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

ANDRÉ LUZ GONÇALVES

Desenvolvimento de um aplicativo Android utilizando banco de dados não-relacional para organização e controle de presença de um time de futebol

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

Orientador: Prof. Dr. Krug Wives Leandro

Porto Alegre
2016

AGRADECIMENTOS

Agradeço a toda minha família que sempre me deu apoio e incentivo desde o início, a todos os amigos que conheci aqui e me ajudaram a concluir esta etapa, ao Void F.C. que sempre será o time mais simpático que já passou pela INF e a todos os professores que tive na vida, que me conduziram até aqui através de seus conhecimentos, em especial ao Prof. Dr. Leandro Wives que fez com que esse trabalho fosse possível.

RESUMO

A realização de esportes e/ou atividades físicas sempre foram de grande importância para a saúde mental e física das pessoas. Dentre eles, o futebol é o esporte preferido dos brasileiros, o qual ainda possui a carência de um aplicativo que auxilie na gerência de uma equipe, em especial no que diz respeito à confirmação de presença dos atletas nas partidas da equipe. Este trabalho foi feito devido à esta necessidade, consistindo na concepção, modelagem e implementação de um aplicativo para smartphones que rodam sobre o sistema operacional Android Jelly Bean (ou superior) utilizando o banco de dados não-relacional Firebase. O aplicativo nomeado de SeuTime, possibilita que os atletas de uma equipe de futebol confirmem sua presença em uma determinada partida, bem como compartilhem as informações dela para os companheiros de equipe através de meios eletrônicos, visando facilitar o acesso à informação das partidas de um clube de futebol entre seus jogadores, reunindo e centralizando todas estas informações em um lugar de fácil acesso para todos

Palavras-chave: SeuTime. android. java. NoSQL. firebase. futebol.

Developing an Android application with NoSQL database to manage and organize a soccer team

ABSTRACT

The practice of sports and/or physical activities has always been of great importance for human mental and physical health. Among them, soccer is Brazilian's favorite sports, but there is a need for an application that can help on the management of a team, especially when it comes to the attendance of their players on their matches. This paper was done due to that necessity, it regards the conception, modeling and implementation of an application for smartphones which runs on Android Jelly Bean operational system (or above) using the Firebase NoSQL database. The application named SeuTime enables that players of a soccer team confirms his presence on a certain match, as well as share its information to teammates by eletronic means, aiming to ease the access to information about a team's matches among its players, gathering all the information and centralizing them in the same place where all can access easily.

Keywords: T.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

NoSQL	Not Only Structured Query Language
API	Application programming interface
ORM	Object-relational mapping

SUMÁRIO

1 INTRODUÇÃO

A utilização de dispositivos móveis vem crescendo cada vez mais ao longo dos anos e, consequentemente, muitas atividades do cotidiano acabam sendo facilitadas para a sociedade através de serviços disponíveis para estes dispositivos, um exemplo que podemos apontar é o UBER, um aplicativo de carona (paga) para viagens rápidas e confiáveis em minutos (GOOGLE, 2016a). O aplicativo proposto neste trabalho visa facilitar a organização de uma equipe de futebol ajudando seus membros a reunir e compartilhar informações sobre partidas e atletas de forma simples. Há alguns anos, antes dos smartphones e aplicativos serem amplamente difundidos, algumas equipes de futebol se organizavam através de e-mails, enviando para uma lista, dados da partida como localização, horário, data, etc, e os membros desta equipe tinham que responder confirmando ou não sua presença, fazendo com que a apresentação dessas informações ficassem muito desorganizadas e as vezes dificultando o entendimento. Depois da chegada do WhatsApp aplicativo de troca de mensagens rápidas, simples e seguras (WHATSAPP, 2016b), o mesmo se tornou o ambiente principal para a troca de informações de diversas equipes, porém por não se tratar de um aplicativo com esse objetivo, muitas vezes os dados de uma partida acabam se perdendo no meio das conversas. Dentre as possíveis soluções encontradas no mercado, nenhuma pôde ser utilizada, pois havia a carência de uma solução que não necessitasse que todos atletas da equipe tivessem a aplicação em seu smartphone, já que nem todas aplicações possuíam versões para os sistemas operacionais móveis mais utilizados, ou então nem todos atletas possuíam espaço virtual para a instalação de mais um aplicativo. Para suprir essa carência, seria necessária uma solução em que um atleta pudesse confirmar a própria presença ou de seus companheiros, e também pudesse compartilhar as informações da partida incluindo a situação presencial do elenco, fazendo com que a necessidade de que todos atletas tivessem que possuir o aplicativo instalado não fosse mais obrigatória. Sentindo a necessidade de um sistema com as características citadas, foi projetado o SeuTime, aplicativo deste trabalho, onde as informações das partidas, inclusive a presença dos jogadores, estão todas reunidas no mesmo lugar, ainda podendo ser compartilhadas através de meios eletrônicos.

1.1 Objetivo

Este trabalho tem por objetivo demonstrar e explicar a criação de um aplicativo que visa facilitar o processo de organização de jogadores e integrantes de um time de futebol, a fim de saber quais estarão presentes ou não em suas partidas e ainda providenciar informações pertinentes sobre as mesmas tudo em um só lugar. As funcionalidades deste aplicativo consistem em: gerenciamento de usuários em que se pode adicionar e remover jogadores da equipe informando seu nome e a posição em que joga, gerenciamento de partidas onde é possível adicionar e remover as partidas do time informando o adversário, a localização e o horário do jogo, confirmação de presença, onde o usuário pode confirmar a presença, ausência ou deixar em dúvida o comparecimento dos jogadores nas partidas para que os integrantes da equipe possam se programar antecipadamente e chamar convidados em caso de falta de atletas, planejar estratégias com base nos jogadores confirmados, entre outras atitudes, compartilhar as informações da partida através de meios eletrônicos, integração com aplicativos de localização para identificar o local da partida e integração com serviços de agendas para criar um evento da partida.

1.2 Estrutura do texto

O trabalho está dividido em diversos capítulos e seções. O capítulo 1 contextualiza o trabalho e as motivações para sua realização, assim como o objetivo do trabalho e como ele foi estruturado. O capítulo 2 faz uma breve descrição de alguns aplicativos que possuem solução semelhante ao SeuTime e no fim do capítulo traz uma comparação desses aplicativos e suas funcionalidades. O capítulo 3 demonstra como foi projetado e modelado o aplicativo para que este pudesse atender as necessidades citadas pelas users stories descritas neste mesmo capítulo. O capítulo 4 relata o desenvolvimento do aplicativo explicando a metodologia utilizada, descrevendo a plataforma de desenvolvimento e esclarecendo onde e como funcionam as classes e pacotes. No capítulo 5 é explicado o funcionamento do aplicativo detalhando as telas apresentadas ao usuário. O capítulo 6 apresenta uma avaliação feita com base nas respostas dos usuários que utilizaram o aplicativo. E no capítulo 7 são apresentadas as conclusões e funcionalidades que possivelmente poderiam ser somadas ao aplicativo.

2 SOLUÇÕES EXISTENTES

Neste capítulo será feita uma introdução dos aplicativos que possuem uma proposta similar ao SeuTime, abordando suas funcionalidades e singularidades, e ao final do capítulo é apresentada uma tabela comparativa das funcionalidades atendidas pelos aplicativos apresentando seus pontos fracos e fortes de cada um.

2.1 Soluções existentes

Depois de ter os objetivos deste trabalho bem definidos, foi feita uma pesquisa e então selecionado alguns aplicativos que possuem funcionalidades que se assemelham às que são oferecidas pelo SeuTime, fazendo uma análise sobre suas semelhanças e peculiaridades. O SeuTime pode ser diferenciado dos outros aplicativos com o mesmo propósito, pois ele foi idealizado a partir da dificuldade do autor e seus companheiros de equipe em manter organizado a confirmação de atletas para as partidas de seu time, assim suas principais características foram projetadas para manter as mesmos aspectos que se tem no modelo atual de confirmação (Whatsapp) que são a possibilidade de confirmar a própria presença ou de algum companheiro, e a possibilidade de compartilhar em modo de texto as informações sobre uma partida e seus jogadores confirmados. Ambas funcionalidades permitem que todos atletas possam ter sua presença definida nas partidas e que eles consigam visualizar essas informações sem precisar ter o aplicativo instalado, basta ele pedir para algum companheiro que tenha o aplicativo, confirme a presença dele e compartilhe os dados da partida com ele.

2.1.1 Difora

O Difora é um aplicativo bastante focado em exibir as partidas de futebol (nomeadas de "peladas") com localização relativamente próximas ao usuário, podendo ser filtradas por sexo dos jogadores (masculino e/ou feminino) o que é um diferencial bem interessante pois nenhum outro aplicativo oferece essa opção, tipo de quadra (campo, society, salão, areia, rua), dias da semana e horário. Este aplicativo ainda possui alguns conceitos de redes sociais tais como a possibilidade de adicionar amigos e a criação e gerenciamento de grupos. Possibilita também a criação de partidas pelo usuário podendo

convidar seus amigos e/ou grupo, enquanto mantém um registro de partidas disputadas e gols marcados pelos jogadores. Pode ser considerado um aplicativo bem completo e balanceado, pois possui funcionalidades em diversas áreas, mas carece uma opção para confirmar a presença de companheiros, e como seu login é feito apenas através do Facebook ele acaba forçando com que todos os atletas tenham obrigatoriamente uma conta no Facebook e o aplicativo instalado, o que muitas vezes é quase impossível de se conseguir, se tornando inviável como solução para o autor.

2.1.2 FINTTA

O FINTTA pode ser considerado um dos aplicativos mais completos dentre os que possuem a proposta de organizar as partidas de um time de futebol, podendo ser considerado como uma solução que contempla diversos dos problemas apresentados anteriormente no capítulo 1. Contendo inúmeras funcionalidades que vão desde criação de partidas, jogadores e times (este último apenas via website), possibilidade de confirmar a presença ou ausência própria ou de companheiros até oferecer um mural onde os jogadores podem conversar. Juntamente com o Difora, oferece soluções para diferentes perfis de usuário, e apesar de possuir inúmeras funcionalidades, ele também não dá a opção ao usuário de compartilhar os dados da partida, uma das funcionalidades essenciais para a solução do problema proposto neste trabalho.

2.1.3 Minha pelada

O Minha pelada possui uma interface simplificada com enfoque para os eventos de uma partida, sendo possível realizar a contagem de gols e de tempo durante a mesma. Devido ao fato de manter o foco voltado para estes tipos de eventos, ele consegue satisfazer um pequeno nicho de usuários que necessitam de um aplicativo de monitoramento partida a partida. O Minha Pelada não possui algumas funcionalidades consideradas importantes como, por exemplo, confirmação de presença e compartilhamento de informações.

2.1.4 Peladeiros

Talvez o aplicativo mais conhecido dentre os supracitados, o Peladeiros possui uma abordagem um pouco diferente dos outros aplicativos para os mesmos problemas. O Peladeiros orienta a grande maioria de suas funcionalidades para que estejam relacionadas às partidas cadastradas. Apesar do Peladeiros possuir vários detalhes e cobrir a maioria das necessidades já comentadas, ele apresenta uma necessidade de ter os jogadores da equipe adversária também cadastrados, isso em uma partida entre amigos não seria empecilho, porém para os times de futebol amador que estão sempre jogando contra adversários de diferentes lugares este pré-requisito acaba tornando a utilização do aplicativo inviável.

2.1.5 Comparativo de funcionalidades

Aqui são apresentadas as principais funcionalidades de cada aplicativo onde, ao final, é feita uma tabela comparativa facilitando a visualização da abrangência das mesmas para cada aplicativo. Tais funcionalidades foram separadas em

- Confirmação de presença individual: engloba a funcionalidade do aplicativo prover a opção do usuário informar aos outros usuários se ele poderá comparecer à partida ou não;
- Confirmação de presença coletiva: engloba a funcionalidade do aplicativo prover a opção do usuário informar aos outros usuários se outro jogador poderá comparecer à partida ou não;
- Gerenciamento de jogadores: engloba as funcionalidades de criar, editar e remover um jogador da equipe;
- Gerenciamento de partidas: engloba as funcionalidades de criar, editar e remover uma partida da lista de partidas apresentadas ao usuário;
- Compartilhamento: engloba a funcionalidade de poder compartilhar as informações de uma partida para outra pessoa através de e-mail, SMS, Whatsapp entre outros;
- Localização: engloba a funcionalidade de identificar a localização de uma partida e transferir essa informação para um aplicativo de mapas automaticamente;
- Agenda: engloba a funcionalidade de transferir as informações de uma partida para um aplicativo de agenda automaticamente;

- Autenticação: consiste na possibilidade do jogador utilizar de um login para utilizar o aplicativo;
- Estatísticas: engloba a possibilidade de armazenar informações dos jogadores referente às partidas disputadas, tais como gols, assistências, cartões amarelos, entre outras;
- Comunicação: engloba a possibilidade dos usuários do aplicativo trocar mensagens seja por modo privado, murais, etc.

Tabela 2.1: My caption

	Difora	FINTTA	Minha pelada	Peladeiros	SeuTime
Confirmação de presença individual	✓	✓	✓	✓	✓
Confirmação de presença coletiva		✓		✓	✓
Gerenciamento de jogadores	✓	✓	✓	✓	✓
Gerenciamento de partidas	✓	✓	✓	✓	✓
Compartilhamento					✓
Localização	✓				✓
Agenda					✓
Autenticação	✓	✓		✓	
Estatísticas	✓	✓		✓	
Comunicação		✓		✓	

3 MODELAGEM E PROJETO DO APLICATIVO

Neste capítulo são abordadas as diretrizes e necessidades que guiaram o processo de desenvolvimento através de user stories, o modelo de entidades que foram utilizados, bem como as tecnologias escolhidas para construir o aplicativo

3.1 User stories

User Stories são frequentemente usadas no processo de desenvolvimento de alguma aplicação, pois nelas são trazidas informações concisas detalhando os objetivos que devem ser alcançados por uma determinada ação e o porquê delas. Desta forma acaba facilitando o entendimento e desenvolvimento das funcionalidades de uma aplicação. Neste trabalho é utilizado o formato: como um <tipo de usuário>, quero <algum objetivo>, para <alguma razão> (USER STORIES AND USER STORY EXAMPLES BY MIKE COHN 2016).

3.1.1 Criar jogador

"Como um usuário, quero poder criar um jogador para poder obter informação sobre sua presença em partidas

3.1.2 Excluir jogador

"Como um usuário, quero poder excluir um jogador para eliminar sua relação com a equipe"

3.1.3 Criar partida

"Como um usuário, quero poder criar uma partida para poder disponibilizar suas informações a todos usuários"

3.1.4 Excluir partida

"Como um usuário, quero poder excluir uma partida para remover sua entrada da lista de partidas"

3.1.5 Presença de jogadores

"Como um usuário, quero poder modificar a situação de presença de qualquer jogador perante uma partida para informar a todos sobre sua situação"

3.1.6 Visualizar localizacao

"Como um usuário, quero poder visualizar em qualquer aplicativo de mapas e localização a localização de uma partida para saber como chegar ao local da partida a partir da minha posição atual"

3.1.7 Agendar evento

"Como um usuário, quero poder criar um evento da partida em um aplicativo de agenda para me ajudar na organização"

3.1.8 Compartilhar informações

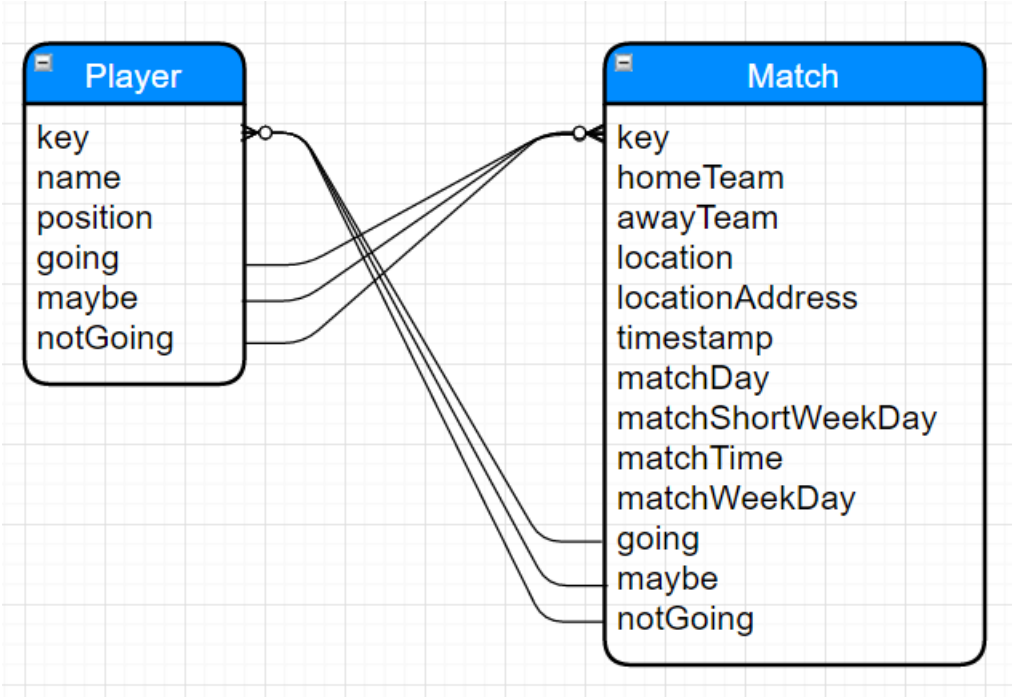
"Como um usuário, quero poder compartilhar os dados relacionado a uma partida do SeuTime em um aplicativo de agenda para me ajudar na organização"

3.2 Entidades

A modelagem de entidades baseada em estórias facilita o desenvolvimento da aplicação pois orienta como suas estruturas devem ser construídas, organizadas e relacionadas para prover um funcionamento conforme esperado. Tendo em vista que para este trabalho foi utilizado um banco de dados não-relacional, entidades de relação e outras menores não

são necessárias, portanto acabaram sendo incorporadas em duas entidades robustas. Na Figura 3.1 é possível de se observar a forma com que as entidades e seus relacionamentos foram projetadas. Por se tratar de um aplicativo que utiliza um banco NoSQL, foi necessário adaptar um diagrama de entidades e relacionamento para que ele pudesse refletir como realmente ocorre esse relacionamento. Uma explicação mais detalhada poderá ser encontrada nas subseções a seguir.

Figura 3.1: Diagrama de entidades e relacionamentos



3.2.1 Jogador

Essa entidade representa um jogador da equipe (Player), onde ela armazena o nome, a posição do jogador e uma relação das partidas em que ele confirmou presença, em que ele é dúvida e em que ele estará ausente. Com o objetivo de tornar o aplicativo mais simples e intuitivo ao usuário, várias informações consideradas extras como idade, apelido, telefone, entre outras, não foram incluídas no projeto. Todavia, em futuros trabalhos, podem ser facilmente adicionadas à entidade jogador.

key: chave única criada pelo Firebase ao inserir o dado no banco;

name: nome do jogador;

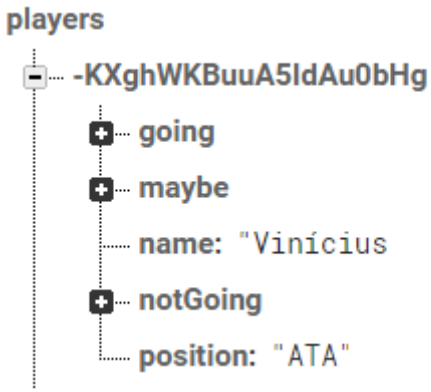
position: posição do jogador;

going: lista contendo as keys das partidas em que o jogador confirmou presença;

maybe: lista contendo as keys das partidas em que o jogador é dúvida;

notGoing: lista contendo as keys das partidas em que o jogador não poderá comparecer;

Figura 3.2: Entidade jogador



3.2.2 Partida

Diferentemente da entidade jogador, a entidade partida (Match) necessita de mais informações para que a aplicação funcione de modo desejado. Essa entidade guarda as informações dos 2 times que vão jogar, da localização, do horário da partida e uma relação dos jogadores que confirmaram presença, são dúvida ou estarão fora da partida.

key: chave única criada pelo Firebase ao inserir o dado no banco;

homeTeam: nome do time que vai jogar em casa;

awayTeam: nome do time que vai jogar como visitante;

location: nome do local da partida;

locationAddress: endereço da partida;

timestamp: timestamp contendo o dia e horário da partida;

matchDay: string contendo o dia da partida no formato dd/mm/aaaa;

matchShortWeekDay: string contendo uma abreviatura do dia da semana da partida;

matchTime: string contendo o horário que irá começar a partida;

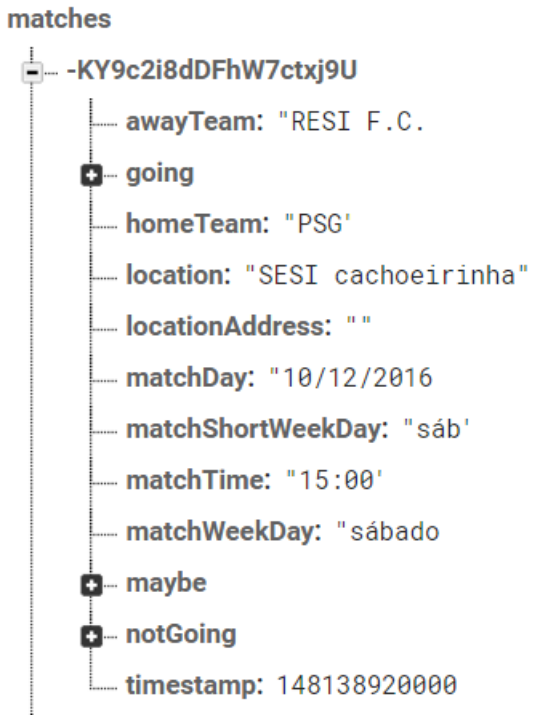
matchWeekDay: string contendo o dia da semana em que ocorrerá a partida;

going: lista contendo as keys dos jogadores que confirmaram presença na partida;

maybe: lista contendo as keys dos jogadores que serão dúvida para a partida;

notGoing: lista contendo as keys dos jogadores que serão desfalque na partida

Figura 3.3: Entidade partida



3.3 Tecnologias

Nesta seção serão apresentadas as características da plataforma Android que é voltada principalmente para dispositivos móveis com telas sensíveis ao toque assim como o sistema de banco de dados não-relacional pertencente à ferramenta Firebase, ambas atualmente desenvolvidas pela Google.

3.3.1 Plataforma Android

O sistema operacional Android é um software open source desenvolvido visando um vasto conjunto de dispositivos com características diferentes. Seu objetivo principal é criar uma plataforma onde desenvolvedores, operadoras e fabricantes possam inserir suas ideias e inovações resultando em um produto que realmente aprimora a experiência do usuário (ANDROID, 2016). Nesta subseção as principais camadas da arquitetura Android serão ilustradas juntamente com os principais conceitos de uma aplicação Android e sua anatomia.

3.3.1.1 Arquitetura

A arquitetura da plataforma Android é dividida em camadas onde cada uma é responsável por seus processos, essas camadas são:

Aplicações: camada onde são encontradas as aplicações que ficam a disposição do usuário, essas aplicações podem ser tanto os aplicativos nativos do sistema operacional como os aplicativos baixados pelo usuário, todos ficam localizados nesta camada.

Android framework: nesta camada se encontram as funções básicas do telefone tais como o sistema de localização, o sistema de chamadas, notificações, etc. A camada de aplicações interage diretamente com essa camada para utilizar essas funções básicas.

Bibliotecas nativas e Android runtime: neste nível de camadas do Android são encontradas os níveis de bibliotecas nativas e a camada de execução (Android runtime). A partir da versão 5.0, cada aplicativo executa seus próprios processos em uma instância própria de execução, o que aumenta a velocidade de resposta e reduz o consumo energético. Anteriormente à versão 5.0 o Android utilizava o Dalvik, uma máquina virtual Java otimizada para ambientes móveis. A camada runtime é responsável por transformar o código compilado pela máquina virtual em instruções nativas. Nesta mesma camada se encontram as bibliotecas nativas do Android, dentre elas a Opengl, Sqlite, entre outras.

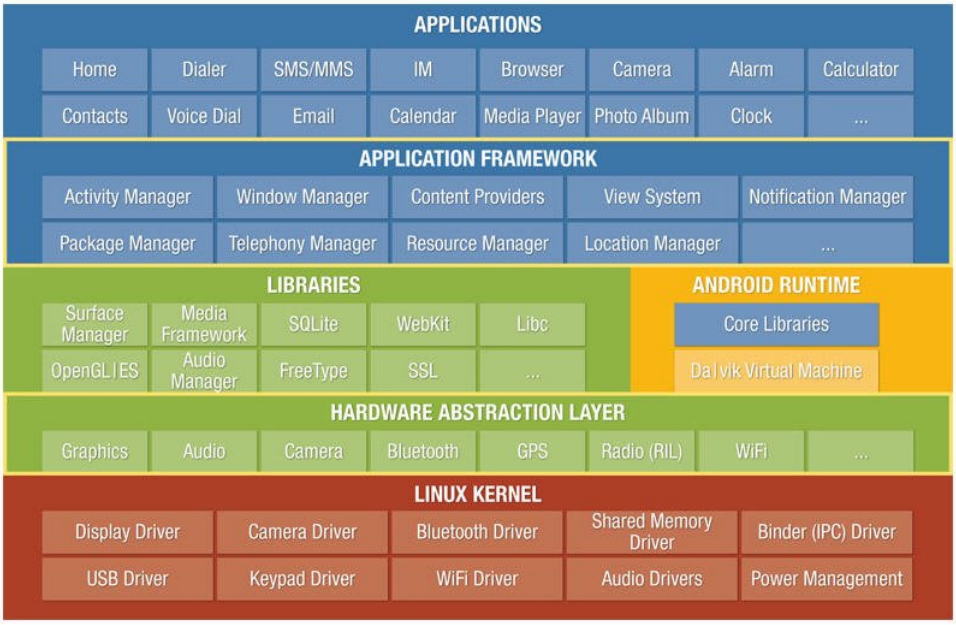
Camada de abstração de hardware: essa camada providencia as interfaces para acesso aos componentes de hardware do smartphone por meio de bibliotecas, assim quando a camada de framework faz uma requisição para utilizar um elemento como, por exemplo, o bluetooth, essa camada que trata este serviço.

Kernel Linux: como camada mais profunda, temos a do kernel do Linux, que nunca interage com os usuários ou desenvolvedores, mas é o coração de todo o sistema. Ela garante a maioria das funcionalidades essenciais tais como: segurança, gerenciamento de memória, configurações de segurança, etc.

3.3.1.2 Conceitos básicos e anatomia

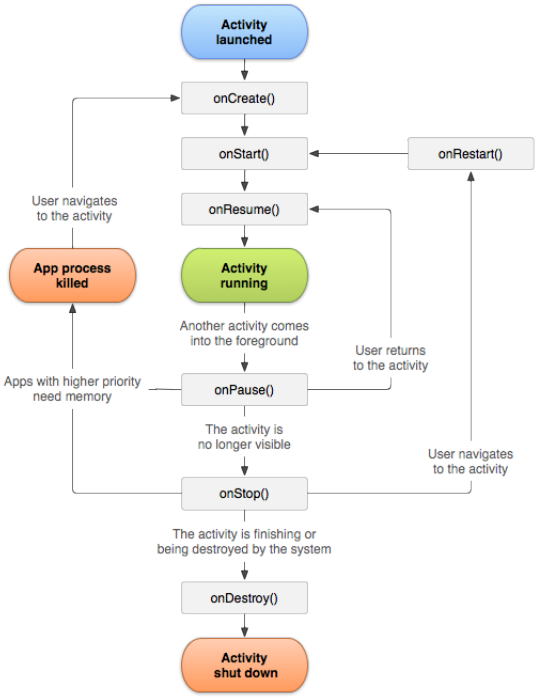
Nesta subseção iremos dissecar a anatomia de um aplicativo Android e destacar seus conceitos de funcionamento mais importantes. As aplicações Android são escritas utilizando a linguagem Java, e cada aplicativo tem seu próprio recipiente fechado de segurança, que o protege e o impede de executar funções fora de seu domínio. Um aplicativo Android possui quatro tipos de componentes, que serão descritos logo abaixo, são eles: atividades, serviços, provedores de conteúdo e receptores de transmissão.

Figura 3.4: Arquitetura Android. Fonte: (GITHUB DEVACADEMY, 2016)



Atividades: uma atividade é uma das classes mais importantes de um aplicativo Android, é ela que gerencia a interface visual apresentada ao usuário. Uma aplicação pode ter várias atividades e normalmente cada atividade representa uma funcionalidade do aplicativo. O entendimento do ciclo de vida de uma atividade é essencial para que o desenvolvedor possa construir um aplicativo que se comporte conforme desejado. Os estados em que uma atividade pode ser encontrada são:

Figura 3.5: Ciclo de vida de uma atividade. Fonte: (ANDROID DEVELOPERS, 2016)



created: estado inicial em que a atividade é criada e inicializa seus dados.

started: estado em que a atividade se encontra logo após sua criação ou então após o aplicativo ser reaberto depois de encerrado.

resumed: estado em que a atividade está sendo projetada na tela em primeiro plano e o usuário tem liberdade de interagir com ela.

paused: estado em que a atividade entra quando ela é parcialmente coberta, por exemplo, quando um menu ou notificação é aberto.

stopped: normalmente a atividade entra nesse estado quando ela é totalmente coberta, como por exemplo, por uma outra atividade.

destroyed: quando a atividade é encerrada ela entra nesse estado.

Sempre que uma atividade troca de estado alguns métodos da classe são chamados:

onCreate: chamado imediatamente após a criação da atividade e deve ser obrigatoriamente implementado, nela é definida o layout utilizado na atividade assim como seus atributos caso existam.

onResume: método chamado sempre que a atividade inicia (em seguida da onCreate) ou reinicia (vindo dos estados pause e created)

onPause: método chamado quando uma atividade é deixada para segundo plano, recomenda-se gravar as informações necessárias, pois apesar de ser possível a retomada da atividade, nem sempre isso acontece.

onStop: método chamado quando uma atividade é totalmente encoberta e vai para segundo plano, também se recomenda salvar as informações desejadas e desalocar recursos.

onDestroy: método chamado para realizar qualquer limpeza final antes da atividade ser completamente destruída.

Serviços: os serviços da plataforma Android executam tarefas de propósito geral, que ficam em segundo plano rodando para todos tipos de motivo. É um componente que não possui uma interface para/com o usuário, por exemplo um aplicativo de música pode executar um serviço para ficar tocando música mesmo que outro aplicativo esteja aberto em primeiro plano.

Receptores de transmissão broadcast: os receptores de transmissão broadcast de uma aplicação funcionam de modo que ela precise ficar rodando para que ela executa alguma instrução caso um evento específico ocorra, basta ela configurar um receptor de transmissão para esse determinado evento, que o sistema faz o tratamento e anuncia à aplicação sempre que ele ocorrer, fazendo com que as instruções configuradas no receptor

sejam executadas.

Provedores de conteúdo: os provedores de conteúdo cuidam da parte de gerenciamento de acesso a um conjunto de dados. Os provedores são projetados para serem utilizados por outras aplicações através de um cliente, podendo fazer com que duas aplicações diferentes compartilhem dos mesmos dados.

3.3.2 Banco de dados não-relacional

Nesta seção serão fornecidas informações sobre o banco de dados utilizado, no qual pertence a classe de bancos não relacionais (NoSQL), uma tecnologia de banco de dados projetada para suportar os requisitos de aplicações em nuvem e arquitetado para superar a escala, desempenho, modelo de dados e as limitações de bancos de dados relacionais (DATA SCIENCE ACADEMY 2016). O banco utilizado foi o Firebase Realtime Database que consistem em um banco de dados não-relacional localizado na nuvem fornecido pela Firebase (FIREBASE DATABASE 2016). Devido ao pequeno tamanho e complexidade do projeto, não havia necessidade da utilização de um banco NoSQL que possui como seu ponto forte a escalabilidade, porém devido ao grande crescimento na utilização de bancos NoSQL que tem se notado, e ao crescimento de tecnologias que se beneficiam com esse tipo de banco, tais como Big Data e Internet of Things, o autor deste trabalho achou que seria uma boa oportunidade para utilizar esse tipo de banco e acabou escolhendo um banco NoSQL para o projeto.

3.3.2.1 Firebase

O Firebase é uma plataforma para a construção de aplicativos mobile e web através de ferramentas e infra estruturas que visam ajudar desenvolvedores a construir aplicativos de qualidade (FIREBASE DATABASE 2016) onde são agrupados diversos serviços importantes tais como o sistema de análise (Firebase Analytics), sistema de autenticação de usuário (Firebase Auth), armazenamento (Firebase Storage), banco de dados (Firebase Realtime Database), hospedagem (Firebase Hosting) entre outros. Porém, para este trabalho foi utilizado apenas o serviço de banco de dados não relacional, o Firebase Realtime Database. Esse banco de dados nada mais é do que uma árvore JSON gigante em que todos seus dados estão armazenados nos nodos, o que facilita uma modelagem simples de dados. O maior benefício do Firebase Database Realtime é que ele já possui um sistema

de sincronização instantânea implementado, fazendo com que, caso ocorra uma modificação no banco, todos os aplicativos que tenham a referência daquele item, o atualizem automaticamente, ao invés de trabalhar com requisição e resposta normalmente utilizado em outros bancos.

3.3.2.2 FirebaseUI

Em conjunto com o Firebase Realtime Database também foi utilizado uma biblioteca FirebaseUI fornecida pela própria Firebase, que possibilita uma integração rápida e fácil entre elementos UI e APIs do Firebase como Firebase Database ou Firebase Authentication (FirebaseUI for Android — UI Bindings for Firebase), abstraindo algumas funções do lado do usuário e automatizando o processo de sincronização entre banco e a UI.

4 PROCESSO DE DESENVOLVIMENTO

Neste capítulo será descrito os passos que foram tomados durante o processo de desenvolvimento do aplicativo deste trabalho, nomeado de SeuTime. O processo de desenvolvimento foi feito sob a premissa de que o aplicativo seria utilizado por alguns integrantes de apenas uma equipe, motivo pelo qual não foi implementado um sistema de criação e associação de equipes de futebol. Durante o desenvolvimento do aplicativo foram escritas pouco mais de 1600 linhas de código, divididas entre 18 classes, nas quais se encontram tanto classes objeto, de baixa complexidade, quanto as atividades, onde as funcionalidades principais estão agrupadas.

4.1 Metodologia de desenvolvimento

Desde alguns anos atrás, o autor do trabalho faz parte de um time de futebol amador fundado em 2002, que joga regularmente aos sábados durante o período da manhã/tarde. Foi reparado que os jogos são marcados por um integrante que entra em contato com os outros times, programando a agenda do time através de uma planilha própria com as informações referentes às datas dos jogos. No início da semana que precede o jogo, essa informação é compartilhada através de um grupo no aplicativo Whatsapp e os integrantes copiam a informação, editam ela informando se estarão presentes ou não na partida e colam o novo texto editado no mesmo grupo. Apesar de ser uma solução funcional, ela não é nada prática, muitas vezes as confirmações se perdem no meio das conversas ou de cópias de uma "versão"antiga da mensagem que não está atualizada com todas confirmações dos jogadores. Com base nessa dificuldade foi notada a necessidade de uma plataforma onde os integrantes da equipe pudessem visualizar todas as partidas já marcadas com antecedência, confirmar ou não sua presença e visualizar a situação de seus companheiros sem precisar rolar inúmeras páginas de conversas em um aplicativo de chat. Juntamente com essa solução, foi idealizada algumas funcionalidades extras que melhorariam a experiência do usuário, como o gerenciamento de jogadores da equipe, integração com aplicativos de mapa para navegação, integração com aplicativos de agenda para criar lembretes e eventos, e a possibilidade de compartilhar em modo de texto as informações pertinentes à partida. Sabendo as características necessárias para o desenvolvimento do aplicativo, foram modeladas as interfaces e suas relações com as funcionalidades do aplicativo durante três semanas até que se tivesse um projeto consistente. Após a projeção

do aplicativo, ele teve sua construção dividida em sprints, espaços de tempo onde um conjunto de atividades deve ser executado (DESENVOLVIMENTO AGIL 2016), que variaram de duas a três semanas aproximadamente, exceto o primeiro no qual despendeu mais tempo, são eles:

Compreensão da plataforma Android: por se tratar de uma tecnologia que possui uma curva de aprendizado razoavelmente rápida, a primeira etapa foi organizada apenas para aprendizado e compreendimento da plataforma Android através de tutoriais, vídeos-aula, textos, etc. Essa etapa teve duração de 6 semanas aproximadamente;

Compreensão e configuração do banco de dados Firebase: apesar do banco de dados Firebase também ser uma tecnologia de fácil aprendizado, foi necessário despender um certo tempo para se ter um entendimento melhor de seu funcionamento. Por não se diferenciar de um banco não relacional tradicional, não foi necessário muito tempo para realizar a configuração de seu ambiente como no sprint anterior. Também foram utilizados tutoriais, vídeos-aula, textos, etc para ajudar na compreensão. Essa etapa teve duração de 2 semanas aproximadamente;

Criação do menu lateral, navegação entre suas opções e customização inicial: a primeira etapa de desenvolvimento real, teve como objetivo criar um aplicativo em que era possível abrir o menu lateral com opções que ao serem escolhidas, desencadeavam a função atrelada àquela opção, para mais tarde serem substituídas pelas devidas funcionalidades. Juntamente foi feita uma customização inicial como as cores, nome, temas, etc. Essa etapa teve duração de 2 semanas aproximadamente;

Criação e remoção de jogadores: foi a etapa de criação e remoção de jogadores, tendo em vista que na versão projetada para esse trabalho, as únicas informações que um jogador contém são seu nome e posição de preferência, dados que praticamente nunca se alteram, não foi implementada a funcionalidade de edição, já que se pode obter o mesmo resultado removendo o usuário a ser editado e adicionando-o novamente com o dado alterado. Essa etapa contou também com a ligação dessa funcionalidade a opção correta no menu lateral, e teve duração de 2 semanas;

Criação e remoção de partidas: foi a etapa de criação e remoção de partidas, utilizando um pensamento similar a criação e remoção de jogadores, essa funcionalidade também não tem a possibilidade de edição, já que depois de inseridas as informações da partida, dificilmente elas irão mudar, e caso mude sempre é possível remover uma partida e adicioná-la novamente com as mudanças feitas. Apesar de grande parte do código da criação e remoção de jogadores ter sido reutilizado, foi gasto algum tempo

personalizando a adição de partida, e adicionando algumas melhorias de interface. Essa etapa teve duração de um pouco menos de 2 semanas;

Lista de partidas e layout para visualização de uma partida nessa etapa foi criada a tela em que a lista de partidas adicionadas pelo usuário é projetada na tela dando a opção de selecionar uma e ir para a tela referente a ela e visualizar suas informações e confirmações. Porém apenas o layout dessa última tela foi feita nesta etapa, deixando as funcionalidades para outra etapa, esta etapa durou aproximadamente 2 semanas;

Implementação das funcionalidades da tela de uma partida: nessa etapa foi criada a funcionalidade que permite ao usuário alterar o status de presença dos jogadores de uma partida atualizando automaticamente no banco e em todos os smartphones que possuem o aplicativo instalado. As funcionalidades de mapa, agenda e compartilhamento tiveram sua tempo de implementação reduzido pois foi possível reaproveitar pequenas partes dos tutoriais feitos na primeira etapa. Essa etapa teve uma duração de 3 semanas;

Testes, correções de bugs e pequenos ajustes: nesta última etapa foi realizado diversos testes que apontaram alguns problemas no aplicativo que precisaram ser consertados, essas ações foram feitas algumas vezes até constatar que o aplicativo estava se comportando da forma adequada e não possuía nenhum bug aparente. Juntamente com esses testes e correções de bugs foram realizados alguns pequenos ajustes, tais como a mudança de alguns ícones, posicionamento de textos, mas também houve ajustes funcionais, como por exemplo a restrição de campos vazios nos cadastros de partidas e jogadores. Essa etapa teve uma duração de 2 semanas.

4.2 Plataforma de desenvolvimento

Para a construção desse projeto foi utilizado o Android Studio, uma IDE que fornece as ferramentas necessárias para a construção de aplicativos para todos tipos de aparelhos Android (ANDROID STUDIO 2016). Desenvolvido pela Google tendo como base a IDE IntelliJ IDEA da JetBrains. Dentre as diversas funcionalidades disponíveis no Android Studio serão ressaltadas apenas algumas mais relevantes:

Emulador: o emulador talvez seja a ferramenta mais interessante dessa IDE, pois ela faz com que o desenvolvedor não precise estar com um smartphone por perto para poder visualizar e testar seu aplicativo, ele também conta com diversos aparelhos de diversas configurações e tamanhos disponíveis para o teste do aplicativo em diferentes cenários;

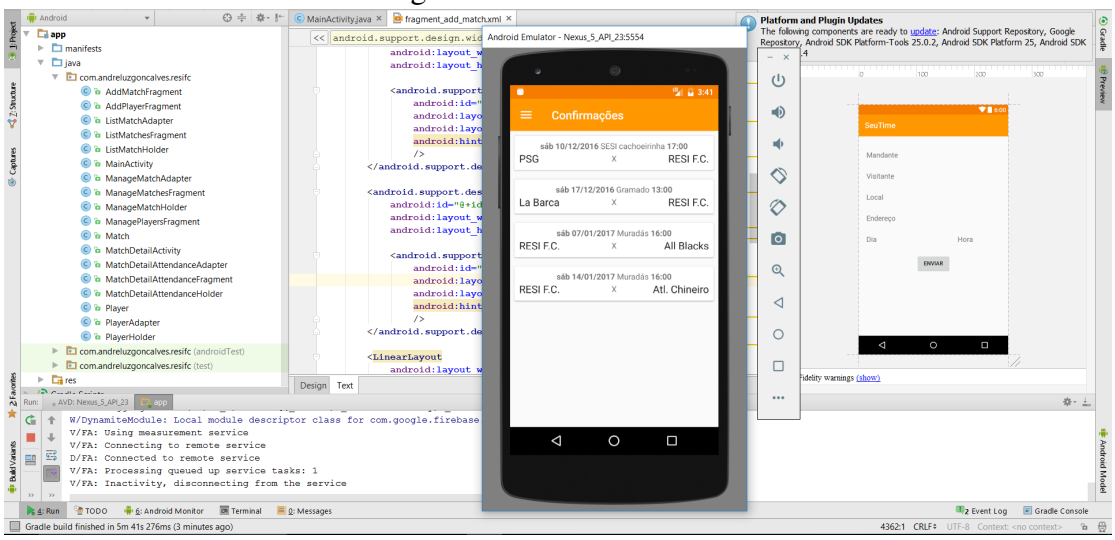
Editor de layout: ao trabalhar com o layout em XML, o Android Studio fornece

uma ferramenta que facilita a criação do layout utilizando um método "drag and drop", em que o usuário arrasta os elementos visuais em um posicionamento de sua preferência, e o editor já projeta esse layout para outros tamanhos de telas e gera o código para o mesmo;

Analisador APK: ferramenta que analisa o conteúdo e tamanho de componentes em seu aplicativo informando quais componentes podem ser alterados. Dentre outras funções ele ajuda a reduzir o tamanho de sua APK;

Editor de linguagens: ferramenta capaz de automatizar o processo de traduzir seu aplicativo para diversos idiomas, basta o usuário preencher um "dicionário"com as palavras e suas respectivas traduções e apontar para o identificador dessa palavra no código que o sistema do escolhe a versão correta.

Figura 4.1: Android Studio



4.3 Estrutura de classes e pacotes

O código fonte desse projeto pode ser dividido em 4 grupos nos quais todas as funcionalidades estão dividas por responsabilidades. Ao passo que as estruturas vão sendo explicadas, alguns trechos de códigos serão salientados para demonstrar como algumas funcionalidades destas estruturas funcionam para um melhor entendimento.

Figura 4.2: Estrutura de arquivos

Atividades: responsáveis por dividir o aplicativo em conjuntos de telas e funcionalidades normalmente com pouca ligação entre elas onde apenas uma é executada por vez. Neste trabalho foram utilizadas 2 atividades:

Figura 4.3: Inicialização da atividade

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FragmentTransaction tx = getSupportFragmentManager().beginTransaction();
    tx.replace(R.id.flContent, new ListMatchesFragment());
    tx.commit();

    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
    drawer.addDrawerListener(toggle);
    toggle.syncState();

    NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
    navigationView.setNavigationItemSelectedListener(this);
}
```

MainActivity e MatchDetailActivity A primeira é encarregada por controlar o menu lateral (Figura 4.4) e gerenciar o fragmento que vai aparecer na tela para o usuário. já a segunda trata das funcionalidades referente a uma partida, sendo responsável pelos eventos que ali ocorrem;

Figura 4.4: Menu lateral

```
@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
    int id = item.getItemId();
    FragmentManager fragManager;

    switch(id) {
        case R.id.nav_matches:
            ListMatchesFragment listMatchesFragment = new ListMatchesFragment();
            fragManager = getSupportFragmentManager();
            fragManager.beginTransaction().replace(
                    R.id.flContent,
                    listMatchesFragment
            ).commit();
            break;
        case R.id.nav_manage_players:
            ManagePlayersFragment managePlayersFragment = new ManagePlayersFragment();
            fragManager = getSupportFragmentManager();
            fragManager.beginTransaction().replace(
                    R.id.flContent,
                    managePlayersFragment
            ).commit();
            break;
        case R.id.nav_manage_matches:
            ManageMatchesFragment manageMatchesFragment = new ManageMatchesFragment();
            fragManager = getSupportFragmentManager();
            fragManager.beginTransaction().replace(
                    R.id.flContent,
                    manageMatchesFragment
            ).commit();
            break;
    }
    DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}
```

Fragmentos: neste trabalho os fragmentos foram utilizados como uma nova tela, porém o que os diferencia das atividades é que eles são chamados quando a tela a ser exibida (no caso o fragmento) possui um propósito similar ou referente a tela anterior. Os fragmentos podem ser agrupados por funcionalidades, são eles:

AddPlayerFragment e *AddMatchFragment* em que ambos consistem em uma tela com inputs de dados, que permitem ao usuário inserir jogadores e partidas respectivamente.

Figura 4.5: Fragmento de adição de jogador

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    final View view = inflater.inflate(R.layout.fragment_add_player, container, false);
    getActivity().setTitle("Adicionar Jogador");
    Button submitButton = (Button) view.findViewById(R.id.btn_player_submit);
    submitButton.setOnClickListener((v) → {
            EditText name = (EditText) view.findViewById(R.id.tiet_player_name);
            TextInputLayout tilName = (TextInputLayout) view.findViewById(R.id.til_player_name);
            Spinner position = (Spinner) view.findViewById(R.id.tiet_player_position);

            InputMethodManager inputManager = (InputMethodManager) getActivity().getSystemService(Context.INPUT_METHOD_SERVICE);
            inputManager.hideSoftInputFromWindow(getActivity().getCurrentFocus().getWindowToken(),
                    InputMethodManager.HIDE_NOT_ALWAYS);

            if(name.getText().toString().trim().equals("")) {
                tilName.setError("Campo obrigatório.");
            }else {
                tilName.setError(null);

                player.setName(name.getText().toString());
                player.setPosition(position.getSelectedItem().toString());
                DatabaseReference mDatabaseRef = FirebaseDatabase.getInstance().getReference().child("players");
                mDatabaseRef.push().setValue(player);
                Toast.makeText(getContext(), "Jogador adicionado", Toast.LENGTH_SHORT).show();
                FragmentManager fragmentManager = getFragmentManager();
                fragmentManager.popBackStack();
            }
    });
    return view;
}
```

ManagePlayersFragment, ManageMatchesFragment e ListMatchesFragment que são responsáveis pela tela de visualização de jogadores e a listas das partidas cadastradas no banco, o managematchfragment difere do listmatchfragment pois o primeiro apresenta a lista de partidas disponíveis para remoção e o segundo apresenta a lista de partidas para o usuário selecionar qual ele gostaria de visualizar.

MatchDetailAttendanceFragment Fragmento encontrado na tela de confirmação de presença em uma partida, sua função é apresentar a lista de jogadores para que o usuário possa confirmar a presença ou ausência do(s) jogadore(s);

Figura 4.6: Fragmento de visualização de partida

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View view = inflater.inflate(R.layout.fragment_match_detail_attendance, container, false);
        Bundle bundle = this.getArguments();
        matchKey = bundle.getString("MATCH_KEY");
        linearLayoutManager = new LinearLayoutManager(getActivity());
        matchDetailAttendanceRecyclerView = (RecyclerView) view.findViewById(R.id.match_detail_attendance_recycler_view);
        matchDetailAttendanceRecyclerView.setHasFixedSize(true);
        mDatabaseRef = FirebaseDatabase.getInstance().getReference().child("players");
        mMatchDetailAttendanceAdapter = new MatchDetailAttendanceAdapter(Player.class, R.layout.player_attendance_view,
MatchDetailAttendanceHolder.class, mDatabaseRef, getContext(), matchKey);
        matchDetailAttendanceRecyclerView.setLayoutManager(linearLayoutManager);
        matchDetailAttendanceRecyclerView.setAdapter(mMatchDetailAttendanceAdapter);

        return view;
    }

}
```

Adaptadores e Viewholders: são as estruturas que facilitam a vida do desenvolvedor no que diz respeito à preencher uma lista com um grupo de dados, o Firebase fornece um adaptador FirebaseUI que automatiza o processo de população de uma lista com dados de um banco de dados Firebase, enquanto os viewholders armazenam as infor-

mações da view de um item para poupar processamento na inflação de um próximo item na lista.

PlayerHolder, PlayerAdapter, ManageMatchHolder, ManageMatchAdapter, ListMatchHolder, ListMatchAdapter, MatchDetailAttendanceHolder e MatchDetailAttendanceAdapter Os view holders armazenam as informações das views de uma lista tendo a diferença que o playerholder e matchdetailattendanceholder armazenam as informações das células da lista de jogadores enquanto os managematchholder e listmatchholder armazenam as informações referentes às partidas. Já os adapters playeradapter e matchdetailattendanceadapter buscam as informações dos jogadores do banco e populam as listas;

Figura 4.7: Funcionamento do adaptador de jogadores

```
@Override
protected void populateViewHolder(final PlayerHolder viewHolder, Player model, int position) {
    viewHolder.playerName.setText(model.getName());
    viewHolder.playerPosition.setText(model.getPosition());
    mDatabaseRef = FirebaseDatabase.getInstance().getReference().child("matches");
    //viewHolder.itemView.setLongClickable(true);
    viewHolder.itemView.findViewById(R.id.player_btn_more).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(final View v) {
            PopupMenu popup = new PopupMenu(viewHolder.itemView.getContext(), v);
            popup.inflate(R.menu.manage);
            // This activity implements OnMenuItemClickListener .
            popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
                @Override
                public boolean onMenuItemClick(MenuItem item) {
                    switch (item.getItemId()) {
                        case R.id.cnt_menu_delete:
                            PlayerAdapter.this.getRef(viewHolder.getAdapterPosition()).removeValue();
                            return true;
                        default:
                            return false;
                    }
                }
            });
            popup.show();
        }
    });
}
```

Classes Objeto responsáveis por fornecer um elo entre o banco de dados e a aplicação dando a possibilidade de manipulação dos dados para o usuário.

Figura 4.8: Classe Player

```
public class Player {

    String name;
    String position;
    Map<String, Boolean> going;
    Map<String, Boolean> maybe;
    Map<String, Boolean> notGoing;

    public Player() {
        this.going = null;
        this.maybe = null;
        this.notGoing = null;
    }
```

Player e Match São classes criadas utilizando a técnica de mapeamento objeto-

relacional para facilitar o entendimento e desenvolvimento do projeto, elas são um "espelho"das entidades de jogadores e partidas do banco de dados, possuindo os getters e setters padrões. Na figura 4.9 pode se notar o método getAttendance que verifica a situação de presença de um dado jogador com a qualquer partida e retorna o recurso adequado para cada situação.

Figura 4.9: Método da classe Player

```
public int getAttendance(String matchKey){

    if(going != null && going.get(matchKey) != null){
        return R.drawable.going;
    }else if(maybe != null && maybe.get(matchKey) != null){
        return R.drawable.maybe;
    }else if(notGoing != null && notGoing.get(matchKey) != null) {
        return R.drawable.not_going;
    }
    return android.R.color.transparent;
}
```

5 FUNCIONAMENTO DO APLICATIVO

Neste capítulo serão apresentadas as telas do aplicativo juntamente com uma breve explicação de seu funcionamento.

5.1 Tela de lista de partidas

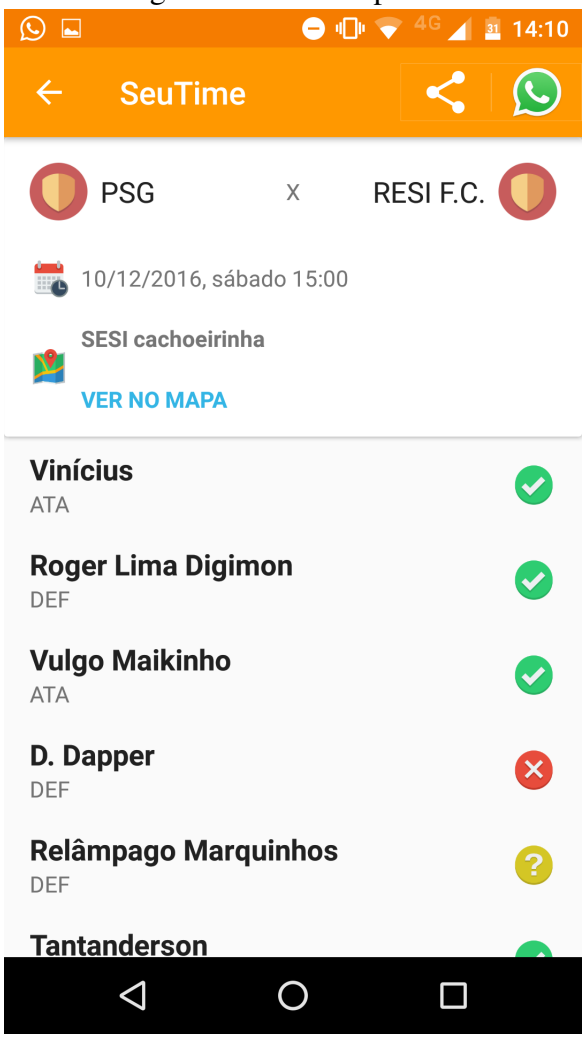
Figura 5.1: Lista de partidas

Esta é a tela em que todas as partidas cadastradas por qualquer jogador aparecem ordenadas pela data da partida, onde é possível visualizar rapidamente o adversário, o dia, horário da partida e o nome do local onde ela será realizada. Caso o usuário queira visualizar mais informações a respeito da partida ele pode clicar no card da partida desejada que uma nova tela com mais detalhes e funcionalidades será aberta. Essa também é a tela inicial e possui um menu lateral onde o usuário pode navegar para outras funções do

aplicativo.

5.2 Tela de partida

Figura 5.2: Tela de partida

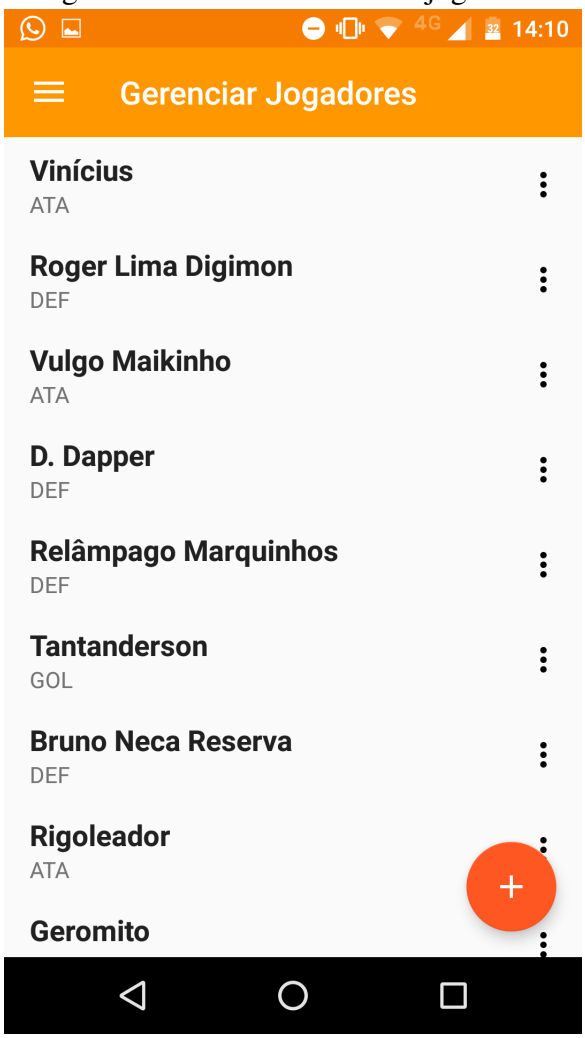


Essa é a tela para qual o usuário é direcionado quando ele clica em uma partida na tela principal, nesta tela o usuário consegue visualizar todas as informações pertinentes à partida, tais como o adversário, localização, horário e jogadores que vão ou não participar do jogo. Caso o usuário pressione no local onde se encontra as informações referentes ao horário da partida ele será levado para outro aplicativo de gerência de agendas para adicionar o evento da partida com as informações da mesma. Caso o usuário pressione na área da tela onde é informada a localização da partida o usuário será solicitado à abrir a localização através de um aplicativo de mapas onde ele poderá verificar com precisão a localização da partida. Mais abaixo na tela de partida encontram-se todos os jogadores

cadastrados na equipe para que possa ser informado aos outros integrantes a sua situação referente à presença da partida selecionada. E por fim na navegação superior à direita se encontra um botão de compartilhamento onde o usuário poderá compartilhar todas as informações da partida inclusive quais jogadores poderão comparecer ou não através de qualquer aplicativo que possa enviar um bloco de texto.

5.3 Tela de gerenciamento de jogadores

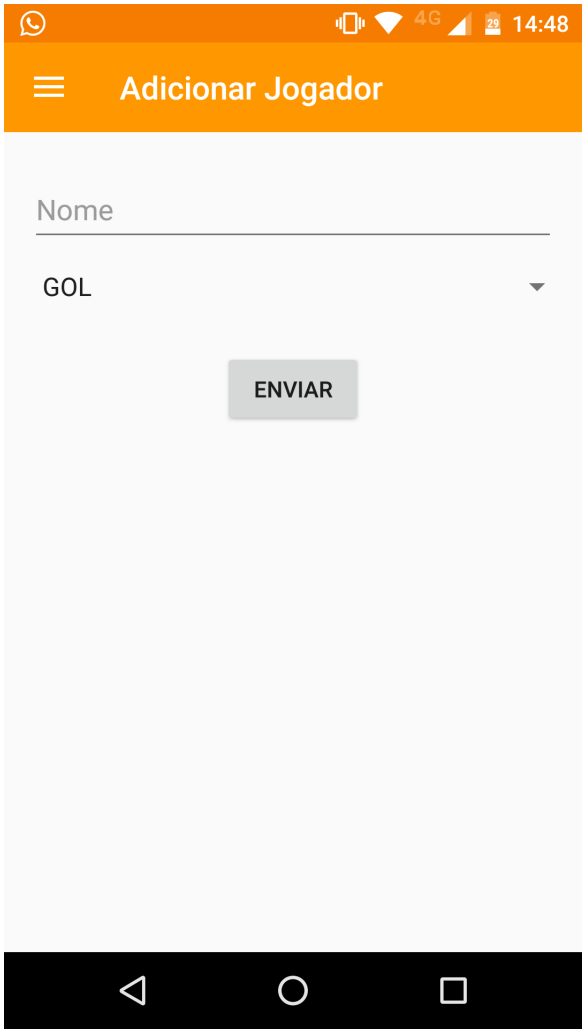
Figura 5.3: Gerenciamento de jogadores



Nesta tela temos o gerenciamento de jogadores da equipe, visualmente temos uma lista contendo todos os jogadores cadastrados, onde o usuário, caso queira, pode pressionar o botão com um "+"no canto inferior direito para adicionar um novo jogador ou então pressionar no botão à direita de cada jogador para abrir um menu dando a opção de remoção do mesmo.

5.4 Tela de adição de jogadores

Figura 5.4: Inserção de jogadores

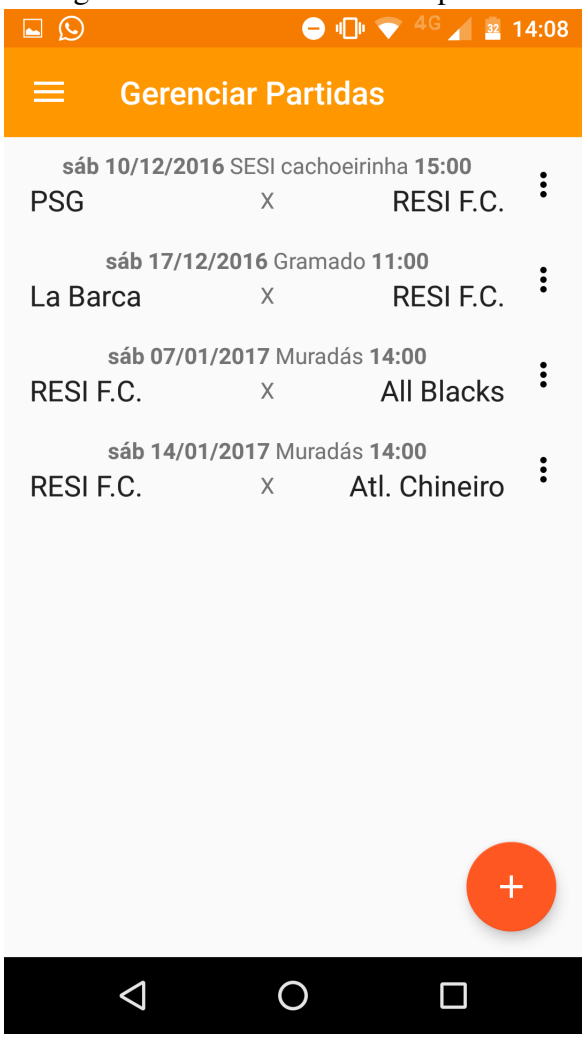


Essa tela é utilizada para adição de jogadores na equipe, existe um formulário contendo 2 campos, sendo um para o nome do jogador e outro para sua posição de preferência, onde depois de preenchidos o usuário será incorporado à equipe.

5.5 Tela de gerenciamento de partidas

Nesta tela temos o gerenciamento de partidas da equipe, visualmente temos uma lista contendo todas as partidas cadastradas, onde o usuário, caso queira, pode pressionar o botão com um "+"no canto inferior direito para adicionar uma nova partida ou então pressionar no botão à direita de cada partida para abrir um menu dando a opção de remover a mesma.

Figura 5.5: Gerenciamento de partidas



5.6 Tela de adição de partida

Essa tela é utilizada para adição de partidas, existe um formulário contendo diversos campos, onde o usuário vai preencher com as informações necessárias para o cadastro da partida no banco. No momento em que uma partida for cadastrada ela já aparecerá automaticamente nos outros dispositivos que possuem o mesmo aplicativo, onde os usuários já poderão verificar a localização ou confirmar presença por exemplo.

Figura 5.6: Inserção de partidas

5.7 Requisitos do sistema

Nesta seção serão descritos os requisitos mínimos necessários para que o aplicativo funcione sem maiores problemas

Dispositivo móvel O aplicativo precisa ser instalado em um dispositivo móvel (tablet ou celular) que utiliza o sistema operacional Android.

Android SDK16 Para o aplicativo obter sucesso em sua instalação e execução é necessário que a versão do Android instalada no celular seja no mínimo a Jelly Bean (4.1.x).

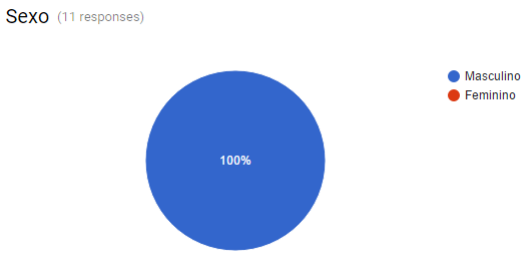
Acesso à rede Para o aplicativo funcionar corretamente sem nenhuma anormalidade, é necessário que o usuário possua uma conexão ativa com a internet, seja ela por 3G/4G ou Wi-Fi, pois as operações de acesso ao banco não podem ser executadas caso essa conexão não exista.

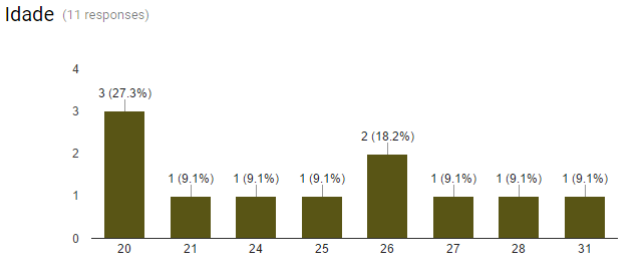
6 AVALIAÇÃO DO APLICATIVO DESENVOLVIDO

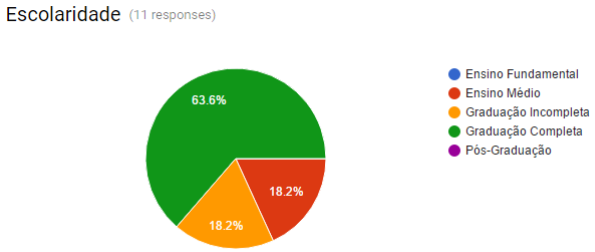
Neste capítulo será feita uma análise dos resultados obtidos através de um questionário feito após os testes de utilização com onze (11) indivíduos. O objetivo desta análise é ter um feedback sobre algumas funcionalidades do aplicativo, facilidade de uso e necessidades atendidas e não atendidas, para poder aprimorar o aplicativo futuramente, e verificar se ele conseguiu cumprir o objetivo pelo qual ele foi feito. Primeiramente os usuários recebiam um documento em que eram passadas algumas instruções sobre como obter o aplicativo, como instalar e o que os usuários deveriam tentar executar no aplicativo. Depois de testar, os usuários deveriam responder a um questionário feito através do Google Forms e dividido em 3 categorias, a primeira faz uma identificação do perfil do usuário fazendo perguntas com relação à idade, conhecimento de informática, quantidade de vezes que pratica futebol em um mês, etc para ter um peso maior as respostas dos potenciais usuários do aplicativo. Na segunda parte do questionário são feitas perguntas com relação à execução das funcionalidades, às dificuldades de encontrar as funcionalidades, etc. Os resultados dessa parte podem ser utilizados para uma possível remodelagem do aplicativo visando melhorar a experiência do usuário. Na terceira e última parte do questionário é apresentada uma área para que o candidato possa dar suas sugestões, críticas ou simplesmente deixar alguma opinião a respeito do aplicativo.

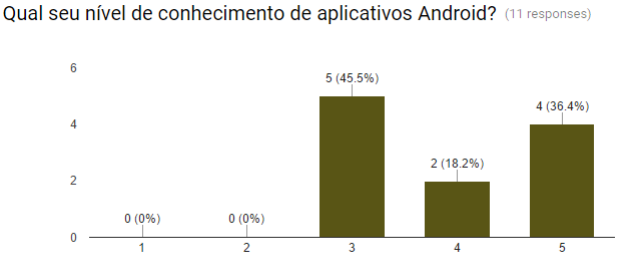
6.1 Identificação de perfil

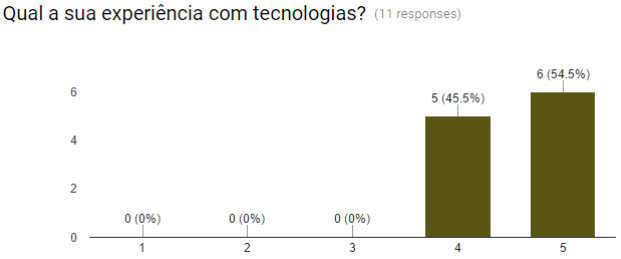
Com base nas respostas da primeira parte do questionário, observa-se que o perfil dos usuários que testaram o aplicativo são na totalidade homens, que variam de 20 a 31 anos com escolaridade entre ensino médio e graduação, justamente o perfil encontrado comumente nas equipes de futebol amador.



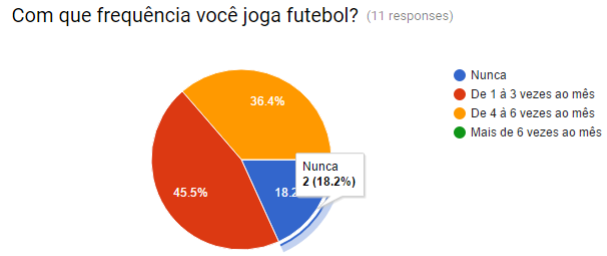


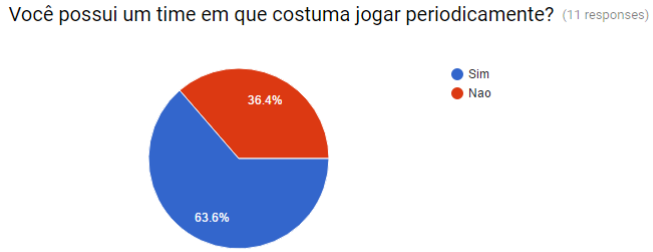






Devido a todos os usuários responderem que possuem conhecimento no mínimo razoável com Android e bastante experiência com tecnologia, quaisquer problemas ou dificuldades que os usuários enfrentaram foram considerados "falha"no aplicativo ao invés de falta de entendimento do usuário.





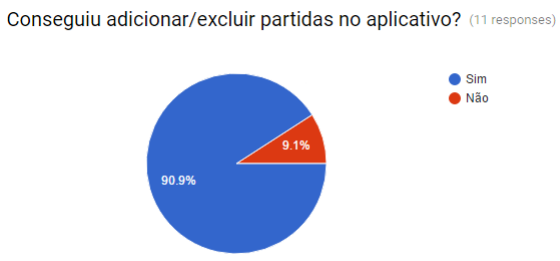
Dos 11 usuários que responderam o questionário apenas 2 não jogam futebol. Desses 9 que jogam, apenas 2 não possuem uma equipe em que jogam regularmente, esses dados reforçam mais ainda a idéia da necessidade de um aplicativo com os propósitos do SeuTime.

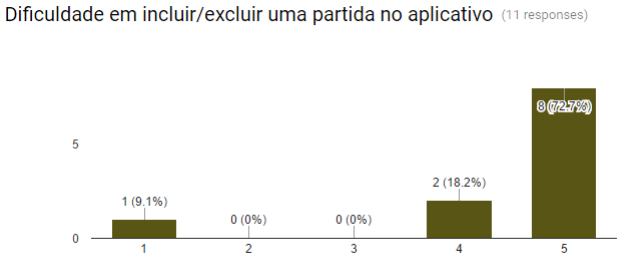
6.2 Experiência do usuário

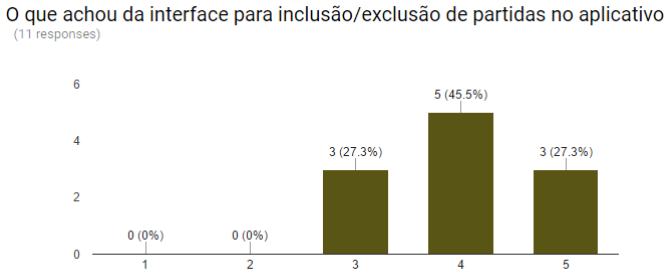
Aqui será observada as respostas que foram dadas na segunda parte do questionário, que diz respeito à experiência do usuário com o aplicativo ao tentar executar as tarefas pedidas nas instruções do questionário.

Adicionar e excluir partidas

Respostas sobre a tarefa em que foi pedido para o usuário tentar adicionar e excluir partidas no aplicativo.

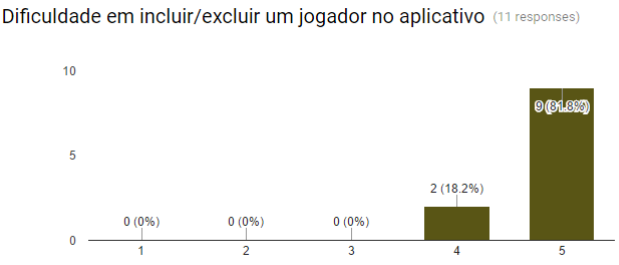


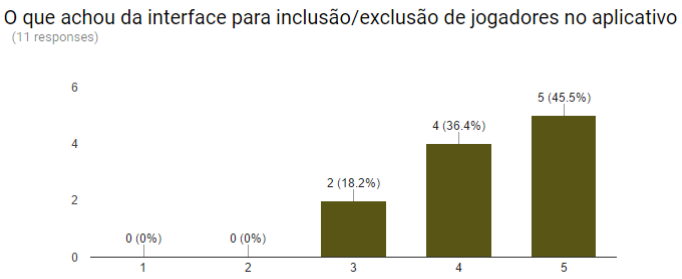




Adicionar e excluir jogadores

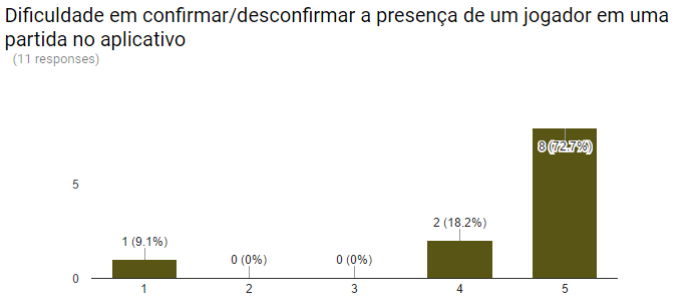
Respostas sobre a tarefa em que foi pedido para o usuário tentar adicionar e excluir jogadores no aplicativo.

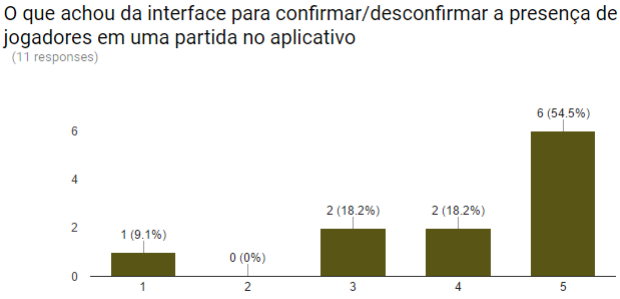




Confirmar e desconfirmar presença de jogadores em uma partida

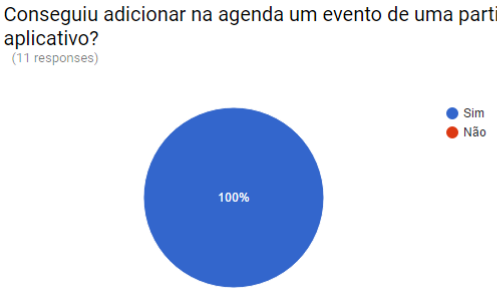
Respostas sobre a tarefa em que foi pedido para o usuário tentar confirmar e desconfirmar sua presença em uma partida no aplicativo.

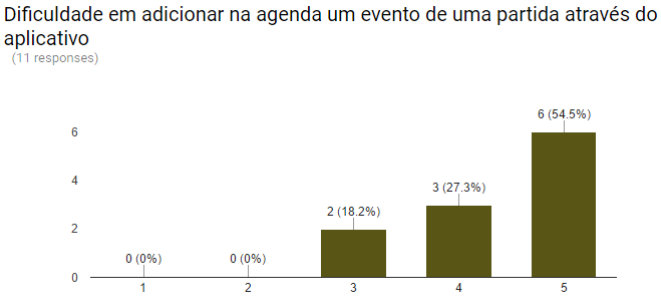


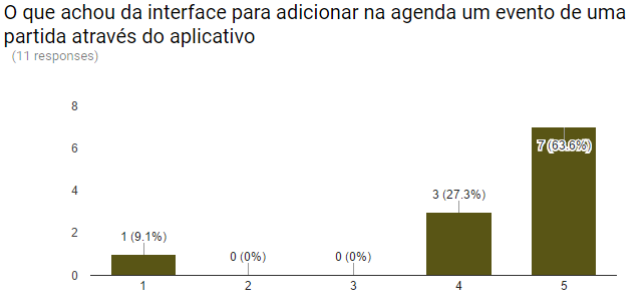


Adicionar evento de uma partida na agenda

Respostas sobre a tarefa em que foi pedido para o usuário tentar adicionar um evento de uma partida em sua agenda através do aplicativo.

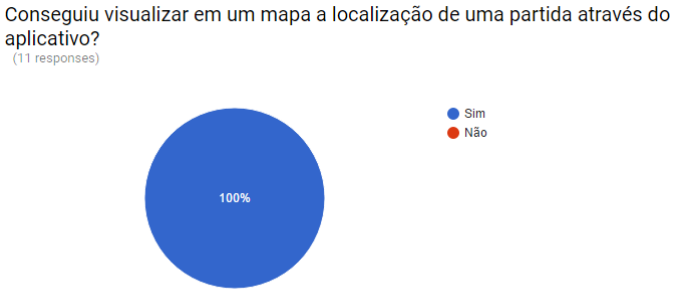


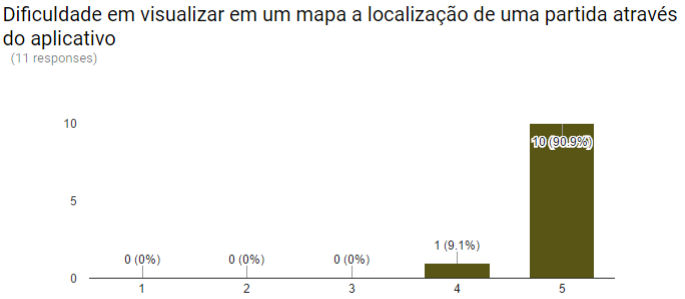


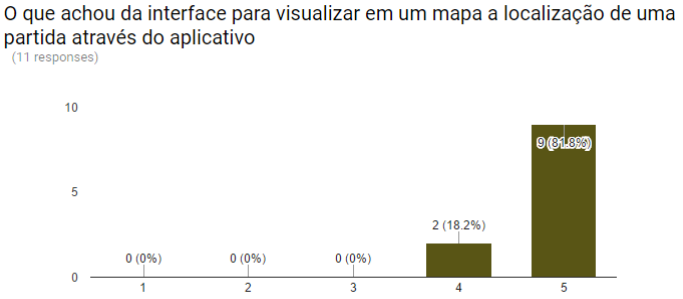


Visualizar em um mapa a localização de uma partida

Respostas sobre a tarefa em que foi pedido para o usuário tentar abrir em um mapa a localização da partida através do aplicativo.

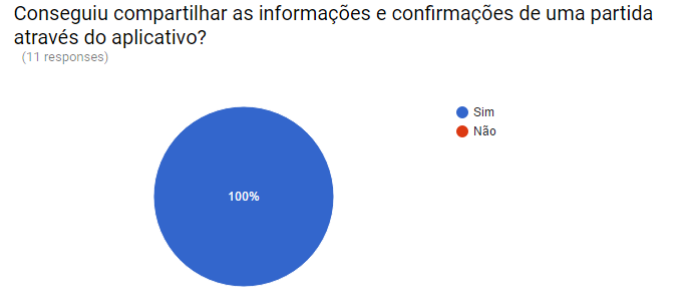


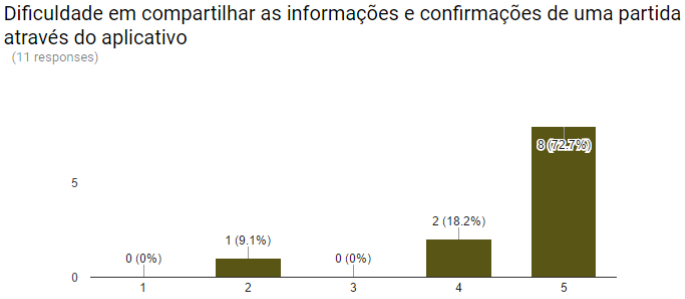


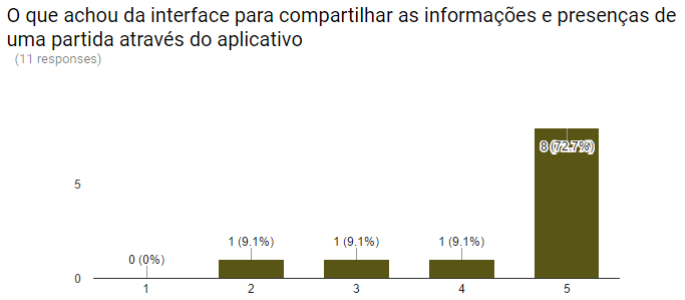


Compartilhar dados de uma partida

Respostas sobre a tarefa em que foi pedido para o usuário tentar compartilhar as informações referente a uma partida.







Ao observar as respostas obtidas, é possível perceber que de um modo geral os usuários conseguiram utilizar o aplicativo e utilizar suas funcionalidades sem maiores problemas, houve apenas alguns casos isolados que se comparados ao escopo geral, não chega a causar preocupações.

6.3 Sugestões e opiniões

Após a finalizar a segunda parte do questionário, é possível que o usuário deixe sua sugestão para melhoria do aplicativo e que ele acha sobre o aplicativo e sua utilidade. Foram obtidas 5 sugestões e 5 feedbacks em relação a utilidade do aplicativo.

Algumas das sugestões dadas eram mudanças e/ou funcionalidades que mudariam um pouco o propósito do aplicativo, assim como também se obteve alguns pedidos de melhoria na interface e correção de alguns bugs, como por exemplo: *"A tela principal do aplicativo poderia ser mais organizada, mais limpa. Poderiam haver filtros para classificar as partidas por times, locais, horários, jogadores, etc."*.

Com relação às opiniões teve-se um parecer bem positivo com todas as 5 respostas sendo positivas, algumas apenas elogiaram, já outras demonstraram a utilização do aplicativo como uma possível solução, como por exemplo: *"É uma ótima idéia e com a correção de alguns problemas e algumas novas funcionalidades poderá dar muito certo.*

Parabéns, André." e *"Creio que o aplicativo seja bastante útil. Deveria ser mais restrito a grupos, não tão aberto ao público. Poderia ter uma mecânica de confirmação só para o criador da partida e o respectivo jogador."*.

7 CONCLUSÃO

Este trabalho teve como objetivo principal apresentar um aplicativo que possa se tornar uma alternativa a outros métodos não tão simples para que participantes de um clube de futebol possam informar seus companheiros a situação de sua presença nas partidas agendadas e possam também visualizar essas informações acerca de seus companheiros. Apesar de carecer de algumas funcionalidades extras tais como os sistemas de estatísticas e pagamento que poderiam ser implementadas para agrupar cada vez mais a administração de uma equipe de futebol e melhorar a experiência do usuário, o aplicativo conseguiu executar o objetivo para que foi construído com êxito. O aplicativo foi apresentado para alguns integrantes da equipe em que o autor do trabalho participa e solicitado uma resposta sobre uma possível implantação do aplicativo como método principal de confirmação, as respostas foram em sua grande maioria positiva juntamente com algumas sugestões para novas funcionalidades que ajudariam o trabalho de alguns integrantes como por exemplo o financeiro caso fosse adicionada uma opção para marcar quem pagou e qual quantia foi paga em cada partida. O autor do trabalho pretende continuar trabalhando em cima deste projeto e incrementando-o para que futuramente sua equipe e quem sabe algumas outras utilizem dele para descomplicar a administração das mesmas.

7.1 Trabalhos Futuros

Sabendo das inúmeras vertentes que existem em volta de uma partida de futebol e de um time, fica fácil enxergar possíveis melhorias para um futuro trabalho em cima deste aplicativo. Algumas funcionalidades consideradas importantes pelo autor são:

Sistema de estatísticas: um sistema em que o usuário possa preencher dados de cada partida como gols, assistências, cartões, jogadores que atuaram, etc. Além destas funcionalidades um local onde o usuário possa visualizar um somatório desses dados dando a possibilidade de ver estatísticas individuais de jogadores ou estatísticas gerais da equipe.

Sistema financeiro: um sistema onde um grupo de usuários pertencentes ao financeiro possam marcar quando e quanto cada jogador tem pago e assim verificar mais facilmente a situação financeira da equipe. Para essa funcionalidade seria interessante implementar algum tipo de senha ou autenticação para que apenas alguns usuários possam alterar esses dados, evitando qualquer eventual problema.

Além das funcionalidades citadas acima existem ainda algumas outras que também poderiam ser implementadas porém não consideradas tão importantes a curto prazo pelo autor, como por exemplo implementar a possibilidade de cadastrar "campos"existentes, possibilitar a criação de times, assim diversas pessoas de times diferentes poderiam utilizar o aplicativo sem conflito de dados, juntamente com a agenda de cada time com intuito de facilitar a marcação de partidas.

REFERÊNCIAS

ANDROID DEVELOPERS. <https://developer.android.com/guide/index.html> Acesso em outubro de 2016.

GITHUB DEVACADEMY. <https://github.com/devacademy/android-fundamental-one/> Acesso em outubro de 2016

FIREBASE DATABASE. <https://firebase.google.com/docs/database/> Acesso em outubro de 2016.

SIMPLE DEVELOPER. <http://simpledeveloper.com/android-architecture/> Acesso em outubro de 2016.

ANDROIDHIVE. <http://www.androidhive.info/2016/01/android-working-with-recycler-view/> Acesso em outubro de 2016.

HIGHLY SCALABLE. <https://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/> Acesso em outubro de 2016.

TUTORIALSPOINT. <https://www.tutorialspoint.com/android/android$_a citivities.htm >$ $Acessoemoutubrode 2016.$

FirebaseUI for Android — UI Bindings for Firebase. <https://github.com/firebase/FirebaseUI-Android> Acesso em novembro de 2016.

GOOGLE. Play Store. (UBER) <https://play.google.com/store/apps/details?id=com.ubercabhl=en> Acesso em novembro de 2016.

WHATSAPP. <https://www.whatsapp.com/> Acesso em novembro de 2016.

ANDROID. <https://source.android.com/> Acesso em novembro de 2016.

CONMEBOL. <http://www.conmebol.com/pt-br/content/futebol-uma-boa-maneira-de-cuidar-da-saude> Acesso em novembro de 2016.

DIFORA. <http://difora.com.br/> Acesso em novembro de 2016.

FINTTA. <https://www.fintta.com/> Acesso em novembro de 2016.

GOOGLE. Play Store. (MINHA PELADA) <https://play.google.com/store/apps/details?id=minhape $pt_B R > Acessoemnovembrode 2016.$

GOOGLE. Play Store. (PELADEIROS) <https://play.google.com/store/apps/details?id=br.com.pela Acesso em novembro de 2016.

USER STORIES AND USER STORY EXAMPLES BY MIKE COHN. <https://www.mountaingoat stories> Acesso em novembro de 2016.

DESENVOLVIMENTO AGIL. <http://www.desenvolvimentoagil.com.br/scrum/> Acesso em novembro de 2016.

ANDROID STUDIO. <https://developer.android.com/studio/index.html> Acesso em novembro de 2016.

DATA SCIENCE ACADEMY. <http://datascienceacademy.com.br/blog/2016/quando-utilizar-rdbms-ou-nosql/> Acesso em novembro de 2016

ANEXO I

7.2 Instruções para teste

Instruções para teste do aplicativo SeuTime

Olá, primeiramente muito obrigado por me disponibilizar um pouco do seu tempo para me ajudar a concluir este projeto. Neste documento vão estar listadas as instruções necessárias para poder executar um teste completo do aplicativo SeuTime.
O primeiro passo é instalar o aplicativo que pode ser baixado nesse link (http://bit.ly/2h5O0hg) em um smartphone que utiliza um sistema android JellyBean (4.1.x) ou superior. Com o aplicativo instalado, abrir ele e tentar executar as tarefas abaixo:
-Adicionar alguns jogadores
-Adicionar algumas partidas
-Confirmar a presença de alguns jogadores
-Modificar o situação referente à presença de alguns jogadores em uma partida
-Abrir a localização de uma partida em algum app de mapas através do aplicativo
-Adicionar na sua agenda um evento referente à uma partida através do aplicativo
-Compartilhar as informações de uma partida por um meio de sua preferência através do aplicativo
-Remover alguns usúarios
-Remover algumas partidas
Depois de concluir essas tarefas pode utilizar o aplicativo um pouco mais se desejar e assim que terminar por favor preencher a pesquisa contida no link:
https://goo.gl/forms/Mpsw1l5APp2x5CWs1

PS: Se possível realizar esses testes em objetos criados por você mesmo, pois talvez exista outra pessoa realizando testes ao mesmo tempo.

Muito Obrigado,
André Gonçalves

ANEXO II

7.3 Questionário pós-teste

Questionário aplicativo SeuTime

Por favor responder este questionário após concluir as instruções contidas no documento: https://docs.google.com/document/d/15Y-cKHUCt2ZsA49s1a4UsPus59a0vD1rZRNn8ZFX59E/edit?usp=sharing

* Required

1. **Email address** *

2. **Nome** *

3. **Sexo** *
   Mark only one oval.
   - ◯ Masculino
   - ◯ Feminino

4. **Idade** *

5. **Escolaridade** *
   Mark only one oval.
   - ◯ Ensino Fundamental
   - ◯ Ensino Médio
   - ◯ Graduação Incompleta
   - ◯ Graduação Completa
   - ◯ Pós-Graduação

6. **Qual a sua experiência com tecnologias?** *
   Mark only one oval.

	1	2	3	4	5	
Pouco Experiente	◯	◯	◯	◯	◯	Muito Experiente

7. **Qual seu nível de conhecimento de aplicativos Android?** *
   Mark only one oval.

	1	2	3	4	5	
Muito Ruim	◯	◯	◯	◯	◯	Muito Bom

8. **Com que frequência você joga futebol?** *
Mark only one oval.

- Nunca
- De 1 à 3 vezes ao mês
- De 4 à 6 vezes ao mês
- Mais de 6 vezes ao mês

9. **Você possui um time em que costuma jogar periodicamente?** *
Mark only one oval.

- Nao
- Sim

10. **Conseguiu adicionar/excluir partidas no aplicativo?** *
Mark only one oval.

- Sim
- Não

11. **Dificuldade em incluir/excluir uma partida no aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Dificil						Muito Fácil

12. **O que achou da interface para inclusão/exclusão de partidas no aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Complicada						Muito Amigável

13. **Conseguiu adicionar/excluir jogadores no aplicativo?** *
Mark only one oval.

- Sim
- Não

14. **Dificuldade em incluir/excluir um jogador no aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Dificil						Muito Fácil

15. **O que achou da interface para inclusão/exclusão de jogadores no aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Complicada						Muito Amigável

16. **Conseguiu confirmar/desconfirmar presença de jogadores em partidas no aplicativo?** *
Mark only one oval.

- ◯ Sim
- ◯ Não

17. **Dificuldade em confirmar/desconfirmar a presença de um jogador em uma partida no aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Dificil	◯	◯	◯	◯	◯	Muito Fácil

18. **O que achou da interface para confirmar/desconfirmar a presença de jogadores em uma partida no aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Complicada	◯	◯	◯	◯	◯	Muito Amigável

19. **Conseguiu adicionar na agenda um evento de uma partida através do aplicativo?** *
Mark only one oval.

- ◯ Sim
- ◯ Não

20. **Dificuldade em adicionar na agenda um evento de uma partida através do aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Dificil	◯	◯	◯	◯	◯	Muito Fácil

21. **O que achou da interface para adicionar na agenda um evento de uma partida através do aplicativo** *
Mark only one oval.

	1	2	3	4	5	
Muito Complicada	◯	◯	◯	◯	◯	Muito Amigável

22. **Conseguiu visualizar em um mapa a localização de uma partida através do aplicativo?** *
Mark only one oval.

- ◯ Sim
- ◯ Não

23. **Dificuldade em visualizar em um mapa a localização de uma partida através do aplicativo** *

Mark only one oval.

	1	2	3	4	5	
Muito Dificil	◯	◯	◯	◯	◯	Muito Fácil

24. **O que achou da interface para visualizar em um mapa a localização de uma partida através do aplicativo** *

Mark only one oval.

	1	2	3	4	5	
Muito Complicada	◯	◯	◯	◯	◯	Muito Amigável

25. **Conseguiu compartilhar as informações e confirmações de uma partida através do aplicativo?** *

Mark only one oval.

- ◯ Sim
- ◯ Não

26. **Dificuldade em compartilhar as informações e confirmações de uma partida através do aplicativo** *

Mark only one oval.

	1	2	3	4	5	
Muito Dificil	◯	◯	◯	◯	◯	Muito Fácil

27. **O que achou da interface para compartilhar as informações e presenças de uma partida através do aplicativo** *

Mark only one oval.

	1	2	3	4	5	
Muito Complicada	◯	◯	◯	◯	◯	Muito Amigável

28. **Sugestões para melhorias e/ou funcionalidades extras.**

.....

.....

.....

.....

29. **Deixe sua opinião com relação ao aplicativo e sua utilidade.**

.....

.....

.....

.....

☐ Send me a copy of my responses.
