para o último objeto adicionado no documento, se existir.

4 DESENVOLVIMENTO DA APLICAÇÃO

O capítulo 4 desenvolve o processo de criação da aplicação, comentando decisões de projeto e as etapas de desenvolvimento do Game Finder. Os tópicos serão abordados em ordem semelhante a como foram desenvolvidos, dando prioridade para cobrir a lista estórias de usuário apresentadas na seção 3.1.

Novamente o autor deste trabalho frisa que esta aplicação tem o intuito de ser produzida sem a necessidade de configurar um servidor Back-End e para tanto as arquiteturas a seguir apresentadas se focam tão somente no lado do cliente.

4.1 Configuração e Criação do Projeto

Um aplicação em AngularJS requer um conjunto básico de bibliotecas para seu correto funcionamento. Entre elas, podemos destacar a biblioteca do AngularJS, as bibliotecas e estilos CSS do Bootstrap, bibliotecas do Firebase (AngularFire), JQuery e outras bibliotecas e módulos integrados ao AngularJS para uma melhor experiência do usuário e facilidade de desenvolvimento.

É comum em muitos projetos neste e outros frameworks JS modernos a utilização de ferramentas de controle de pacotes, ou *Package Managers*. Em especial, dois grandes softwares fazem o controle de pacotes para projetos executados com Node.js, sendo eles o *NPM* (NPM, 2009) e o *Bower* (BOWER, 2012). Estes softwares fazem uso de um arquivo JSON que descreva as dependências de bibliotecas e módulos que a aplicação possui para ser executada corretamente em qualquer servidor (NGLEARN). A figura 4.1 mostra a configuração do arquivo `package.json` que é utilizado para fazer o setup de dependências da aplicação.

A figura 4.2 mostra como é definido o módulo da aplicação e suas dependências. A variável `app` contém o módulo AngularJS do Game Finder e todos os componentes vinculados à esta aplicação (Controladores, Serviços, etc.) são gerados a partir dela.

Node.js não foi utilizado nesta aplicação com o intuito de fornecer o Back-End do sistema (como é comum em projetos JS), mas foram utilizadas algumas funcionalidades para uma execução local da aplicação durante seu desenvolvimento, como por exemplo a instanciação de um servidor local rodando em Node.js para servir arquivos estáticos.

Figura 4.1: Arquivo de configuração de dependência do projeto - `package.json`

```
{
  "name": "game-finder",
  "description": "A sports manager made with AngularJS",
  "version": "1.0.0",
  "repository": "https://bitbucket.org/27raphalupi/game-finder",
  "license": "MIT",
  "private": true,
  "devDependencies": {
    "http-server": "^0.6.1",
    "bower": "^1.3.1",
    "shelljs": "^0.2.6",

    "angular": "1.4.4",
    "angularfire": "1.1.2",
    "firebase": "2.2.1",
    "jquery": "2.1.4",
    "bootstrap": "3.3.5",
    "angular-ui-router": "0.2.15",
    "angular-auto-validate": "1.18.14",
    "angular-ladda": "0.3.1"
  },
  "scripts": {
    "prestart": "npm install",
    "start": "http-server -a localhost -p 8000 -c-1",

    "pretest": "npm install",
    "test": "karma start karma.conf.js",
    "test-single-run": "karma start karma.conf.js  --single-run",

    "preupdate-webdriver": "npm install",
    "update-webdriver": "webdriver-manager update",

    "preprotractor": "npm run update-webdriver",
    "protractor": "protractor e2e-tests/protractor.conf.js"
  }
}
```

Figura 4.2: Definição do módulo central do Game Finder

```
var app = angular.module("gamefinderApp", [
    "firebase",
    "ui.router",
    "ui.bootstrap",
    "jcs-autoValidate",
    "angular-ladda",
    "ngTagsInput"
]);
```

Figura 4.3: Factory *Auth* para acesso ao servidor de autenticação do Firebase

```
app.factory("Auth", ["$firebaseAuth", function ($firebaseAuth) {
    var ref = new Firebase("https://game-finder.firebaseio.com/");
    return $firebaseAuth(ref);
}]);
```

4.2 Autenticação

O sistema de autenticação fornecido pelo Firebase funciona por meio de chamadas específicas a serviços oferecidos pelo módulo AngularFire. Para esta aplicação foi desenvolvido o modo de se autenticar utilizando e-mail e senha, sem a possibilidade de fazer uma autenticação utilizando redes sociais (o que é oferecido também no plano gratuito do Firebase).

A figura 4.3 apresenta o modo como a aplicação se conecta ao servidor de autenticação. Uma *Factory* é criada para armazenar o objeto que contém as funções de autenticação do servidor e, por uma particularidade do AngularJS, pode ser injetada em outros componentes da aplicação sem ser re-instanciado. Aqui, `ref` é um apontador para o servidor no Firebase e `$firebaseAuth` é um serviço fornecido pelo AngularFire para prover acesso ao servidor de autenticação.

As funções de autenticação fornecidas (com parâmetros omitidos) pela então criada *factory* Auth são:

- `$createUser`: Cria um usuário no sistema de autenticação.
- `$authWithPassword`: Autentica um usuário com e-mail e senha.
- `$getAuth`: Verifica se o usuário está atualmente autenticado com o servidor.
- `$onAuth`: Escuta pela autenticação de um usuário no sistema.
- `$offAuth`: Deixa de escutar por autenticações no sistema.

4.3 Rotas

Como descrito no título deste trabalho, a aplicação desenvolvida se trata de uma *Aplicação de Página Única* (SPA). Este conceito se refere à capacidade de uma aplicação web se assemelhar a um aplicativo de *smartphone*, onde todas as *views* de uma aplicação estão contidas em um só aplicativo, e não há a necessidade de buscar por novas páginas HTML no servidor (TAKADA, 2013). Em resumo, aplicações que não se valem deste

modelo necessitam implementar todas as minúcias de estados intermediários no lado do servidor e somente conseguem uma alteração parcial da tela utilizando diversas diretivas de JQuery e manipulações no DOM.

Com a utilização deste paradigma de desenvolvimento, podemos fazer alterações complexas em porções da tela com o mínimo de esforço, mantendo o código limpo e manutenível, graças aos serviços oferecidos pelo AngularJS. Primeiramente será apresentado o modo como as páginas da aplicação foram pensadas e em seguida será mostrado como a aplicação implementa estas páginas.

4.3.1 Diagrama de Rotas

Seguindo os modelos apresentados nos diagramas de entidade (seção 3.2), a aplicação foi organizada de modo a separar semanticamente as porções de código que trabalham com usuário, grupos e eventos. A figura 4.4 mostra como os componentes chave da aplicação são relacionados entre si e como é feita a navegação entre eles.

O usuário do sistema acessa a página inicial (*Home*), de onde pode se registrar ou se autenticar. Passada a etapa de autenticação, ao usuário é permitido acesso aos módulos que controlam usuários, grupos, eventos e busca, além de dar acesso a um painel (*Dashboard*) com informações relevantes.

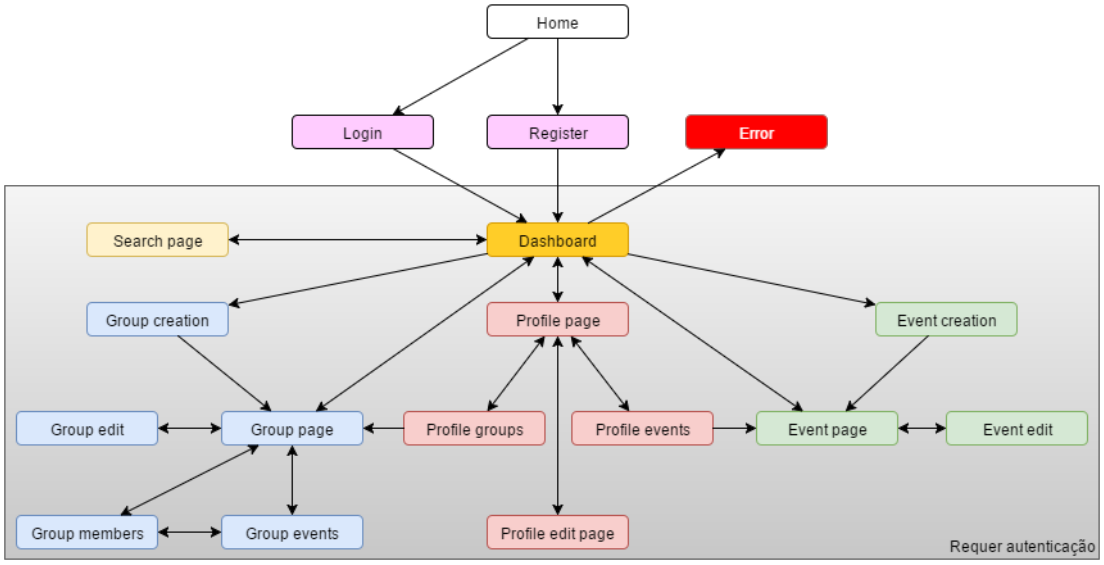
É importante salientar neste ponto que todas as diferentes *views* da página são carregadas sob demanda do servidor e aparecem em uma porção reservada da tela. Deste modo a aplicação se torna capaz de reduzir o consumo de rede e dados de um usuário para que esse possa acessar o conteúdo desejado sem precisar carregar todas as diversas *views* da aplicação na sua inicialização.

4.3.2 AngularJS UI-Router

AngularJS fornece, com seus componentes base, a capacidade de gerenciar rotas da aplicação por meio do controle de URLs. Para facilitar no desenvolvimento de uma SPA utilizando estes módulos, o programador precisa lançar mão de recursos como a inclusão de *templates* para a geração de *views* parciais.

Afim de cobrir as dificuldades proporcionadas pela biblioteca base, foi desenvolvido o módulo UI-Router,que trata as *views* da aplicação como uma máquina de estados,

Figura 4.4: Diagrama de rotas do Game Finder



permitindo o cascateamento de atributos para estados-filho, visualização de estados paralelos e encadeados, entre outros (ANGULAR-UI-ROUTER, 2013).

4.3.3 Construção de Rotas

Baseando-se na utilização e funcionalidades do módulo UI-Router (ANGULAR-UI-ROUTER, 2013), a aplicação foi estruturada de modo que a autenticação pudesse ser coberta a partir do encadeamento de estados. Assim, definindo um estado abstrato (que não possui uma *view* associada) da aplicação que requer que o usuário esteja autenticado pelo Firebase, há a possibilidade que todos outros estados aninhados se valham desta restrição automaticamente.

A figura 4.5 mostra como as rotas e estados do Game Finder foram construídas. Inicialmente é injetado o serviço que faz o gerenciamento de rotas e estados (respectivamente *$stateProvider* e *$urlRouterProvider*). Em seguida se define dentro destes serviços a rota padrão da aplicação (caso um estado inválido seja acessado) e todos os demais estados que a aplicação dará suporte.

Um estado é definido pelo método `state` passando como argumentos o nome do estado e um objeto de configuração. Neste objeto são definidas propriedades como a URL que aparecerá no navegador, qual arquivo HTML será carregado como *template* do estado, qual controlador será vinculado ao estado, etc. Um aninhamento de estados é definido pelo nome do estado criado, utilizando uma nomenclatura hierárquica no formato `estadoPai.estadoFilho`.

Figura 4.5: Configuração de rotas do Game Finder

```
app.config(["$stateProvider", "$urlRouterProvider", "$provide",
    function ($stateProvider, $urlRouterProvider, $provide) {
    // For any unmatched url, redirect to #/
    $urlRouterProvider.otherwise("/dashboard");

    // Now set up the states
    $stateProvider
        .state('main', {"url": "/"...})

        .state('login', {
            url: "/login",
            templateUrl: "app/auth/login.html",
            controller: "authController"
        })
        .state('register', {"url": "/register"...})

        .state('root', { //every state under root will require authentication.
            abstract: true,
            data: {requireLogin: true},
            templateUrl: "app/extra/main.html"
        })

        /* DASHBOARD STATES */
        .state('root.dashboard', {"url": "/dashboard"...})

        /* PROFILE STATES */
        .state('root.profile', {"abstract": true...})
        .state('root.profile.page', {"url": "/profile/:username"...})
        .state('root.profile.edit', {"url": "/profile/:username/edit"...})
        .state('root.profile.groups', {"url": "/profile/:username/groups"...})
        .state('root.profile.events', {"url": "/profile/:username/events"...})
```

É possível também definir variáveis que serão passadas na URL usando a sintaxe `page/:parameter`, e pode-se então navegar até o estado utilizando a chamada `$state.go('stateName', variable: value)` para renderizar o estado com a URL `www.website.com/#!/page/value`.

Para o Game Finder foi definido um estado abstrato chamado `root` que requer a autenticação do usuário para ser acessado. Caso o usuário não esteja autenticado, este é levado a um estado público para efetuar a autenticação (`login`). Os demais estados que requerem autenticação são definidos como um aninhamento do estado `root`, como por exemplo as páginas de perfil (`root.profile`), grupo (`root.group`) e estados internos a estas, como a visualização de grupos de um usuário (`root.profile.groups`) ou eventos que um grupo promove (`root.group.events`).

Figura 4.6: Apontamento à API do Firebase

```
var ref = new Firebase("https://game-finder.firebaseio.com/");
```

Figura 4.7: Busca de um usuário no banco de dados pelo seu ID

```
// Select * from users where _uid = userID
this.selectUserByUserID = function (userID) {
    return ref.child('users').child(userID);
};
```

4.4 Queries

Diferente do modo como queries funcionam em bancos de dados relacionais, onde o usuário faz o acesso aos dados de uma tabela, o Firebase oferece em sua API métodos para escutar uma porção do documento JSON sendo armazenado. O sistema de consulta requer uma referência para uma coleção de dados, um tipo de evento para escutar e um callback que será invocado toda vez que o Firebase disparar um evento do tipo em questão (FIREBASE, 2009c).

As referências são apontadores para coleção de dados no banco, onde uma série de filtros e ordenamentos são feitos para conseguir os dados específicos. A referência é definida como um objeto de conexão com o servidor (figura 4.6) instanciado como um objeto `Firebase`. Um caminhamento até a porção do documento que será consultado (figura 4.7) se dá pela utilização dos métodos `child`, referenciando atributos dos objetos JSON armazenados. As operações de filtragem e ordenação (figura 4.8) necessitam primeiramente de um método de ordenação de dados (como por exemplo o `orderByChild`) e em seguida uma série de restrições aos dados sendo retornados, como a filtragem do conteúdo de uma *string* (método `equalTo`).

Uma consulta é feita no banco por meio de um entre dois métodos:

- `on`: Executado ativamente sobre uma referência do banco, recebendo um tipo de evento e um callback para ser invocado com o snapshot do banco toda vez em que o evento é disparado no Firebase.
- `once`: Executado uma única vez sobre uma referência do banco, recebendo um tipo de evento e um callback para ser invocado com o snapshot do banco quando o

Figura 4.8: Busca de um grupo no banco de dados pelo nome

```
// Select * from groups where _uglyName = flatGroupName and rownum = 1 order by _uglyName
this.selectUniqueGroupByFlatName = function(flatGroupName) {
    return ref.child("groups").orderByChild("_uglyName").equalTo(flatGroupName).limitToFirst(1);
};
```

evento é disparado no Firebase.

4.5 Organização dos Dados

Um dos tipos mais comuns de inserção de dados utilizados neste trabalho se baseia na execução ordenada de dois métodos. Primeiramente é chamado o método `push` que cria um objeto no local da referência e devolve um ID gerado dinamicamente. Em seguida se chama o método `set` passando como parâmetro o JSON que será armazenado nesta referência. Para a deleção de um documento no banco é possível simplesmente executar um método `set` na referência que será apagada passando um objeto JSON nulo (*null*).

Para o armazenamento dos objetos JSON no banco foram utilizadas as seguintes estruturas: Todos os objetos que representam entidades ficam vinculados à uma raiz do documento, sendo nomeados os objetos de *users*, *sports*, *groups* e *events*.

A figura 4.9 mostra como é feito o armazenamento dos dados de uma usuário do sistema, incluindo seus dados cadastrais, lista de grupos que faz parte (contendo suas permissões no mesmo), lista de eventos que foi convidado (incluindo seu estado de confirmação) e esportes favoritos. A figura 4.10 apresenta a estrutura dos eventos, guardando o nome do esporte, ícone e número recomendados de jogadores.

Na figura 4.11 mostra como os grupos são armazenados. Nestes são registrados a lista de dados do cadastro do grupo, os eventos promovidos, membros, etc. As figuras 4.12 e 4.13 mostram o armazenamento de eventos no sistema, que é dividido em dois grupos de dados por questões de experimentação em performance de queries. Na figura 4.12 são mostrados os dados armazenado do grupo no momento do cadastro, como seu nome, jogadores, local, preço e esporte. Na figura 4.13 é mostrado o armazenamento da lista dos usuários participantes ou não da atividade.

4.6 Componentes

Como comentando na sub-subseção 3.3.1.2, os arquivos foram organizados de forma mista nos diretórios do projeto, afim de separar componentes tanto de entidades diferentes quanto de camadas diferentes da arquitetura. Em se tratando de uma arquitetura MVC, os grupos de arquivos HTML e JS que faziam parte de uma mesma entidade foram deixados juntos em um diretório com o nome da entidade. Temos então diretórios como

Figura 4.9: Estruturação dos objetos de usuários

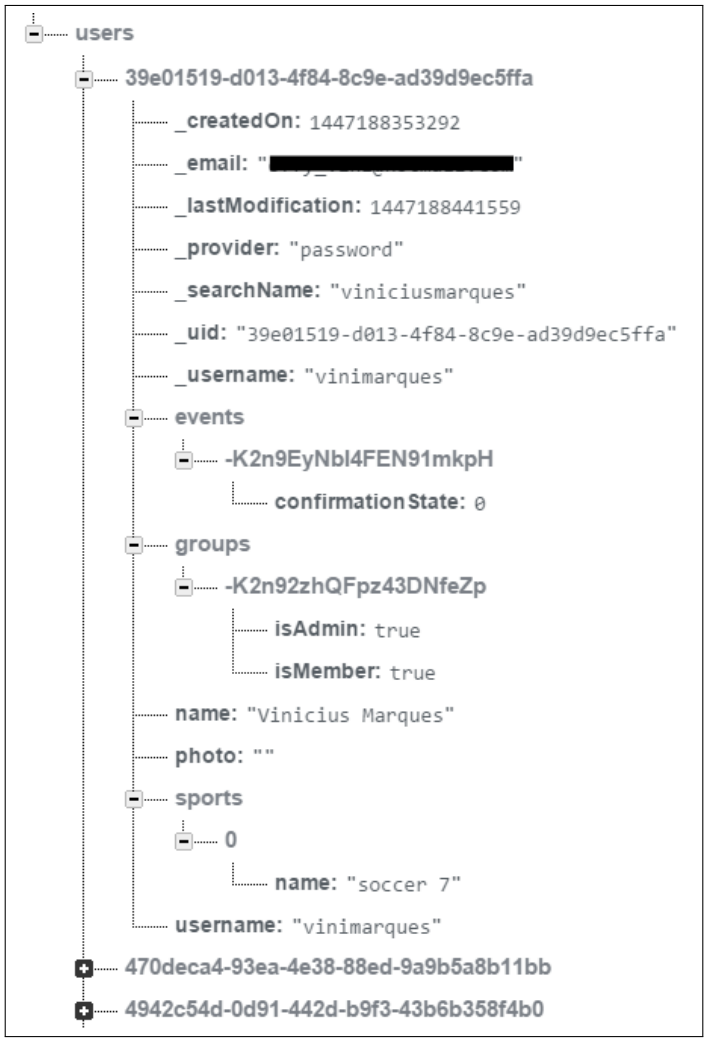


Figura 4.10: Estruturação dos objetos de esportes

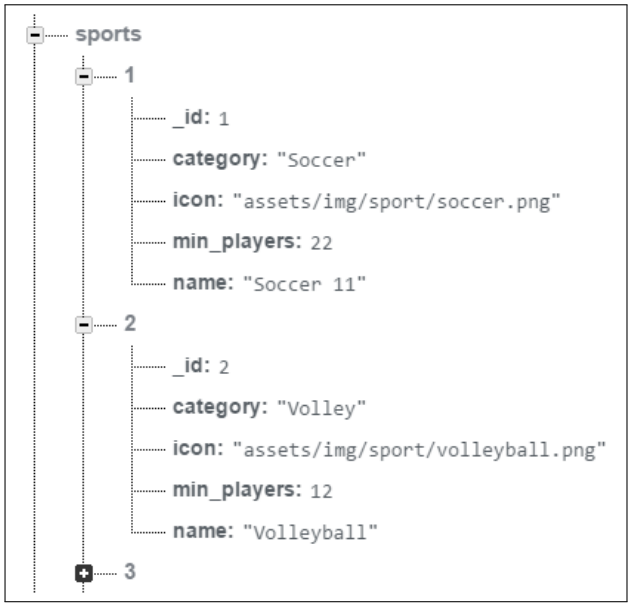
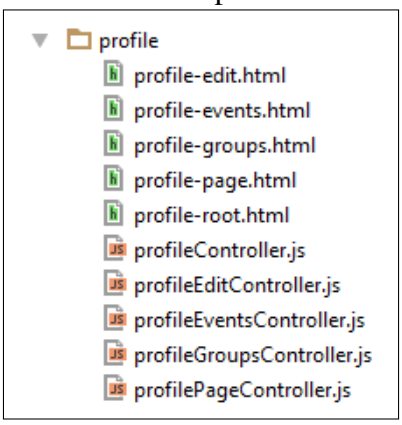


Figura 4.11: Estruturação dos objetos de grupos

Figura 4.12: Estruturação dos objetos de eventos

Figura 4.13: Estruturação dos objetos de participantes de um evento

Figura 4.14: Estrutura de pastas da entidade Perfil



`profile`, `group` e `event` contendo os controladores para cada estado da aplicação.

A figura 4.14 Mostra como os arquivos são estruturados na pasta de uma entidade, mostrando a lista dos arquivos HTML de *views* parciais da tela e a lista dos arquivos JS responsáveis por fornecer controladores para cada *view*. Afim de fornecer as regras de negócio de modo estruturado ao sistema, um outro conjunto de arquivos foi organizado afim de guardar arquivos JS relacionados aos serviços AngularJS criados para tratar das manipulações de dados oriundos do banco para cada entidade. Foram criados então os serviços `ProfileService`, `SportService`, `GroupService` e `EventService` que externalizam métodos que facilitam o acesso aos dados armazenados no Firebase, além de calcular regras de negócio e oferecer manipulações de modelos de modo transparente para os controladores.

4.7 Sistema de Busca

Afim de fornecer um sistema onde o usuário pode pesquisar por outros participantes do Game Finder e encontrar grupos em que possa participar, foi pensado um sistema de busca para a aplicação. Nele, o usuário é capaz de encontrar usuários e grupos pelos seus nomes.

Como um grande fator limitador para buscas mais complexas, o Firebase oferece momentaneamente somente buscas por intervalos de valores de atributos em objetos dentro de um documento. O que isso quer dizer é que as tradicionais diretivas de filtragem presentes em *RDBMS* como o `LIKE '%string%'` não estão disponíveis para o uso, deixando para o desenvolvedor somente a possibilidade de encontrar usuários e grupos onde seus nome são lexicograficamente maiores ou iguais ao termo consultado.

Esta característica funciona muito bem para obtenção de valores entre intervalos de datas (*timestamps*) e geolocalização, onde pode-se definir como máximo e mínimo de um intervalo a coordenada em latitude ou longitude mais um raio definido. No entanto, para strings, foi somente possível desenvolver um sistema onde os nomes eram limitados inferiormente pelo termo pesquisado. Uma consulta por usuários chamados "Miguel"retornariam outros usuários na busca como "Paulo", "Sandro"e "Tomás".

5 GUIA DE USO

Este capítulo faz uma análise mais aprofundada sobre a aplicação desenvolvida, o Game Finder, e como o usuário a utiliza, navega e se organiza nos seus grupos e eventos esportivos. Para tanto, as próximas seções mostrarão o que é oferecido para o usuário em questão de funcionalidades e em seguida listar como as funcionalidade foram implementadas.

5.1 Navegação

Assim como abordado na seção 4.3, o Game Finder oferece um conjunto de páginas que abordam diferentes componentes de uma mesma entidade, as renderizando com uma *view* parcial e utilizando o módulo AngularJS-UI-Router para controlar uma máquina de estados da navegação.

Para um maior controle da autorização do usuário são feitas uma série de validações para garantir que o usuário possui acesso à página que está tentando acessar. O primeiro tipo, também já comentado na seção 4.3, é a autenticação, que valida as credenciais de um usuário no sistema de autenticação para que essa possa acessar as funcionalidades do Game Finder.

Para os demais componentes e entidades é feita uma verificação interna ao banco de dados afim de garantir que o usuário tem acesso aos dados sendo requisitados. Como exemplo, ao tentar acessar via URL a página de edição de perfil de outro usuário, o sistema rejeita a requisição, pois valida que o usuário autenticado é diferente do dono do perfil, e leva o usuário para a página do perfil novamente. Esse comportamento é comum também nas páginas de edição de grupos e eventos (caso o usuário não seja um administrador do grupo em questão ou do grupo promovendo a atividade) ou nas páginas de evento caso o usuário não seja membro do grupo que promove a atividade.

Na figura 5.1 é mostrado como exemplo o acesso à página de um evento na aplicação, mostrando os passos que são tomados pelo controlador da página até que seja definido se o cliente pode ver seu conteúdo ou não. São feitas algumas consultas no banco até descobrir se o usuário tem permissão para acessar o que está sendo requisitado, e só aí a aplicação renderiza o conteúdo na tela.

Uma série de outras estórias de usuário foram elaboradas no começo deste trabalho para cobrir aspectos como privacidade de grupos e eventos, mas foram deixadas em

Figura 5.1: Fluxograma de acesso a um evento dentro do Game Finder

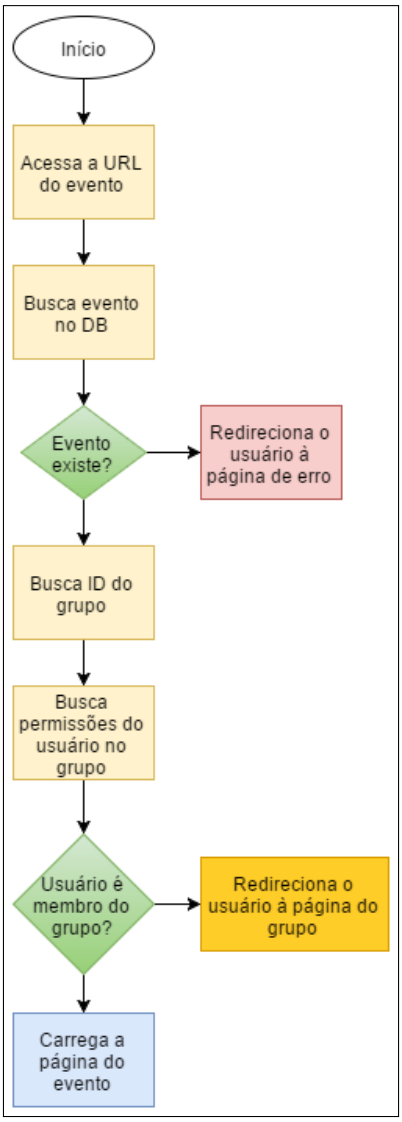
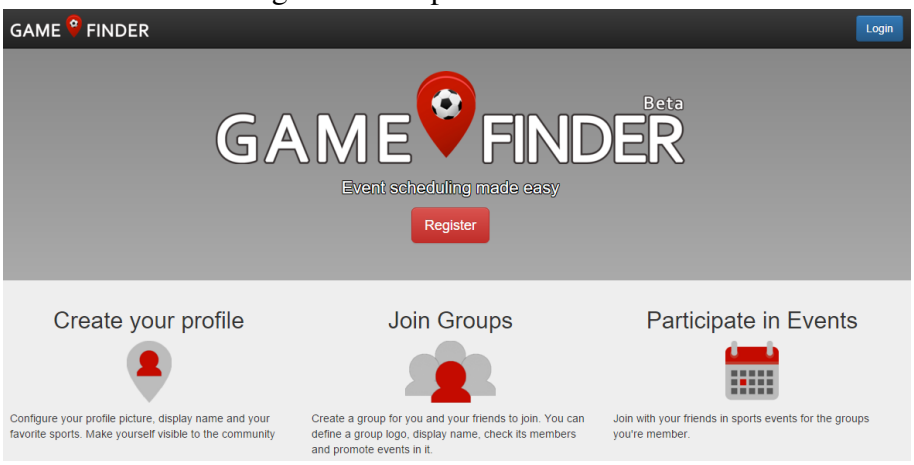


Figura 5.2: Capa do Game Finder



segundo plano e não foram implementadas para simplificar o escopo.

5.2 Componentes

Nesta seção serão mostrados com mais detalhes o funcionamento de cada porção do Game Finder, mostrando os componentes HTML utilizados e o que cada elemento da tela representa, além do exemplificar uma rota de uso do sistema. Será dada preferência para mostrar somente o conteúdo da *view* parcial que estará sendo comentada.

5.2.1 Capa, Login e Registro

O design desta aplicação foi inspirado em uma tendência entre desenvolvedores Front-End de mostrar, de modo objetivo, o conjunto principal de ações logo na capa do site. A figura 5.2 mostra como os elementos que caracterizam minimamente a aplicação foram dispostos na capa do sistema. Nesta tela é feita uma explicação superficial e objetiva das funcionalidades presentes, além de exemplificar o que pode ser feito no sistema e como. É também adicionado à esta tela botões para navegar até as páginas de Login e Registro do sistema.

As figuras 5.3 e 5.4 mostram como o usuário se autentica e se registra no sistema. Para o registro é necessário que o usuário informe um e-mail e senha, para autenticação, além de um nome de usuário que deve ser único no sistema. Com este nome de usuário podemos fazer a navegação até a página de perfil de uma pessoa por meio da rota `/profile/<username>`.

Figura 5.3: Tela de login do Game Finder

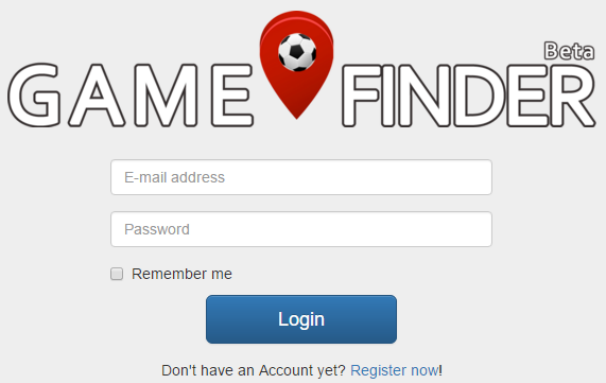


Figura 5.4: Tela de registro do Game Finder

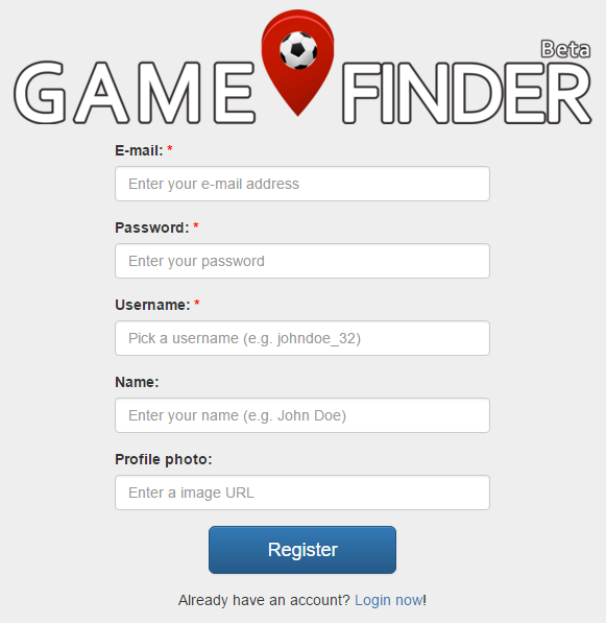
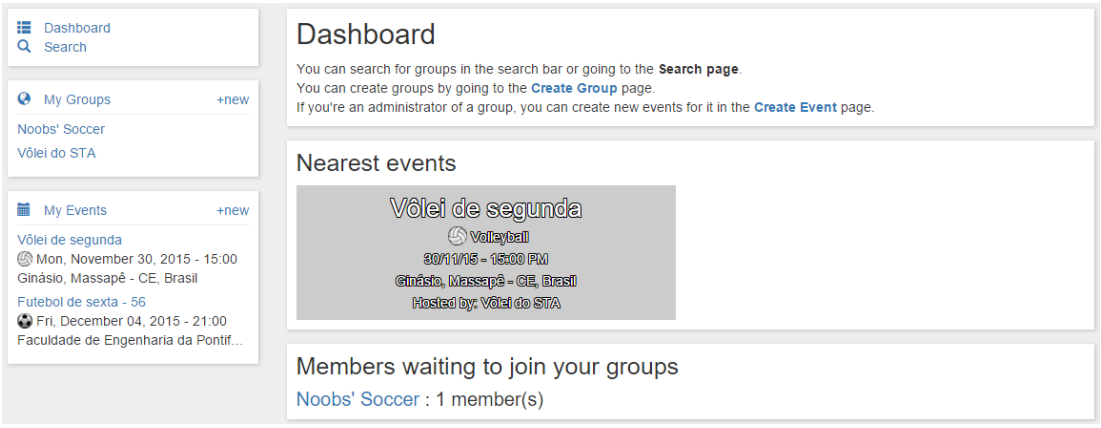


Figura 5.5: Painel do Game Finder - Tela inicial



5.2.2 Painel

O painel da aplicação mostra informações relevantes para um usuário que acessa o Game Finder. Entre as informações listadas estão algumas dicas de uso, eventos que ocorrerão nos próximos três dias sendo organizados pelos grupos que o usuário faz parte e a quantidade de pessoas esperando por aprovação nos grupos que o usuário é administrador.

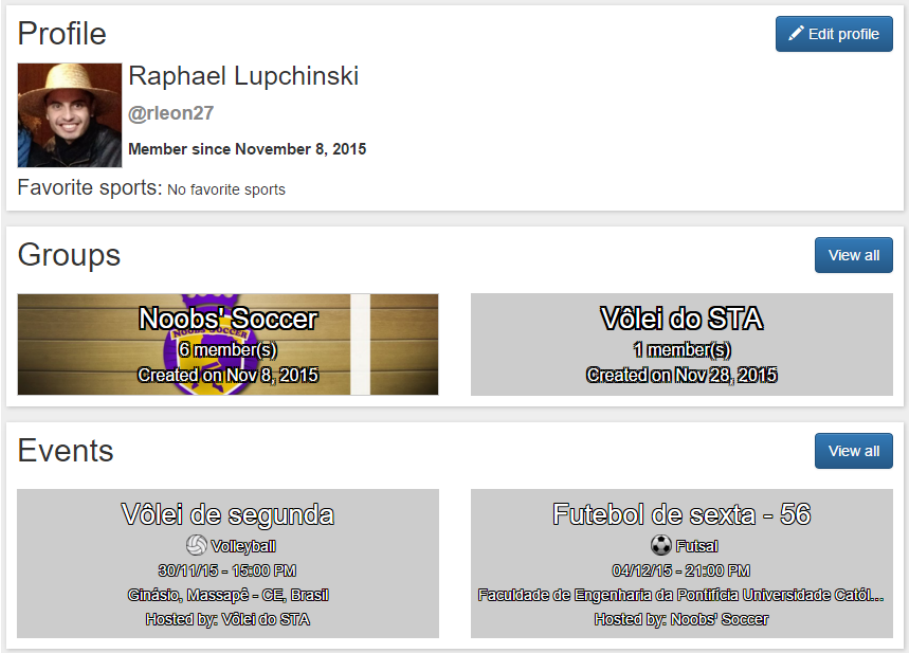
É também oferecido para o usuário que acessa o Game Finder em um tablet ou no desktop um menu lateral de navegação, incluindo outras informações como uma listagem rápida dos grupos que o usuário faz parte ou os próximos 5 ventos que acontecerão. A figura 5.5 mostra como os dados recém listados são apresentados para o usuário.

5.2.3 Perfil

O conjunto de telas que definem o perfil de um usuário na aplicação é composto pela tela de perfil central, a listagem dos grupos do usuário e a listagem de eventos do usuário. Adicionalmente o usuário tem acesso a uma página de edição de perfil para alterar seus dados cadastrais, como nome, imagem de perfil e esportes favoritos.

Na figura 5.6 são mostrados os dados principais do usuário em um cartão inicial, os quatro grupos mais antigos que este usuário faz parte e os quatro próximos eventos que o usuário foi convidado. Clicando nas opções *View all* do grupo o usuário é levado para a página de grupos do perfil sendo visualizado e na do evento o usuário é levado para a página dos evento do perfil sendo visualizado.

Figura 5.6: Página de perfil de um usuário



5.2.4 Grupos

Entrando na página de um grupo o usuário consegue ver as informações deste grupo, como seu nome, descrição, data de criação, lista de membros que fazem parte do grupo e lista dos eventos próximos que o grupo hospeda. Caso o grupo possua uma fila de espera de usuários para serem aceitos e o usuário acessando a página do grupo é um administrador deste grupo, é mostrada uma mensagem dizendo para o administrador se dirigir à página de membros do grupo para aceitar ou rejeitar os pedidos de participação.

A figura 5.7 mostra os principais componentes da tela de grupos, incluindo a mensagem informando quantos usuários estão na espera para entrar no grupo, membros e eventos próximos.

5.2.5 Eventos

A página de um evento contém informações acerca da atividade esportiva que foi organizada por um grupo. Em sua criação, o administrador de um grupo preenche um formulário com as informações da partida (custo, local, data, etc.) e é em seguida redirecionado para a página do evento recém criado. A figura 5.8 mostra a página de um evento, mostrando as informações que foram preenchidas durante seu cadastro, um botão para confirmar ou não sua participação e uma listagem dos membros participantes.

Figura 5.7: Página inicial de um grupo

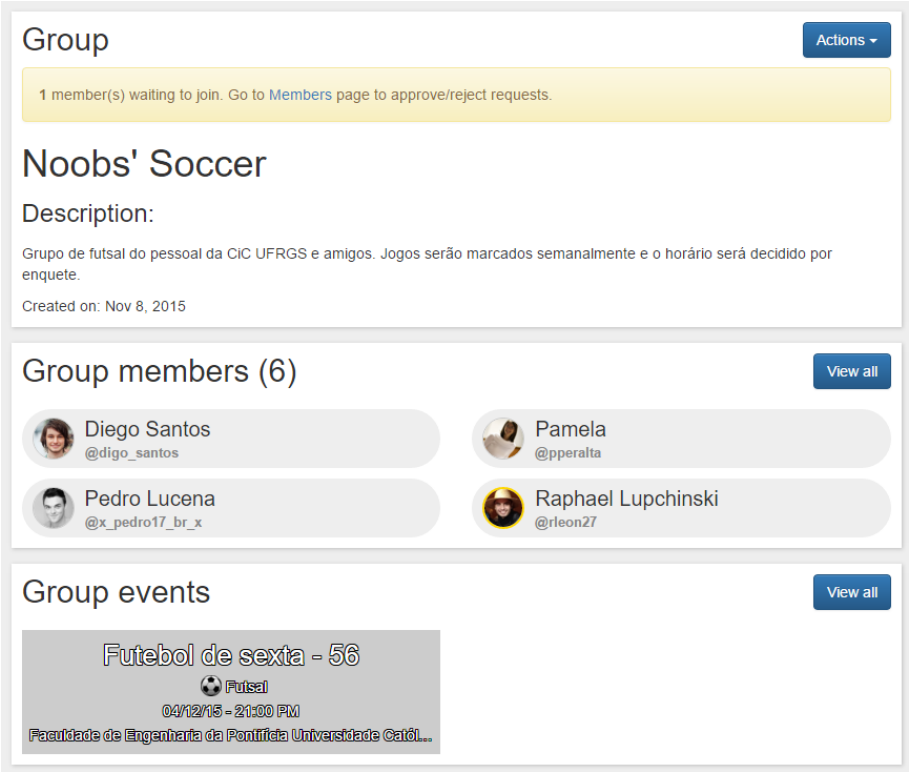
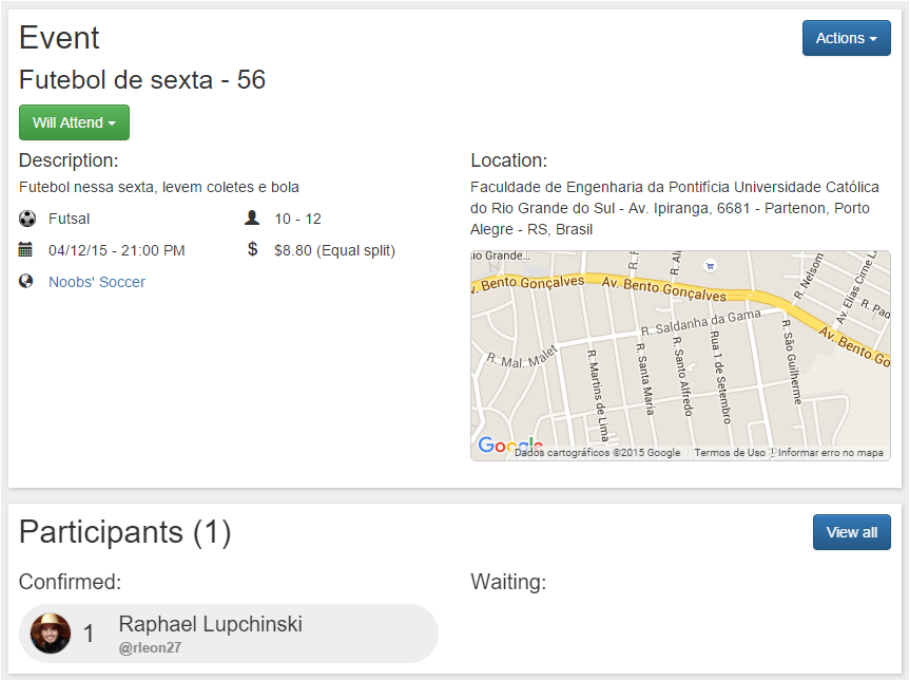


Figura 5.8: Página de um evento



6 AVALIAÇÃO E PESQUISA

Este capítulo fala sobre a pesquisa realizada sobre o Game Finder com diversas pessoas afim de obter feedback sobre determinados comportamentos da aplicação, seu estilo, facilidade de uso e entender as necessidades dos usuários do sistema para desenvolver um produto melhor no futuro. Ao todo, vinte e seis (26) pessoas responderam o questionário produzido no Google Forms e divulgado via Facebook no perfil do autor deste trabalho e nos grupos esportivos que o autor deste trabalho participava.

A pesquisa foi dividida em três partes: A primeira identifica o perfil dos participantes, perguntando a eles sobre seus conhecimentos em informática e organização de eventos esportivos, se pratica esportes ou não e se teria interesse em utilizar uma aplicação de organização de eventos esportivos. Usuários que não tinham interesse em utilizar a aplicação tiveram seus questionários encerrados para evitar vícios nas respostas seguintes.

A segunda parte pedia para que o usuário acessasse a aplicação e fizesse uma série de operações, para posteriormente dar seu feedback sobre a facilidade de executar a ação e permitir que fossem feitos comentários sobre a funcionalidade em questão. Como roteiro foi escolhido a trilha de se registrar na aplicação, editar a lista de esportes favoritos, criar um grupo, criar um evento e pesquisar por um grupo.

A terceira parte do questionário envolvia perguntas gerais sobre a aplicação, permitindo que os participantes pudessem dar um parecer final sobre o produto e, dentre uma lista de outras funcionalidades da aplicação, escolher quais as que ele ou ela gostariam de ver sendo implementados no futuro. Tais funcionalidades englobavam tanto o controle de pagamentos quanto a capacidade de fazer amigos na aplicação.

Nesta pesquisa o usuário era direcionado para um conjunto específico de perguntas dependendo de suas respostas anteriores. Com isso, no momento da geração das estatísticas, é possível que os números de total (ou *100%*) sejam discrepantes. do total de pessoas que efetivamente responderam o questionário.

6.1 Identificação de Perfil

Na primeira seção do questionário foi avaliado o perfil dos participantes da pesquisa. A figura 6.1 apresenta a idade dos pesquisados, que em sua maioria possuem entre 21 e 25 anos. As figuras 6.2 e 6.3 relacionam o conhecimento do entrevistado em informática e acerca do domínio da organização de eventos esportivos, que possuem em média

Figura 6.1: Idade dos pesquisados

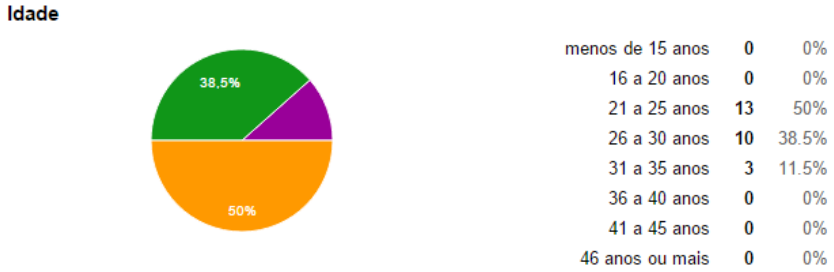


Figura 6.2: Nível de conhecimento em informática dos pesquisados

um perfil alto conhecimento em informática e médio conhecimento em organização de eventos.

Adicionalmente foi compilado que 88% (23) dos pesquisados praticam algum esporte e 100% destes possuem interesse em uma aplicação de organização dos eventos esportivos. Os demais participantes que não tiveram interesse na aplicação foram removidos do conjunto das respostas das perguntas seguintes.

A figura 6.4 mostra a distribuição dos esportes favoritos de cada pesquisado, sendo sua grande maioria o futebol o esporte escolhido. A figura 6.5 mostra a frequência com que cada pesquisado realiza uma atividade física ou participa de algum evento esportivo, onde a maioria pratica algum esporte menos de uma vez por semana. Foi coletado também que 78% (18) dos pesquisados participam de um grupo para realizar suas atividades esportivas, e as principais ferramentas utilizadas para organização são o Facebook (15), WhatsApp (14) e o Peladeiro.com (2). Para os pesquisados que não participam de grupos, as principais ferramentas são o WhatsApp (4), Facebook (3) e SMS (2).

Ainda sobre os pesquisados que participam de um grupo, 61% (11) acham que as

Figura 6.3: Nível de conhecimento do domínio dos pesquisados

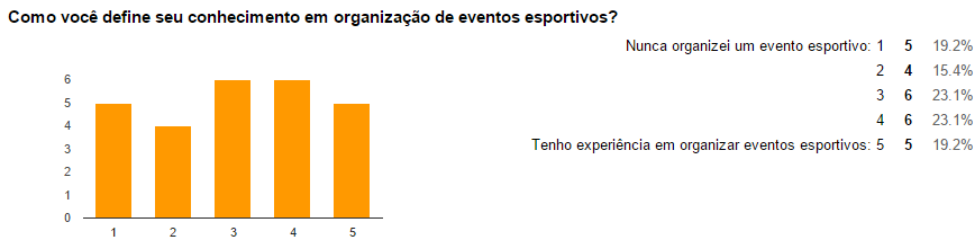


Figura 6.4: Esportes favoritos dos pesquisados

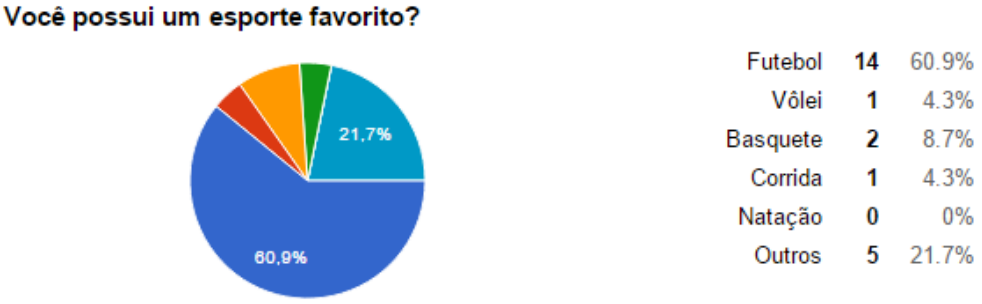
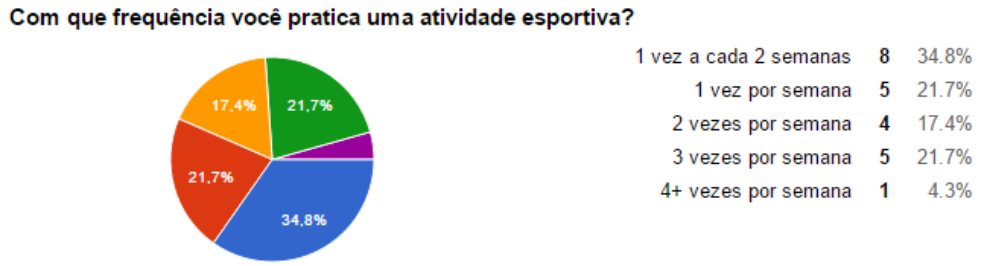


Figura 6.5: Frequência da prática esportiva dos pesquisados



ferramentas atuais são suficientes para a organização dos eventos esportivos, utilizando-as para basicamente ser informado sobre quando uma partida acontecerá, se foi cancelado ou não e para coletar a lista dos jogadores que podem participar.

6.2 Funcionalidades

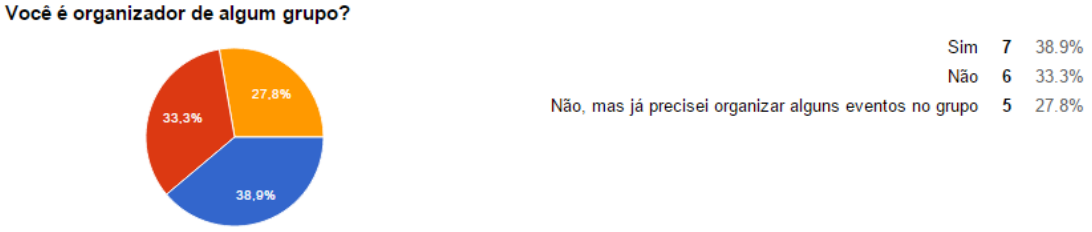
Foram preparadas uma série de perguntas sobre funcionalidades que os praticantes acham fundamentais ou não na hora de organizar um evento esportivo. Entre elas se encontram o controle de pagamento, controle de tempo da partida ou a troca de mensagens entre os participantes. A seguir se encontra uma listagem do interesse médio dos pesquisados em cada funcionalidade apresentada.

- **66%** (12) dos pesquisados acham a *troca de mensagens entre membros do grupo* muito importante, utilizando basicamente o WhatsApp para se comunicar.
- **66%** (12) dos pesquisados acham a *ordem de confirmação dos participantes* muito importante.
- **44%** (8) dos pesquisados acham que *dividir os participantes em times* é muito importante.
- **77%** (14) dos pesquisados acham que *controlar o pagamento dos participantes* é muito importante.

- **44%** (8) dos pesquisados acham que *controlar o tempo do evento* é muito importante, utilizando basicamente o cronômetro do celular para tal tarefa..

Finalmente, conforme apresentado na figura 6.6, mostra-se que basicamente dois terços (2/3) dos pesquisados já tiveram que organizar algum evento esportivo, sendo que metade deles organiza regularmente as atividades de seus grupos.

Figura 6.6: Pesquisados que organizam eventos para seus grupos



6.3 Uso da Aplicação

Baseando-se em uma preferência pessoal do autor deste trabalho, a aplicação foi desenvolvida em Inglês. Sabendo que o questionário seria aplicado para pessoas no Brasil foi perguntado aos participantes seu domínio da língua inglesa, afim de identificar se algum dos passos para utilizar a aplicação corretamente não seriam afetados pelo seu conhecimento da língua. em média, o conhecimento dos participantes varia em torno de 4.04/5, considerado forte.

Para o uso da aplicação foi pedido que os usuários acessassem o Game Finder, hospedado no Firebase, e realizasse as seguintes cinco tarefas: Registrar-se na aplicação (1), editar seu perfil para adicionar um esporte favorito (2), criar um grupo (3), criar um evento para este grupo (4) e pesquisar pelo grupo no sistema de busca (5).

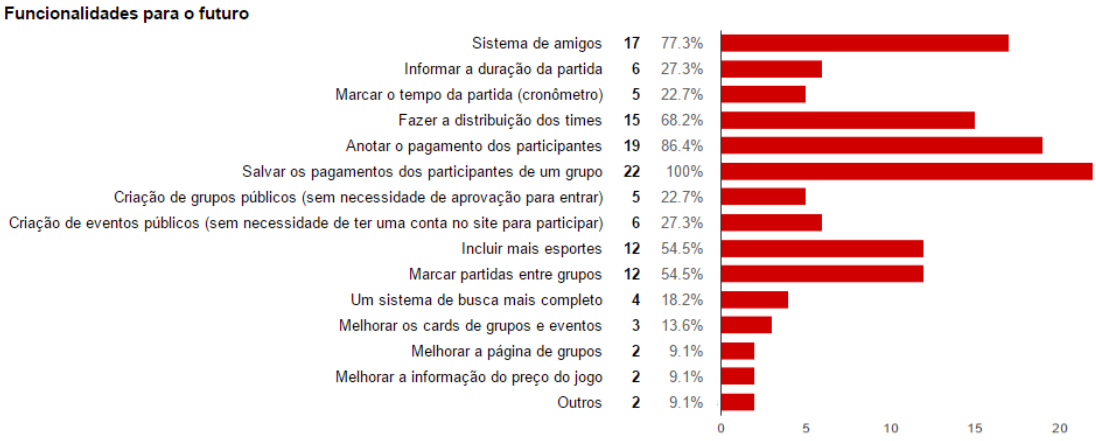
A seguir se encontra os dados de feedback dos usuários para cada tarefa proposta - lembrando que neste ponto haviam 23 participantes da pesquisa:

- **Tarefa 1**: 69% (16) dos pesquisados acharam o sistema de **registro** simples e/ou intuitivo (nota 5/5). Como sugestões se destacam: "*Associar a criação da conta a conta google/facebook*", "*Poderia haver um campo para confirmar a senha, senão a pessoa pode digitar errado e registrar, e aí depois não consegue mais acessar.*"e "*Poderia ser em português.*".
- **Tarefa 2**: 78% (18) dos pesquisados acharam fácil e/ou intuitivo o acesso até o menu de **edição de perfil** e 82% (19) consideram o sistema de edição simples e/ou

intuitivo (nota 5/5). Como sugestões se destacam: "*Gostei da idéia do App, muito mais fácil de organizar jogos por ali, sei que é um público pequeno mas se possivel seria legal usar esse app pra organizar jogos nerds também, ou até mesmo eventos.*"e "*Apesar de fácil de entender e intuitivo, é ruim não poder fazer o upload de uma foto salva no HD. Poderia haver sincronização com o Facebook para achar fotos e colocar o nome de usuário automaticamente. Quando você digita o esporte já aparecem as tags com ícones, o que é excelente.*"

- **Tarefa 3**: 91% (21) dos pesquisados acharam fácil e/ou intuitivo o acesso até o menu de **criação de grupos** e 82% (19) consideram o sistema de criação de grupos simples e/ou intuitivo (nota 5/5). Como sugestões se destacam: "*Talvez esteja muito em bloco, deixar tudo na vertical pode deixar melhor pro smartphone (a menos que tu já tenha feito uma versão pro smartphone que eu não vi)*", "*Consegui criar um grupo, mas fiquei sem saber como entrar num grupo já existente e como convidar outros usuários para o meu grupo.*"e "*Simples e direto ao ponto, talvez expandir o processo de criação , permitindo adicionar pessoas logo nesta tela.*".
- **Tarefa 4**: 73% (17) dos pesquisados acharam fácil e/ou intuitivo o acesso até o menu de **criação de eventos** e 69% (16) consideram o sistema de criação de grupos simples e/ou intuitivo (nota 5/5). Como sugestões se destacam: "*Quando põe o local o esquema já puxa no Google Maps, muito bom. Não entendi de primeira as opções de preço de quadra, talvez colocar breves descrições ao passar o mouse por cima (talvez testar "Total price:"ao invés de "Price:"e "Price per member:"ao invés de "Member pays:")*", "*Parece muito cheio de opções, talvez fazer por camadas fique melhor, ir indo passo a passo. Espacialmente a parte do pagamento eu achei confusa.*"e "*Fiquei com dúvidas na parte de valores. Quanto é para cada um ou quanto é o evento.*".
- **Tarefa 5**: 69% (16) dos pesquisados acharam fácil e/ou intuitivo o acesso até o seu grupo pela **busca** e 56% (13) consideram o sistema de busca simples e/ou intuitivo (nota 5/5). As reclamações nesta etapa foram que ao buscar um termo outros dados eram retornados. Este problema está intimamente relacionado com as dificuldades apresentadas na seção 4.7, dado que a implementação corrente é feita por limitação de intervalo de valores em atributos. Como sugestões se destacam: "*Ao começar a digitar no sistema de busca ja ter sugestões para selecionar (sim, que nem o facebook :P)*", "*Por alguma razão o app pegou um grupo de voley junto.*", "*Sem auto-complete no search ou preview da search no campo ali em cima, sem advanced*

Figura 6.7: Funcionalidades desejadas para o futuro do Game Finder



options, e eu sou a unica pessoa que colocou "kayaking", mas sou o terceiro usuário a aparecer na lista. Clicar num usuario deveria mostrar o perfil, não outro menu para ver o perfil."e "*Resultados contem o termo buscado e muito mais informacoes, o que fica um pouco confuso*".

Finalmente a figura 6.7 mostra a preferência que os participantes da pesquisa têm sobre outras funcionalidades não implementadas. Entre as opções mais votadas estão àquelas de controle de pagamento, distribuição de times, inclusão de esportes e partidas entre grupos. Como surpresa para o autor deste trabalho, um grande número de participantes sugeriram que fosse também implementado um sistema de amigos dentro da aplicação. Não fora previsto no desenvolvimento da aplicação tentar cobrir funcionalidades já bem desenvolvidas por grandes redes sociais (como o sistema de amigos), mas são funcionalidades que possivelmente ajudariam na divulgação da aplicação.

Como *feedback* geral da aplicação, é possível destacar: "*No geral ta bom, só achei alguns menus muito cheios de opções*", "*Uma ótima e nova ideia para combinar eventos esportivos dinamicamente*", "*Muito intuitivo. Ótimo para organizar eventos. Com um bom layout e, se pudesse ser acessado do facebook, ficaria perfeito*", "*No geral, app bacana, ainda em estágio inicial, eu usaria.*", "*Sistema intuitivo e design simples e fácil de usar*", "*Seria muito util, tipo meetup.com para esportes. Adorei*", "*Teria como após criar o evento, mandar um link para as pessoas? Seria bacana, não cheguei a achar isso. Dai as pessoas recebem um email com o convite.*"e "*A aplicação ficou bem completa e fácil de utilizar, preenchendo uma lacuna que não era muito bem coberta pelas redes sociais atuais. O maior desafio seria o de popularizar o sistema, contudo, trata-se de algo que acrescentaria muito àqueles que praticam esporte*".

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho apresentou uma aplicação que tenta solucionar algumas dificuldades que praticantes de esporte têm na hora de organizar um evento esportivo. Adicionalmente esta aplicação também serviu como uma *prova de conceito*, cujo foco foi mostrar que é possível desenvolver com AngularJS e Firebase uma aplicação que não necessita de um Back-End explicitamente desenvolvido.

Com este trabalho o autor tenta cobrir lacunas que as ferramentas atuais de comunicação e organização possuem. Tais ferramentas, como o Facebook, são utilizadas por uma imensa quantidade de pessoas, mas não necessariamente auxiliam gestores de grupos a realizar uma organização simplificada de suas atividades. A tabela 7.1 agrupa novamente as funcionalidades das aplicações semelhantes à este trabalho e a incrementa com o conjunto de funcionalidades que o Game Finder cobre.

Tabela 7.1: Comparativo de funcionalidades das aplicações relacionadas em relação ao Game Finder

	Funcionalidades / Aplicativo	*Facebook*	*Joga +1*	*Peladeiro.com*	*Peladeiro App*	*Peladeiros*	*Futeba*	*Minha Pelada*	*Do League*	*Timpik*	***Game Finder***
Distrib.	Versão gratuita	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Versão paga			✓	✓	✓			✓		
	Versão web	✓	✓	✓					✓	✓	✓
	Versão móvel	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Social	Login com redes sociais	-	✓	✓			✓			✓	
	Criação de perfil	✓	✓	✓		✓	✓	✓	✓	✓	✓
	Esportes favoritos		✓							✓	✓
	Sistema de amigos	✓		✓					✓	✓	
	Grupos/Times	✓		✓	✓	✓	✓	✓	✓	✓	✓
	Trocar mensagens	✓		✓					✓	✓	
Gerência	Criar Partidas	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ordenar participantes			✓	✓	✓		✓	✓		✓
	Distribuição de times			✓	✓	✓	✓				
	Controle de pagamento			✓	✓	✓				✓	
	Tempo da partida				✓	✓		✓			
Esportes	Esportes com interação	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Esportes sem interação	✓	✓						✓	✓	
	Esportes coletivos	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Esportes individuais	✓	✓						✓	✓	

7.1 Aspectos Técnicos

Como anteriormente mencionado, ente trabalho serviu também como uma *prova de conceito*, onde o objetivo era mostrar que é possível desenvolver uma aplicação Front-End sem a necessidade de manipular códigos no servidor. Para tanto, todo o Game Finder foi desenvolvido no Front-End, utilizando os recursos oferecidos pelo Firebase para a hospedagem de conteúdo estático, sistema de autenticação e banco de dados não-relacional para entregar um produto que roda no lado do cliente.

As curvas de aprendizado das duas ferramentas principais utilizadas foram bem diferentes. AngularJS oferece uma curva de aprendizado suave, onde é possível ver grandes ganhos logo no começo do seu estudo. Tarefas bem complicadas são realizadas com facilidade sem a necessidade de escrever código JS com JQuery explicitamente, por exemplo. AngularJS foi a espinha desta aplicação, oferecendo nativamente em seu framework uma forma simples de trabalhar com as camadas MVC da aplicação.

Firebase, por outro lado, apresentou uma curva de aprendizado difícil de ser superada logo de início. O autor deste trabalho atribui esta dificuldade ao vício lógico de estruturar os dados de uma aplicação de forma normalizada e relacional. O principal desafio de trabalhar com um banco de dados não-relacional é estudar quais serão as consultas que serão feitas na aplicação, e dependendo da resposta, a estrutura do documento JSON no banco de dados pode ser alterada totalmente.

Adicionalmente, o AngularJS e o Firebase são tecnologias novas e em constante mudança. Suas comunidades ativas garantem que diversas melhorias e funcionalidades virão com o tempo. AngularJS está prestes a lançar sua versão 2.0, adotando os novos padrões do ES6, a nova versão de JavaScript. Firebase segue trabalhando para ampliar seu sistema de busca, oferecendo métodos e técnicas para tornar a busca por dados mais eficiente. Por hora, o que foi oferecido por estas ferramentas cobre de modo muito satisfatório as necessidades do autor deste trabalho.

7.2 Trabalhos Futuros

Há uma série de outras funcionalidades que esta aplicação pode cobrir em suas versões futuras. Assim como mencionado na listagem da estórias de usuário, uma série de funcionalidades foram abandonadas para reduzir o escopo da aplicação. Funcionalidades como a integração com redes sociais para o registro de usuários no sistema, níveis de

visibilidade em grupos, criação de eventos públicos, controle de pagamento, etc. podem ser trabalhadas nas versões futuras para cobrir melhor as necessidades dos usuários.

Novamente a figura 6.7 se mostra importante aqui. Apesar de ser uma listagem pequena de funcionalidades sugeridas, estas possuem ordem de preferência para a implementação. Nela, é listado como principal funcionalidade desejada para esta aplicação um sistema de controle de pagamentos. É possível expandir este trabalho de modo a alterar e repensar a estrutura do banco de dados para cobrir um controle de pagamentos de modo simplificado que auxilie organizadores e participantes de eventos esportivos.

Outra sugestão é criar um modo de organizar os dados de modo que seja possível extrair estatísticas de eventos esportivos, como os usuários que mais participam, suas pontuações nos jogos, usuários com pagamentos atrasados, etc. É possível também fazer um melhor uso dos dados de localização de eventos e usuários para sugerir eventos esportivos na sua redondeza. Para isso, é possível utilizar a biblioteca GeoFire (FIREBASE, 2014) ou outra abordagem em banco NoSQL ou relacionais.

É prudente também levar em consideração que para desenvolver estas novas funcionalidades uma reengenharia da estrutura do banco de dados é necessária. Recomenda-se o estudo amplo de tecnologias NoSQL ou relacionais presentes no mercado afim de encontrar uma que resolva os problemas apresentados de forma mais fácil.

Finalmente, como sugerido por alguns participantes da pesquisa, a aplicação também pode evoluir ao incluir internacionalização entre suas funcionalidades, dando suporte, preferencialmente, ao Português. Desse modo, a aplicação ganha em cobertura, pois rompe a barreira que diversas pessoas têm ao utilizar uma língua que não a materna. Para isso, é possível utilizar serviços oferecidos pelo próprio AngularJS como o `$translate`, que têm como entrada um arquivo de *labels* que são aplicados nas *views* para traduzir o conteúdo de um texto.

REFERÊNCIAS

AMBLER, S. W. **User Stories: An Agile Introduction**. 2009. <http://www.agilemodeling.com/artifacts/userStory.htm>. [Online: Acessado 18 de Novembro de 2015].

ANGULAR-UI-ROUTER. **AngularUI Router**. 2013. <https://github.com/angular-ui/ui-router>. [Online: acessado 28 de Agosto de 2015].

ANGULARJS. **AngularJS**. 2010. <https://github.com/angular/angular.js>. [Online: Acessado 28 de Agosto de 2015].

ANGULARJS. **Concepts**. 2010. <https://docs.angularjs.org/guide/concepts>. [Online: Acessado 12 de Setembro de 2015].

ANGULARJS. **Controller**. 2010. <https://docs.angularjs.org/guide/controller>. [Online: Acessado 13 de Setembro de 2015].

ANGULARJS. **Data Binding**. 2010. <https://docs.angularjs.org/guide/databinding>. [Online: Acessado 29 de Agosto de 2015].

ANGULARJS. **Services**. 2010. <https://docs.angularjs.org/guide/services>. [Online: Acessado 13 de Setembro de 2015].

BAASBOX. **What is a Backend as a Service?** 2015. <http://www.baasbox.com/what-is-backend-as-a-service/>. [Online: Acessado 17 de Outubro de 2015].

BOOTSTRAP. **About Bootstrap**. 2015. <http://getbootstrap.com/about/>. [Online: Acessado 29 de Agosto de 2015].

BOWER. **Bower - A package manager for the web**. 2012. <http://bower.io/>. [Online: acessado 28 de Agosto de 2015].

CLASSEN, D. **Comparison of 4 popular JavaScript MV* frameworks (part 2)**. 2015. <http://www.developereconomics.com/comparison-4-popular-javascript-mv-frameworks-part-2/>. [Online - Acessado 18 de Outubro de 2015].

COHN, M. **Agile Estimating and Planning**. 2008. <https://www.mountaingoatsoftware.com/uploads/presentations/Agile-Estimating-Planning-Agile-Development-Practices-2008.pdf>. [Online: Acessado 18 de Novembro de 2015].

COHN, M. **Agile User Stories, Epics and Themes**. 2014. <https://www.scrumalliance.org/community/spotlight/mike-cohn/march-2014/agile-user-stories-epics-and-themes>. [Online: Acessado 18 de Novembro de 2015].

COUCHBASE. **Comparing document-oriented and relational data**. 2011. <http://docs.couchbase.com/developer/dev-guide-3.0/compare-docs-vs-relational.html>. [Online: Acessado 2 de Novebro de 2015].

COYIER, C. **What is the DOM?** 2013. <https://css-tricks.com/dom/>. [Online: Acessado 20 de Setembro de 2015].

DB-ENGINES. **DBMS popularity broken down by database model**. 2014. <http://db-engines.com/en/ranking_categories>. [Online: Acessado 12 de Novembro de 2015].

ESPORTE, M. do. **Diagnóstico Nacional do Esporte**. 2013. <http://www.esporte.gov.br/diesporte/>. [Online: Acessado 2 de Novemvro de 2015].

FIREBASE. **About Firebase**. 2009. <https://www.firebase.com/about.html>. [Online: Acessado 28 de Agosto de 2014].

FIREBASE. **Features**. 2009. <https://www.firebase.com/features.html>. [Online: Acessado 28 de Agosto de 2015].

FIREBASE. **Retrieving Data**. 2009. <https://www.firebase.com/docs/web/guide/retrieving-data.html>. [Online: Acessado 28 de Agosto de 2015].

FIREBASE. **GeoFire — Realtime location queries with Firebase**. 2014. <https://github.com/firebase/geofire>. [Online: acessado 13 de Outubro de 2015].

FRANKLIN, A. **Tableless - Tenha o DOM**. 2011. <http://tableless.com.br/tenha-o-dom/>. [Online: Acessado 20 de Setembro de 2015].

GIT. **About Version Control**. 2015. <https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>. [Online: Acessado 30 de Agosto de 2015].

GITHUB. **Front-end Javascript frameworks**. 2014. <https://github.com/showcases/front-end-javascript-frameworks>. [Online: Acessado 16 de Novembro de 2015].

GONZALEZ, F. J. Sistema de classificação de esportes com base nos critérios: cooperação, interação com o adversário, ambiente, desempenho comparado e objetivos táticos da ação. **Revista Digital**, v. 10, n. 71, apr 2004.

GOOGLE. **Trends - AngularJS, EmberJS, ReactJS e BackboneJS**. 2015. <https://www.google.com.br/trends/explore#q=angular%20js%2C%20react%20js%2C%20backbone.js%2C%20ember.js&date=1%2F2010%2072m&cmpt=q&tz=Etc%2FGMT%2B2>. [Online: Acessado 5 de Novembro de 2015].

GREEN, B.; SESHADRI, S. **AngularJS**. first. 1005 Gravenstein Highway North, Sebastopol, CA 95472: O'Reilly, 2013.

KUKIC, A. **AngularJS Best Practices: Directory Structure**. 2014. <https://scotch.io/tutorials/angularjs-best-practices-directory-structure>. [Online: Acessado 29 de Agosto de 2015].

MONGODB. **Document Databases**. 2015. <https://www.mongodb.com/document-databases>. [Online: Acessado 28 de Outubro de 2015].

MOZILLA. **JavaScript**. 2015. <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. [Online: Acessado 20 de Setembro de 2015].

MOZILLA. **Promise**. 2015. <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise>. [Online: Acessado 30 de Agosto de 2015].

MOZILLA. **Web Socket**. 2015. <https://developer.mozilla.org/en-US/docs/Web/API/WebSocket>. [Online: Acessado 3 de Novembro de 2015].

NODEJS. **NodeJS**. 2015. <https://nodejs.org/en/>. [Online: Acessado 20 de Setembro de 2015].

NPM. **NPM - Node Package Manager**. 2009. <https://docs.npmjs.com/getting-started/what-is-npm>. [Online: acessado 28 de Agosto de 2015].

ROUSSET, D. **Introduction to HTML5 Web Workers: The JavaScript Multi-threading Approach**. 2015. <https://msdn.microsoft.com/en-us/hh549259.aspx>. [Online: Acessado 20 de Setembro de 2015].

TABLELESS. **O que é client-side e server-side?** 2012. <http://tableless.github.io/iniciantes/manual/obasico/o-que-front-back.html>. [Online: Acessado 20 de Outubro de 2015].

TAKADA, M. **Single page apps in depth**. 2013.