UNIVERSIDADE FEDERAL DE ALFENAS - MG

ÁLISSON JOSÉ OLIVEIRA DE FARIA

APLICATIVO ANDROID PARA ACOMPANHAMENTO DE CAMPEONATOS DE FUTEBOL AMADOR

Alfenas/MG

2017

ÁLISSON JOSÉ OLIVEIRA DE FARIA

APLICATIVO ANDROID PARA ACOMPANHAMENTO DE CAMPEONATOS DE FUTEBOL AMADOR

Projeto de pesquisa apresentado ao curso de Ciência da Computação da Universidade Federal de Alfenas como requisito parcial para obtenção do Título de Bacharel em Ciência da Computação. Orientador: Prof. Flávio Barbieri Gonzaga.

Alfenas/MG

2017

ÁLISSON JOSÉ OLIVEIRA DE FARIA

APLICATIVO ANDROID PARA ACOMPANHAMENTO DE CAMPEONATOS DE FUTEBOL AMADOR

A Banca examinadora abaixo-assinada aprova a monografia apresentada como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação pela Universidade Federal de Alfenas.

Aprovada em:

Profº.
Instituição: Assinatura:

Profº.
Instituição: Assinatura:

Profº.
Instituição: Assinatura:

Alfenas/MG

2017

DEDICATÓRIA

Dedico este trabalho a toda minha família: Maria José de Oliveira Faria, Dagilberto José de Faria e Anderson Oliveira de Faria, pelo total apoio e confiança que tiveram em mim durante esse caminho. Em especial a minha namorada Daniela de Carvalho Batista que participou e ajudou durante todo desenvolvimento desse trabalho. E a Deus, meu muito obrigado por mais esta conquista.

AGRADECIMENTOS

Agradeço primeiramente aos professores responsáveis pela minha formação e por todos os conhecimentos que adquiri ao longo destes anos. Agradeço principalmente ao meu orientador Flávio Barbieri que colaborou com desenvolvimento deste trabalho. Agradeço ainda ao Leornado Ciscon e Michelle Andrade que me proporcionaram grandes experiências profissionais.

RESUMO

Neste trabalho foi desenvolvido um aplicativo de Gerenciamento de Campeonatos de Futebol Amador. O sistema visa facilitar a gestão, inscrição e o acompanhamento de competições esportivas como futebol de campo, futsal e futebol society. Todas as etapas necessárias para se criar um campeonato podem ser realizadas pelo aplicativo, bem como as inscrições das equipes e o cadastro dos torcedores que acompanham os torneios esportivos. Todas as informações do campeonato são organizadas em tabelas de classificação, que podem ser visualizadas por todos os tipos de usuário. Este aplicativo ou *app* (abreviação para *application* em inglês) também permite maior interação da torcida com o campeonato, equipes e jogadores por meio de votações em jogadores que se destacaram durante as competições popularizando ainda mais o torneio esportivo. O aplicativo desenvolvido neste trabalho proporciona a facilidade de se criar, gerir e acompanhar um campeonato de maneira rápida e eficiente evitando perdas, erros e questionamentos existentes em competições esportivas administradas de maneira manual.

Palavras-chave: Aplicativo. Acompanhamento. Futebol Amador. Gerenciamento. Campeonato.

ABSTRACT

In this work, an Amateur Soccer Championships Management application was developed. The system was developed to facilitate the management, registration and monitoring of sports competitions such as field soccer, futsal and soccer society. All the necessary steps to create a championship can be realized by the application, as well as the inscriptions of the teams and the register of the fans that accompany the sporting tournaments. All championship information is organized into rating tables, which can be viewed by all types of users. This application or app (short for application in English) also allows greater interaction of the fans with the championship, teams and players through polls in players who stood out during the competitions, further popularizing the sports tournament. The application developed in this work provides the facility to create, manage and follow a championship quickly and efficiently avoiding losses, errors and questioning existing in sports competitions administered manually.

Keywords: Application. Follow up. Football Amateur. Management. Championship.

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

APP **-** Aplicativo

CBF - Confederação Brasileira de Futebol

FIFA - Federação Internacional de Futebol

IDE - Ambiente de Desenvolvimento Integrado

HTTP - Protocolo de Transferência de Hipertexto

UML - Linguagem Unificada de Modelagem

JSON - Notação de Objetos JavaScript

URI – Identificador Uniforme de Recursos

XML – Linguagem Extensível de Marcação Genérica

DCU – Design Centrado no Usuário

UI – Interface do Usuário

API – Interface de Programação de Aplicativos

URL – Localizador Padrão de Recursos

SUS – Escala de Usabilidade do Sistema

SUMÁRIO

1 INTRODUÇÃO

Segundo a *IDC - International Data Corporation* (2012), empresa especializada no mercado de tecnologia e telecomunicações, o crescimento do mercado de dispositivos móveis tem gerado oportunidades comerciais e sociais em diversas áreas. Esse tipo de dispositivo é considerado um computador de bolso com acesso a milhões de aplicativos. Segundo o grupo Gartner (2017) em 2016 o número de aplicativos baixados em smartphones aproximou - se de 309 bilhões.

O crescimento no número de *app* baixados se deve principalmente à facilidade com que esses aplicativos podem ser acessados em suas respectivas lojas virtuais. Desse modo, desenvolver soluções computacionais no formato de aplicativos móveis representa um meio eficaz de disponibilizar a ferramenta e atingir o público-alvo desejado (TIBES; DIAS, 2014).

A margem de toda esta facilidade está à forma de gerenciamento e criação de campeonatos amadores no Brasil, que utilizam um modelo arcaico de súmulas e fichas de papel para gerir os torneios. O futebol é um esporte altamente popular e internacional. Este tem o poder de unir as pessoas e inspirar o mundo através de suas competições (FIFA, 2016). Os torneios de futebol para jogadores amadores acontecem cotidianamente em todo país, este trabalho foi desenvolvido com objetivo de facilitar a criação, gestão, inscrição e o acompanhamento destas competições.

Sendo assim, neste trabalho foi desenvolvido um aplicativo *Android* de gerenciamento e acompanhamento de campeonatos de futebol amador, visto que atualmente os campeonatos são criados e gerenciados na maioria das vezes de maneira manual, podendo haver perdas ou erros por parte do organizador do campeonato. O aplicativo desenvolvido possibilita ao administrador do campeonato criar o torneio, permitir as inscrições de equipes e organizar as informações dos jogos. Aos jogadores e torcida o *app* permite o acompanhamento do campeonato com o apoio de grande quantidade de informações, tais como a tabela de classificação que é gerada automaticamente conforme os critérios de desempate adotados pelo aplicativo, estatísticas sobre o desempenho do time, datas e horários dos jogos,

permitindo também o acesso aos dados registrados na súmula de cada um dos jogos disputados, dando transparência e evitando questionamento de possíveis erros cometidos.

1.1 JUSTIFICATIVA

Neste trabalho foi desenvolvido um aplicativo *Android* de gerenciamento e acompanhamento de campeonatos de futebol amador. Atualmente as competições esportivas de futebol amador são criadas e gerenciadas na maioria das vezes de maneira manual, o que pode ocasionar perdas ou erros por parte do organizador do campeonato. Com o aplicativo desenvolvido neste trabalho o campeonato pode ser criado de forma rápida e eficiente garantindo maior transparência e evitando questionamento de possíveis erros cometidos, proporcionando maior facilidade para administradores de torneios esportivos para geri-los e permitindo que todos os usuários tenham acesso às informações dos jogos, informações estas que normalmente estão disponibilizadas apenas para os organizadores.

2 OBJETIVOS

2.1 OBJETIVOS GERAIS

O presente trabalho tem como objetivo geral desenvolver um aplicativo *Android mobile* de gerenciamento e acompanhamento para campeonatos de futebol amador, de modo que o aplicativo poderá ser utilizado em várias modalidades do futebol tais como: futsal, futebol society, e o futebol de campo tradicional.

2.2 OBJETIVOS ESPECÍFICOS

- Desenvolver um aplicativo *Android* para criar, gerenciar e acompanhar campeonatos de futebol amador;
- Desenvolver um sistema de gerenciamento de informações e exposição de dados, por meio de tabelas autoexplicativas, resultados dos jogos e estatísticas em geral de todos os times;
- Desenvolver um *WebService RESTfull* na linguagem de programação *Java*, que atenderá as requisições *HTTP* provenientes do aplicativo;
- Aplicar conceitos de usabilidade permitindo ao usuário maior facilidade na interação com o *app*.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 REGULAMENTO DE CAMPEONATOS DE FUTEBOL

Uma competição de futebol tem sua organização fundamentada no Regulamento Geral das Competições organizadas pela CBF[1] (Confederação Brasileira de Futebol). De acordo com o tipo de campeonato disputado é definido um sistema de competição. A seguir são brevemente explicadas às formas de disputa para competições de futebol de acordo com Rezende (2007):

- Campeonato por acúmulo de pontos ou pontos corridos: são campeonatos onde os pontos ganhos em cada jogo são acrescidos. Também denominado “todos contra todos”. Campeonatos onde todas as equipes jogam o mesmo número de partidas. Vence a equipe que somar mais pontos. É um tipo de campeonato muito fácil de organizar e pode ser realizado num curto período de tempo ou durante toda uma temporada, incluindo-se várias rodadas.
- Campeonatos por eliminação: em cada rodada é eliminada a metade dos participantes, se os perdedores não possuírem outra oportunidade de disputar o campeonato, este é chamado de eliminatória simples. Porém, se os perdedores disputam outro campeonato paralelo este é chamado de eliminatória dupla.
- Campeonato de Grupos: nesta modalidade pode-se criar um campeonato de grupos podendo ser com dois ou mais grupos contendo em cada um no mínimo duas esquipes. Sistema parecido com a Copa do Mundo, onde na primeira fase as equipes de cada grupo se enfrentam e no final as melhores equipes de cada grupo passam para a próxima fase, fase eliminatória (“mata-mata”), nesta fase as equipes poderão disputar as vagas em apenas um jogo ou realizarem jogos de ida e volta, pode-se ainda escolher se gol marcado fora de casa conta como critério de desempate.

3.1.1 Pontuação e Tabela de Classificação

De acordo a CBF[1][1] (Confederação Brasileira de Futebol) em seu Regulamento Geral das Competições, as competições são regidas pelo sistema de pontos ganhos, seguindo os critérios de pontuação: por vitória três pontos, por empate um ponto.

Esta pontuação é adotada pela maioria dos campeonatos de futebol profissional e não seria diferente em campeonatos amadores.

A soma da pontuação alcançada pelas equipes em um campeonato é organizada pela tabela de classificação. A tabela de classificação expõe informações como a posição do time no torneio, número de vitórias, gols marcados, gols sofridos etc.

Algumas das informações da tabela de classificação, são utilizadas como critérios de desempate e estes, na maioria das vezes, são os mesmos para todos os campeonatos. O que pode variar é a ordem de importância destes critérios.

De acordo com o Regulamento Geral das Competições da CBF[1] (2016), em geral, os campeonatos adotam como primeiro critério o número de pontos ganhos. O segundo critério, para o caso de empate pode variar entre o número de vitórias e o saldo de gols dependendo do campeonato.

3.1.2 Registros de informações em campeonatos e atual gerenciamento

Em todas as partidas de futebol existe um documento de registros de informações oficial conhecido como súmula, onde são registrados todos os acontecimentos do jogo. A súmula é preenchida pelos árbitros da partida.

Neste documento são contidas informações como: times que disputaram a partida, relação dos jogadores de cada equipe, data e horário do jogo, resultado do jogo, substituições feitas, jogadores que marcaram gols e que receberam cartões e a classificação das punições dada pela cor dos cartões (amarelo e vermelho). Por meio das informações contidas em cada súmula são acrescidos os dados necessários à tabela de classificação do campeonato.

[1] htt/www.cbf.com.br

A organização e o gerenciamento de torneios de futebol amadores são realizados de forma manual. Desde as inscrições que na maioria das vezes é feita por meio de fichas, até o chaveamento e sorteio dos grupos são utilizados métodos sujeitos a alterações que possam beneficiar determinada equipe. Além disso, o registro de informações manual dificulta a recuperação destes dados e também abre espaço para perdas, erros de anotação e interpretação dos dados contidos nos registros de partida.

3.2 DESENVOLVIMENTO DE SOFTWARE

Segundo Pressman (2011, p.48) “a engenharia de software engloba processos, métodos e ferramentas que possibilitam a construção de sistemas complexos baseados em computador dentro do prazo e com qualidade”. A estrutura de um processo de software incorpora cinco atividades estruturais: comunicação, planejamento, modelagem, construção e implantação; e elas se aplicam a todos os projetos de software (PRESSMAN, 2011). A seguir estão descritas as fases do desenvolvimento de um software.

3.2.1 Comunicação

Esta fase do processo de estruturação do projeto de software promove a comunicação e colaboração entre cliente e usuário, buscando compreender os objetivos do cliente para com o projeto e fazer o levantamento e análise de requisitos necessários para definir as características e funções do software (PRESSMAN, 2011). Abaixo destaca-se a importância do levantamento e análise de requisitos.

Levantamento e Análise de Requisitos: De acordo com Sommerville e Sawyer (1997), esta atividade tem como objetivo levantar e priorizar características que o sistema precisa ter, para ser capaz de realizar sua função. O processo de levantamento de requisitos deve estabelecer a definição de requisitos como um processo onde, o que deve ser feito é elicitado, modelado e analisado, considerando os diferentes pontos de vista, objetivando a validar um modelo, a partir do qual um documento de requisitos é produzido (SOMMERVILLE; SAWYER, 1997).

No processo de levantamento é que se define o objetivo do sistema, e como este será utilizado no dia-a-dia. Com a análise destes requisitos definidos na etapa de levantamento minimizam-se problemas comuns identificados durante o desenvolvimento do software, tais como: problemas de escopo em que se definem os limites do sistema de forma precária ou especificação de detalhes técnicos desnecessários que confundem ao invés de esclarecer os objetivos do sistema; problemas de entendimento onde são especificados requisitos ambíguos, gerando desentendimento entre cliente e analista; por fim, temos os problemas de volatilidade em que os requisitos mudam com o tempo (CHRISTEL; KANG, 2012).

De acordo com a norma técnica 610.12 do IEEE (Instituto de Engenheiros Elétricos e Eletrônicos) (1990), a etapa de análise de requisitos é um processo que envolve o estudo das necessidades do usuário para se encontrar uma definição correta e completa do sistema. Primeiro avalia-se o problema identificando as informações que serão necessárias ao sistema para a obtenção da melhor solução. Depois realiza-se a especificação dos requisitos para consolidar funções, interfaces, desempenho, o contexto e as restrições do sistema; por fim faz-se a revisão dos requisitos para eliminar possíveis inconsistências do sistema.

Os requisitos colhidos fornecerão a referência para validar o produto final, estabelecerão o acordo entre cliente e fornecedor sobre o que o software fará e consequentemente reduzirão os custos de desenvolvimento, pois requisitos mal definidos implicam num retrabalho (IEEE, 1990).

Após a análise e modelagem dos requisitos inicia-se a fase de construção do projeto que pode ser simplificada pela atividade de planejamento.

3.2.2 Planejamento

“O plano de projeto de software define o trabalho de engenharia de software, descrevendo as tarefas técnicas a serem conduzidas, os riscos prováveis, os recursos que serão necessários, os produtos resultantes a ser produzidos e um cronograma de trabalho” (PRESSMAN, 2011, p.40).

Realizando a etapa de planejamento, simplifica-se o processo de construção do projeto de software, neste, é gerada uma representação computacional significativa, mencionando o que o software deve fazer.

> O projeto de software é um processo iterativo através do qual os requisitos são traduzidos em uma “planta” para construir o software. Inicialmente, a planta representa uma visão holística do software. O projeto é representado em um alto nível de abstração, um nível que pode ser associado diretamente ao objetivo específico do sistema e aos requisitos mais detalhados de dados, funcionalidade e comportamento (PRESSMAN, 2011, p.209).

Ainda segundo Pressman (2011, p.209), “um bom projeto de software deve implementar todos os requisitos explícitos contidos no modelo de requisitos e deve acomodar todos os requisitos implícitos desejados pelos interessados”. O projeto deve ser um guia legível, compreensível para aqueles que desenvolvem e testam e, subsequentemente, dão suporte ao software, permitindo uma visão completa do software.

“Um projeto é produzido a partir de requisitos. Ele exclui o código. Uma notação extremamente útil para a documentação de um projeto é a Linguagem Unificada de Modelagem (UML - *Unified Modeling Language*)” (BRAUDE, 2005, p.28). A linguagem UML é comumente utilizada na fase de modelagem do processo de desenvolvimento do projeto do software, esta será descrita no item seguinte.

3.2.3 Modelagem

De acordo com Braude (2005), a modelagem irá conter as entidades e relacionamentos que representam características do software, que ajudam os profissionais a construí-lo efetivamente considerando a arquitetura, a interface do usuário e os componentes que o sistema irá disponibilizar.

Para a etapa de modelagem, geralmente, é utilizado a UML (Linguagem Unificada de Modelagem) que auxilia no processo de desenvolvimento a fim de contribuir com o projeto através de ilustração de itens, relacionamentos e

diagramas padronizados que demonstram todas as funções que o mesmo irá conter (BRAUDE, 2005).

A UML é uma linguagem destinada à estruturação de projetos. Um dos principais objetivos da UML é tornar o processo de desenvolvimento totalmente transparente e organizado, por meio da construção de modelos para comunicar a estrutura e o comportamento desejados do sistema, para este fim, utiliza-se o diagrama entidade-relacionamento (DER), que permite visualizar e controlar a arquitetura do software garantindo uma melhor compreensão do sistema que estamos elaborando (BRAUDE, 2005).

3.2.4 Construção

"Essa atividade combina geração de código (manual ou automatizada) e testes necessários para revelar erros na codificação" (PRESSMAN, 2011, p.41). A etapa de construção consiste na implementação e realização de testes.

3.2.5 Implementação e testes

A implementação envolve as atividades de codificação e testes, esta pode ser descrita como uma forma de especificação de algoritmos e estruturas de dados e deve então ser codificada em uma linguagem de programação. Deve ser implementado tudo o que foi especificado durante a análise de requisitos e desenvolvimento do projeto. Os componentes do *software* devem ser implementados de forma independente e depois integrados conforme a arquitetura interna do *software* (PFLEEGER, 2004). Durante a fase de implementação podem ser iniciados os testes.

De acordo com Rocha (2001), o software é validado quando o programa implementado satisfaz à sua especificação. Para realizar a validação do software, este deve ser submetido a uma fase de testes. É necessário testar cada funcionalidade de cada módulo, considerando a especificação feita na fase de projeto.

Os testes devem ser planejados antes de serem realizados. Um plano de testes deve ser utilizado para guiar todas as atividades de teste e deve

incluir os objetivos de cada teste, como serão executados e quais critérios a serem utilizados para determinar quando o teste está finalizado. O resultado é documentado em um relatório de testes, que contém as informações sobre erros encontrados no sistema, e o modo como este se comporta sobre vários aspectos. Assim os diversos módulos do sistema são integrados, resultando no produto de software (PFLEEGER, 2004). Os testes são realizados a fim de validar o software, visando determinar se a especificação do mesmo satisfaz os requisitos do usuário.

3.3 APLICATIVOS MOBILE-ANDROID

O uso de dispositivos móveis vem crescendo consideravelmente no mundo, e o Brasil é um dos países que mais se destaca nesse assunto. Segundo a Anatel[2][2], em janeiro de 2014, o número de *smartphones* cresceu 99%, considerando o período de um ano. Neste mesmo ano, o acesso à internet por dispositivos móveis superou o acesso por desktops.

De acordo com a *International Data Corporation* (IDC) (2016), no Brasil o crescimento do tráfego de dados móveis até 2018 aumentará cerca de 11 vezes, com crescimento maior que 60% ao ano. O mercado de aplicativos movimentou uma renda de US$ 27 bilhões em 2013, de acordo com pesquisas recentes da Gartner (2016), o mercado de aplicativos pode atingir US$ 77 bilhões em 2017.

A primeira geração de telefone *Android* foi lançada em outubro de 2008. Dois anos depois as vendas de telefones que possuíam o *Android* como sistema operacional aumentaram 707%. Em março de 2011, o *Android* já se encontrava com 37% do mercado de *smartphones* nos Estados Unidos (DEITEL *et al*., 2013). Em 2014 a participação no mercado do sistema operacional *Android* chegou a 81,5% de acordo com a Tabela 1.

> Uma vantagem de desenvolver aplicativos Android é a franqueza (ou grau de abertura) da plataforma. O sistema operacional é de código-fonte aberto e gratuito. Isso permite ver o código-fonte do Android e como seus recursos são implementados (DEITEL *et al,* 2013, p.04).

[2] http/www.anatel.gov.br/institucional

De acordo com Deitel *et al* (2013), os aplicativos *Android* são desenvolvidos com Java (a linguagem de programação mais usada do mundo). Essa linguagem foi uma escolha lógica para a plataforma Android, porque é poderosa, gratuita e de código-fonte aberto. O Java é usado para desenvolver aplicativos empresariais de larga escala, melhorar a funcionalidade de servidores Web, fornece aplicativos para aparelhos celulares e para muitos outros propósitos.

A distribuição dos aplicativos *Android* é disponibilizada pelo site Google Play[3]. Neste é possível encontrar uma variedade de aplicações para diferentes fins, tais como jogos, redes sociais, filmes, etc. As aplicações possuem versões pagas e outras gratuitas que podem ser baixadas e instaladas normalmente em plataformas *Android* desde que o aplicativo seja compatível com o dispositivo.

Tabela 1: Participação(em milhões) do Android no mercado em relação a outros sistemas operacionais-IDC-2015

Sistema Operacional	Unidades de 2014	Participação de mercado em 2014	Unidades de 2013	Participação de mercado em 2013	Variação ano a ano
Android	**1 059,3**	**81,5%**	**802,2**	**78,7%**	**32,0%**
IOS	**192,7**	**14,8%**	**153,4**	**15,1%**	**25,6%**
Outros	**48,4**	**3,7%**	**55**	**5,1%**	**169,2%**
Total	**1300,4**	**100,0%**	**1018,7**	**100,0%**	**27,7%**

Fonte: Corporação de Dados Internacionais - IDC

3.4 CLIENTE SERVIDOR

A arquitetura Cliente/Servidor é definida como o processamento da informação dividido em módulos ou processos distintos. Um processo é responsável pela manutenção da informação (Servidor) e outros responsáveis

[3] http://play.google.com/store

pela obtenção dos dados (Cliente) (BATTISTI, 2001). Ou seja, clientes fazem requisições de serviço a um servidor (uma máquina geralmente bastante potente, em termos de capacidades de entrada/saída).

Cliente é um processo que interage com o usuário através de uma interface gráfica ou não, permitindo consultas ou comandos para análise e recuperação de dados. Além disso, apresenta algumas características distintas: é o processo ativo na relação Cliente/Servidor, inicia e termina as conversações com os Servidores, solicitando serviços distribuídos e torna a rede transparente ao usuário (AMARAL FILHO, 1993).

O servidor fornece um determinado serviço que fica disponível para todo cliente que o necessita. A natureza e escopo do serviço são definidos pelo objetivo da aplicação cliente/servidor. Além disso, ele apresenta ainda algumas propriedades distintas: é o processo passivo na relação cliente/servidor, possui uma execução contínua, recebe e responde às solicitações dos clientes, não se comunica com outros servidores enquanto estiver fazendo o papel de servidor, presta serviços distribuídos atendendo a diversos clientes simultaneamente (AMARAL FILHO, 1993).

Neste trabalho, o cliente é o aplicativo *mobile* (*Android*) que faz requisições por meio do *Json - JavaScript Object Notation* (formato comum para intercambio de dados) ao servidor (*WebService RESTful Java*) que disponibiliza os resultados requisitados pelo cliente. O banco de dados utilizado é o *PostgreSQL,* localizado no servidor, e nele são armazenados todos os dados do sistema, tais como: campeonatos, equipes participantes e torcedores usuários do aplicativo.

3.5 WEB SERVICES

Os *Web services* são definidos como unidades de lógica de aplicação autosuficiente que fornecem funcionalidade a outras aplicações através da internet (SRIVASTAVA; KOEHLER, 2003).

De acordo com Aho, Sethi e Ullman (1986), web services oferecem serviços para aplicações e podem ser vistos como objetos ou chamadas de funções através da internet, por meio de protocolos desenvolvidos para a web,

podendo então, serem acessados de qualquer lugar, estes também são utilizados para integrar sistemas e aplicações distintas através da internet.

Neste trabalho, o *web service* foi desenvolvido para receber as requisições vindas do aplicativo *Android*, sendo o *web service* responsável pela comunicação entre aplicativo e o banco de dados. Este recebe as requisições vindas do aplicativo, acessando o servidor para manipular informações conforme as necessidades do usuário.

3.6 WEB SERVICE RESTFUL

"*Web service* RESTful são aplicações de *web* desenvolvidas sobre a arquitetura REST, expondo recursos (dados e funcionalidades) através do Identificador Uniforme de Recursos (URI) *web*, possibilitando criar, recuperar, atualizar e deletar recursos através de um protocolo de comunicação" (HAMAD; SAAD; ABED, 2010, p.74), neste caso o HTTP.

Segundo Hamad Saad e Abed (2010, p.74), "na arquitetura REST, clientes e servidores trocam representações de recursos, utilizando interface e protocolo padronizados, proporcionando simplicidade, leveza e alta performance às aplicações REST".

Comparando o web service RESTful com o SOAP, evidencia-se claramente a superioridade do RESTful sobre o SOAP, conforme mostra a figura 1 (HAMAD; SAAD; ABED, 2010).

Figura 1:Comparação de desempenho entre o RESTful e SOAP

Number of array elements	Message Size (byte)				Time (Milliseconds)			
	SOAP/HTTP		REST (HTTP)		SOAP/HTTP		REST (HTTP)	
	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition	String Concatenation	Float Numbers Addition
2	351	357	39	32	781	781	359	359
3	371	383	48	36	828	781	344	407
4	395	409	63	35	828	922	359	375
5	418	435	76	39	969	1016	360	359
6	443	461	93	43	875	953	359	359
7	465	487	104	47	875	875	469	360
8	493	513	127	51	984	875	437	344

Fonte: HAMAD; SAAD; ABED (2010, p. 75).

Na tabela é possível observar que no *web service RESTful* o tamanho da *string* enviada é menor em relação ao SOAP, com isso pode-se verificar o menor tempo de transmissão da *string* no REST(HTTP).

3.7 **JSON**

De acordo com Nurseitov *et al* (2009), JSON[4] (*JavaScript Object Notation* – Notação de Objetos *JavaScript*) é uma linguagem de intercâmbio de dados, que foi projetada para facilitar a análise, a troca e o uso de dados entre computadores, sendo também a mais adequada para aplicações *JavaScript*, proporcionando um ganho de desempenho significativo se comparado à linguagem XML. Além disso, o JSON usa contexto para evitar ambiguidade, a sintaxe deste é legível/compreensível para humanos e para máquinas é fácil de interpretar e gerar.

Neste trabalho foi utilizado o JSON como linguagem padrão de comunicação entre o aplicativo *android* e o *web service*.

3.8 USABILIDADE

De acordo com a Associação Brasileira de Normas Técnicas (ABNT) (2002), usabilidade é um conjunto de atributos de software relacionado ao esforço necessário para seu uso e para o julgamento individual de tal uso por determinado conjunto de usuários,

.

> A usabilidade garante a própria continuidade e afirmação competitiva de um site, de um software ou de um sistema de informação na perspectiva de interação com o usuário. É pela interação com o usuário, a partir do seu desempenho e da sua satisfação, que se evidencia a sobrevivência de um sistema de informação (COSTA; RAMALHO, 2010, p. 110).

Assim destaca-se a seguir alguns dos atributos de usabilidade descritos por Nielsen (1993, p.26) em seu livro *Usability Engineering*:

[4] http://www.json.org/json-pt.html

- "Facilidade de aprendizado que prioriza a simplicidade do sistema e facilidade de aprendizagem para que o usuário conheça o sistema rapidamente e possa desenvolver suas atividades" (NIELSEN, 1993, p.26);
- "Eficiência de uso, o sistema tem que ser hábil para que o usuário ao desenvolver suas atividades atinja altos níveis de produtividade" (NIELSEN, 1993, p.26);
- "Facilidade de memorização garantindo a possibilidade do usuário de reegressar ao sistema e realizar suas tarefas mesmo tendo estado sem fazer uso do sistema por um bom tempo" (NIELSEN, 1993, p.26);
- "Baixa taxa de erros permitindo que o usuário utilize o sistema sem grandes problemas e com baixo índice de erros" (NIELSEN, 1993, p.26);
- "Satisfação subjetiva que permite ao usuário satisfação ao interagir com o sistema" (NIELSEN, 1993, p.26).

"O aplicativo terá o design centrado no usuário (DCU). As atividades de DCU prezam pelo envolvimento ativo do usuário por meio da compreensão clara da tarefa e dos requisitos para sua realização" (ALVES; BATTAIOLA, 2014, p. 27).

O teste de usabilidade foi utilizado para compreender melhor a interação do usuário com o produto, considerando as medidas de usabilidade determinadas pela Norma Brasileira Regulamentadora (NBR) número 9241-11 de 2002, da Associação Brasileira de Normas Técnicas (ABNT), que diz que todo produto deve garantir eficiência, eficácia e satisfação.

A construção de um sistema com usabilidade depende de uma análise cuidadosa de seus componentes considerando o contexto de uso e da participação ativa do usuário nas decisões do projeto, levando em consideração as opiniões dos usuários do aplicativo expressas após os testes de usabilidade.

Assim o conceito de usabilidade amplamente aplicado no desenvolvimento deste aplicativo objetivou garantir que os usuários do sistema tivessem suas necessidades plenamente atendidas em um contexto particular de uso, que é a organização de campeonatos de futebol amador.

4 MATERIAL E MÉTODOS

Este capítulo apresenta os materiais e a metodologia que foram utilizados para o desenvolvimento do sistema de gerenciamento e acompanhamento de campeonatos de futebol amador.

4.1 FERRAMENTAS PARA O DESENVOLVIMENTO DO APLICATIVO

4.1.1 Banco de dados PostgreSQL

O PostgreSQL[5] é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. É altamente escalável, tanto na quantidade enorme de dados que pode gerenciar, quanto no número de usuários concorrentes que pode acomodar. Possui um sofisticado mecanismo de bloqueio (MVCC - *Multi-Version Concurrency Control* é um controle de simultaneidade multiversão que oferece o isolamento de transações), aceita vários tipos de subconsultas e conta com um bom mecanismo de *FAILSAVE* (segurança contra falhas, por exemplo, no desligamento repentino do sistema).

O PostgreSQL se destaca por ser mais robusto e possuir muito mais recursos que o MySQL, como por exemplo o sofisticado planejador de consultas (otimizador) que garantiu ao PostgreSQL uma velocidade equivalente a oferecida com o uso do MySQL.

O pgAdmin[6] é a plataforma de administração e desenvolvimento *open source* para acesso ao PostgreSQL. O pgAdmin foi desenvolvido para responder às necessidades de todos os utilizadores, desde escrever consultas SQL simples até desenvolver bases de dados complexas. A interface gráfica pode ser executada no desktop ou em um servidor web e suporta todos os recursos comuns do PostgreSQL.

Para este trabalho foi utilizado o PostgreSQL devido sua eficiência e suas funcionalidades atenderem aos requisitos necessários ao desenvolvimento do aplicativo e para administrar o banco de dados foi empregado o pgAdmin.

[5] https://www.postgresql.org.br/
[6] https://www.pgadmin.org/

4.1.2 Componentes de layout para criação da interface gráfica

Abaixo estão sendo explicados os componentes de layout que foram utilizados para criar a interface gráfica.

***LinearLayout*:** é um grupo de exibições que alinha todos os filhos em uma única direção vertical ou horizontal. A direção do layout foi especificada com o atributo *android: orientation* (DEVELOPERS, 2016). O *LinearLayout* foi utilizado como base para o desenvolvimento das telas. Este recebeu os outros componentes de layout (imagens, textos, caixa de texto etc) usados no desenvolvimento da interface gráfica.

***RelativeLayout*:** é um grupo de exibição, semelhante ao *LinearLayout*, em que os componentes são orientados em relação a posição relativa de outros componentes. É um utilitário usado para projetar interfaces de usuário, pode eliminar grupos de exibição aninhados e manter sua hierarquia de layout plana, o que melhora o desempenho (DEVELOPER, 2016). Ao invés de se usar vários grupos *LinearLayout* aninhados, é possível substituí-los por um único RelativeLayout, considerando esta vantagem foi utilizado este recurso para criação das telas.

ScrollView: é um contêiner de *layout* para uma hierarquia de exibição que pode ser rolada pelo usuário, permitindo que o *layout* seja maior que a exibição física (DEVELOPER, 2016). Foi utilizado em conjunto com o *LinearLayout* ou *RelativeLayout* todas as vezes que a dimensão do *layout* foi maior que o espaço físico de exibição disponível.

***TextView*:** exibe texto para o usuário. Foi utilizado como modo de exibição de texto no desenvolvimento das telas do aplicativo.

***EditText*:** é um editor de texto. Foi utilizado para obter informações do usuário no formato de texto.

***Button*:** consiste em texto e/ou ícone que comunica a ação que ocorre quando o usuário o toca. Foi utilizado como botões de confirmação de informações.

***Spinner*:** fornece uma maneira rápida de selecionar um item de um conjunto. No estado padrão, um spinner mostra o item atualmente selecionado. Ao tocar no botão giratório, é exibido um menu suspenso com todos os outros itens disponíveis, a partir dos quais o usuário pode selecionar um novo

(DEVELOPER, 2016). Foi utilizado para exibição de itens predeterminados pelo aplicativo.

***ImageView*:** Exibe uma imagem, como por exemplo um ícone. A classe *ImageView* pode carregar imagens de várias fontes, cuida de computar a dimensão da imagem para que ela possa ser usada em qualquer componente de layout e fornece várias opções de exibição (DEVELOPER, 2016). Para tanto este recurso foi utilizado.

***TableLayout*:** é um *layout* que organiza seus filhos em linhas e colunas. Consiste em um número de objetos *TableRow*, cada um definindo uma linha (DEVELOPER, 2016). Utilizado para exibir informações em formato de tabela, como a tabela de classificação de um campeonato.

***RecyclerView*:** usado para exibir itens por meio de listas. O *RecyclerView* é muito mais eficiente que as duas antigas implementações (*ListView* e *GridView*). Uma de suas grandes vantagens é a flexibilidade de poder mudar o modo de exibição dos itens, sem a necessidade de grandes alterações no código (DEVELOPER, 2016). Neste trabalho foi utilizado para listar, por exemplo, os jogos dos campeonatos.

***ViewPager*:** gerenciador de layout que permite ao usuário deslizar a tela de exibição da direita para esquerda, e vice-versa, através de páginas de dados. O *ViewPager* é mais usado em conjunto com o *Fragment*, que é uma maneira conveniente de fornecer e gerenciar o ciclo de vida de cada página. Há adaptadores padrão implementados para usar fragmentos com o *ViewPager*, que cobrem os casos de uso mais comuns (DEVELOPER, 2016). Recurso utilizado para exibir informações das equipes e do campeonato, onde as informações de cada campeonato foram separadas nas abas: tabelas, jogos e estatísticas.

***CheckBox*:** conjunto de opções de caixa de seleção que permite que o usuário selecione vários itens, cada caixa de seleção é gerenciada separadamente (DEVELOPER, 2016). Foi utilizado para selecionar as equipes cabeças de chave dos grupos do campeonato, sendo esta apenas uma de suas aplicações neste trabalho.

***ProgressDialog*:** é uma caixa de diálogo mostrando um indicador de progresso e uma mensagem de texto opcional (DEVELOPER, 2016).

***AlertDialog*:** é uma caixa de dialogo apresentada ao usuário para confirmar uma ação ou reportando algum erro (DEVELOPER, 2016).

4.1.3 Biblioteca Retrofit

Retrofit[7] é um cliente REST seguro para *Android* desenvolvido pela *Square .* A biblioteca fornece uma estrutura poderosa para autenticar e interagir com APIs (*webservice*) e enviar solicitações de rede com o OkHttp. Esta biblioteca torna o download de dados JSON de uma API da web bastante simples. Uma vez que os dados são baixados, em seguida, ele é analisado em um simples *Plain Old Java Object* (POJO) que deve ser definido para cada "recurso" na resposta (SQUARE, 2016).

Neste trabalho a biblioteca Retrofit foi usada para criar a conexão HTTP cliente no aplicativo *android*, para que este se comunique com o *webservice* (JAVA).

4.1.4 Picasso

Picasso[8] é uma biblioteca de dowload de imagens de código aberto amplamente utilizadas no *Android*. Ele é criado e mantido pela *Square*. É uma poderosa biblioteca de transferência e de *caching* de imagem para *Android*.

O Picasso simplifica o processo de carregar imagens de URLs externas e exibir em aplicações *Android*. Além de oferecer o download das imagens ele também oferece o cache automatico da imagem na aplicação.

4.1.5 AsyncTask

AsyncTask permite o uso adequado e fácil do *thread* UI (Interface do Usuário). Esta classe permite executar operações em segundo plano e publicar resultados no *thread* da interface do usuário sem ter que manipular threads. É projetado para ser uma classe auxiliar em torno de *thread* e manipulador. Este idealmente deve ser usado para operações curtas (alguns segundos no máximo) (DEVELOPER, 2016).

[7] http://square.github.io/retrofit/
[8] http://square.github.io/picasso/

Uma tarefa assíncrona é definida por uma computação que é executada em segundo plano e cujo resultado é publicado no *thread* da interface do usuário. Uma tarefa assíncrona é definida por três tipos genéricos, chamados *Params, Progress* e *Result*, e quatro etapas, chamadas *onPreExecute, doInBackground, onProgressUpdate* e *onPostExecute* (DEVELOPER, 2016), estes são descritos a seguir.

- ***OnPreExecute* ():** invocado na thread da interface do usuário antes que a tarefa seja executada. Esta etapa é normalmente usada para configurar a tarefa, por exemplo, mostrando uma barra de progresso na interface do usuário (DEVELOPER, 2016).

- ***DoInBackground* (*Params*) :** invocado na thread de plano de fundo imediatamente após onPreExecute () terminar a execução. Esta etapa é usada para executar a computação em segundo plano que pode levar muito tempo. Os parâmetros da tarefa assíncrona são passados para esta etapa. O resultado do cálculo deve ser retornado por esta etapa e será passado para a última etapa (DEVELOPER, 2016).

- ***OnProgressUpdate* (*Progress*):** invocado no thread da UI após uma chamada para publishProgress (Progress ...). O tempo da execução é indefinido. Esse método é usado para exibir qualquer forma de progresso na interface do usuário enquanto a computação em segundo plano ainda está sendo executada. Por exemplo, ele pode ser usado para animar uma barra de progresso ou mostrar *logs* em um campo de texto (DEVELOPER, 2016).

- ***OnPostExecute* (*Result*):** invocado no thread UI (Interface do Usuário) após a conclusão do cálculo de background. O resultado da computação em segundo plano é passado para esta etapa como um parâmetro (DEVELOPER, 2016).

Neste trabalho o OnPreExecute foi aplicado para informar ao usuário a ação que está acontecendo, por meio de uma barra de progresso. A partir do DoInBackground foram realizadas as chamadas no *webservice* através da

biblioteca Retrofit. Assim que recebe a resposta o DoInBackground a envia para o OnPostExecute que retorna as informações para o usuário.

4.2 ANÁLISE E LEVANTAMENTO DE REQUISITOS

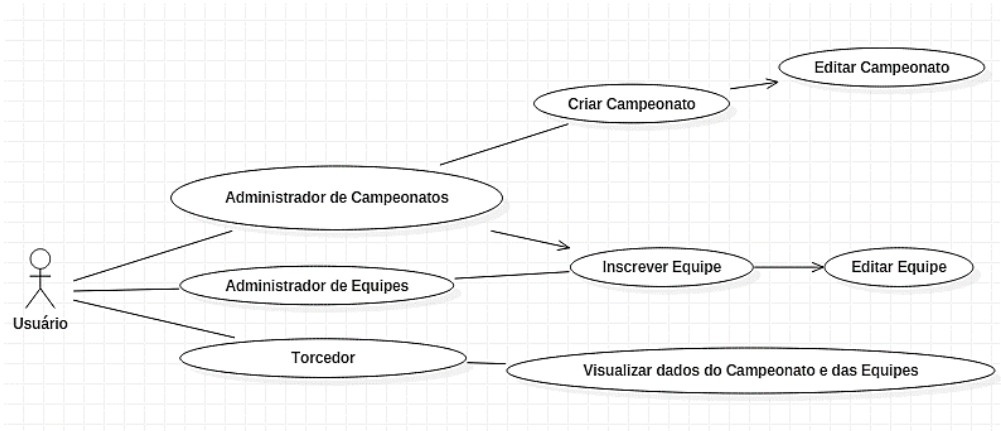
A etapa de análise de requisitos envolveu o estudo das necessidades do usuário para se encontrar uma definição correta e completa do sistema. Determinaram-se primeiramente os requisitos funcionais (Quadro 1), estes estão listados abaixo com seus respectivos diagramas de casos de uso, nestes há três tipos de usuários (Figura 2).

Quadro 1: Requisitos funcionais do aplicativo

RF1	Criar campeonato.	Figura 6
RF2	Inscrever equipes.	Figura 7
RF3	Confirmar inscrições das equipes.	Figura 8
RF4	Configurar modo de disputa.	Figura 9
RF5	Atualização de resultados e horários.	Figura10
RF6	Acompanhar campeonato.	Figura 11

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

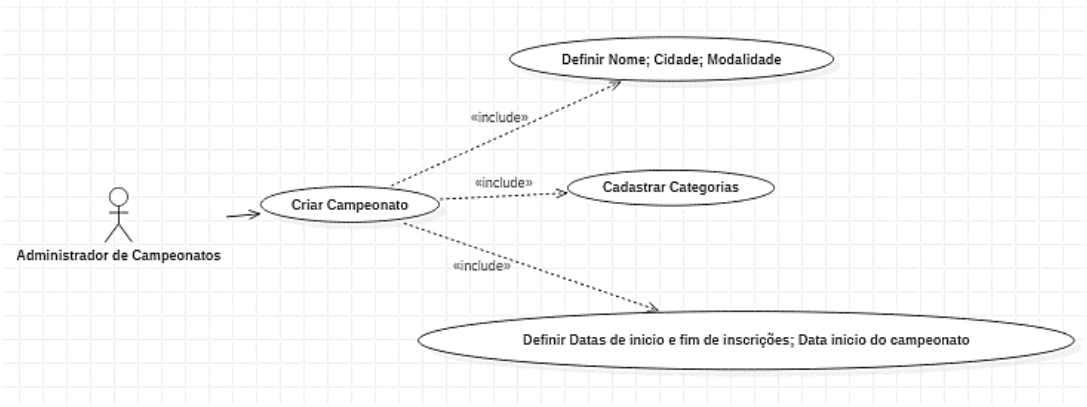
Figura 2: Tipos de usuário



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

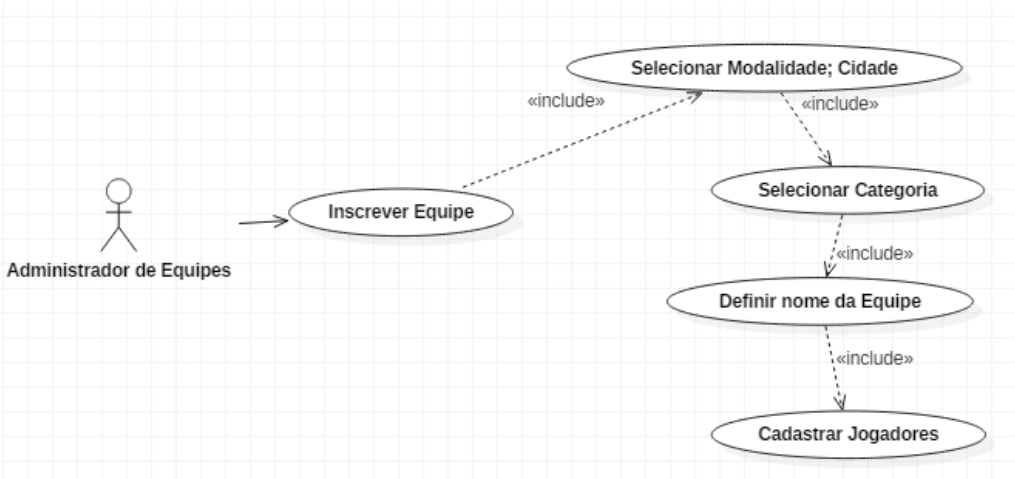
- Diagramas de casos de uso:

Figura 3: Criar campeonatos



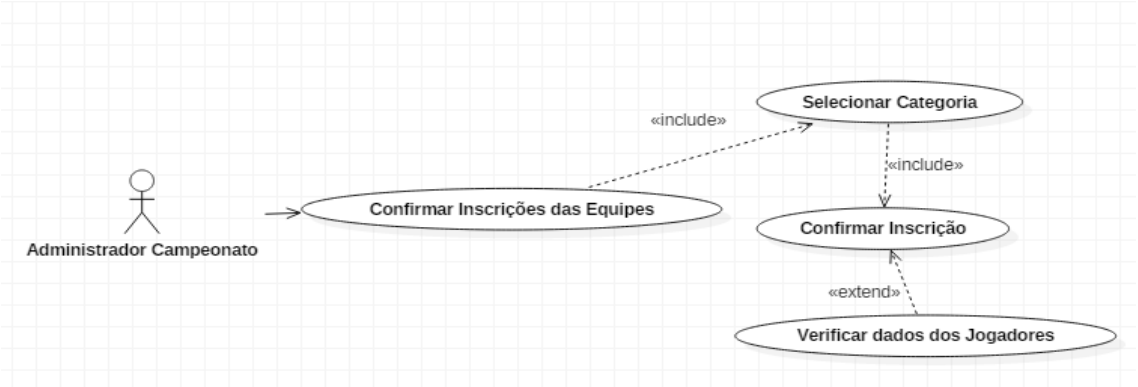
Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Figura 4: Inscrever equipe



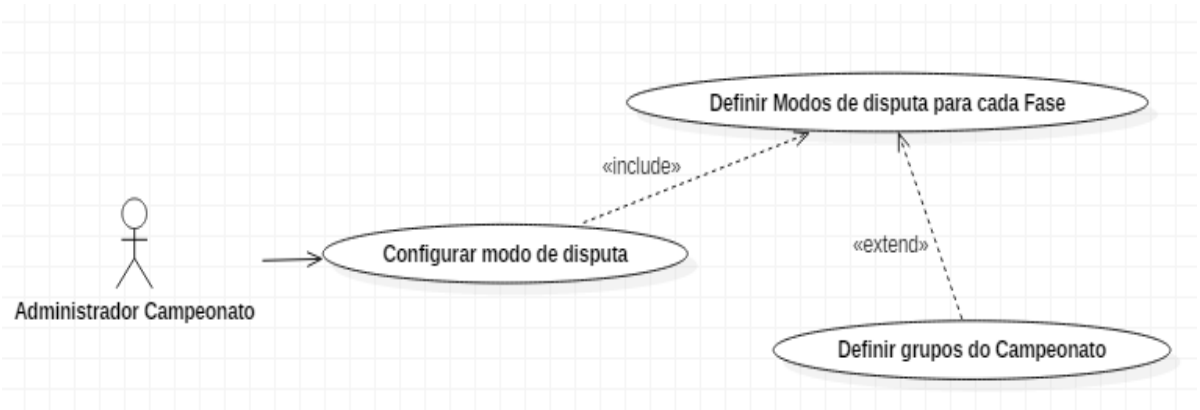
Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Figura 5: Confirmar inscrições da equipe



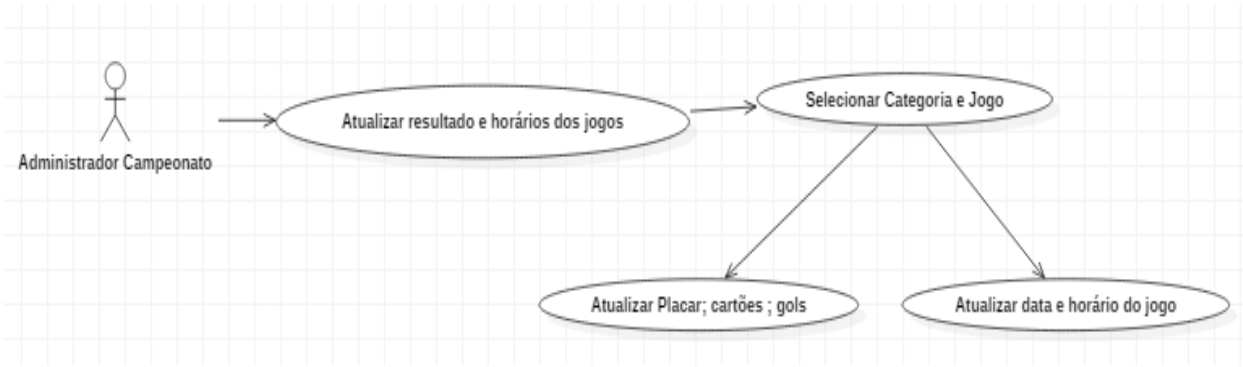
Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Figura 6: Configurar modo de disputa



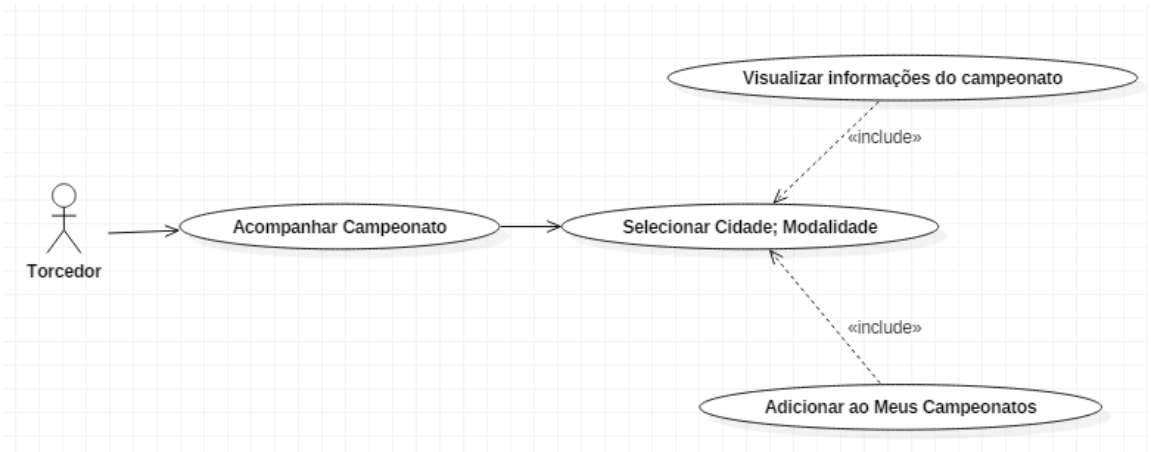
Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Figura 7: Atualização de resultados e horários dos jogos



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Figura 8: Acompanhar campeonato



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Depois foram analisados os requisitos elicitados para consolidar suas funções. Em seguida, foram levantados os requisitos de interface aplicando conceitos de usabilidade seguindo o modelo utilizado no gerenciamento de campeonatos tradicionais, visando o uso intuitivo do aplicativo desenvolvido. A seguir estão listados os requisitos de interface (Quadro 2) (requisitos não funcionais):

Quadro 2: Requisitos não funcionais do aplicativo

RNF1	Restringir a quantidade de funcionalidades, diminuindo a chance dos usuários se confundirem diante das opções oferecidas.
RNF2	Criar uma interface onde o usuário esteja familiarizado e se sinta confortável diante das opções oferecidas.
RNF3	Propiciar o fácil reconhecimento e memorização de palavras, símbolos e imagens.
RNF4	Navegação seguindo modelo mental simples e intuitivo.
RNF5	Informar ao usuário qual ação está em processamento.
RFN6	Qualidade visual e distribuição dos componentes.
RFN7	Visualização de informações por tabelas.
RFN8	Criar o campeonato em etapas de forma sequencial.

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

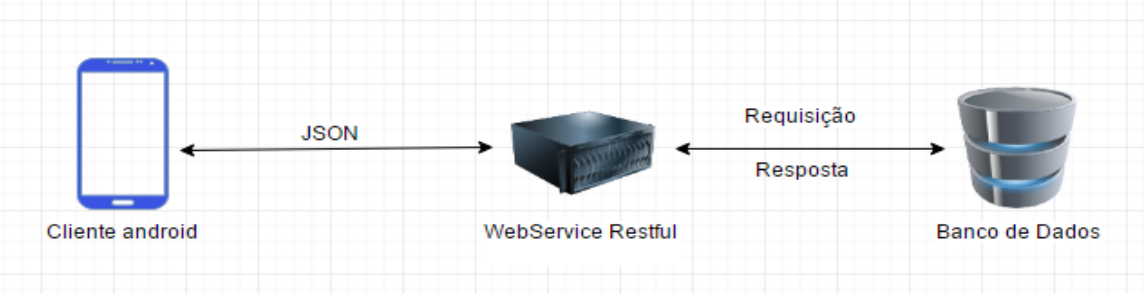
Por fim fez-se a revisão dos requisitos para eliminar possíveis inconsistências do sistema, permitindo o início da etapa de projeto e modelagem.

4.3 PROJETO

Nesta etapa foi construído o projeto do software a partir dos resultados obtidos na análise de requisitos, por meio deste, foi possível ter uma visão completa das entidades e das funcionalidades do sistema, também foi possível aperfeiçoar as inconsistências que surgiram durante a fase de projeção e acrescentar funcionalidades para aprimorar a qualidade do aplicativo. Foram feitos esboços do aplicativo que facilitaram a etapa de modelagem das

entidades e seus relacionamentos e posteriormente a fase de implementação (Figura 9).

Figura 9: Arquitetura do funcionamento do sistema

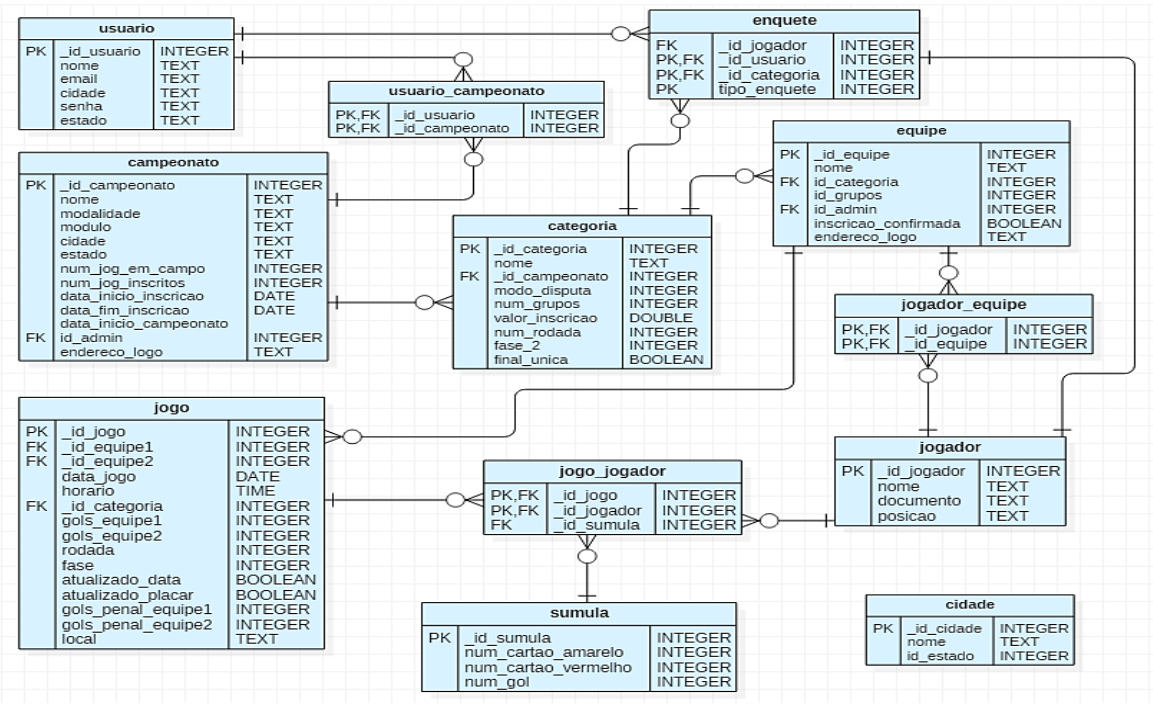


Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

4.4 MODELAGEM

Nessa fase do desenvolvimento do aplicativo foi feita a modelagem para representar as características do sistema. Utilizou-se a UML (Linguagem Unificada de Modelagem) para auxiliar no processo de desenvolvimento contribuindo com o projeto através de ilustração de itens, relacionamentos e diagramas padronizados (Figura 10) que demonstraram todas as funções que o mesmo irá conter.

Figura 10: Diagrama de entidade e relacionamento



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

4.5 IMPLEMENTAÇÃO

Foram estabelecidos parâmetros para compor a criação e gerenciamento dos campeonatos, cadastro das equipes participantes e acesso dos usuários em geral. Para o desenvolvimento do aplicativo de gerenciamento de campeonatos de futebol amador foi utilizado à linguagem de programação JAVA juntamente com o *Android SDK* (kit de desenvolvimento para *Android*). Também foram ultilizados os ambientes de desenvolvimento integrado (IDE) *Android Studio 1.4* para o desenvolvimento do aplicativo e a IDE *NetBeans 8.0.2* para o desenvolvimento do *webservice*.

O aplicativo de gerenciamento e acompanhamento de campeonatos de futebol amador deste trabalho nomeado NATA F.C. foi desenvolvido a partir da implementação das telas utilizando as ferramentas de interface gráfica citadas anteriormente, posteriormente implementou-se a lógica de negócios para receber, processar e armazenar as informações do usuário, visando criar e configurar campeonatos de acordo com as necessidades do mesmo. Logo, foi implementado o *web service* para receber e retornar as informações cadastradas pelo usuário.

4.5.1 Desenvolvimento do aplicativo Nata F.C.

Nesta subseção são apresentados os métodos utilizados para o desenvolvimento do aplicativo.

4.5.2 Sessão do usuário

Foi implementado a classe *SessionManager* para armazenar dados do usuário globalmente através de todo o aplicativo. Foram utilizadas para armazenar os dados, preferências compartilhadas que permitem salvar e recuperar dados de pares de chaves e valores onde mesmo que o usuário encerre o aplicativo as informações serão persistentes.

Ao realizar o *login* no aplicativo é instanciado um objeto da classe *SessionManager* que recebe todos os dados do usuário, onde este objeto pode ser acessado de qualquer parte do aplicativo, como por exemplo, para buscar

os campeonatos do atual usuário. Para maior facilidade ao usuário essa sessão é salva persistindo seus dados mesmo que feche o aplicativo, com isso depois de realizado o primeiro *login* no aplicativo não será mais necessário realizar tal tarefa, ele será automaticamente redirecionado ao menu principal da aplicação. Caso queira encerrar a sessão é necessário acessar seu perfil e sair do aplicativo.

4.5.3 Informações em forma de lista

Para exibir informações em forma de lista de itens no aplicativo foi utilizado o componente de *layout RecyclerView*, como por exemplo, os campeonatos, categorias, equipes, jogos, jogadores etc.

Para utilizar o *RecyclerView* foi necessário implementar uma classe *Adapter*, que irá conseguir ligar a *view* dos itens da lista com as informações dos objetos que serão populados. Ainda no *Adapter* também foi necessário implementar uma classe *ViewHolder*, onde é declarado as *views*, como *TextView*, *ImageView*, *CheckBox* etc, que serão usadas posteriormente pelo *Adapter*. A classe *Adapter* possui três principais métodos:

- onCreateViewHolder: onde é atribuído o layout ,que possui as views, no *Adapter* que está sendo populado.
- *onBindViewHolder:* neste método que o objeto da lista é transferido para a *view*, onde é exibido os dados daquele objeto.
- *getItemCount*: esse método simplesmente retorna o número de objetos da lista;

Para implementar uma função de clique em cada item na lista, foi desenvolvido a Interface *RecyclerViewOnClickListener* com os métodos onClickListener que representa um clique simples no item e o *onLongClickListener* que representa um clique longo. Essa *Interface* foi utilizada para todos os *Adapters* utilizados no aplicativo, de modo que cada atividade implementou a *Interface* realizando ações conforme a necessidade de uso.

4.5.4 Comunicação com WebService

Para realizar chamadas ao WebService foram desenvolvidas classes que herdam a Classe *AsyncTask*, de modo que essas requisições sejam executadas em um processo diferente da *thread* principal da interface gráfica do usuário como é recomendado pela *Google* para o desenvolvimento *Android*. Foi necessário implementar classes que serializam os objetos do modelo da aplicação para serem utilizados no formado *Json* pela biblioteca *Retrofit* .

Foram desenvolvidas *Interfaces Api* com os métodos que realizam as requisições ao *WebService*, estes são chamados pelo Retrofit conforme a necessidade. Cada método da Interface Api deve ter uma anotação HTTP que forneça o método de solicitação e o URL (*Uniform Resource Locator* - Localizador Padrão de Recursos) relativo. Há cinco anotações internas: *GET, POST, PUT, DELETE* e *HEAD*. O URL relativo do recurso é especificado na anotação (GUIDES, 2016).

Abaixo um exemplo (Figura 11) do método utilizando a anotação HTTP/GET, usada no aplicativo para listar os jogadores de uma equipe específica:

Figura 11: Método utilizando a anotação HTTP/GET

```
@GET("listarJogadoresEquipeInscricao/Equipe{idEquipe}/idEquipe")
Call<List<Jogador>> listarJogadoresEquipeInscricao (@Path("idEquipe") int idEquipe);
```

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Também foram implementadas Interfaces *AsyncResponse*, para retornar para o usuário a resposta requisitada, como por exemplo listar as equipes do usuário. Para realizar a requisição desejada foi utilizado os principais métodos da *AsyncTask* como demostrado abaixo:

OnPreExecute: este método foi utilizado para criar um diálogo para informar ao usuário que o aplicativo está realizando uma determinada tarefa (Figura 12).

Figura 12: Método OnPreExecute

```
@Override
protected void onPreExecute() {
    this.dialog = ProgressDialog.show(this.context, "Buscando Equipes", " Aguarde... ", true, true);
    this.dialog.setCanceledOnTouchOutside(false);
}
```

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

doInBackground: este método foi utilizado para realizar a tarefa de chamar a requisição ao *WebService*. É neste método que é feita a solicitação da serialização do objeto , é instanciado o objeto *Retrofit* que realiza a construção da conexão na base do URL do *WebService*, e instancia o objeto serializado no método *addConverterFactory*. Por fim, é instanciado ainda a *Interface* Api do objeto para realizar a chamada desejada. Para receber a resposta vinda do *webservice* é instanciado um objeto *Call*, da biblioteca Retrofit, designada por *response* que executa o método *execute* () e o método *body* () para receber o que foi solicitado e retornar ao método onPostExecute como mostra a figura 13 abaixo.

Figura 13: Método doInBackground

```
@Override
protected List<Equipe> doInBackground(Void... params) {

    Gson gson = new GsonBuilder().registerTypeAdapter(Equipe.class, new EquipeDes()).create();

    Retrofit retrofit = new Retrofit
            .Builder()
            .baseUrl(API)
            .addConverterFactory(GsonConverterFactory.create(gson))
            .build();

    final EquipeApi equipeApi = retrofit.create(EquipeApi.class);

    try{
        Call<List<Equipe>> response = equipeApi.minhasEquipes(idUsuario);
        return response.execute().body();

    }catch (Exception e){
        e.printStackTrace();
    }

    return null;
}
```

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

onPostExecute: este método foi utilizado para receber a chamada realizada pelo *Retrofit* e retornar a atividade principal as informações ao

usuário através do objeto *delegate* da *Interface AsyncReponse* e encerrar o diálogo demonstrando que a tarefa foi finalizada , como mostra a figura 14 abaixo:

Figura 14: Método onPostExecute

```
@Override
protected void onPostExecute(List<Equipe> list){

    delegate.processFinishListEquipe(list);
    this.dialog.dismiss();

}
```

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

4.5.5 Geração dos grupos campeonato

Para configurar o modo de disputa estão disponíveis todas as opções de modo de disputa usuais e tradicionais aos campeonatos de futebol (Tabelas 2 e 3), sendo que as opções oferecidas podem ser combinadas e as fases seguintes do campeonato são geradas automaticamente após a escolha inicial do modo de disputa feita pelo usuário.

Quando o usuário optar pela opção Grupos, o aplicativo disponibiliza a opção de escolher o número de grupos variando de 2 a 8 e de selecionar quais serão os cabeças de chave de cada grupo. É disponibilizado a lista de todas as equipes participantes e assim o administrador seleciona a partir de um checkBox quais serão então os cabeças de chave.

O sorteio é realizado de forma que sendo n grupos definidos pelo usuário, as equipes selecionadas para cabeças de chave são sorteadas utilizando a classe *Random* de modo que cada uma fique atribuída a um grupo distinto. A partir das equipes restantes de maneira similar é sorteada e atribuída a um grupo de cada vez, assim garantindo que cada grupo tenha números iguais de equipes, exceto quando o número de equipes não seja divisível pelo número de grupos, ou seja, haverá um ou mais grupos com equipes a mais ou a menos. Como por exemplo, um campeonato com 8 equipes se for dividido em dois grupos, cada grupo terá quatro equipes, mas

caso tenha três grupos, dois grupos terão três equipes e um grupo terá duas equipes.

Tabela 2: Opções de modo de disputa primeira fase

Modo de disputa para 1ª fase
Grupos – Somente ida
Grupos – Ida e volta
Pontos corridos – Somente ida
Pontos corridos – Ida e volta

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Tabela 3: Opções de modo de disputa segunda fase

Modo de disputa para 2ª fase
Não possui
Oitavas de final – Somente ida
Quartas de final – Somente ida
Semifinal – Somente ida
Final – Somente ida
Oitavas de final – Ida e volta
Quartas de final – Ida e volta
Semifinal – Ida e volta
Final – Ida e volta

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

4.5.6 Métodos utilizados para gerar os jogos

- **Turno único e turno e returno:** no turno único cada equipe enfrenta seus adversários apenas uma vez. Onde se o número de equipes for ímpar todas as equipes jogaram o mesmo número de jogos com mando de campo, caso seja par haverá equipes com um jogo a mais tanto com mando de campo ou jogando fora. No turno e returno as equipes se enfrentam duas vezes alternando-se o mando de campo.
- **Rodadas e número de jogos:** sendo n o número de equipes participantes no campeonato, cada uma delas deve disputar (n -1) partidas em um turno completo, onde todas as equipes se enfrentam. Se n for par, todas as

equipes jogam em todas as rodadas e o número de rodadas será igual a (n – 1). Caso contrário, n sendo número ímpar, o número de rodadas será igual a n, e sempre haverá uma equipe sem jogar a cada rodada.

A fórmula para o número de jogos é:

j = n * (n - 1) / 2

j = jogos

n = número de equipes

4.5.7 Algoritmo para geração dos jogos

Abaixo segue o algoritmo desenvolvido para gerar os jogos do campeonato.

1- Cada equipe será representada por uma letra: A, B, C, etc. No exemplo abaixo, foram utilizadas as letras de A a H, totalizando 8 equipes.

2 - Deve-se colocar as equipes em uma matriz com 2 linhas e (n / 2) colunas.

As equipes devem ser dispostas nas linhas no sentido horário: na primeira linha da esquerda para a direita e na segunda linha da direita para a esquerda.

Como no exemplo abaixo:

A	B	C	D
H	G	F	E

Os jogos da primeira rodada são definidos pelas colunas da matriz.

- Coluna 1 (A,H), Equipe A x Equipe H;
- Coluna 2 (B,G), Equipe B x Equipe G;
- Coluna 3 (C,F), Equipe C x Equipe F;
- Coluna 4 (D,E), Equipe D x Equipe E.

4 – Para a segunda rodada, mantém- se a equipe **A** fixa e giram-se as demais equipes no sentido horário.

A	H	B	C
G	F	E	D

- Coluna 1 (A,G), Equipe A x Equipe G;
- Coluna 2 (G,F), Equipe G x Equipe F;

- Coluna 3 (B,E), Equipe B x Equipe E;
- Coluna 4 (C,D), Equipe C x Equipe D.

5 – Repete – se o processo 4 até a última rodada, ou seja, até que a matriz volte a posição original.

A	C	D	E
B	H	G	F

6 – Para determinar os mandos de campo, é necessário fazer algumas inversões nas colunas depois de todas as rodadas:

1. Inverter sempre a ordem das equipes das colunas pares; e inverter a ordem da primeira coluna apenas nas rodadas pares, como no exemplo abaixo da primeira rodada:

A	G	C	E
H	B	F	D
	(Inverter)		(Inverter)

Exemplo segunda rodada:

G	F	B	D
A	H	E	C
(Inverter)	(Inverter)		(Inverter)

Quando o número de equipes é par, como no exemplo acima, o número de jogos de cada equipe é ímpar, logo, não é possível que em turno único, cada equipe tenha o mesmo número de jogos em casa e fora. Nesse algoritmo desenvolvido as equipes com letras ímpares (A, C, E) vão ter um jogo a mais em casa (no exemplo, 4 jogos em casa e 3 fora), enquanto as equipes pares (B, D, F), vão ter um jogo a mais fora (no exemplo, 3 jogos em casa e 4 fora).

Se o número de equipes for ímpar, foi criada uma equipe “fantasma” e incluída as equipes do campeonato, deste modo quando uma equipe “real” enfrentar a equipe “fantasma” essa equipe “real” ficará sem jogar na rodada, logo, todas as equipes do campeonato ficarão sem jogar uma rodada.

4.5.8 Tabela de Classificação

Para exibir a tabela de classificação com todas as informações necessárias ordenadas conforme os critérios de desempate estabelecidos pelo aplicativo foi necessário realizar consultas no banco de dados referentes a todos os jogos disputados por cada equipe a fim de obter os seguintes dados:

- Número de vitórias: identificar jogos que a equipe fez mais gols que o adversário;
- Empates: identificar jogos que a equipe fez o mesmo número de gols que o adversário;
- Derrotas: identificar jogos que a equipe sofreu mais gols que a equipe adversária;
- Gols marcados: somar todos os gols marcados;
- Gols sofridos: somar todos os gols sofridos
- Saldo de gols: o saldo de gol é contabilizado a partir da diferença entre os gols marcados e gols sofridos;
- Pontos: o número de pontos de cada equipe é contabilizado a partir do número de vitórias e empates, sendo que a cada vitória é somado três pontos e o empate um ponto;
- Aproveitamento: o aproveitamento da equipe se refere ao número de jogos disputados e pontos contabilizados. Sendo que cada jogo vale três pontos, deve-se multiplicar o número de pontos da equipe por cem e dividir pelo número de jogos disputados multiplicados por três, assim obtendo o aproveitamento em porcentagem.

Com todas essas informações obtidas através dos jogos, as equipes são separadas por grupos, quando houverem, e ordenadas pelos seguintes critérios de desempate: número de pontos, número de vitórias, saldo de gols, gols marcados, gols sofridos e número de derrotas. Para tanto foi implementado a interface *Comparable* na classe Equipe.

Para exibir a tabela de classificação foi utilizado os componentes de layout *RecyclerView* e *TableLayout*, a fim de separar em cada tabela um grupo e as suas respectivas equipes. Onde em cada linha da tabela se encontra uma equipe e nas colunas as informações de pontos, saldo de gols etc. Cada linha da tabela possui uma ação de clique do usuário, e ao ser acionada o usuário é direcionado ao perfil da equipe selecionada.

4.5.9 Estatísticas do Campeonato

O aplicativo além de disponibilizar as informações contidas na tabela de classificação referentes a cada equipe, também dispõe estatísticas de artilheiro, melhor ataque e melhor defesa. Essas estatísticas são identificadas da seguinte maneira:

- Melhor Ataque: é necessário realizar uma busca em todos os jogos já disputados e contabilizar o número de gols marcados por cada equipe e ordena-las em ordem decrescente;
- Melhor Defesa: é necessário realizar uma busca em todos os jogos já disputados e contabilizar o número de gols sofridos por cada equipe e ordena-las em ordem crescente.
- Artilheiro: para identificar os jogadores que mais fizeram gols em todo campeonato é necessário verificar a súmula de cada um dos jogos disputados pela sua equipe e contabilizar os gols de cada um dos jogadores do campeonato e ordena-los em ordem decrescente.

4.5.10 Enquete do Campeonato

Foi implementado uma enquete para cada uma das categorias do campeonato. Essa enquete possibilita a qualquer usuário votar nos melhores e piores jogadores do campeonato. Para tal funcionalidade foi necessário criar uma tabela no banco de dados para contabilizar os números de votos de cada jogador.

A enquete foi dividida nas seguintes modalidades: melhor jogador, melhor goleiro, bola murcha, mascarado e pipoqueiro. Foi definido que cada usuário pode votar apenas uma vez em cada modalidade. Ao realizar o voto é feita uma consulta a fim de identificar se o usuário já realizou seu único voto na modalidade e caso ele já tenha votado é avisado através de uma caixa de diálogo se desejar trocar seu voto, caso não tenha votado seu voto é contabilizado ao jogador escolhido. O resultado da enquete pode ser acessado

a qualquer momento, mostrando os jogadores votados, número de votos e a porcentagem de votos em relação aos votos totais.

4.5.11 Personalização do escudo do campeonato e equipes

Cada campeonato e equipe podem adicionar um escudo próprio, porém há também um escudo padrão para as equipes que não optarem por personalizar o perfil da equipe ou para o gerenciador de campeonato que não optar por personificar seu campeonato. Para esta funcionalidade, ao clicar na imagem padrão o usuário será direcionado a sua galeria de fotos do dispositivo e selecionar qual será o escudo da equipe ou do campeonato.

Para realizar o *upload* da imagem para o *WebService* foi necessário transformar a imagem em um *array* de *byte*. No *WebService* o escudo é transformado em um objeto *Image* do *java* e salva em um diretório específico. No banco de dados é somente persistido o endereço na qual se encontra o logo no servidor.

A biblioteca Picasso foi utilizada para realizar o download das imagens através do endereço no servidor e atribuir em qual *ImageView* será mostrada a imagem.

4.5.12 Implementação do WebService Restful

Para este trabalho o *webservice* foi desenvolvido na linguagem JAVA, sendo que este recebe as requisições vindas do aplicativo. O *webservice* é o intermediário que faz a comunicação entre o banco de dados e o aplicativo.

A implementação do *webservice* foi dividida em três pacotes: *ws*, *modelo* e *dao*. O pacote *modelo* representa os dados do aplicativo, provenientes das entidades do banco de dados. As classes referentes no pacote *ws* são o *Webservice* REST, e para interagir com este foi utilizado os métodos HTTP, POST, GET e DELETE como apresentado nas imagens Figura 15 e Figura 16 respectivamente.

Figura 15: Método HTTP POST

```java
@Path("campeonatoRes")
public class CampeonatoRes {

    @POST
    @Consumes({"application/json"})
    @Path("criar/Campeonato")
    public int criarCampeonato(String campeonato) throws Exception{
        Gson g = new Gson();
        Campeonato c = (Campeonato) g.fromJson(campeonato, Campeonato.class);
        CampeonatoDAO dao = new CampeonatoDAO();
        int id = dao.criarCampeonato(c);
        return id;
    }
```

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Figura 16: Método HTTP GET

```java
@GET
@Consumes({"application/json"})
@Path("buscarEstatisticaCampeonato/Campeonato{idCampeonato}/idCampeonato/{idCategoria}/idCategoria")
public String buscarEstatisticaCampeonato(@PathParam("idCampeonato") int idCampeonato, @PathParam("idCategoria") int idCategoria) throws Exception{

    CampeonatoDAO dao = new CampeonatoDAO();
    EstatisticaCampeonato ec = dao.getEstatistica(idCampeonato, idCategoria);
    Gson gson = new Gson();
    String jsonResponse = gson.toJson(ec);
    return jsonResponse;

}
```

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Nestes métodos as informações são trocadas no formato Json, representado por *@Consumes ({"application/json"})*, e para converter os objetos Java em Json ou vice-versa foi utilizado a API Gson[9] do Google.

No pacote *dao* estão as classes de entidade e seus respectivos métodos referentes a todos os procedimentos realizados no banco de dados PostgreSQL.

4.6 TESTE DE USABILIDADE

4.6.1 Questionário SUS

[9] https://sites.google.com/site/gson/gson-user-guide

Para avaliar a usabilidade e a satisfação do aplicativo NATA F.C. utilizou-se a aplicação do questionário SUS (*System Usability Scale*) (ANEXO 1). Este questionário foi desenvolvido pela *Digital Equipment CO Ltd.*, para avaliar a usabilidade dos sistemas e produtos desenvolvidos na empresa. É um questionário simples e de rápida aplicação que demonstra uma visão geral e subjetiva da avaliação da usabilidade de um produto e também avalia a satisfação do usuário em relação ao produto (SIMÕES; MORAES, 2010).

Utiliza-se a escala *Likert* para medir as opiniões. É composto de 10 questões e cada uma tem uma escala de avaliação que está entre 1 (discordo plenamente), 2 (discordo), 3 (neutro), 4 (concordo) e 5 (concordo plenamente). As 10 questões avaliam os seguintes itens:

1. Frequência de uso do sistema;
2. Complexidade do sistema;
3. Facilidade de uso;
4. Assistência para usar o sistema;
5. Funções integradas do sistema;
6. Inconsistência do sistema;
7. Rápida aprendizagem;
8. Sistema é incômodo e complicado para usar;
9. Segurança e confiança para usar o sistema;
10. Aprendizagem de outras informações para usar o sistema.

Para calcular a pontuação do questionário, deve-se somar a contribuição de cada questão. O valor de cada contribuição muda de acordo com a característica da questão, para as questões 1, 3, 5,7 e 9, a pontuação na escala é de menos 1. Para as questões de número 2, 4, 6, 8 e 10, a pontuação na escala é de menos 5. Após determinado o valor de cada questão, é necessário somar todos os valores e multiplicar por 2,5 para obter o resultado global do SUS. Este resultado global está inserido numa escala de 0 a 100.

4.6.2 Aplicação

Foram selecionados para responder o questionário SUS, 15 usuários com conhecimento em gerenciamento de campeonatos que participam comumente de eventos esportivos, visando opiniões respaldadas na experiência dos mesmos. Para realizar a aplicação do questionário SUS utilizou-se o método analítico percurso cognitivo, que avalia uma proposta de projeto de Interação Homem Computador (IHC) no contexto de tarefas específicas do usuário. Antes de realizar a avaliação, foi criada uma fase de preparação, para se definir: os cenários de tarefas construídos a partir de uma seleção de tarefas importantes e de tarefas frequentes e a sequência “correta” de ações para completar cada tarefa. As tarefas propostas para os usuários foram as seguintes:

1. Criar um campeonato;
2. Inscrever equipe;
3. Confirmar inscrições das equipes;
4. Configurar modo de disputa;
5. Atualizar datas/horários e resultados;
6. Visualizar campeonato;
7. Buscar e acompanhar campeonato.

Depois de realizado o percurso cognitivo os usuários responderam o questionário SUS, que gerou uma pontuação referente à avaliação do aplicativo Nata F.C. Os resultados das avaliações estão expressos em um gráfico referente à média de cada uma das 10 questões respondidas pelos usuários e outro referente à pontuação final da avaliação do aplicativo de cada usuário (ambos estão disponíveis na subseção 10.2 da seção resultados).

4.7 APLICATIVOS SEMELHANTES

Após o término do desenvolvimento do aplicativo de gerenciamento de campeonatos de futebol amador NATA F.C. comparou-se o mesmo com outros

aplicativos semelhantes disponíveis para *download* no Google Play[10]. Os aplicativos escolhidos são Bolla - Gerador de Chaves e App Liga e Criação de Torneios que possuem um grande número de downloads no Google Play. Estes foram escolhidos por possuírem funcionalidades semelhantes ao do aplicativo deste trabalho.

Foi feita uma análise das funcionalidades dos aplicativos escolhidos e se estes atendiam as necessidades de seus usuários. Isto foi possível devido ao acesso e análise das resenhas de usuário sobre os aplicativos que estão disponíveis na página de download de cada um dos aplicativos escolhidos no Google Play. Observou-se que algumas das necessidades dos usuários para se criar e gerenciar um campeonato não eram atendidas, tais como: estatística sobre jogadores e times não estavam disponíveis, dados da súmula não estavam acessíveis para o usuário, datas e horários não podiam ser cadastradas e quando podiam era uma funcionalidade paga, há poucas opções de modo de disputa de campeonatos.

Em comparação aos aplicativos analisados observaram-se funcionalidades exclusivas ao aplicativo deste trabalho, sendo elas: opção de separar o campeonato em várias categorias, as equipes podem buscar e se inscrever no campeonato que querem disputar, as estatísticas e informações da súmula estão disponíveis para o usuário torcedor, administrador de equipe e de campeonato, o aplicativo disponibiliza mais de 30 opções de modo de disputa, datas e horários podem ser cadastrados e estão acessíveis a todos os usuários.

Enfim, devido ao aplicativo NATA F.C. ter sido desenvolvido de acordo com padrão de disputa brasileiro, este pode assim atender melhor os usuários que gerenciam campeonatos amadores baseados neste modo de disputa, isto tendo em vista que os dois aplicativos analisados para efeito de comparação são aplicativos estrangeiros e podem possuir algumas diferenças na maneira de criar e gerenciar campeonatos.

[10] https://play.google.com/store

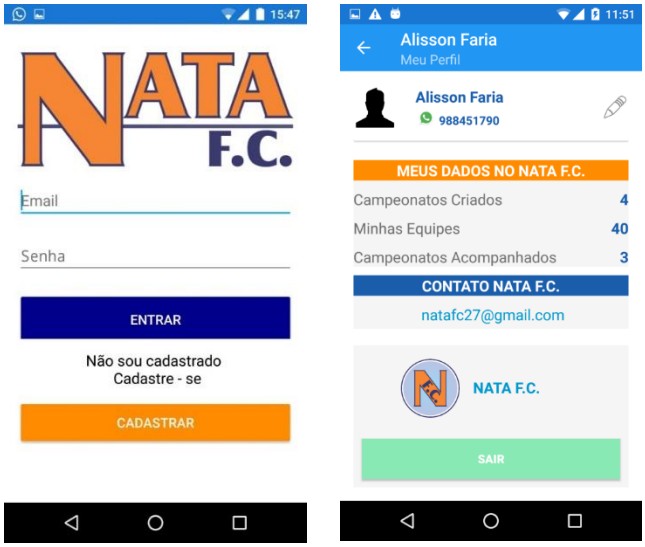
5 RESULTADOS

Neste capítulo são exibidas as telas finais do aplicativo Nata F.C. e como as funcionalidades do *app* podem ser executadas pelo usuário, bem como os resultados das avaliações feitas pelos participantes do teste de usabilidade em que foi utilizado o questionário SUS.

5.1 TELAS E FUNCIONALIDADES DO NATA F.C.

Nesta subseção são apresentadas as telas implementadas bem como as funcionalidades do aplicativo. A figura 17 apresenta a tela de *login* do aplicativo para autenticação do usuário e a tela do perfil do usuário. As informações solicitadas são o endereço de *e-mail* e senha, que são enviadas ao servidor para verificação no banco de dados, caso o usuário ainda não esteja cadastrado no aplicativo, este deverá realizar o cadastro, onde serão solicitadas informações que são utilizadas para criar o perfil do usuário. Depois de efetuado o cadastro o usuário deve voltar à tela de *login*, realizar o *login* com seu *e-mail* e senha recém-cadastrados, para assim acessar as funcionalidades do aplicativo gerenciador de campeonatos.

Figura 17: Tela Login/ Perfil do usuário



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

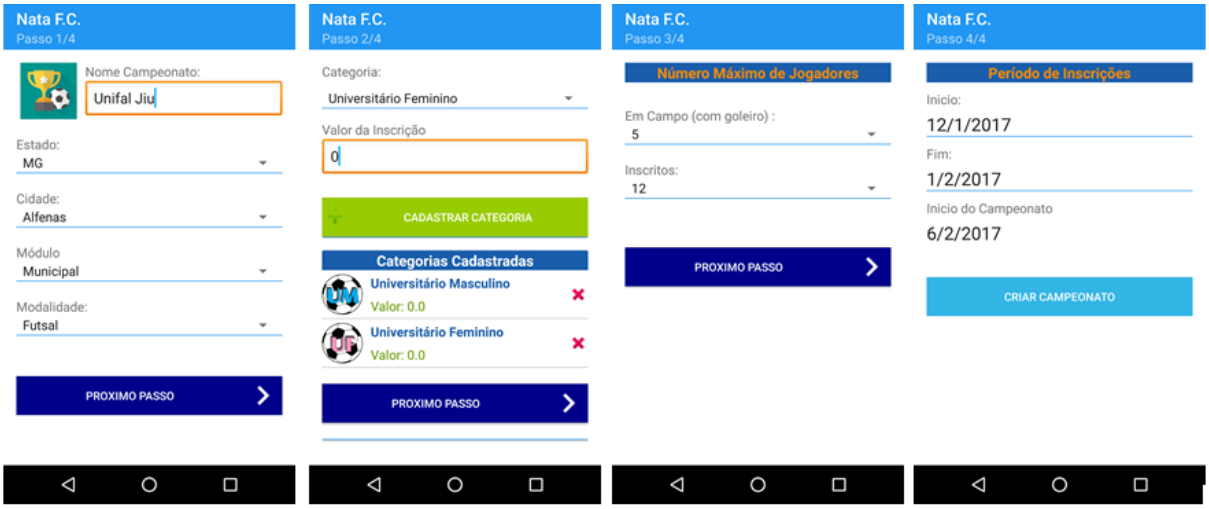
Para acessar as funcionalidades do *app* o usuário é direcionado para a tela de menu principal apresentada na figura 18, nesta tela o usuário pode optar por criar e gerenciar campeonatos, inscrever sua equipe em torneios vigentes ou mesmo acompanhar informações do campeonato que ele deseja seguir.

Figura 18: Menu principal

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Para criar o campeonato o usuário deve seguir alguns passos necessários para esta finalidade como demonstrado na figura 19. As etapas para se gerar um campeonato são: cadastro do campeonato, escolha das categorias, determinação do número de jogadores e período de inscrições, estas informações só podem ser editadas pelo usuário gerenciador do campeonato.

Figura 19: Sequência de passos para criar campeonato



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Depois de realizados esses passos as informações fornecidas pelo usuário são enviadas para o servidor para que elas sejam armazenadas no banco de dados. Com a confirmação da criação do campeonato o aplicativo retorna para interface de usuário as opções de cadastrar as equipes no campeonato ou aguardar que as equipes sejam cadastradas por outros usuários. O usuário administrador do campeonato ao acessar o item meus campeonatos do menu têm retornado a sua tela os campeonatos por ele criados e tem a opção de editar informações como datas, horários e resultados dos jogos, podendo também confirmar as inscrições das equipes que solicitaram participar do seu campeonato, pode ainda configurar o modo de disputa do campeonato por ele administrado e tendo ainda a possibilidade de inscrever uma equipe no seu campeonato como mostra a figura 20.

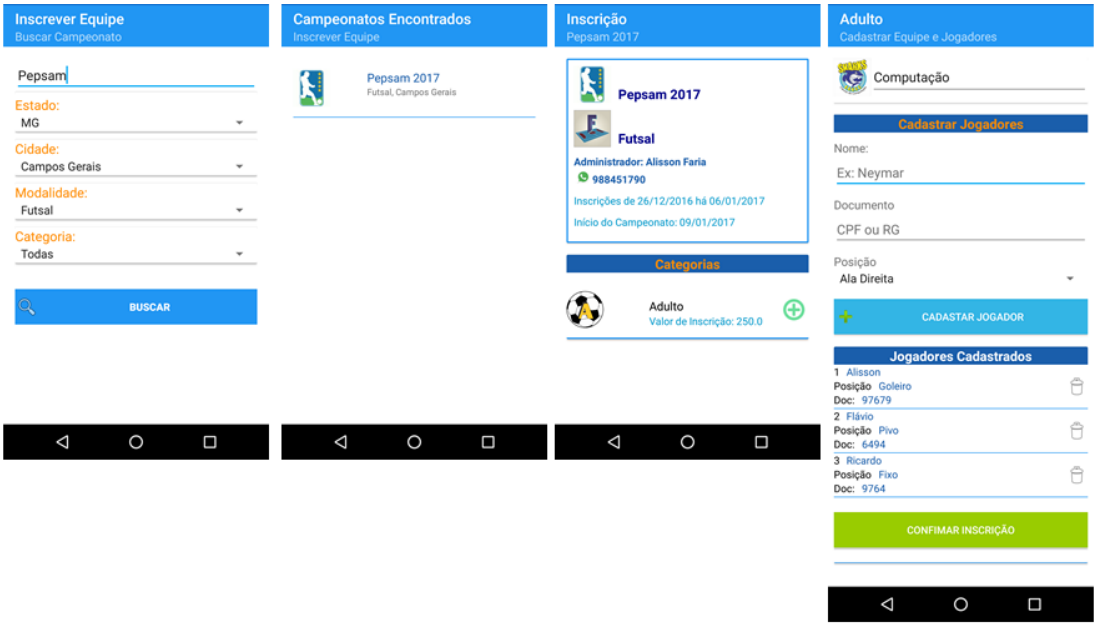
Figura 20: Opções fornecidas ao administrador do campeonato

Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Para que as equipes sejam cadastradas o usuário deve realizar uma busca do campeonato, local, modalidade e categoria em que desejam competir como apresenta a figura 21.

Esta busca é feita no banco de dados que retorna uma lista dos campeonatos disponíveis para inscrição, com isso o usuário deve apenas selecionar qual o campeonato deseja disputar e a categoria em que a equipe será inscrita, um mesmo usuário pode inscrever mais de uma equipe por campeonato ou inscrever a mesma equipe em mais de um campeonato podendo também editar os jogadores da equipe inscrita. O usuário também deve cadastrar os jogadores e as posições dos mesmos, respeitando o número máximo de jogadores permitido pelo campeonato, o administrador da equipe ainda pode personalizar sua equipe inserindo um logo para seu time e terá a opção de acessar e editar o perfil da sua equipe.

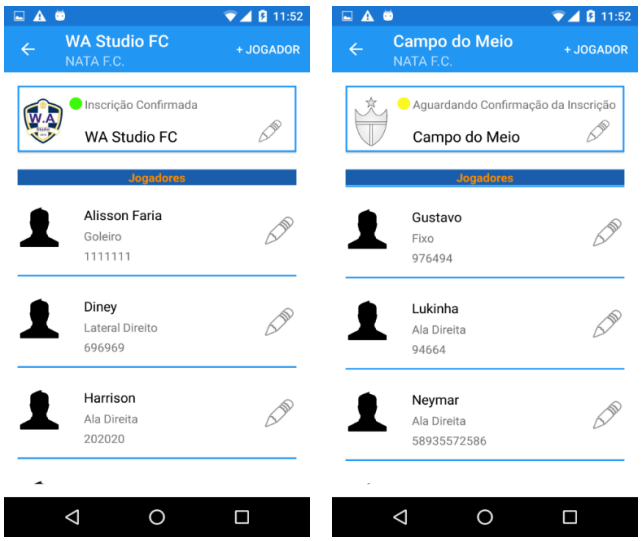
Figura 21: Passos necessários para inscrições de equipes



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

A inscrição da equipe só será efetivada quando o usuário gerenciador do campeonato confirmar as informações dos jogadores cadastrados e o pagamento da inscrição (caso haja) feito pela equipe validando assim a participação da equipe no campeonato, a confirmação da inscrição da equipe pode ser visualizada no perfil do time, como mostra a figura 22.

Figura 22: Confirmação da inscrição no perfil da equipe

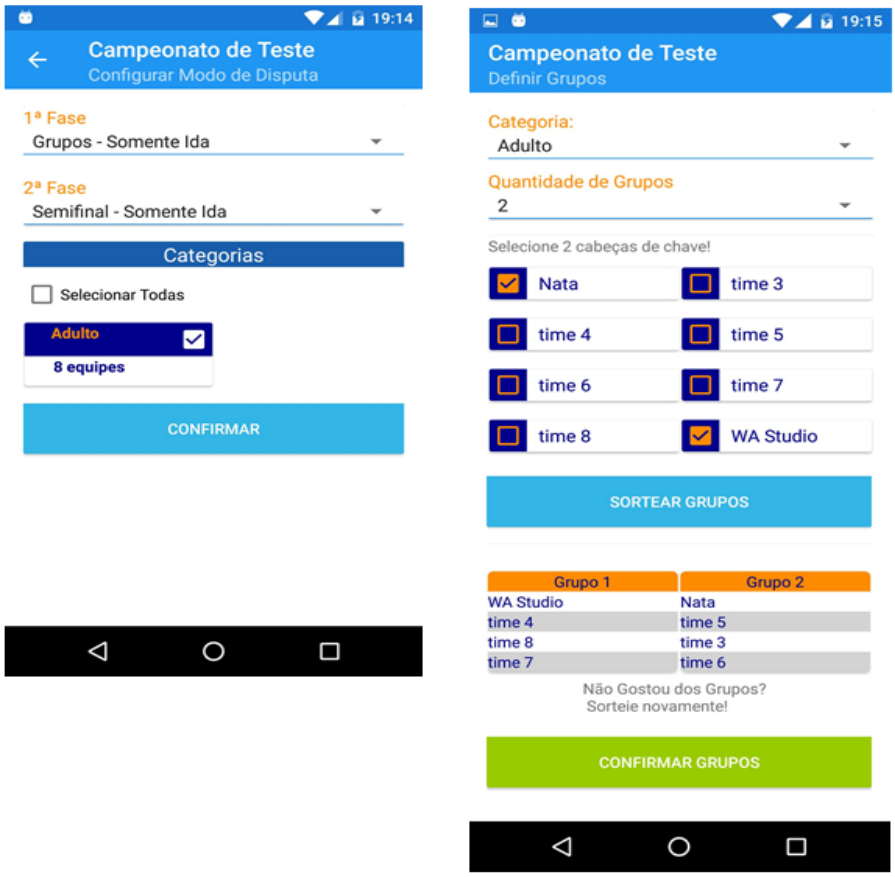


Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

É apresentada na figura 23 a opção de configuração do modo de disputa que pode ser configurado de acordo com a necessidade do usuário, onde este tem as opções de selecionar modos de disputas diferentes para as duas fases (1ª e 2ª fase) do campeonato. Encontram-se todas as alternativas referentes a modo de disputa usual e tradicional aos campeonatos de futebol, sendo que as opções oferecidas podem ser combinadas e as fases seguintes do campeonato são geradas automaticamente após a escolha inicial do modo de disputa feita pelo usuário.

Também está salientado na figura 23, a seleção dos cabeças de chave de cada grupo, caso o usuário escolha a opção grupos na tela configurar modo de disputa.

Figura 23: Configurar modo de disputa

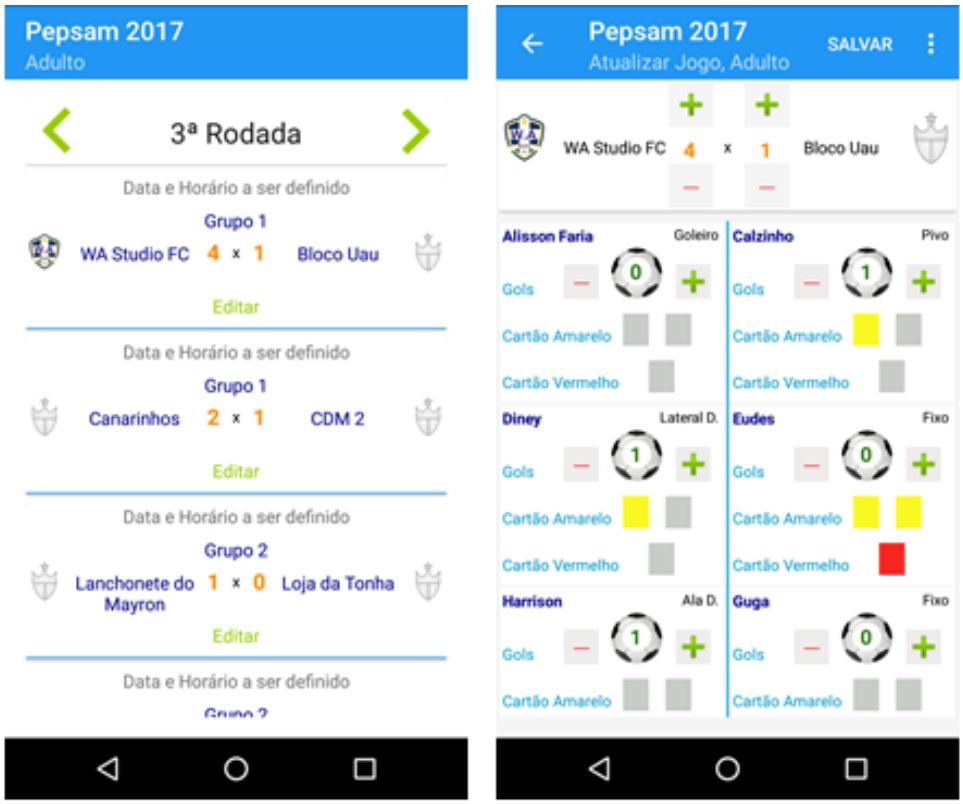


Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Com a definição do modo de disputa os jogos do campeonato podem ser gerados. Cada jogo gerado tem uma súmula e cada jogador (das duas

equipes) tem uma súmula individual com informações referentes aquele jogo. Depois de gerados os jogos, o usuário administrador do campeonato define datas e horários para realização dos mesmos, sendo que os jogos são ordenados de acordo com a data e o horário, o administrador do campeonato pode também editar a súmula adicionando às informações dos jogos e jogadores tais como placar, cartões e gols, como exibido na figura 24. Para tanto foi desenvolvido uma interface de forma a listar todos os jogadores das duas equipes onde é possível editar individualmente as informações dos jogadores. O aplicativo Nata F.C. permite também que em um jogo eliminatório o usuário possa informar o resultado dos pênaltis.

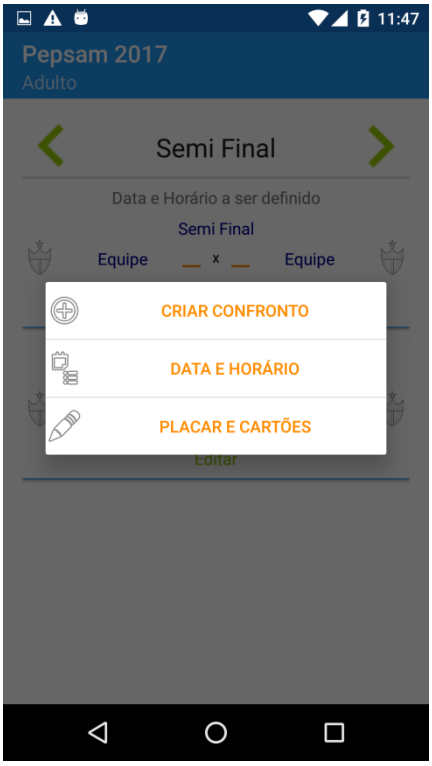
Figura 24: Alterar informações da súmula



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

A figura 25 mostra a tela que permite ao usuário criar os confrontos das fases finais do campeonato. Estes confrontos são criados baseados nas informações da tabela de classificação, onde o usuário pode selecionar as duas equipes de cada jogo conforme o critério de seleção de times seguido pelo campeonato.

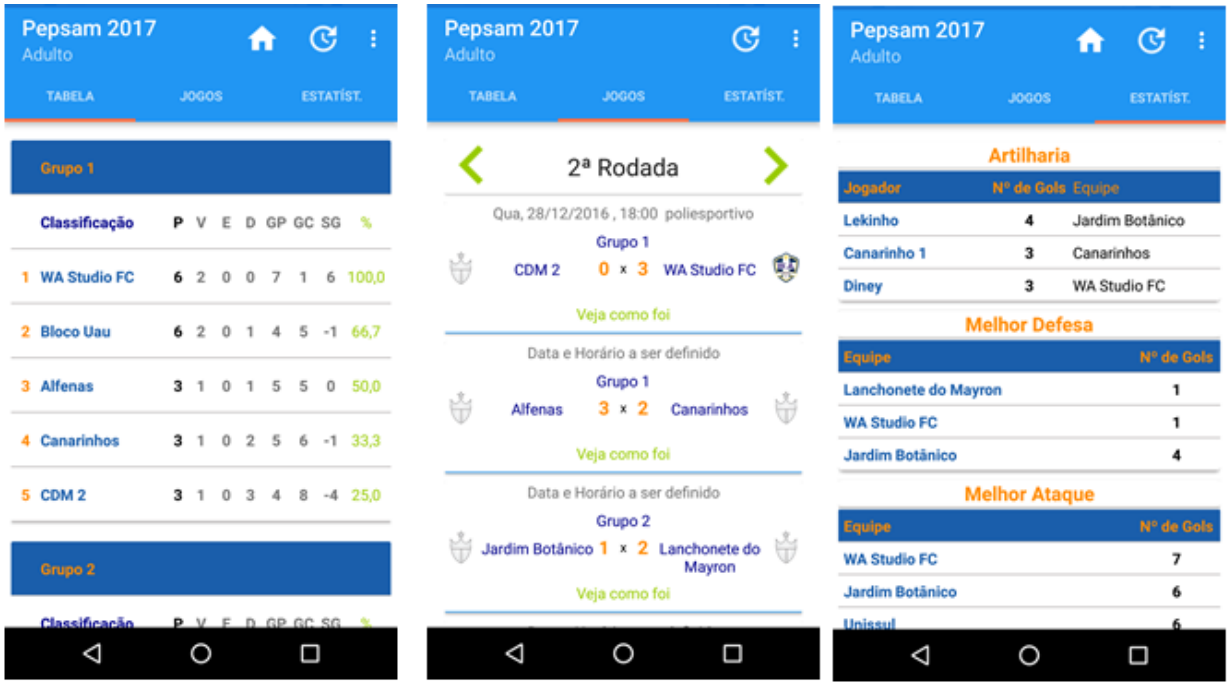
Figura 25: Criar confrontos das fases finais



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

O aplicativo apresenta as informações em tabelas de classificação, como apresentado na figura 26. A tabela de classificação obedece a critérios de desempate, sendo estes: número de pontos, número de vitórias, saldo de gols, gols marcados e gols sofridos. No aplicativo Nata F.C. os usuários que acompanham o campeonato tem acesso as informações da súmula de cada jogo, este disponibiliza ainda estatísticas sobre o desempenho das equipes e dos jogadores .

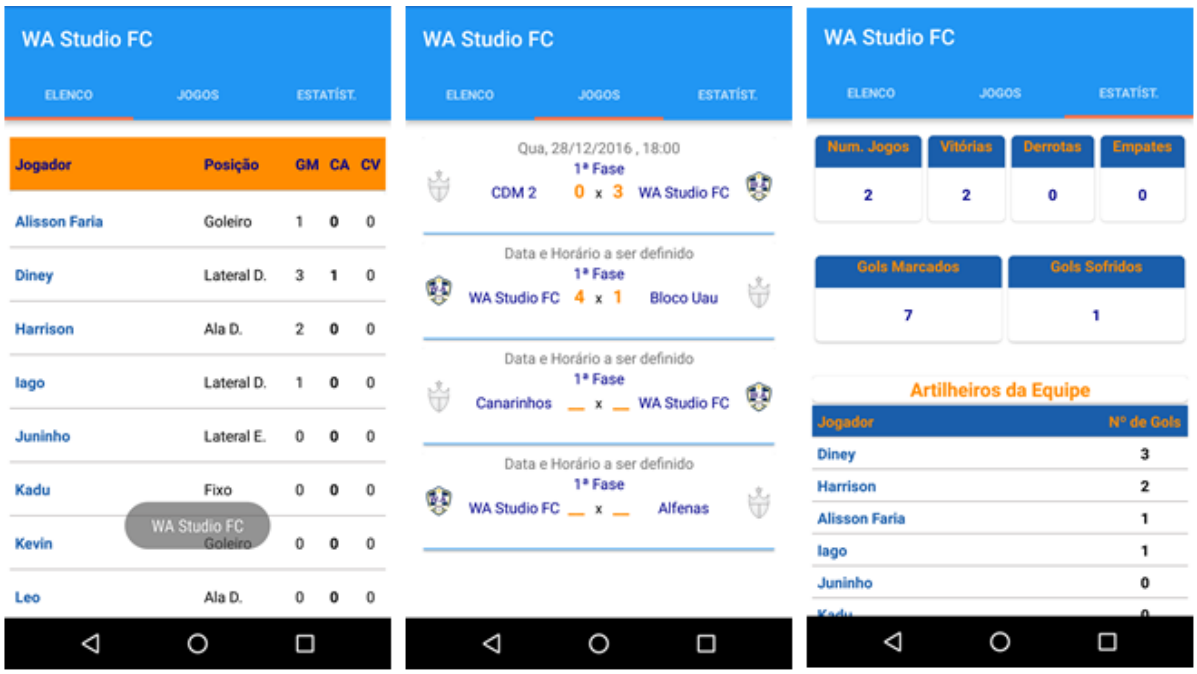
Figura 26: Informações disponibilizadas pelo aplicativo para os tipos de usuários



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Cada equipe inscrita pode acessar informações relativas aos seus jogadores, aos jogos disputados pela equipe no campeonato e estatística geral da equipe, como apresentado na Figura 27.

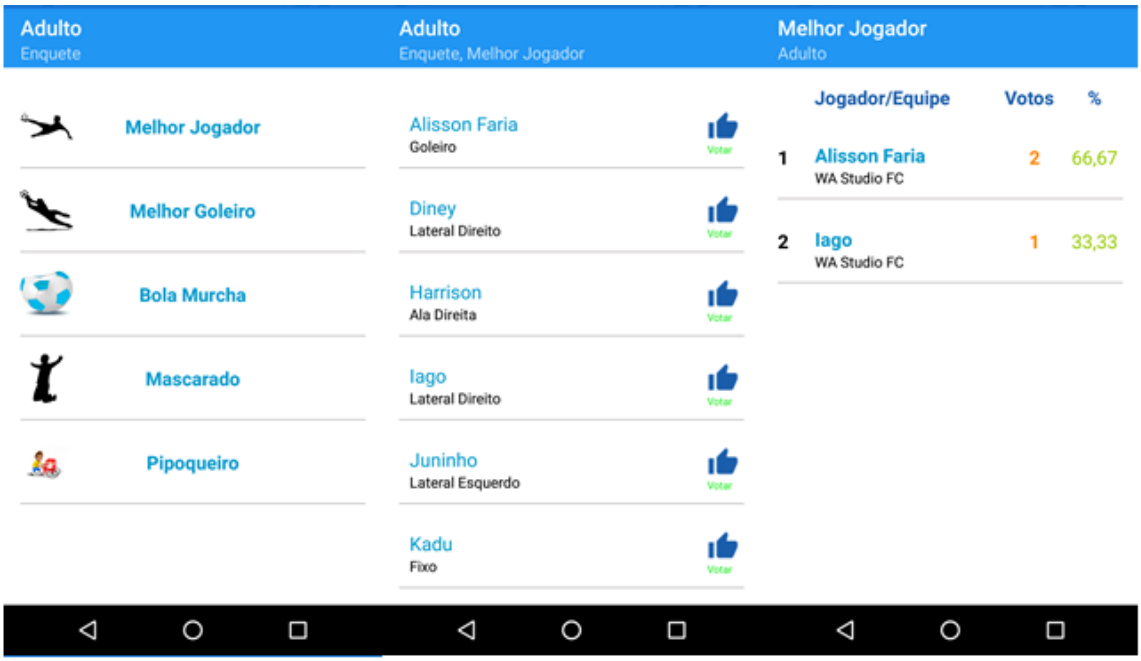
Figura 27:Informações disponíveis por equipe



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

No aplicativo Nata F.C. o usuário tem a opção de escolher e votar nos jogadores do campeonato que mais se destacam em quesitos comuns a torneios de futebol, tais como: melhor jogador, melhor goleiro, bola murcha, mascarado e pipoqueiro, proporcionando assim maior interação e entretenimento a todos os usuários envolvidos no campeonato. Entretanto cada usuário poderá votar somente uma vez no jogador escolhido em cada um dos quesitos listados acima. A Figura 28 mostra as telas da enquete apresentadas ao usuário, onde são disponibilizadas as opções de votação ou a visualização do resultado.

Figura 28: Enquete sobre os jogadores

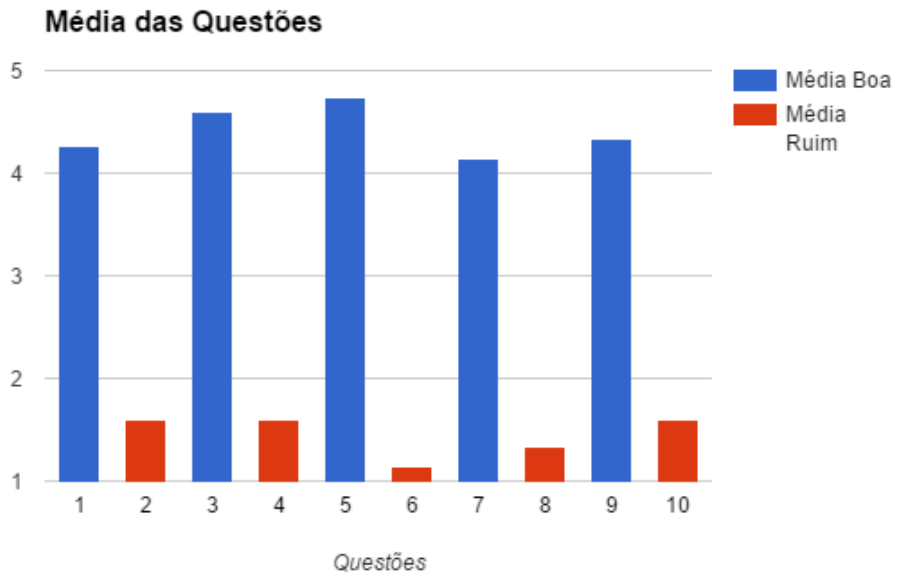


Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

5.2 TESTE DE USABILIDADE

Nesta subseção estão expressos os resultados do teste de usabilidade, em que foi utilizado o questionário SUS. Foram selecionados 15 usuários para realizar tarefas no aplicativo, visando testar as funcionalidades do Nata F.C., após isto, os usuários responderam ao questionário que gerou uma pontuação das avaliações feitas pelos mesmos. As figuras 29 e 30 abaixo exibem os gráficos da média de cada questão respondida pelo usuário e as pontuações finais das avaliações realizadas, respectivamente.

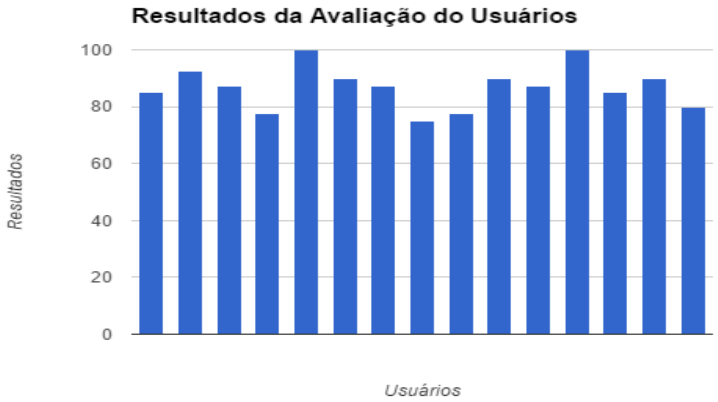
Figura 29: Média das questões



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

No gráfico acima estão às médias de cada questão respondida pelos 15 usuários participantes da avaliação. Em azul estão expressas as médias das questões que avaliam positivamente o aplicativo Nata F.C., sendo que as médias boas quanto mais próximas do valor 5 melhor a avaliação do *app*, já em vermelho estão as médias que avaliam negativamente o aplicativo, estas quanto mais distantes do valor 1 pior será a avaliação do aplicativo.

Figura 30: Resultados da avaliação dos usuários



Fonte: FARIA, Alisson José Oliveira de; GONZAGA, Flávio Barbieri.(2016)

Acima estão as pontuações finais das avaliações do aplicativo feitas pelos 15 usuários. Em geral, o aplicativo foi muito bem avaliado pelos usuários participantes obtendo como média de avaliação 87 pontos, com um desvio padrão próximo a +/- 7, 2.

6 CONCLUSÃO E TRABALHOS FUTUROS

O aplicativo de gerenciamento e acompanhamento de futebol amador desenvolvido oferece maior facilidade, praticidade e confiabilidade na criação, gerenciamento e acesso a informações de um campeonato. O sistema foi desenvolvido de modo a permitir que todas as etapas necessárias para se gerar um torneio esportivo possam ser realizadas no aplicativo, permitindo também que o administrador de uma equipe possa inscrever a mesma utilizando o *app.* O aplicativo ainda disponibiliza as informações de datas e horários de jogos, tabela de classificação, resultado dos jogos e estatísticas dos times e jogadores.

As formas de se criar e gerir uma competição esportiva atualmente está alheia a toda tecnologia e facilidade existente, tendo em vista que a maioria dos torneios esportivos são criados e administrados de forma manual, desde a fase de inscrições até o sorteio dos jogos. Com isso é comum haverem erros e perdas por parte do gerenciador do campeonato, o que pode causar questionamentos e comprometer a confiabilidade frente ao participante e torcedor do campeonato. Mas para resolver os problemas associados a isto, foi desenvolvido o aplicativo Nata F.C. que cria campeonatos amadores de forma rápida e eficiente, permitindo que equipes que estejam em cidades distantes da cidade sede do campeonato possam visualizar e se inscrever nas competições, ou mesmo um torcedor que não possa comparecer no jogo pode acessar as informações deste por meio do aplicativo.

Por esse motivo pode-se concluir que os objetivos de se desenvolver um sistema capaz de criar e gerenciar e acompanhar um campeonato de futebol amador foi alcançado. Entretanto como trabalhos futuros serão implementados alguns requisitos que são considerados essenciais para maior interação do usuário com o aplicativo, tais como:

- Implementação de um chat onde os usuários possam discutir sobre os fatos e resultados dos jogos;
- Permitir que o jogador possa se identificar e personalizar seu perfil do usuário por meio de fotos;
- Permitir que o usuário escolha os critérios utilizados na tabela de classificação;
- Divulgar os resultados em redes sociais;

- Salvar os dados do usuário no dispositivo, para que este possa trabalhar off-line.

Com essas novas implementações acredita-se que o aplicativo Nata F.C., se tornaria cada vez mais completo garantindo uma maior satisfação dos seus usuários.

REFERÊNCIAS BIBLIOGRÁFICAS

AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D. **Compilers, Principles,** Services. pp. 28-35., 2003.

ALVES, Márcia Maria; BATTAIOLA, André Luiz. **Design Centrado no Usuário e concepções pedagógicas como guia para o desenvolvimento da animação educacional. Revista Brasileira de Design da Informação**, São Paulo, v. 11, n. 1, pp. 21-35, 2014;

AMARAL FILHO, Wilson Henriques do. **Arquitetura Cliente/Servidor Orientada a Objeto**. Tese de Mestrado, DES-IME – Departamento de Engeharia de Sistemas, Instituto Militar de Engenharia, 1993.

ANATEL, disponível em: https:// anatel.gov.br/institucional acesso em: 23/11/2016 as 15:00.

ASYNC. **Developer**, disponível em: https://developer.android.com/reference/android/os/AsyncTask.html acesso em: 24/11/2016 as 16:12

BATTISTI, Júlio. **SQL Server 2000: Administração e Desenvolvimento**. 2ª edição. Rio de Janeiro. Editora Axcell Books, 2001.

BRAUDE, Eric. **Projeto de Software:** Da programação à arquitetura: Uma abordagem baseada em Java. Porto Alegre. Editora S.A. Bookman, 619p., 2005;

CBF- **Confederação Brasileira de Futebol**, disponível em:http://www.cbf.com.br/ acesso em: 21/11/2016 as 14:45.

CHRISTEL, Michael; KANG Kyun C. **"Issues in Requirements Elicitation".** Technical Report CMU/SEI-92-TR-012. CMU / SEI. Software Engineering Institute. Retrieved January 14, 2012.

COSTA, Luciana Ferreira da; RAMALHO, Francisca Arruda. **A usabilidade nos estudos de uso da informação: em cena usuários e sistemas interativos de informação. Revista Perspectivas em Ciências da Informação**, Belo Horizonte, v. 15, n.1, pp.92-117, 2010;

DEITEL, Paul J.; DEITEL, Abbey; DEITEL, Harvey; MORGANO, Michael. **Android para programadores: Uma Abordagem Baseada em Aplicativos.** Porto Alegre. Editora S.A. Bookman, 512p., 2013;

DEVELOPER, disponível em: https://developer.android.com/index.html acesso em: 23/11/2016 as 15:45

FIFA - Federação Internacional de Futebol. Disponível em: http://www.fifa.com/ acessado em: 10/10/2016 as 17:25

GARTNER- **Executive Summary: Capturing Business Value From Mass-Market Mobile Technologies.** Stamford, 2016. Disponível em: https://www.gartner.com/doc/1779628?ref=SiteSearch&sthkw=MOBILE%20APPLICATION%20MARKET&fnl=search&srcId=1-3478922254 acesso em: 17/11/2016 as 16:11.

GATNER, disponível em: http://www.gartner.com/technology/home.jsp acesso em: 20/02/2017 as 11:35.

GOOGLE PLAY, **App Liga** disponível em: https://play.google.com/store/apps/details?id=com.mileyenda.manager&hl=pt_BR acesso em: 06/12/2016 as 21:00.

GOOGLE PLAY, **Bolla – Gerador de Chaves** disponível em: https://play.google.com/store/apps/details?id=com.inova.bolla&hl=pt_BR acesso em: 06/12/2016 as 19:45.

GSON, disponível em: https://sites.google.com/site/gson/gson-user-guide acesso em: 25/11/2016 as 16:00.

GUIDES, disponível em https://guides.codepath.com/android/Consuming-APIs-with-Retrofit acesso em: 24/11/2016 as 15:12

HAMAD, Hatem; SAAD, Motaz; ABED, Ramzi. Performace Evolution of RESTful Web Services for Mobile Devices. **International Arab Journal of e-Technology**, Palestina, v.1, n.3, p74., 2010.

IDC – International Data Corporation. **IDC Releases: O mercado brasileiro de celulares voltou a crescer**. São Paulo, 2015. Disponível em: http://br.idclatin.com/releases/news.aspx?id=2083 acessado em: 17/11/2016 as 15:50.

IDC- International Data Corporation . **IDC press release: China to Become the Largest Market for Smartphones in 2012 with Brazil and India Forecast to Join the Top 5 Country-Level Market**. Framingham. 2012.

IEEE – Institute of Eletrical and Electronic Engineers: Standards Glossary of Software Engineering Terminology. Nova Iorque, 84p., 1990;

JSON, disponível em: http://www.json.org/json-pt.html acesso em: 25/11/2016 as 15:30.

NIELSEN, Jakob. **Usability Engineering.** Califórnia. Editora Morgan Kaufmann, 362p., 1993.

NORMAS TÉCNICAS, Associação Brasileira de. **Requisitos Ergonômicos para Trabalho de Escritório com Computadores:** Orientações sobre Usabilidade. Edição digital. Rio de Janeiro. Editora ABNT, 21p., 2002;

NURSEITOV,Nurzhan; PAULSON, Michael; REYNOLDS, Randall ; IZURIETA, Clemente. **Case Study: Comparison of JSON and XML Data Interchange Formats**. CAINE 2009: pp.157-162., 2009.

PFLEEGER, Shari Lawrence. **Engenharia de Software:** Teoria e Prática. 2ª edição. São Paulo. Editora Prentice Hall, 560p., 2004;

PGADMIN, disponível em: https://www.pgadmin.org/ acesso em: 24/11/2016 as 17:15.

PICASSO, disponível em: http://square.github.io/picasso/ acesso em: 24/11/2016 as 18:00.

POSTGRESQL, disponível em: https://www.postgresql.org.br/ acesso em: 23/11/2016 as 22:12

PRESSMAN, Roger S. **Engenharia de Software:** Uma abordagem profissional. 7ª edição. Porto Alegre. Editora AMGH, 780p., 2011;

REZENDE, José Ricardo. **Sistemas de Disputa para Competições Esportivas**: Campeonatos & Torneios. São Paulo. Editora Phorte, 168p., 2007.

ROCHA, Ana Regina Cavalcanti da. **Qualidade de Software:** Teoria e Prática. São Paulo. Editora Prentice Hall. 303p., 2001.

SIMÕES, Aliana Pereira; MORAES, Ana Maria de. **Aplicação do Questionário SUS para Avaliar a Usabilidade e a Satisfação do Software de EAD**. Congresso Internacional de Ergonomia e Usabilidade de Interfaces Humano-Computador**.** Rio de Janeiro, 6p., 2010.

Solutions and open problems. In: ICAPS 2003 workshop on Planning for Web

SOMMERVILLE, Ian; SAWYER, Pete. **Requirements Engineering:** Good Pratice Giuide. Nova Jersey. Editora Wiley, 404p., 1997.

SQUARE RETROFIT, disponível em http://square.github.io/retrofit/ acesso em:24/11/2016 as 14:46

SRIVASTAVA, Biplav; KOEHLER, Jana. **Web service composition-current Techniques.** Boston. Editora Addison Wesley, 796p., 1986.

TIBES, Chris Mayara dos Santos; DIAS, Jessica David. **Aplicativos móveis desenvolvidos para a área da saúde no Brasil: revisão integrativa da literatura.** Revista Mineira de Enfermagem**.** Belo Horizonte, v.18, n.2, pp. 471-478; 2014.

Anexo

Teste de Usabilidade no App Nata F.C.						
		Discordo fortemente				Concordo fortemente
Item	Parâmetro para concordância	1	2	3	4	5
1	Eu acho que gostaria de usar o sistema com frequência.					
2	Eu achei o sistema desnecessariamente complexo.					
3	Eu achei que o sistema fácil de usar.					
4	Eu acho que precisaria de ajuda do suporte técnico para ser capaz de usar este sistema.					
5	Eu achei as que várias funções deste sistema foram bem integrados.					
6	Eu achei que havia muitas inconsistências no sistema.					
7	Eu imagino que a maioria das pessoas aprenderia a usar este sistema muito rapidamente.					
8	Eu achei o sistema muito complicado de usar.					
9	Eu me senti muito confiante usando o sistema.					
10	Eu precisaria aprender muitas coisas antes que eu pudesse usar bem o sistema.					