



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Alex Adriel Barreira de Souza
Isaque Dias Coelho

REDE SOCIAL - SOCIALDIBRE

Americana, SP
2018



FACULDADE DE TECNOLOGIA DE AMERICANA
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Alex Adrihel Barreira de Souza
Isaque Dias Coelho

REDE SOCIAL - SOCIALDIBRE

Trabalho de Conclusão do Curso Superior de
Tecnologia em Análise e Desenvolvimento de
Sistemas, sob a orientação do Prof. Dr. Kleber
de Oliveira Andrade

Área de concentração: Engenharia de Software

Americana, SP
2018

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

S713r SOUZA, Alex Adrihel Barreira de

Rede social - SocialDibre. / Alex Adrihel Barreira de Souza, Isaque
Dias Coelho. – Americana, 2018.

124f.

Monografia (Curso de Tecnologia em Análise e Desenvolvimento de
Sistemas) - - Faculdade de Tecnologia de Americana – Centro Estadual de
Educação Tecnológica Paula Souza

Orientador: Prof. Dr. Kleber de Oliveira Andrade

1 Engenharia de software 2. Android - aplicativos I. COELHO, Isaque
Dias II. ANDRADE, Kleber de Oliveira III. Centro Estadual de Educação
Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.3.05

681.519

ALEX ADRIHEL BARREIRA DE SOUZA
ISAQUE DIAS COELHO

REDE SOCIAL - SOCIALDIBRE

Trabalho de Graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia - FATEC/AMERICANA.

Área de concentração: Engenharia de Software.

Americana, 10 de dezembro de 2018.

Banca Examinadora:



Kleber de Oliveira Andrade
Doutor

Faculdade de Tecnologia de Americana



Wagner Siqueira Cavalcante

Mestre

Faculdade de Tecnologia de Americana



Rossano Pablo Pinto

Mestre

Faculdade de Tecnologia de Americana

RESUMO

O presente trabalho se concentrou na análise e desenvolvimento de uma rede social, acessível via dispositivo *mobile* Android, que permita a conexão e interação de jogadores amadores e/ou profissionais de futebol, a fim de que os mesmos possam marcar partidas, disputar posições de um *ranking* e até mesmo se ler notícias sobre este esporte.

A construção da rede social se deu através de procedimentos da engenharia de software, destacando-se a análise de requisitos funcionais e não funcionais e projeção de diagramas UML como caso de uso e diagrama de classe além do diagrama de entidade relacionamento para o banco de dados.

A aplicação, permite marcar/receber desafios de partidas, apresentar um ranking, criar times e adicionar jogadores ao time, tudo construído de forma a ser acessível de qualquer lugar com acesso à internet, tendo *design* intuitivo ao usuário e sendo escalável para uma grande quantidade de usuários.

Palavras Chave: Rede Social; Futebol; Jogadores; Desenvolvimento de Software.

ABSTRACT

The present work focused on the analysis and development of a social network, accessible via Android mobile device, allowing the connection and interaction of amateur players and / or football professionals, so that they can mark matches, compete for positions of one ranking and even read news on this sport.

The construction of the social network took place through procedures of software engineering, highlighting the analysis of functional and non-functional requirements and projection of UML diagrams as use case and class diagram in addition to the relationship entity diagram for the database.

The application allows you to score / receive match challenges, present a ranking, create teams and add players to the team, all built in a way that is accessible from anywhere with internet access, having intuitive user design and being scalable to a large amount of users.

Keywords: Social Network; Football; Players; Software Development.

SUMÁRIO

1	Introdução.....	16
2	Projeto do Sistema	18
2.1	Levantamento de Requisitos	18
2.1.1	Requisitos Funcionais.....	19
2.1.2	Requisitos Não Funcionais	20
2.2	Recursos e Ferramentas.....	22
3	Modelagem.....	26
3.1	Diagramas de Casos De Uso	26
3.2	Documentação dos Casos de Uso.....	29
3.3	Diagrama de Classe do <i>WebService</i>.....	42
3.4	Diagrama de Entidade e Relacionamento	51
3.4.1	Dicionário de Dados.....	52
3.5	Diagrama de Classe do Aplicativo	57
4	Desenvolvimento	93
4.1	Etapas de Desenvolvimento	93
4.1.1	<i>Sprint 1</i>.....	94
4.1.2	<i>Sprint 2</i>.....	96
4.1.3	<i>Sprint 3</i>.....	98
4.1.4	<i>Sprint 4</i>.....	100
4.1.5	<i>Sprint 5</i>.....	102

4.2	Interfaces de Usuário	104
5	Considerações Finais	120
	Referências.....	122

LISTA DE FIGURAS

Figura 1 - Diagrama de caso de uso da tela entrar no sistema e cadastrar usuário.	27
Figura 2 - Diagrama de caso de uso da tela principal do aplicativo	27
Figura 3 - Diagrama de caso de uso da tela de Perfil do Usuário.	28
Figura 4 - Diagrama de caso de uso da tela Perfil do Time.	28
Figura 5 - Diagrama de classe de usuário	42
Figura 6 - Diagrama de classe de Times.	44
Figura 7 - Diagrama de classe de amizade.	45
Figura 8 - Diagrama de classe de participação do usuário no time.	46
Figura 9 - Diagrama de classe de Notícias.	47
Figura 10 - Diagrama de classe de Partida.	48
Figura 11 - Diagrama de classe de Desafio.	49
Figura 12 - Diagrama de classe de Log	50
Figura 13 - Diagrama de Entidade e Relacionamento.	51
Figura 14 - Diagrama de classe de login.	57
Figura 15 - Diagrama de classe de Register.	60
Figura 16 - Diagrama de classe de perfil do usuário.	63
Figura 17 - Diagrama de classe de times do usuário.	66
Figura 18 - Diagrama de classe de amizades do usuário.	68
Figura 19 - Diagrama de classe de perfil do time.	70
Figura 20 - Diagrama de classe de jogadores do time.	72
Figura 21 - Diagrama de classe de histórico do time.	75
Figura 22 - Diagrama de classe de notícias.	77
Figura 23 - Diagrama de classe de desafio.	80

Figura 24 - Diagrama de classe de <i>ranking</i>	82
Figura 25 - Diagrama de classe de <i>Match</i>	85
Figura 26 - Diagrama de classe de marcar partida/escolher campo.....	87
Figura 27 - Diagrama de classe de marcar partida/selecionar meu time.	88
Figura 28 - Diagrama de classe de marcar partida/selecionar time adversário.	90
Figura 29 - Diagrama de classe de marcar partida/marcar desafio.	91
Figura 30 - <i>Burndown</i> 1	95
Figura 31 - <i>Burndown</i> 2.....	97
Figura 32 - <i>Burndown</i> 3.....	99
Figura 33 - <i>Burndown</i> 4.....	101
Figura 34 - <i>Burndown</i> 5.....	103
Figura 35 – Captura da tela de login e da tela de apresentação (<i>splash screen</i>)....	104
Figura 36 - Captura da Tela de cadastro.	105
Figura 37 - Captura da Tela de Notícias.	106
Figura 38 - Captura da Tela de Desafios.	107
Figura 39 - Captura da Tela de <i>Ranking</i>	108
Figura 40 - Captura da Tela de Acessar Partidas.	109
Figura 41 - Captura da Tela Criar Partidas.....	110
Figura 42 - Captura da Tela Perfil Pessoal.....	112
Figura 43 - Captura da Tela Amizades.	113
Figura 44 - Captura da Tela Meus Times.	114
Figura 45 - Tela Criar Time.	115
Figura 46 - Tela Perfil do Time.	116
Figura 47 - Tela Jogadores do Time.....	117
Figura 48 - Tela Histórico de Partidas.	118

LISTA DE TABELAS

Tabela 1 – Comparativo entre Fintta, Appito e SocialDibre	16
Tabela 2 - Requisitos funcionais do projeto.	19
Tabela 3 – Requisitos não funcionais do projeto	21
Tabela 4 – Caso de uso “Entrar no Sistema / Cadastrar Usuário”	29
Tabela 5 – Caso de uso Principal “Ver Notícias Específicas”	29
Tabela 6 – Caso de uso Principal “Ver Desafios Específicos”	30
Tabela 7 - Caso de uso Principal “Acessar <i>Ranking</i> ”	31
Tabela 8 - Caso de uso Principal “Acessar Partidas”	32
Tabela 9 - Caso de uso Principal “Ver Perfil”	33
Tabela 10 - Caso de uso Principal “Criar Partidas”	34
Tabela 11 - Caso de uso Perfil do Usuário “Ver <i>Feed</i> ”	35
Tabela 12 - Caso de uso Perfil do Usuário “Editar Perfil”	35
Tabela 13 - Caso de uso Perfil do Usuário “Ver Amizades”	36
Tabela 14 - Caso de uso Perfil do Usuário “Ver Time Específico”	37
Tabela 15 - Caso de uso Perfil do Usuário “Criar Time”	38
Tabela 16 - Caso de uso Perfil do Time “Ver <i>Feed</i> ”	38
Tabela 17 - Caso de uso Perfil do Time “Editar Perfil”	39
Tabela 18 - Caso de uso Perfil do Time “Ver Jogadores”	40
Tabela 19 - Caso de uso Perfil do Time “Ver Histórico”	41
Tabela 20 - Diagrama de classe de usuários	43
Tabela 21 - Diagrama de classe de usuários	43
Tabela 22 - Diagrama de classe de times	44
Tabela 23 - Diagrama de classe de amizade	45

Tabela 24 - Diagrama de classe de participação do usuário no time.	46
Tabela 25 - Diagrama de classe de Notícias.	47
Tabela 26 - Diagrama de classe de Partida.	48
Tabela 27 - Diagrama de classe de Desafio.	49
Tabela 28 - Diagrama de classe de Log.	50
Tabela 29 - Dicionário de Dados da entidade <i>User</i>	52
Tabela 30 - Dicionário de Dados da entidade <i>Relationship</i>	53
Tabela 31 - Dicionário de Dados da entidade <i>Participates</i>	53
Tabela 32 - Dicionário de Dados da entidade <i>News</i>	53
Tabela 33 - Dicionário de Dados da entidade <i>Log</i>	54
Tabela 34 - Dicionário de Dados da entidade <i>Team</i>	54
Tabela 35 - Dicionário de Dados da entidade <i>Challenge</i>	55
Tabela 36 - Dicionário de Dados da entidade <i>Match</i>	55
Tabela 37 - Dicionário de Dados da entidade <i>Address</i>	56
Tabela 38 – Métodos do Diagrama de classe de login.	58
Tabela 39 - Métodos do Diagrama de classe de login.	58
Tabela 40 - Métodos do Diagrama de classe de login.	58
Tabela 41 - Métodos do Diagrama de classe de login.	59
Tabela 42 - Métodos do Diagrama de classe de login.	59
Tabela 43 - Métodos do Diagrama de classe de login.	60
Tabela 44 – Métodos do Diagrama de classe de Register.	61
Tabela 45 - Métodos do Diagrama de classe de Register.	61
Tabela 46 - Métodos do Diagrama de classe de Register.	62
Tabela 47 - Métodos do Diagrama de classe de Register.	62
Tabela 48 - Métodos do Diagrama de classe de Register.	62

Tabela 49 - Métodos do Diagrama de classe de Register.	63
Tabela 50 – Métodos do Diagrama de classe de perfil do usuário.	64
Tabela 51 - Métodos do Diagrama de classe de perfil do usuário.	64
Tabela 52- Métodos do Diagrama de classe de perfil do usuário.	65
Tabela 53 - Métodos do Diagrama de classe de perfil do usuário.	65
Tabela 54 – Métodos do Diagrama de classe de times do usuário.....	66
Tabela 55 - Métodos do Diagrama de classe de times do usuário.	67
Tabela 56- Métodos do Diagrama de classe de times do usuário.....	67
Tabela 57 - Métodos do Diagrama de classe de times do usuário.	67
Tabela 58 – Métodos do Diagrama de classe de amizades do usuário.....	68
Tabela 59 - Métodos do Diagrama de classe de amizades do usuário.....	69
Tabela 60- Métodos do Diagrama de classe de amizades do usuário.	69
Tabela 61 – Métodos do Diagrama de classe de perfil do time.	70
Tabela 62 - Métodos do Diagrama de classe de perfil do time.	71
Tabela 63- Métodos do Diagrama de classe de perfil do time.	71
Tabela 64- Métodos do Diagrama de classe de perfil do time.	72
Tabela 65 – Métodos do Diagrama de classe de jogadores do time.	73
Tabela 66 - Métodos do Diagrama de classe de jogadores do time.	73
Tabela 67- Métodos do Diagrama de classe de jogadores do time.	74
Tabela 68- Métodos do Diagrama de classe de jogadores do time.	74
Tabela 69- Métodos do Diagrama de classe de jogadores do time.	74
Tabela 70 – Métodos do Diagrama de classe de histórico do time.....	75
Tabela 71 - Métodos do Diagrama de classe de histórico do time.....	76
Tabela 72- Métodos do Diagrama de classe de histórico do time.	76
Tabela 73 – Métodos do Diagrama de classe de news.....	78

Tabela 74 - Métodos do Diagrama de classe de notícias.	78
Tabela 75- Métodos do Diagrama de classe de News.	78
Tabela 76- Métodos do Diagrama de classe de News.	79
Tabela 77 – Método do Diagrama de classe de Challenge.	80
Tabela 78 - Método do Diagrama de classe de Challenge.	81
Tabela 79- Método do Diagrama de classe de Challenge.	81
Tabela 80 – Método do Diagrama de classe de <i>ranking</i>	82
Tabela 81 - Método do Diagrama de classe de <i>ranking</i>	83
Tabela 82- Método do Diagrama de classe de <i>ranking</i>	83
Tabela 83- Método do Diagrama de classe de <i>ranking</i>	84
Tabela 84 - Método do Diagrama de classe de <i>ranking</i>	84
Tabela 85 – Métodos do Diagrama de classe de Match.	85
Tabela 86 - Métodos do Diagrama de classe de Match.	86
Tabela 87 – Método do Diagrama de classe de marcar partida/escolher campo.	87
Tabela 88 - Método do Diagrama de classe de marcar partida/escolher campo.	87
Tabela 89 – Métodos do Diagrama de classe de marcar partida/selecionar meu time.	89
Tabela 90 - Métodos do Diagrama de classe de marcar partida/selecionar meu time.	89
Tabela 91 – Método do Diagrama de classe de marcar partida/selecionar time adversário.	90
Tabela 92 - Método do Diagrama de classe de marcar partida/selecionar time adversário.	91
Tabela 93 – Métodos do Diagrama de classe de marcar partida/escolher campo. ...	92
Tabela 94 – Planejamento da <i>Sprint 1</i>	94

Tabela 95 - Planejamento da <i>Sprint 2</i>	96
Tabela 96 - Planejamento da <i>Sprint 3</i>	98
Tabela 97 - Planejamento da <i>Sprint 4</i>	100
Tabela 98 - Planejamento da <i>Sprint 5</i>	102

1 Introdução

O futebol é um componente que influencia a cultura brasileira (GOERG, 2010), moldando os aspectos sociais e as formas de relacionamentos das pessoas. Há ações desenvolvidas pelos aficionados pelo esporte para que se sintam integrantes de uma mesma comunidade como a promoção de grandes campeonatos e até mesmo simples partidas entre grupo de amigos. Segundo uma pesquisa realizada no Brasil no ano de 2016 pela empresa Ipsos, 16,9 milhões de brasileiros têm interesse no futebol, do qual, 76% destes possuem acesso à internet e 48% são usuários de *smartphone* (IPSOS, 2018).

Atualmente, entre os sistemas mais populares que operam neste segmento pode-se citar o website “Fintta” que permite ao usuário a locação online de um campo, trazendo uma lista dos campos mais próximos que podem ser alugados de modo avulso, mensalmente e até mesmo ser pago de forma dividida entre os jogadores. Outro sistema que atua no setor é o aplicativo “Appito”, que oferece aos seus usuários a possibilidade de encontrar jogos próximos, marcar partidas com outros times, ter um *ranking* da partida e até mesmo manter um perfil por onde pode-se definir suas características.

Ao aplicativo desenvolvido nesse projeto deu-se o nome “SocialDibre”, resultado da união entre a palavra “Social” pelo significado de relacionamento entre indivíduos, e a palavra “Dibre” que é uma informalidade da palavra “Drible” comum entre os jovens, principalmente entre os praticantes e admiradores do futebol.

Levando estes aspectos em consideração, foram descritas na Tabela 1 as principais diferenças entre o Fintta, Appito e o aplicativo desenvolvido neste trabalho.

Tabela 1 – Comparativo entre Fintta, Appito e SocialDibre

Funcionalidade	Fintta	Appito	SocialDibre
Facilidade de uso	X		X
Perfil de jogador		X	X
Pagamento online	X		
Estatísticas de jogador		X	
Funcionalidade	Fintta	Appito	SocialDibre

Inter-relação entre usuários		X	X
Informes ao usuário sobre atualidades do futebol			X
Sistema de Ranking		X	X
Perfil de time			X
Agenda das partidas		X	X
Página de gestão de locação de campos	X	X	
Acesso online	X	X	X

Fonte: Elaborado pelo autor

Este trabalho tem por motivação, criar um meio em que apreciadores de futebol, principalmente jogadores amadores, possam se conectar e interagir entre si, podendo através de times marcar partidas com outros times, acompanhando pelo *feed* de notícias as ações das suas amizades, as notícias e até mesmo post patrocinados.

Quanto aos objetivos específicos são:

- Utilizar a metodologia *Scrum* e expor os procedimentos realizados.
- Permitir a conexão entre apreciadores e jogadores amadores de futebol.
- Proporcionar a criação de grupos virtuais (times).
- Propiciar o agendamento de partidas reais.
- Possibilitar a pontuação de partidas realizadas.
- Exibir *ranking* de times e usuários.
- Apresentar notícias de futebol.

2 Projeto do Sistema

Este capítulo detalha o processo do desenvolvimento do aplicativo, levando em consideração que sua construção foi realizada através de um conjunto de requisitos funcionais e não funcionais sob recursos e ferramentas familiares aos desenvolvedores deste projeto.

2.1 Levantamento de Requisitos

A engenharia de requisitos (RE – *Requirements Engineering*) é o processo de descobrir, analisar, documentar e verificar requisitos de um sistema. Um requisito pode ser definido como uma descrição dos serviços fornecidos pelo sistema e as suas restrições operacionais (SOMMERVILLE, 2011). Tradicionalmente, os requisitos são divididos em dois tipos: requisitos funcionais e requisitos não funcionais.

Para estabelecer a prioridade dos requisitos, foram adotadas as denominações “essencial”, “importante” e “desejável”.

- **Essencial** é o requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, que têm que ser implementados impreterivelmente.
- **Importante** é o requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implementados, mas, se não forem, o sistema poderá ser implantado e usado mesmo assim.
- **Desejável** é o requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis podem ser deixados para versões posteriores do sistema, caso não haja tempo hábil para implementá-los na versão que está sendo especificada.

2.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem o que o sistema deve fazer, isto é, definem a funcionalidade desejada do software (SOMMERVILLE, 2011). A Tabela 2 apresenta os requisitos funcionais deste projeto.

Tabela 2 - Requisitos funcionais do projeto.

Identificação	Requisito Funcional	Prioridade
RF001	Cadastrar Usuário	Essencial
RF002	Logar usuário	Essencial
RF003	Ver Notícias de futebol	Essencial
RF004	Abrir Notícia específica de futebol	Importante
RF005	Ver Desafios recebidos	Essencial
RF006	Aceitar Desafios recebidos	Essencial
RF007	Rejeitar Desafios recebidos	Essencial
RF008	<i>Ver Ranking</i>	Essencial
RF009	<i>Ver Ranking</i> de Times	Essencial
RF010	<i>Ver Ranking</i> de Usuário	Importante
RF011	Ver Partidas	Essencial
RF012	Criar Partida	Essencial
RF013	Aceitar Partida	Essencial
RF014	Rejeitar Partida	Essencial
RF015	Ver Agenda	Essencial
RF016	Ver Perfil Pessoal/Jogador	Essencial
RF017	Editar Perfil	Importante
RF018	<i>Ver Feed</i>	Importante
RF019	Ver Amizades	Essencial
RF020	Ver Seguidores	Importante
RF021	Ver seguindo	Importante
RF022	Adicionar amigo	Importante
RF023	Remover amigo	Importante
RF024	Ver lista de Times	Importante
RF025	Ver Perfil Time	Importante
RF026	Ver lista de jogadores	Importante

Identificação	Requisito Funcional	Prioridade
RF027	Adicionar Jogador ao Time	Essencial
RF028	Remover Jogador do Time	Essencial
RF029	Editar Perfil do Time	Desejável
RF030	Ver Histórico	Importante
RF031	Sair do time	Essencial
RF032	Criar time	Essencial

Fonte: Elaborado pelo autor

2.1.2 Requisitos Não Funcionais

“Os requisitos não funcionais são aqueles não diretamente relacionados às funções específicas fornecidas pelo sistema” (SOMMERVILLE, 2011). A Tabela 3 apresenta os requisitos não funcionais deste projeto.

Para estabelecer uma descrição mais clara, foram adotadas as categorias “usabilidade”, “Confiabilidade”, “Desempenho”, “Segurança”, “Distribuição”, “Padrões” e “Hardware e Software”.

- **Usabilidade:** Esta seção descreve os requisitos não funcionais associados à facilidade de uso da interface com o usuário, material de treinamento e documentação do sistema.
- **Confiabilidade:** Esta seção descreve os requisitos não funcionais associados à frequência, severidade de falhas do sistema e habilidade de recuperação das mesmas, bem como à corretude do sistema.
- **Desempenho:** Esta seção descreve os requisitos não funcionais associados à eficiência, uso de recursos e tempo de resposta do sistema.
- **Segurança:** Esta seção descreve os requisitos não funcionais associados à integridade, privacidade e autenticidade dos dados do sistema.
- **Distribuição:** Esta seção descreve os requisitos não funcionais associados à distribuição da versão executável do sistema.

- **Padrões:** Esta seção descreve os requisitos não funcionais associados a padrões ou normas que devem ser seguidos pelo sistema ou pelo seu processo de desenvolvimento.
- **Hardware e Software:** Esta seção descreve os requisitos não funcionais associados ao hardware e software usados para desenvolver ou para executar o sistema.

Tabela 3 – Requisitos não funcionais do projeto.

Identificação	Requisito não funcional	Categoria	Prioridade
RNF001	Responsividade	Usabilidade	Essencial
RNF002	Cores leves	Usabilidade	Essencial
RNF003	Letras legíveis	Usabilidade	Essencial
RNF004	Listas bonitas	Usabilidade	Essencial
RNF005	Carregamento evidente	Usabilidade	Essencial
RNF006	Erros e Exceções entendíveis	Usabilidade	Essencial
RNF007	Tratamento de <i>input</i>	Confiabilidade	Essencial
RNF008	Carregamento de erro	Confiabilidade	Essencial
RNF009	Lista do <i>Feed</i> rápida	Desempenho	Essencial
RNF010	Lista do <i>Ranking</i> rápida	Desempenho	Essencial
RNF011	Lista de Notificações rápida	Desempenho	Essencial
RNF012	Lista de Agenda rápida	Desempenho	Essencial
RNF013	Lista do Amizades rápida	Desempenho	Essencial
RNF014	Lista de Times rápida	Desempenho	Essencial
RNF015	Lista do Histórico rápida	Desempenho	Essencial
RNF016	Lista de Conquistas rápida	Desempenho	Essencial
RNF017	Autenticidade de usuário	Segurança	Essencial
RNF018	Privacidade do usuário	Segurança	Essencial
RNF019	Integridade dos dados	Segurança	Essencial
RNF020	Distribuição do aplicativo	Distribuição	Essencial
RNF021	<i>Design</i> limpo	Padrões	Essencial

Identificação	Requisito não funcional	Categoria	Prioridade
RNF022	Comunicação simplificada	Padrões	Essencial
RNF023	Ambiente de execução limitado	Hardware e Software	Essencial

Fonte: Elaborado pelo autor

2.2 Recursos e Ferramentas

Esta seção contempla as ferramentas de programação e os conceitos necessários para o desenvolvimento do aplicativo.

- **Web Service Rest:** Este padrão de comunicação foi adotado porque atualmente pela alta demanda em reutilização de processos principalmente por dispositivos móveis, é comumente utilizada a tecnologia *Web Service* que, segundo a Organização Internacional que coordena os padrões de tecnologias utilizado na internet [WTC] – *World Wide Web Consortium*, é um software projetado para suportar interação máquina-a-máquina interoperáveis sobre uma rede, ou seja, é uma arquitetura que permite a comunicação entre máquinas independentemente da plataforma. A tecnologia *Web Service* permite estabelecer uma comunicação através de um dos dois tipos de dados suportados, *SOAP* ou *Rest*, embora a arquitetura *SOAP* possa utilizar qualquer meio de transporte existente para enviar suas requisições, a arquitetura *Rest* é mais vantajosa quando há limitação de recurso de banda e de processamento (FERREIRA e MOTA, 2014).
- **PHP:** esta linguagem foi utilizada para a construção do webservice porque é gratuita e de código aberto, além de permitir dinamismo e praticidade na sua utilização, suportando também diversos Bancos de Dados, desde PostgreSQL, MySQL, SQL Server e até mesmo Oracle (NIEDERAUER, 2011).
- **Android:** foi abordado este sistema operacional pois é amplamente utilizado além de ser baseado no *kernel* do Linux, tendo como uma de suas maiores características a capacidade de poder ser utilizado em inúmeras

plataformas, desde *smartphones*, *tablets*, *netbooks*, relógios e outros, tornando o mesmo o sistema operacional para aparelhos portáteis mais utilizado do mundo (GOMES, Rafael C.; FERNANDES, Jean Alves R.; FERREIRA, Vinicius C., 2012).

- **JAVA:** esta ferramenta foi utilizada pois “Java é uma linguagem de programação independente de plataforma desenvolvida pela *Sun Microsystems*. Originalmente feita para integrar circuitos de eletrodomésticos, ganhou a Internet, sendo utilizada largamente na *Web* com objetivo de dinamizar sites e integrar servidores” garantindo a vantagem de poder construir nativamente aplicações para o sistema Android (HOPSON e INGRAM, 1998).
- **MySQL:** Este Sistema Gerenciador de Banco de Dados (SGBD) foi escolhido para armazenar todas as informações pelo fato de que, entre os SGBD Relacionais mais conhecidos do mercado, o MySQL é o que mais se destaca, sua compatibilidade com PHP e sua licença livre lhe concederam o grande reconhecimento, pode-se destacar também que o MySQL tem todas as características dos principais SGBDs pagos existentes no mercado, e por isso é encontrado até mesmo em grandes empresas como o próprio *Facebook* (MILANI, 2006).
- **Xampp:** Para executar localmente o webservice em PHP o XAMPP foi utilizado por ser um servidor *Web* livre que usa os principais servidores *open source* do mercado, incluindo FTP (*File Transfer Protocol*), SGBD MySQL e Apache, que suporta as linguagens PHP e Perl. É compatível com os sistemas operativos Windows, Linux, Mac OS X e Solaris. Para utilizar o XAMPP, é apenas necessário fazer download, descompactar e executar. Este servidor foi desenvolvido com o objetivo de ser uma ferramenta que permita aos programadores *Web* testar os seus códigos no seu próprio computador sem que seja necessário aceder à Internet. Contudo, atualmente, é por vezes usado como servidor *Web*, adicionando-lhe uma ferramenta de proteção para as partes vitais do programa (SOUZA, 2013).
- **PHP Microframework Slim 3:** Afim de facilitar o desenvolvimento do web servisse utilizou-se o *Slim*, isso porque este é um *micro-framework* PHP que ajuda a escrever rapidamente aplicações web e APIs simples, porém poderosas. Em essência, o *Slim* é um *dispatcher* que recebe uma

solicitação HTTP, invoca uma rotina de retorno de chamada apropriada e retorna uma resposta HTTP (SLIM, 2018).

- **Visual Studio Code:** Para codificação do webservice o Visual Studio Code foi utilizado por ser um editor de texto multi plataforma disponibilizado pela Microsoft para o desenvolvimento de aplicações web, com suporte principalmente a ASP.NET 5 e Node.js. Conhecer essa ferramenta é importante para os desenvolvedores que pretendem trabalhar em ambientes multi plataforma, por exemplo, seguindo a tendência de desenvolvimento web em ambiente Mac e Linux, ao mesmo tempo em que mantém o projeto compatível com Windows (DEVMEDIA, 2018).
- **Composer:** O Composer é uma ferramenta para gerenciamento de dependências em PHP que foi utilizado na construção do webservice. Ele permite declarar as bibliotecas das quais um projeto depende e as gerenciará (instalará / atualizará) (COMPOSER, 2018).
- **Android Studio:** É um ambiente de desenvolvimento integrado (IDE) para desenvolver para a plataforma Android. Foi anunciado em 16 de maio de 2013 na conferência Google I/O. Android Studio é disponibilizado gratuitamente sob a Licença Apache 2.0 (DUCROHET, 2013).
- **Astah Community:** Para produzir a diagramação de forma mais facilitada foi utilizado o Astah, pois este é uma ferramenta CASE (*Computer-Aided Software Engineering*) vastamente utilizada para a modelagem de soluções de software fazendo uso da UML. Ela dispõe de uma versão free “community” e de uma versão paga “professional”. Astah é desenvolvido na plataforma JAVA e permite que seja modelado soluções de software fazendo uso de uma linguagem que seja mais próxima do pensamento humano. Ela admite que os modelos criados sejam transformados em códigos, conceito conhecido na computação como engenharia à frente. Astah também aceita que códigos já criados, sejam transformados em modelos UML, através da engenharia reversa (NETO, 2017).
- **Enterprise Architect:** Para gerar a diagramação mais facilitada diretamente do código fonte do aplicativo foi utilizado o Enterprise Architect, que é uma plataforma gráfica para auxiliar equipes a construir sistemas

robustos através da geração de relatórios e documentos de alta qualidade (SPARXSYSTEMS, 2018).

- **MySQL Workbench:** este é uma ferramenta visual para arquitetos de Banco de Dados, pelo qual foi possível realizar modelagem de dados, desenvolvimento de SQL e ferramentas de administração e configuração de servidores, usuários e Banco de Dados(MYSQL, 2018).
- **Photoshop:** Para auxiliar na formalização do Design geral do aplicativo foi utilizado o Photoshop, que é um dos programas da Adobe, que tem como funcionalidade base a edição de imagens. Só está disponível para Windows ou Mac OS X. Este software é líder de mercado na edição e manipulação de imagens, mantendo o desenvolvimento contínuo. Para além da edição de imagens este software permite também criar imagens (SOUZA, 2013).
- **CorelDraw:** Afim de auxiliar na formação do logotipo essa ferramenta de design vetorial se tornou muito útil, isso pois é uma das mais completa no mercado, ajudando a trazer resultados com os desenhos gráficos. Está sempre em atualização, e é ideal para planejar desenhos e visualizar projetos. (CORELDRAW, 2018).
- **GitHub:** Este foi utilizado para o versionamento do código, isso porque este é um serviço web que oferece diversas funcionalidades extras aplicadas ao *git*, com ele é possível armazenar códigos em nuvem e revisar entre as versões do código, gerenciar projetos, gerenciar equipes, se conectar com outros usuários e manter atualizada a documentação do projeto (GITHUB, 2018).
- **API Bing News:** Para buscar as notícias de futebol da internet foi utilizado esta API de Pesquisa do *Bing News*, esta fornece uma experiência semelhante ao *Bing News*, retornando uma lista de artigos de notícias que o *Bing* considera relevantes para a consulta de um usuário (MICROSOFT, 2018).

3 Modelagem

Na fase da modelagem é feita a documentação do aplicativo, se trata de diagramas que facilitam na compreensão e documentação dos projetos orientados a objetos de forma padronizada (BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar, 2006).

A documentação deste trabalho utilizará a *Unified Modeling Language*¹ (UML) para modelar os casos de uso e os diagramas de classes.

3.1 Diagramas de Casos De Uso

Os diagramas de caso de uso descrevem um cenário de funcionalidades do ponto de vista do usuário, catalogando os requisitos funcionais do sistema. Dentro do diagrama são retratados os atores (representado pelos bonecos), as funcionalidades (representadas pelos balões com a ação escrita por dentro) e as relações (representadas pelas linhas).

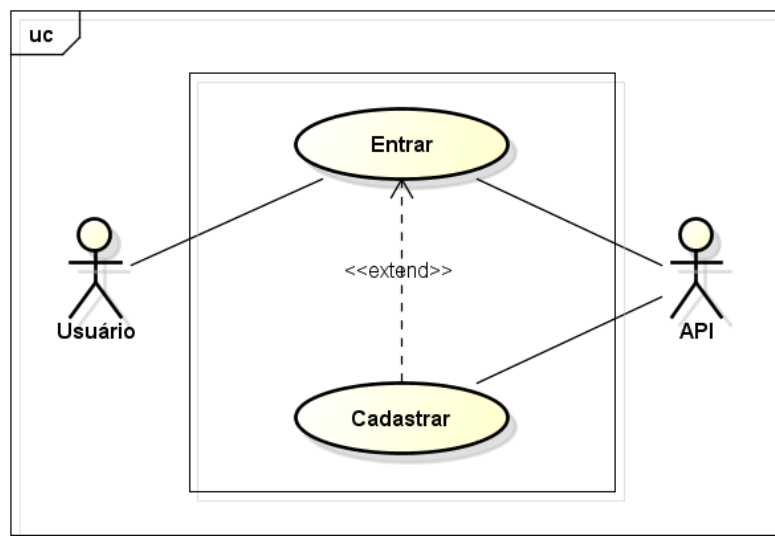
Os atores que interagem com o sistema são: O Usuário, a API do Sistema. O sistema é um caso de uso explícito e se trata do sistema em si em que os casos de uso acontecem:

- **Usuário** é o ator que representa os utilizadores deste aplicativo. Um ator pode, por exemplo, buscar times, ver perfil, marcar partidas, entre outros.
- **API do Sistema** representa o ator da API que permite a interação entre o aplicativo e o Banco de Dados, por exemplo, o login com as credenciais da do usuário específico.

A Figura 1 apresenta o caso de uso para a entrada do usuário no sistema e/ou cadastro do mesmo.

¹ *Unified Modeling Language* ou Linguagem Unificada de Modelagem (UML) é uma linguagem padrão para modelagem e documentar os sistemas orientados a objetos.

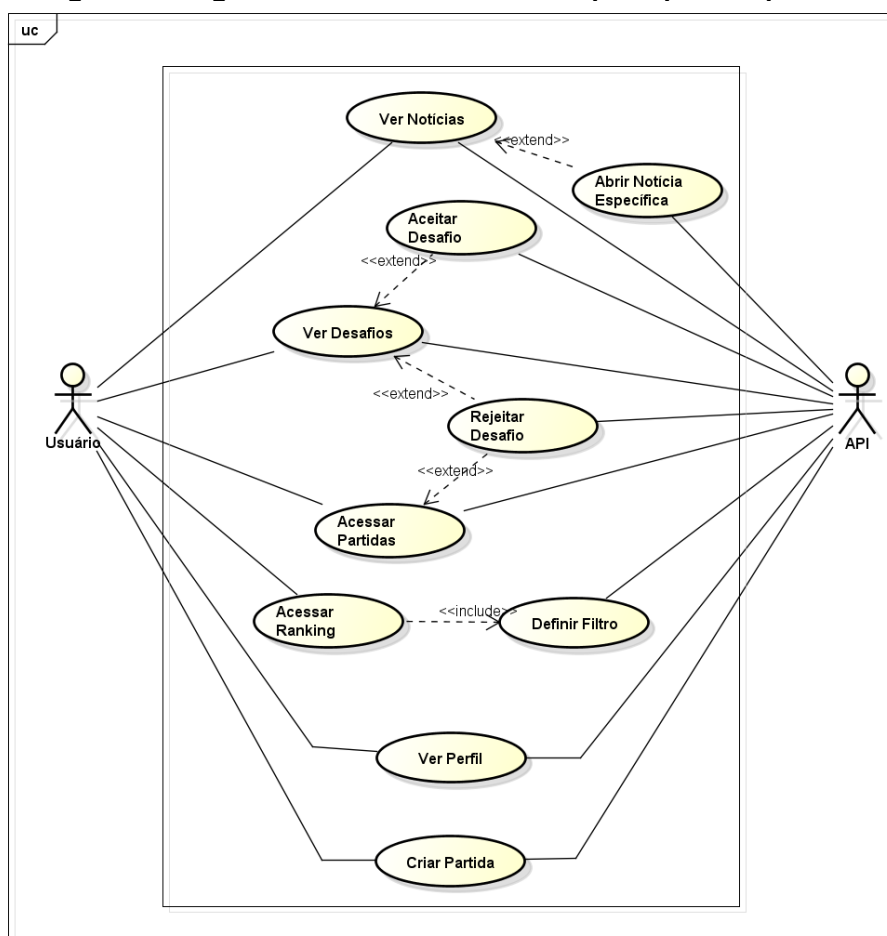
Figura 1 - Diagrama de caso de uso da tela entrar no sistema e cadastrar usuário.



Fonte: Elaborado pelo autor

A Figura 2 apresenta o caso de uso para a tela principal do sistema.

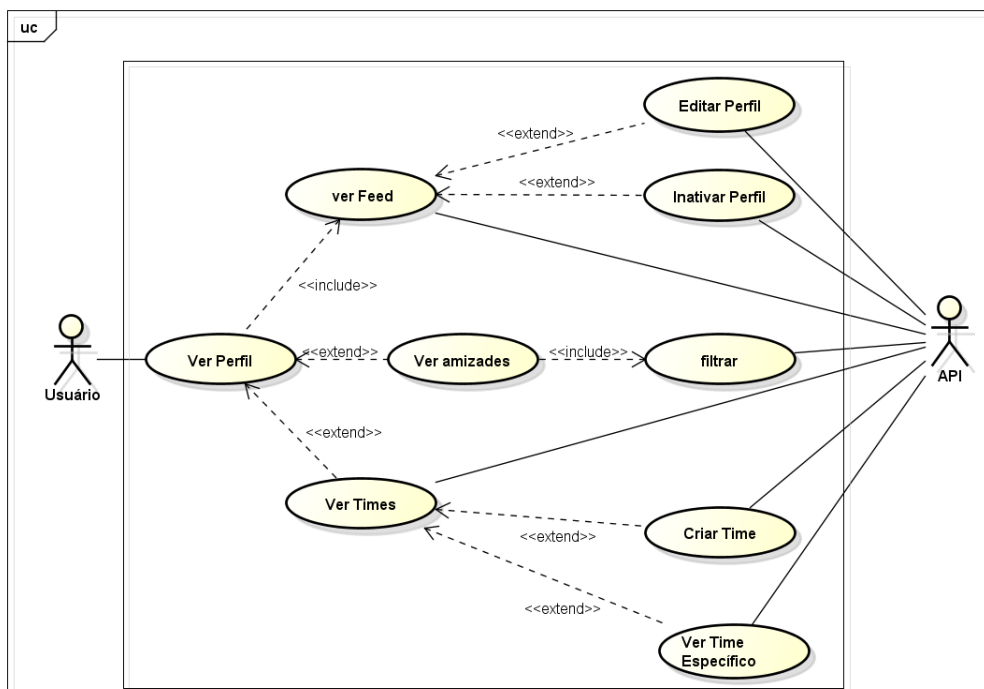
Figura 2 - Diagrama de caso de uso da tela principal do aplicativo



Fonte: Elaborado pelo autor

A Figura 3 apresenta o caso de uso para a tela de perfil do usuário.

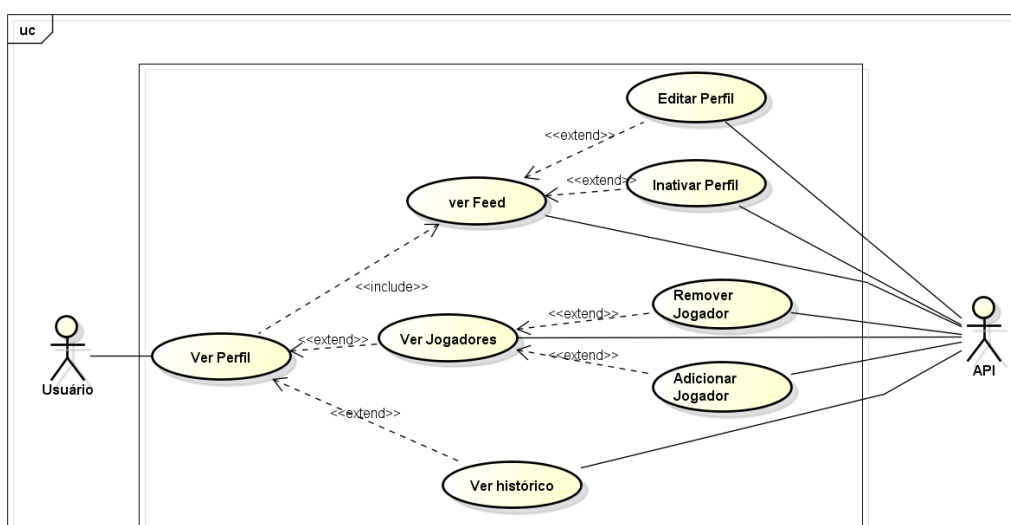
Figura 3 - Diagrama de caso de uso da tela de Perfil do Usuário.



Fonte: Elaborado pelo autor

A Figura 4 apresenta o caso de uso para a tela de perfil do time.

Figura 4 - Diagrama de caso de uso da tela Perfil do Time.



Fonte: Elaborado pelo autor

Na Seção 3.2 é apresentado a documentação dos casos de uso do projeto deste trabalho.

3.2 Documentação dos Casos de Uso

As funcionalidades do diagrama de caso de uso “Entrar no sistema/ Cadastrar usuário” são descritas na Tabela 4.

Tabela 4 – Caso de uso “Entrar no Sistema / Cadastrar Usuário”.

Nome do caso de uso	Entrar no Sistema / Cadastrar Usuário
Atores envolvidos	Usuário, API do Sistema
Objetivo	Este caso de uso descreve os passos do login e / ou cadastro de um usuário no sistema
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O Usuário entra com login e senha	
	2. A API leva para o Banco de dados, onde será validado sob condição.
	3. Se existir, entra no sistema, se não informa a invalidade dos dados.
	4. Após a autenticação ou cadastro, o sistema redireciona para a tela de Início do aplicativo.
Validações	Para que o login seja efetuado, o usuário deve entrar com seu usuário e senha.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso Principal “Ver Notícias Específicas” são descritas na Tabela 5.

Tabela 5 – Caso de uso Principal “Ver Notícias Específicas”.

Nome do caso de uso	Ver Notícias
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela principal do sistema, especificamente notícias.
Prioridade de desenvolvimento	Alta

Ações do ator	Ações do Sistema
1. O usuário clica em “home” (tela inicial)	
2. O usuário clica em Notícias	
	3. O sistema chama a tela específica
	4. A API Traz pelo banco informações relacionadas a ele ou seu time, e notícias de interesses pessoais.
	5. O sistema recebe os dados e os manipula para serem visualizados pelo usuário em forma de post.
6. O usuário clica na notícia da lista	
	7. O sistema abre uma tela que carrega a página completa da notícia ou abre um browser com a página completa da notícia.
Validações	Precisa clicar em Home e depois em notícias para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso Principal “Ver Desafios Específicos” são descritas na Tabela 6.

Tabela 6 – Caso de uso Principal “Ver Desafios Específicos”.

Nome do caso de uso	Ver Desafios
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela principal do sistema, especificamente Desafios.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário clica em “home” (tela inicial)	
2. O usuário clica em Desafios	
	3. A API puxa todos desafios que fizeram a algum dos times cujo usuário seja criador, e as informações do desafio.

Ações do ator	Ações do Sistema
	4. O sistema lista os desafios, deixando a opção de aceitar ou rejeitar.
5. O usuário clica em aceitar	
	6. A API reconhece o comando e joga para a agenda de jogos o desafio aceito.
	7. O sistema retira o desafio da lista e atualiza.
8. O usuário clica em rejeitar	
	9. A API reconhece o comando e rejeita o desafio.
	10. O sistema retira o desafio da lista e atualiza.
Validações	Precisa clicar em Home e depois em Desafios para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso Principal “Acessar *Ranking*” são descritas na Tabela 7.

Tabela 7 - Caso de uso Principal “Acessar *Ranking*”.

Nome do caso de uso	Acessar <i>Ranking</i>
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela principal do sistema, especificamente <i>Ranking</i> .
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário clica um ícone de <i>ranking</i> para ver <i>ranking</i>	
	2. O sistema chama a tela de <i>Ranking</i> de Times, por tipo de campo normal, <i>society</i> ou salão (condicionada ao filtro).
	3. A API puxa todos os times daquele campo do banco dados, em ordem decrescente de pontos.

Ações do ator	Ações do Sistema
	4. O Sistema trata essas informações de forma a trazê-las em lista para melhor visualização.
5. O usuário clica em um ícone de <i>ranking</i> para ver <i>ranking</i> individual	
	6. O sistema chama a tela de <i>Ranking</i> Individual, por cidade, estado ou região (condicionada ao filtro).
	7. A API puxa todos os usuários daquele filtro do banco dados, em ordem decrescente de pontos.
	8. O Sistema trata essas informações de forma a trazê-las em lista para melhor visualização.
Validações	Precisa clicar em <i>Ranking</i> para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso Principal “Acessar Partidas” são descritas na Tabela 8.

Tabela 8 - Caso de uso Principal “Acessar Partidas”.

Nome do caso de uso	Acessar Partidas
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela principal do sistema, especificamente Partidas.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário clica no ícone de agenda para ver as suas partidas	
	2. O sistema chama a tela de Partidas.
	3. A API puxa todos eventos já marcados (que foram aceitos no desafio), com suas respectivas informações do banco de dados.
	4. O Sistema trata essas informações de forma a trazê-las em lista para melhor visualização, com a opção de rejeitar.
5. O usuário clica em Rejeitar	

Ações do ator	Ações do Sistema
	6. A API reconhece o comando e rejeita o desafio
	7. O sistema retira o desafio da lista e atualiza.
Validações	Precisa clicar em Partidas para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso Principal “Ver Perfil” são descritas na Tabela 9.

Tabela 9 - Caso de uso Principal “Ver Perfil”.

Nome do caso de uso	Ver Perfil
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela principal do sistema, especificamente Meu Perfil.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário clica um ícone de perfil para ver seu perfil	
	2. O sistema chama a tela de Perfil pessoal
	3. A API busca no Banco de dados todas informações preenchidas no cadastro a respeito desse usuário conectado. Tanto pessoais (Nome, Foto de perfil, CEP e etc.) como as relacionadas a ele (Times, Amizades, Pontos e etc.).
	4. O sistema organiza as informações pessoais de forma conveniente dentro de um Painel.
	5. O Sistema organiza todas as informações relacionadas a ele em forma de lista a baixo por ordem de data
Validações	Precisa clicar no <i>float button</i> na guia <i>home</i> , <i>ranking</i> ou partidas para que haja a possibilidade de clicar em Perfil, e gerar a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso Principal “Criar Partidas” são descritas na Tabela 10.

Tabela 10 - Caso de uso Principal “Criar Partidas”.

Nome do caso de uso	Criar Partidas
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela principal do sistema, especificamente Partidas.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário clica um ícone de partidas para criar partidas	
	2. O sistema chama a tela de Escolha de tipo de campo
	3. Após validar tipo de campo, o sistema chama a tela de escolha seu time que irá jogar.
	4. A API puxa seus times do banco, e o sistema lista-os.
	5. Após validar seu time, o sistema chama a tela de Escolha de adversário.
	6. A API puxa do banco de dados todos adversários possíveis condicionáveis ao filtro de cidades, região ou estado para a escolha.
	7. Após validar seu adversário, o sistema chama a tela de agendamento (dados da partida: Horário, Data e Local).
8. Usuário informa dados da partida	
	9. API salva informações no banco de dados.
Validações	Precisa clicar no <i>float button</i> na guia <i>home, ranking</i> ou partidas para que haja a possibilidade de clicar em Partidas, e gerar a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Usuário “Ver Feed” são descritas na Tabela 11.

Tabela 11 - Caso de uso Perfil do Usuário “Ver Feed”.

Nome do caso de uso	Ver <i>Feed</i>
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Usuário, especificamente Ver <i>Feed</i> .
Prioridade de desenvolvimento	Alto
Ações do ator	Ações do Sistema
1. O usuário abre a tela de início em Perfil	
	2. O sistema chama a tela inicial do Perfil pessoal
	3. A API além de buscar no Banco de dados todas informações pessoais, busca também todas informações preenchidas no cadastro a relacionadas com esse usuário conectado e ao seu time.
	4. O sistema organiza as informações pessoais de forma conveniente dentro de um Painel.
	5. O Sistema organiza todas as informações de <i>feed</i> relacionadas a ele em forma de <i>post</i> a baixo por ordem de data
Validações	Precisa clicar no Início da guia Perfil para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Usuário “Editar Perfil” são descritas na Tabela 12.

Tabela 12 - Caso de uso Perfil do Usuário “Editar Perfil”.

Nome do caso de uso	Editar Perfil
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Usuário, especificamente Editar Perfil.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário abre a tela de início em Perfil	

Ações do ator	Ações do Sistema
2. O usuário clica no ícone de Editar Perfil	
	3. O sistema chama a tela de editar
	4. API puxa informações desse perfil
5. Usuário edita informações	
	6. API salva as alterações
Validações	Precisa clicar no Editar Perfil na guia Perfil para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Usuário “Ver Amizades” são descritas na Tabela 13.

Tabela 13 - Caso de uso Perfil do Usuário “Ver Amizades”.

Nome do caso de uso	Ver Amizades
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Usuário, especificamente Amizades.
Prioridade de desenvolvimento	Alto
Ações do ator	Ações do Sistema
1. O usuário abre a tela de amizades em Perfil	
	2. O sistema chama a tela dos seus Seguidores
	3. A API puxa a lista do banco de todos seus amigos que te seguem
	4. O sistema organiza-os em forma de <i>cards</i> separados.
5. Usuário clica em “seguindo”	
	6. A API puxa a lista do banco de todos seus amigos que você segue

Ações do ator	Ações do Sistema
	7. O sistema organiza-os em forma de <i>cards</i> separados.
Validações	Precisa clicar no Amizades na guia Perfil para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Usuário “Ver Time Específico” são descritas na Tabela 14.

Tabela 14 - Caso de uso Perfil do Usuário “Ver Time Específico”.

Nome do caso de uso	Ver Time Específico
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Usuário, especificamente Times.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário abre a tela de Meus Times em Perfil	
	2. O sistema chama a tela dos seus Times
	3. A API puxa a lista do banco de todos seus Times criados e que participa
	4. O sistema organiza-os em forma de lista separados.
5. Usuário clica em algum time	
	6. O sistema chama a tela de Perfil do Time
	3. A API busca no Banco de dados todas informações preenchidas no cadastro a respeito desse Time clicado (Nome, Pontos, Jogadores, Posição).
	4. O sistema organiza as informações do time de forma conveniente dentro de um Painel.
	5. O Sistema organiza todas as informações relacionadas a ele em forma de lista a baixo por ordem de data

Validações	Precisa clicar em Times na guia Perfil para que haja a ação.
-------------------	--

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Time “Criar Time” são descritas na Tabela 15.

Tabela 15 - Caso de uso Perfil do Usuário “Criar Time”.

Nome do caso de uso	Criar Times
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Usuário, especificamente criar um time.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário abre a tela de Meus Times em Perfil	
2. O usuário clica no ícone de criar um time	
	2. O sistema chama a tela de criar time
3. Usuário insere as informações	
	4. A API salva as informações no banco de dados.
Validações	Precisa clicar em Times/Criar Time na guia Perfil para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Time “Ver Feed” são descritas na Tabela 16.

Tabela 16 - Caso de uso Perfil do Time “Ver Feed”.

Nome do caso de uso	Ver Feed
Atores envolvidos	Usuário, API do sistema

Objetivo	Este caso de uso descreve os passos da tela Perfil de Time, especificamente Ver <i>Feed</i> do Time.
Prioridade de desenvolvimento	Alto
Ações do ator	Ações do Sistema
1. O usuário abre a tela de início em Perfil do Time	
	2. O sistema chama a tela inicial do Perfil do Time
	3. A API além de buscar no Banco de dados todas informações pessoais do time, busca também todas informações preenchidas no cadastro a relacionadas com esse time.
	4. O sistema organiza as informações pessoais de forma conveniente dentro de um Painel.
	5. O Sistema organiza todas as informações de <i>feed</i> relacionadas a ele em forma de <i>post</i> a baixo por ordem de data
Validações	Precisa clicar no Início da guia Perfil do Time para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Time “Editar Perfil” são descritas na Tabela 17.

Tabela 17 - Caso de uso Perfil do Time “Editar Perfil”.

Nome do caso de uso	Editar Perfil
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Time, especificamente Editar Perfil do Time.
Prioridade de desenvolvimento	Essencial
Ações do ator	Ações do Sistema
1. O usuário abre a tela de início em Perfil	
2. O usuário clica no ícone de Editar Perfil	
	3. O sistema chama a tela de editar

Ações do ator	Ações do Sistema
	4. API puxa informações desse perfil de time
5. Usuário edita informações	
	6. API salva as alterações
Validações	Precisa clicar no Editar Perfil na guia Perfil do Time para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Time “Ver Jogadores” são descritas na Tabela 18.

Tabela 18 - Caso de uso Perfil do Time “Ver Jogadores”.

Nome do caso de uso	Ver Jogadores
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Time, especificamente Jogadores.
Prioridade de desenvolvimento	Alto
Ações do ator	Ações do Sistema
1. O usuário abre a tela de Jogadores em Perfil do Time	
	2. O sistema chama a tela dos Jogadores do time
	3. A API puxa a lista do banco de todos os jogadores daquele time.
	4. O sistema organiza-os em forma de lista separados.
5. Usuário clica em Retirar	
	6. A API retirar aquele jogador, daquele time.
	7. O sistema reorganiza-os em forma de lista separados.

Ações do ator	Ações do Sistema
8. Usuário clica em Adicionar	
	9. O sistema redireciona para a tela de escolha de jogador
	10. A API lista todos jogadores que não participam daquele time.
11. Usuário seleciona um deles	
	11. A API atrela o jogador escolhido ao time presente.
Validações	Precisa clicar no Jogadores na guia Perfil de times para que haja a ação.

Fonte: Elaborado pelo autor

As funcionalidades do diagrama de caso de uso do Perfil do Time “Ver Histórico” são descritas na Tabela 19.

Tabela 19 - Caso de uso Perfil do Time “Ver Histórico”.

Nome do caso de uso	Ver Histórico
Atores envolvidos	Usuário, API do sistema
Objetivo	Este caso de uso descreve os passos da tela Perfil de Times, especificamente Histórico.
Prioridade de desenvolvimento	Alto
Ações do ator	Ações do Sistema
1. O usuário abre a tela de Histórico em Perfil do time	
	2. O sistema chama a tela de Histórico
	3. A API puxa a lista do banco de todos os eventos de partidas jogadas, e suas respectivas informações.
	4. O sistema organiza-os em forma de lista separados.

Validações	Precisa clicar no Histórico na guia Perfil de times para que haja a ação.
-------------------	---

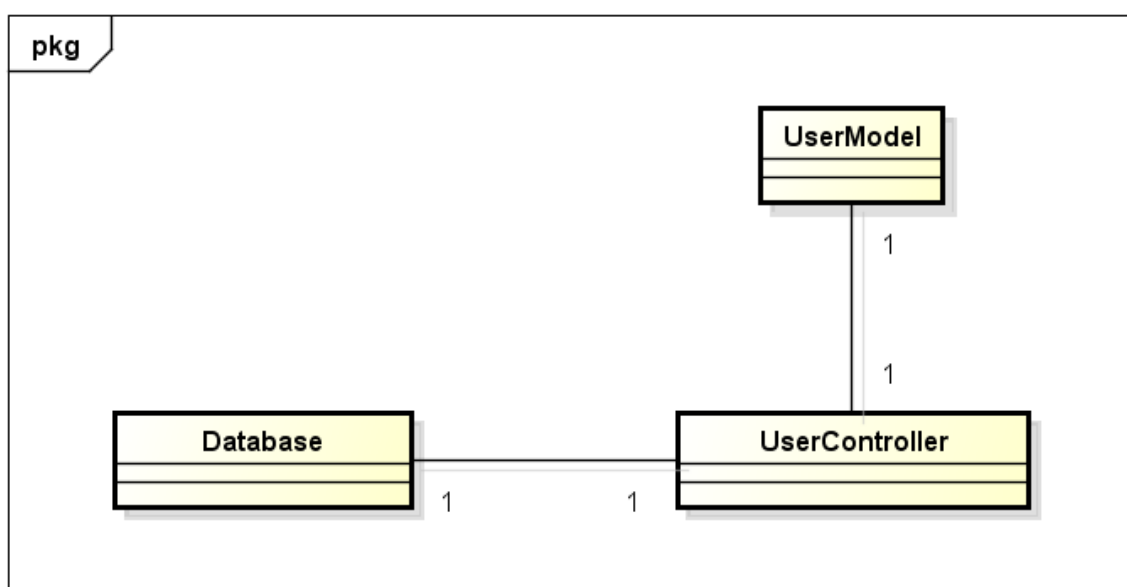
Fonte: Elaborado pelo autor

3.3 Diagrama de Classe do *WebService*

O diagrama de classe é responsável por oferecer uma representação da estrutura e relações das classes como também as operações solicitadas pelos atores que servem de modelo para os objetos.

UserModel: Esta classe tem por objetivo fornecer a classe *UserController* os atributos necessários para as regras de negócio (Figura 5). Não há métodos nesta classe.

Figura 5 - Diagrama de classe de usuário.



Fonte: Elaborado pelo autor

Database: Esta classe tem por objetivo realizar a conexão com o Banco de Dados (Figura 5). Os métodos identificados para esta classe são apresentados na Tabela 20:

Tabela 20 - Diagrama de classe de usuários.

Método	Descrição
Conexão	Abre conexão com o banco de dados

Fonte: Elaborado pelo autor

UserController: Esta classe tem por objetivo realizar todas as ações referentes aos dados do usuário (Figura 5). Os métodos identificados para esta classe são apresentados na Tabela 21:

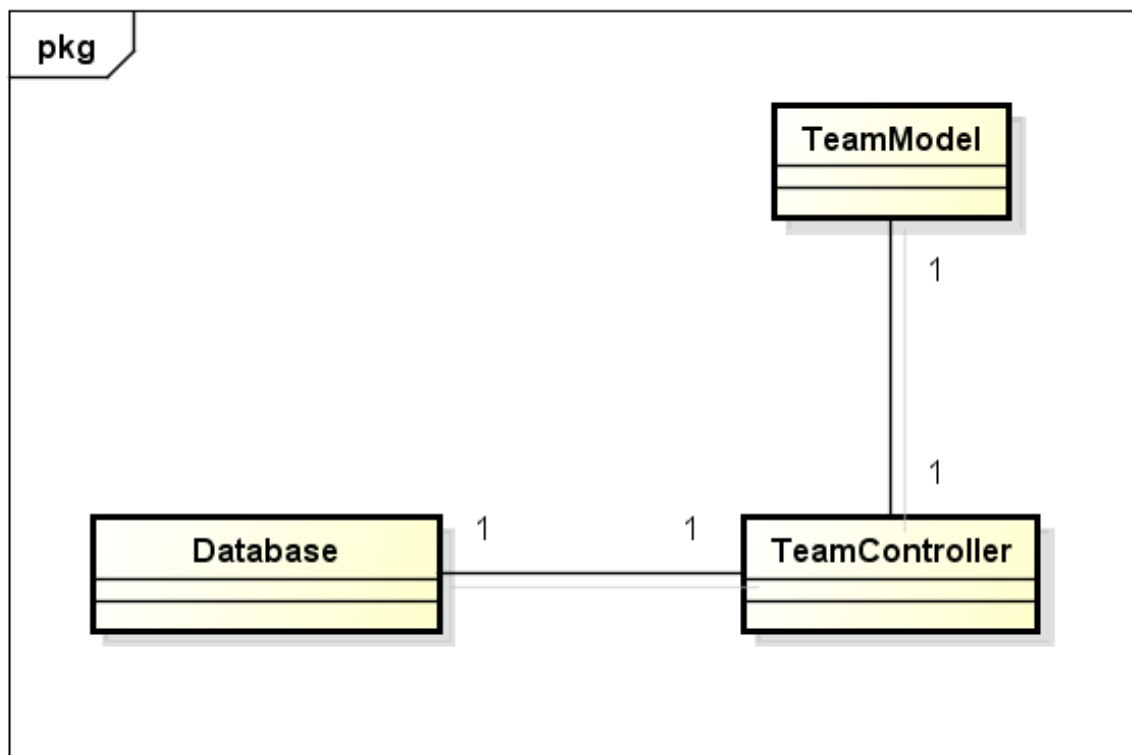
Tabela 21 - Diagrama de classe de usuários.

Método	Descrição
<i>login</i>	Realiza uma validação de usuário no sistema.
<i>get</i>	Busca um usuário.
<i>getAll</i>	Lista uma lista de usuários ordenando-os por pontuação.
<i>post</i>	Cria um novo usuário.
<i>put</i>	Atualiza os dados de um usuário.
<i>delete</i>	Deleta a conta de um usuário.

Fonte: Elaborado pelo autor

TeamModel: Esta classe tem por objetivo fornecer a classe *TeamController* os atributos necessários para as regras de negócio (Figura 6). Não há métodos nesta classe.

Figura 6 - Diagrama de classe de Times.



Fonte: Elaborado pelo autor

TeamController. Esta classe tem por objetivo realizar as ações em um time (Figura 6). Os métodos identificados para esta classe são apresentados na Tabela 22:

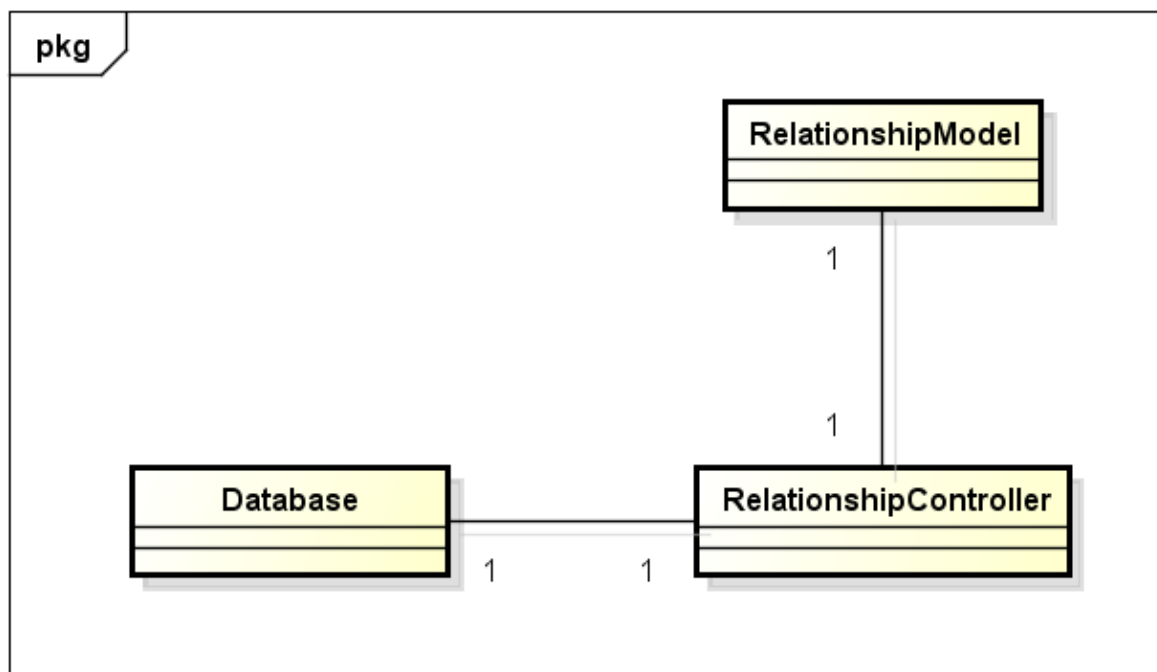
Tabela 22 - Diagrama de classe de times.

Método	Descrição
<i>get</i>	Busca um time.
<i>getAll</i>	Busca todos os times ordenados por pontuação.
<i>post</i>	Cria um novo time.
<i>put</i>	Atualiza um time.
<i>delete</i>	Remove um time.

Fonte: Elaborado pelo autor

RelationshipModel: Esta classe tem por objetivo fornecer a classe *RelationshipController* os atributos necessários para as regras de negócio (Figura 7). Não há métodos nesta classe.

Figura 7 - Diagrama de classe de amizade.



Fonte: Elaborado pelo autor

RelationshipController. Esta classe tem por objetivo formar uma relação entre usuários (Figura 7). Os métodos identificados para esta classe são apresentados na Tabela 23:

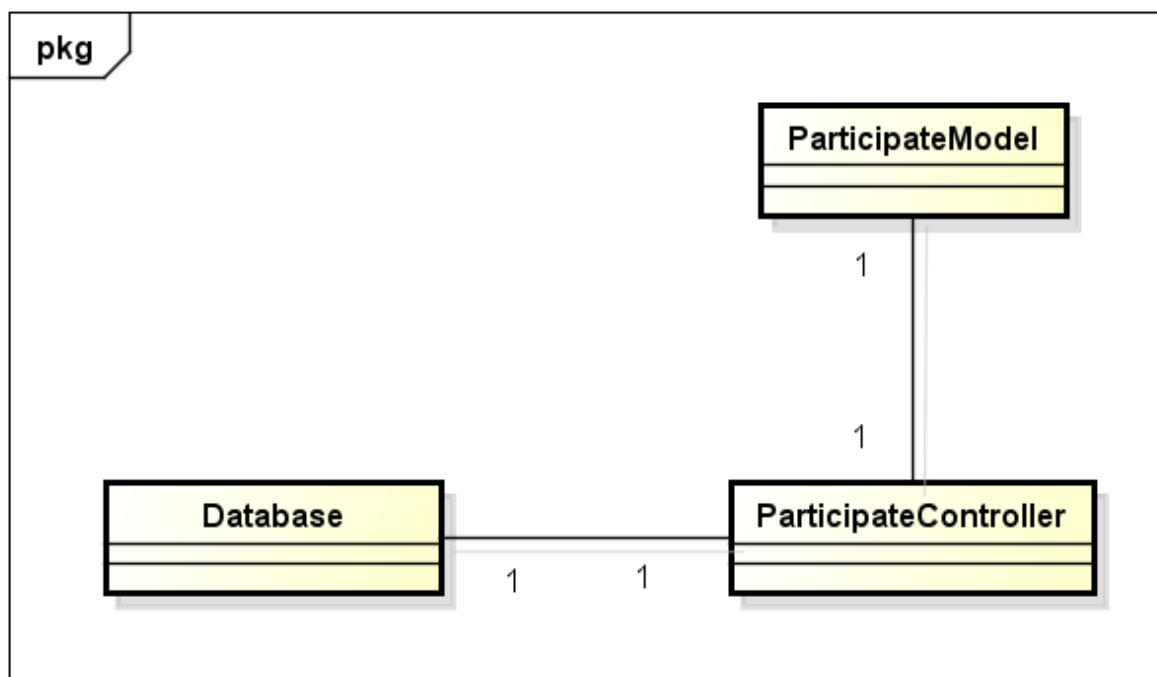
Tabela 23 - Diagrama de classe de amizade.

Método	Descrição
<i>getAll</i>	Busca todas as amizades de um usuário.
<i>get</i>	Busca relação de amizade com outro usuário
<i>post</i>	Cria uma nova amizade.
<i>delete</i>	Remove uma amizade entre usuários.

Fonte: Elaborado pelo autor

ParticipateModel: Esta classe tem por objetivo fornecer a classe *ParticipateController* os atributos necessários para as regras de negócio (Figura 8). Não há métodos nesta classe.

Figura 8 - Diagrama de classe de participação do usuário no time.



Fonte: Elaborado pelo autor

ParticipateController. Esta classe tem por objetivo realizar a conexão entre um usuário e um time (Figura 8). Os métodos identificados para esta classe são apresentados na Tabela 24:

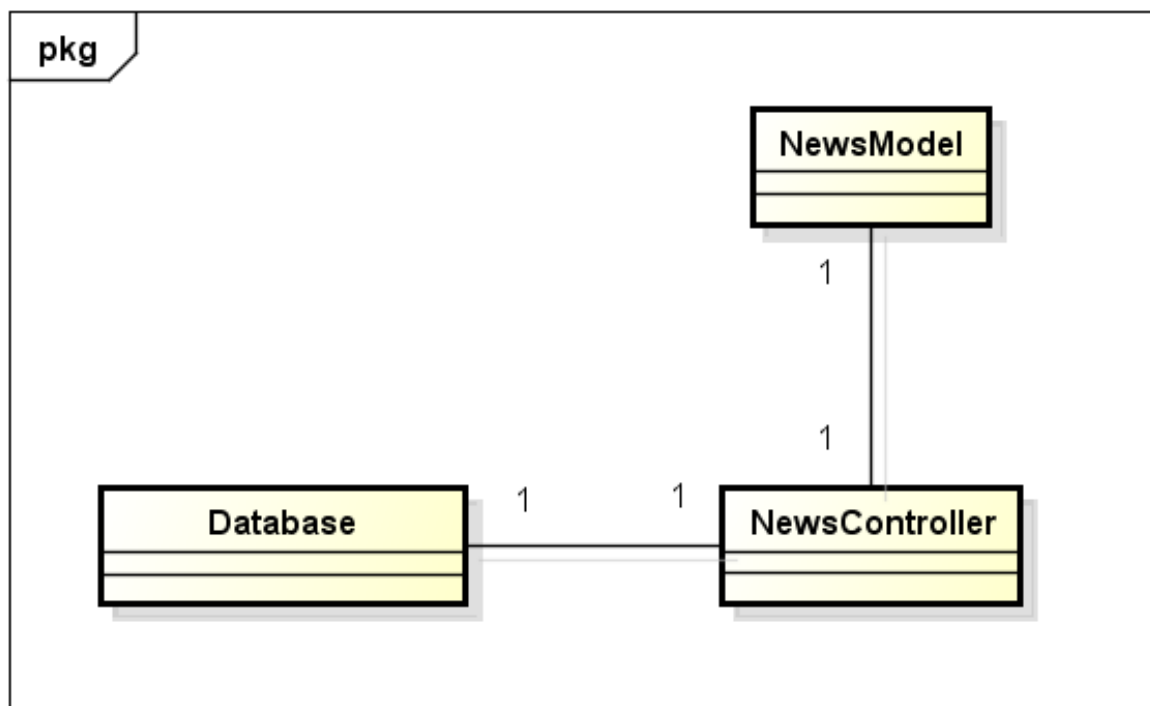
Tabela 24 - Diagrama de classe de participação do usuário no time.

Método	Descrição
<i>get</i>	Busca um time de um usuário.
<i>post</i>	Cria uma nova participação de um usuário em um time.
<i>put</i>	Atualiza o usuário Capitão do time, caso o anterior saia dele.
<i>delete</i>	Remove a participação de um usuário em um time.

Fonte: Elaborado pelo autor

NewsModel: Esta classe tem por objetivo fornecer a classe *NewsController* os atributos necessários para as regras de negócio (Figura 9). Não há métodos nesta classe.

Figura 9 - Diagrama de classe de Notícias.



Fonte: Elaborado pelo autor

NewsController. Esta classe tem por objetivo realizar as ações das notícias para os usuários (Figura 9). Os métodos identificados para esta classe são apresentados na Tabela 25:

Tabela 25 - Diagrama de classe de Notícias.

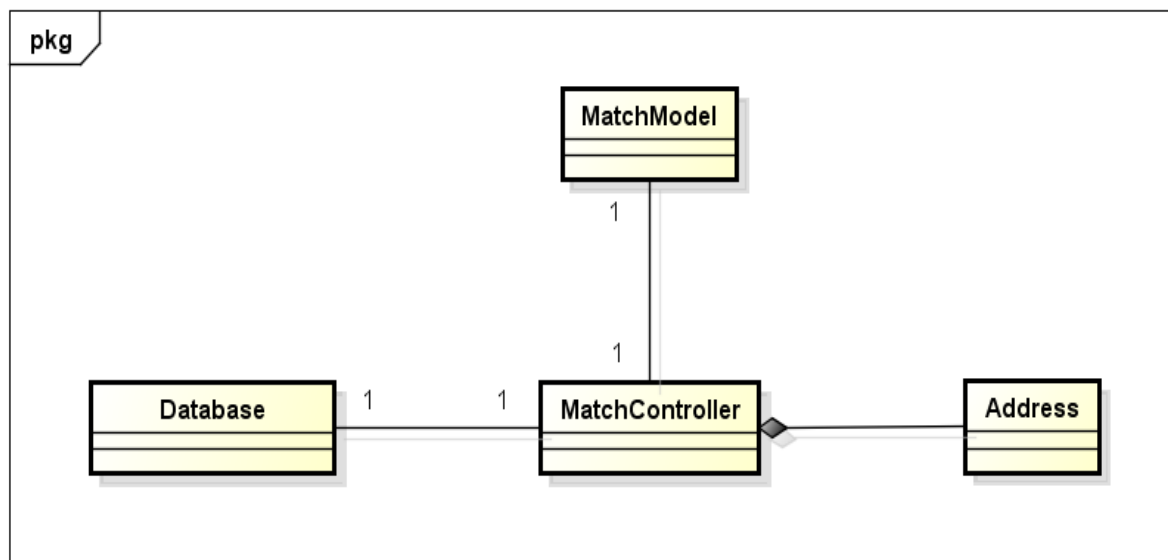
Método	Descrição
<i>getAll</i>	Busca todas as notícias de futebol retornadas pela API de Bing News.
<i>get</i>	Busca uma notícia.
<i>post</i>	Cria uma nova notícia.
<i>put</i>	Atualiza uma notícia.
<i>delete</i>	Remove uma notícia.

Fonte: Elaborado pelo autor

MatchModel: Esta classe tem por objetivo fornecer a classe *MatchController* os atributos necessários para as regras de negócio (Figura 10). Não há métodos nesta classe.

Address: Esta classe é uma classe dependente que existe apenas para disponibilizar os dados de endereço para a classe *MatchController* (Figura 10)

Figura 10 - Diagrama de classe de Partida.



Fonte: Elaborado pelo autor

MatchController: Esta classe tem por objetivo realizar as ações em uma partida marcada entre times (Figura 10). Os métodos identificados para esta classe são apresentados na Tabela 26:

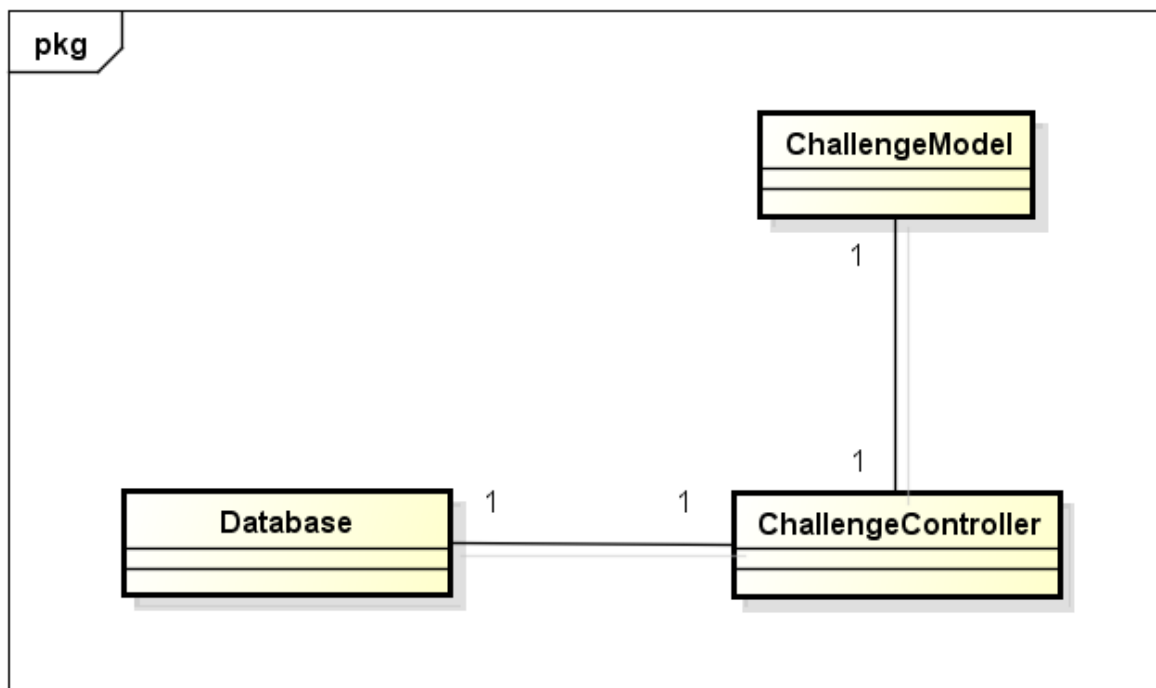
Tabela 26 - Diagrama de classe de Partida.

Método	Descrição
<i>get</i>	Busca os dados de uma partida.
<i>post</i>	Cria uma nova partida.
<i>delete</i>	Remove uma partida.

Fonte: Elaborado pelo autor

ChallengeModel: Esta classe tem por objetivo fornecer a classe *ChallengeController* os atributos necessários para as regras de negócio (Figura 11). Não há métodos nesta classe.

Figura 11 - Diagrama de classe de Desafio.



Fonte: Elaborado pelo autor

ChallengeController. Esta classe tem por objetivo realizar as ações referentes a um desafio entre times (Figura 11). Os métodos identificados para esta classe são apresentados na Tabela 27:

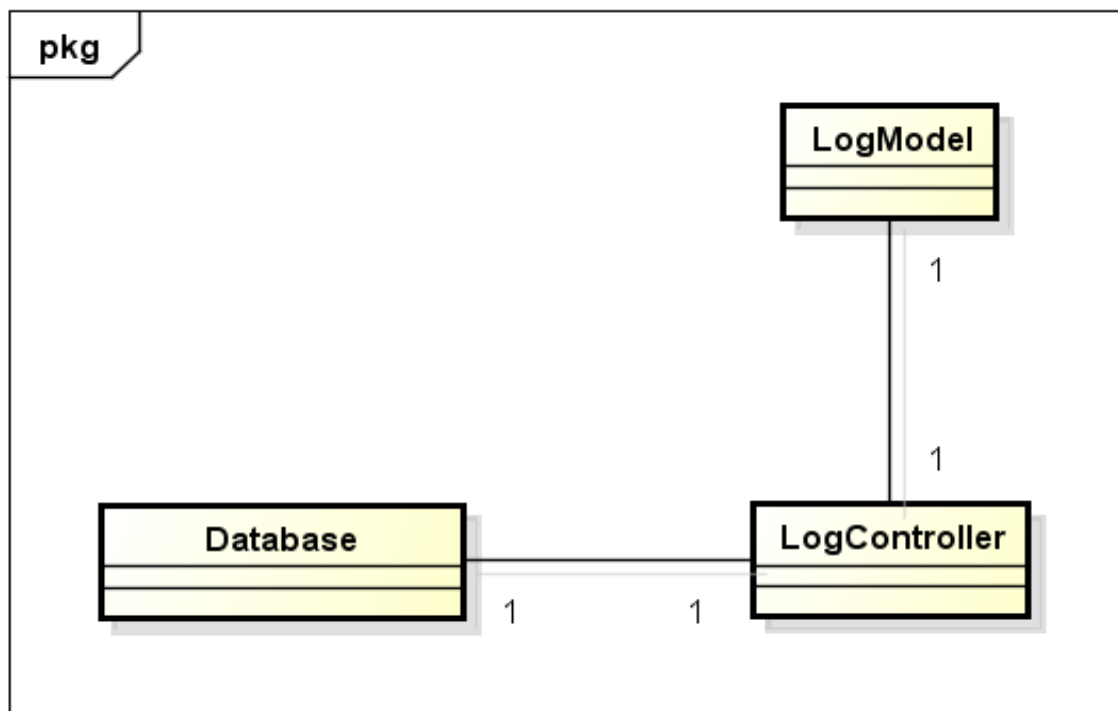
Tabela 27 - Diagrama de classe de Desafio.

Método	Descrição
<i>getAll</i>	Busca os Desafios de um time.
<i>get</i>	Busca as informações de um Desafio em específico.
<i>post</i>	Cria um novo Desafio.
<i>delete</i>	Remove um desafio.

Fonte: Elaborado pelo autor

LogModel: Esta classe tem por objetivo fornecer a classe *LogController* os atributos necessários para as regras de negócio (Figura 12). Não há métodos nesta classe.

Figura 12 - Diagrama de classe de Log.



Fonte: Elaborado pelo autor

LogController. Esta classe tem por objetivo realizar as ações referentes a um desafio entre times (Figura 12). Os métodos identificados para esta classe são apresentados na Tabela 28:

Tabela 28 - Diagrama de classe de Log.

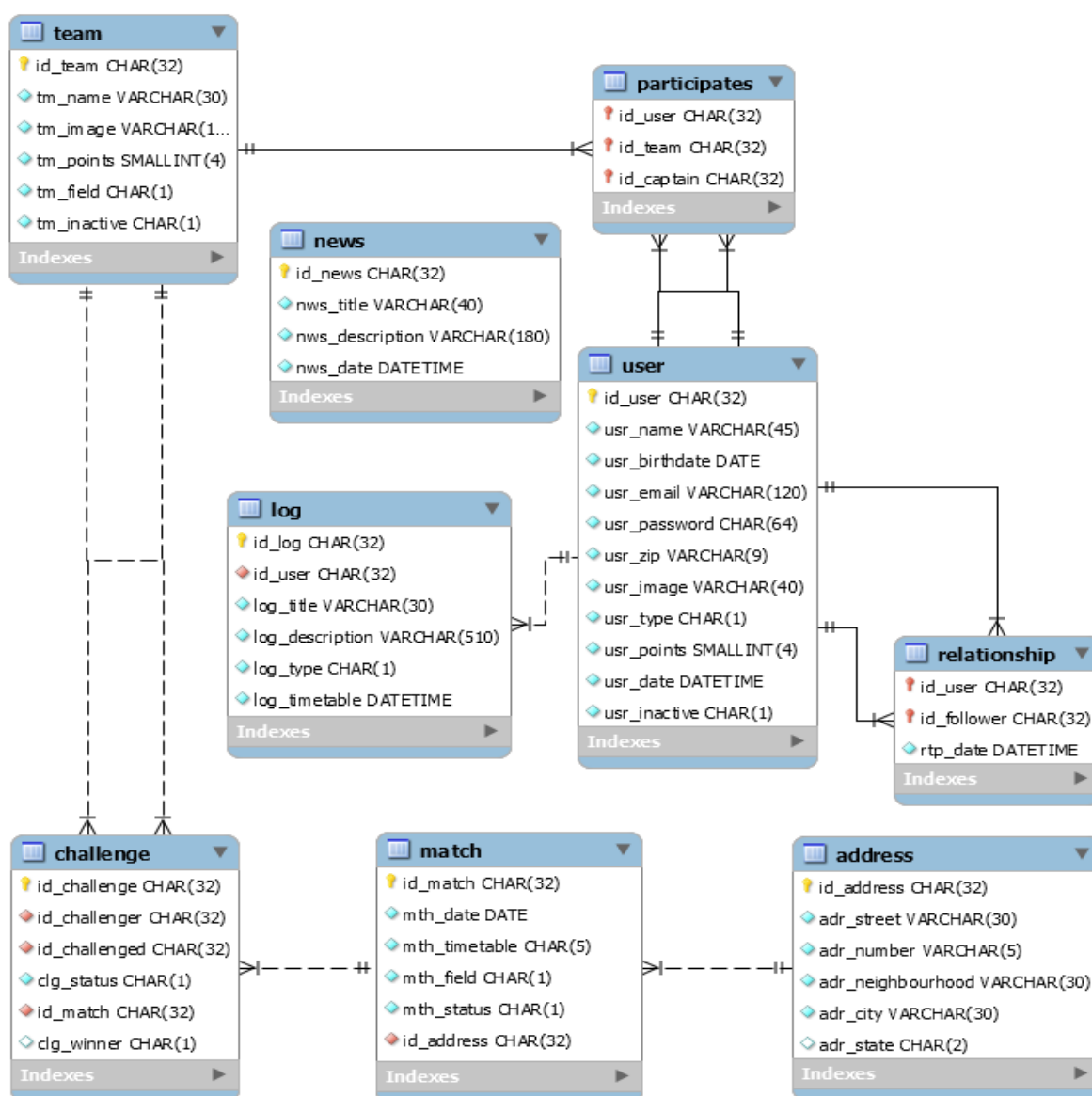
Método	Descrição
<i>getAll</i>	Busca todos Log completo de acordo com número de página passada
<i>get</i>	Busca os dados de um Log específico.
<i>post</i>	Cria um novo log.
<i>put</i>	Atualiza um log.
<i>delete</i>	Remove um desafio.

Fonte: Elaborado pelo autor

3.4 Diagrama de Entidade e Relacionamento

Diagrama Entidade Relacionamento (DER) é um modelo diagramático que descreve o modelo de dados de um sistema com alto nível de abstração. Ele é a principal representação do Modelo de Entidades e Relacionamentos. Sua maior aplicação é visualizar o relacionamento entre tabelas de um banco de dados, no qual as relações são construídas através da associação de um ou mais atributos destas tabelas (SOMMERVILLE, 2011). A Figura 13 apresenta o DER do sistema proposto.

Figura 13 - Diagrama de Entidade e Relacionamento.



Fonte: Elaborado pelo autor

3.4.1 Dicionário de Dados

O Dicionário de Dados (DD) consiste numa lista organizada de todos os elementos de dados que são pertinentes ao sistema. As tabelas devem conter os seguintes campos: Entidade, Atributo, Classe, Domínio, Tamanho e Descrição. A Tabela 29 apresenta o DD da entidade *User*.

Tabela 29 - Dicionário de Dados da entidade *User*.

Entidade: <i>User</i>				
Atributo	Classe	Domínio	Tamanho	Descrição
id_user	Determinante	Texto	32	Identificador universal composto por 32 caracteres
usr_name	Simples	Texto	45	Nome do usuário
usr_birthdate	Simples	Data		Data de Nascimento
usr_email	Simples	Texto	120	E-mail
usr_password	Simples	Texto	64	Senha
usr_zip	Simples	Texto	9	CEP
usr_image	Simples	Texto	40	Endereço da imagem de perfil
usr_type	Simples	Texto	1	Identifica o tipo de usuário (Administrador, proprietário ou jogador)
usr_points	Simples	Numérico	4	Pontos do time
usr_date	Simples	Data		Data de cadastro do usuário.
usr_inactive	Simples	Texto	1	Conta está ativa ou inativa.

Fonte: Elaborado pelo autor

A Tabela 30 apresenta o DD da entidade *Relationship*.

Tabela 30 - Dicionário de Dados da entidade *Relationship*.

Entidade: <i>Relationship</i>				
Atributo	Classe	Domínio	Tamanho	Descrição
id_user	Determinante	Texto	32	Identificador universal composto por 32 caracteres
id_follower	Determinante	Texto	32	Identificador universal composto por 32 caracteres
rtp_date	Simples	Data		Data que amizade iniciou.

Fonte: Elaborado pelo autor

A Tabela 31 apresenta o DD da entidade *Participates*.

Tabela 31 - Dicionário de Dados da entidade *Participates*.

Entidade: <i>Participates</i>				
Atributo	Classe	Domínio	Tamanho	Descrição
id_user	Determinante	Texto	32	Identificador universal composto por 32 caracteres
id_team	Determinante	Texto	32	Identificador universal composto por 32 caracteres
id_captain	Determinante	Texto	32	Identificador universal composto por 32 caracteres

Fonte: Elaborado pelo autor

A Tabela 32 apresenta o DD da entidade *News*.

Tabela 32 - Dicionário de Dados da entidade *News*.

Entidade: <i>News</i>				
Atributo	Classe	Domínio	Tamanho	Descrição

id_news	Determinante	Texto	32	Identificador universal composto por 32 caracteres
nws_title	Simple	Texto	40	Título da notícia
nws_description	Simple	Texto	180	Descrição da notícia
Atributo	Classe	Domínio	Tamanho	Descrição
nws_date	Simple	Data		Data de publicação

Fonte: Elaborado pelo autor

A Tabela 33 apresenta o DD da entidade *Log*.

Tabela 33 - Dicionário de Dados da entidade *Log*.

Entidade: <i>Log</i>				
Atributo	Classe	Domínio	Tamanho	Descrição
id_log	Determinante	Texto	32	Identificador universal composto por 32 caracteres
id_user	Simple	Texto	32	Identificador universal composto por 32 caracteres
log_title	Simple	Texto	30	Título do log
log_description	Simple	Texto	510	Descrição do log
log_type	Simple	Texto	1	Se é operação interna ou <i>Exception</i>

Fonte: Elaborado pelo autor

A Tabela 34 apresenta o DD da entidade *Team*.

Tabela 34 - Dicionário de Dados da entidade *Team*.

Entidade: <i>Team</i>				
Atributo	Classe	Domínio	Tamanho	Descrição
id_team	Determinante	Texto	32	Identificador universal composto por 32 caracteres

tm_name	Simples	Texto	30	Nome do time
tm_image	Simples	Texto	19	Imagem do perfil do time
tm_points	Simples	Numérico	4	Pontos do time
tm_field	Simples	Texto	1	Tipo de campo do time
tm_inactive	Simples	Texto	1	Se time está ativo ou inativo

Fonte: Elaborado pelo autor

A Tabela 35 apresenta o DD da entidade *Challenge*.

Tabela 35 - Dicionário de Dados da entidade *Challenge*.

Entidade: <i>Challenge</i>				
Atributo	Classe	Domínio	Tamanho	Descrição
id_challenge	Determinante	Texto	32	Identificador universal composto por 32 caracteres
id_challenger	Simples	Texto	32	Identificador universal composto por 32 caracteres
id_challenged	Simples	Texto	32	Identificador universal composto por 32 caracteres
clg_status	Simples	Texto	1	Partida aconteceu.
id_mach	Simples	Texto	32	Identificador universal composto por 32 caracteres
clg_winner	Simples	Texto	1	Define quem ganhou.

Fonte: Elaborado pelo autor

A Tabela 36 apresenta o DD da entidade *Match*.

Tabela 36 - Dicionário de Dados da entidade *Match*.

Entidade: <i>Match</i>

Atributo	Classe	Domínio	Tamanho	Descrição
id_match	Determinante	Texto	32	Identificador universal composto por 32 caracteres
mth_date	Simples	Data		Data da partida
mth_timetable	Simples	Texto	5	Horário da partida
mth_field	Simples	Texto	1	Tipo de campo
mth_status	Simples	Texto	1	Se já ocorreu ou não
id_address	Simples	Texto	32	Identificador universal composto por 32 caracteres

Fonte: Elaborado pelo autor

A Tabela 37 apresenta o DD da entidade *Address*.

Tabela 37 - Dicionário de Dados da entidade *Address*.

Entidade: <i>Address</i>				
Atributo	Classe	Domínio	Tamanho	Descrição
id_address	Determinante	Texto	32	Identificador universal composto por 32 caracteres
adr_street	Simples	Texto	30	Rua
adr_number	Simples	Texto	5	Número
adr_neighborhood	Simples	Texto	30	Bairro
adr_city	Simples	Texto	30	Cidade
adr_estate	Simples	Texto	2	Estado

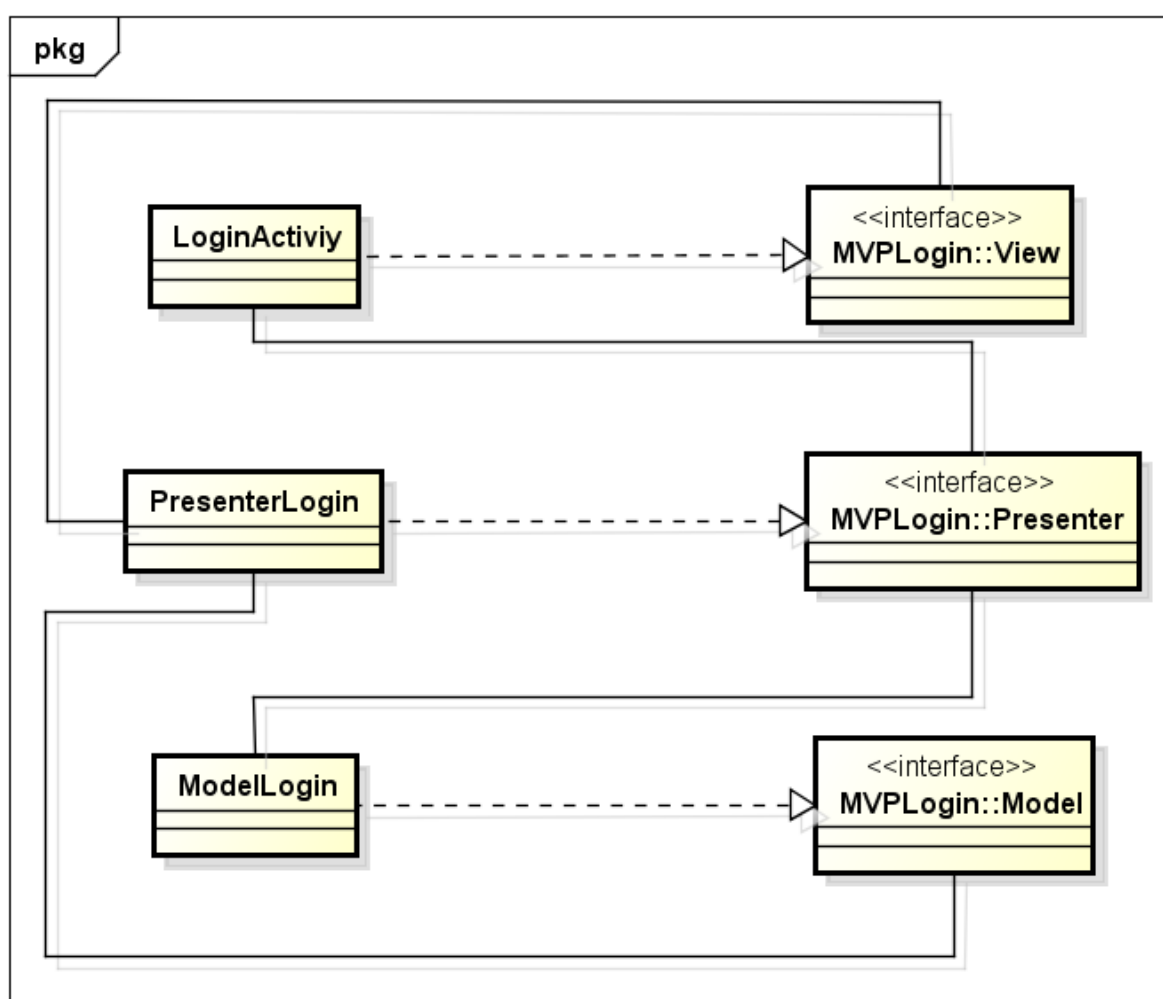
Fonte: Elaborado pelo autor

3.5 Diagrama de Classe do Aplicativo

Os diagramas de classe seguintes são representações da estrutura, das relações das classes e também das operações solicitadas pelos atores que servem de modelo para os objetos referentes ao aplicativo *mobile*.

A Figura 14 apresenta o diagrama de classe de login.

Figura 14 - Diagrama de classe de login.



Fonte: Elaborado pelo autor

LoginActivity: Esta classe tem por objetivo receber os dados do usuário (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 38:

Tabela 38 – Métodos do Diagrama de classe de login.

Método	Descrição
<i>Cadastrar</i>	Carrega a tela de cadastrar
<i>Error</i>	Emita uma janela de diálogo para informar de erro
<i>login_invalido</i>	Emita uma janela de diálogo para informar de falha nos dados inseridos
<i>login_valido</i>	Abre a tela de <i>Splash</i>
<i>validar</i>	Valida os campos
<i>onCreate</i>	Estabelece contato com o Layout da <i>Activity</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

PresenterLogin: Esta classe tem por objetivo representar a camada *Presenter* da arquitetura MVP (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 39:

Tabela 39 - Métodos do Diagrama de classe de login.

Método	Descrição
<i>checkLogin</i>	<i>presenter</i> do método da classe <i>Model</i>
<i>error</i>	<i>presenter</i> do método da classe <i>LoginActivity</i>
<i>getContext</i>	<i>presenter</i> do método que retorna o contexto para a classe que o chamar
<i>login_invalido</i>	<i>presenter</i> do método da classe <i>LoginActivity</i>
<i>login_valido</i>	<i>presenter</i> do método da classe <i>LoginActivity</i>
<i>PresenterLogin</i>	construtor
<i>setView</i>	<i>presenter</i> para construção da janela de diálogo

Fonte: Elaborado pelo autor

ModelLogin: Esta é classe de modelagem (regra de negócio) (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 40:

Tabela 40- Métodos do Diagrama de classe de login.

Método	Descrição
<i>checkLogin</i>	realiza requisição API para realizar Login
<i>computeHash</i>	gerador de criptografia SHA-256
<i>ModelLogin</i>	construtor da classe

Fonte: Elaborado pelo autor

Interface MVPLogin::View: Esta classe tem por objetivo realizar a interface para a view (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 41:

Tabela 41 - Métodos do Diagrama de classe de login.

Método	Descrição
<i>Error</i>	Interface para o método <i>error</i>
<i>login_invalido</i>	Interface para o método <i>login_invalido</i>
<i>login_valido</i>	Interface para o método <i>login_valido</i>

Fonte: Elaborado pelo autor

Interface MVPLogin::Presenter: Esta classe tem por objetivo realizar a interface para o *presenter* (Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 42:

Tabela 42 - Métodos do Diagrama de classe de login.

Método	Descrição
<i>checkLogin</i>	Interface para o método <i>checkLogin</i>
<i>error</i>	Interface para o método <i>error</i>
<i>getContext</i>	Interface para o método <i>getContext</i>
<i>login_invalido</i>	Interface para o método <i>login_invalido</i>
<i>login_valido</i>	Interface para o método <i>login_valido</i>
<i>setView</i>	Interface para o método <i>setView</i>

Fonte: Elaborado pelo autor

Interface MVPLogin:: Model: Esta classe tem por objetivo realizar a interface para o *model*(Figura 14). Os métodos identificados para esta classe são apresentados na Tabela 43:

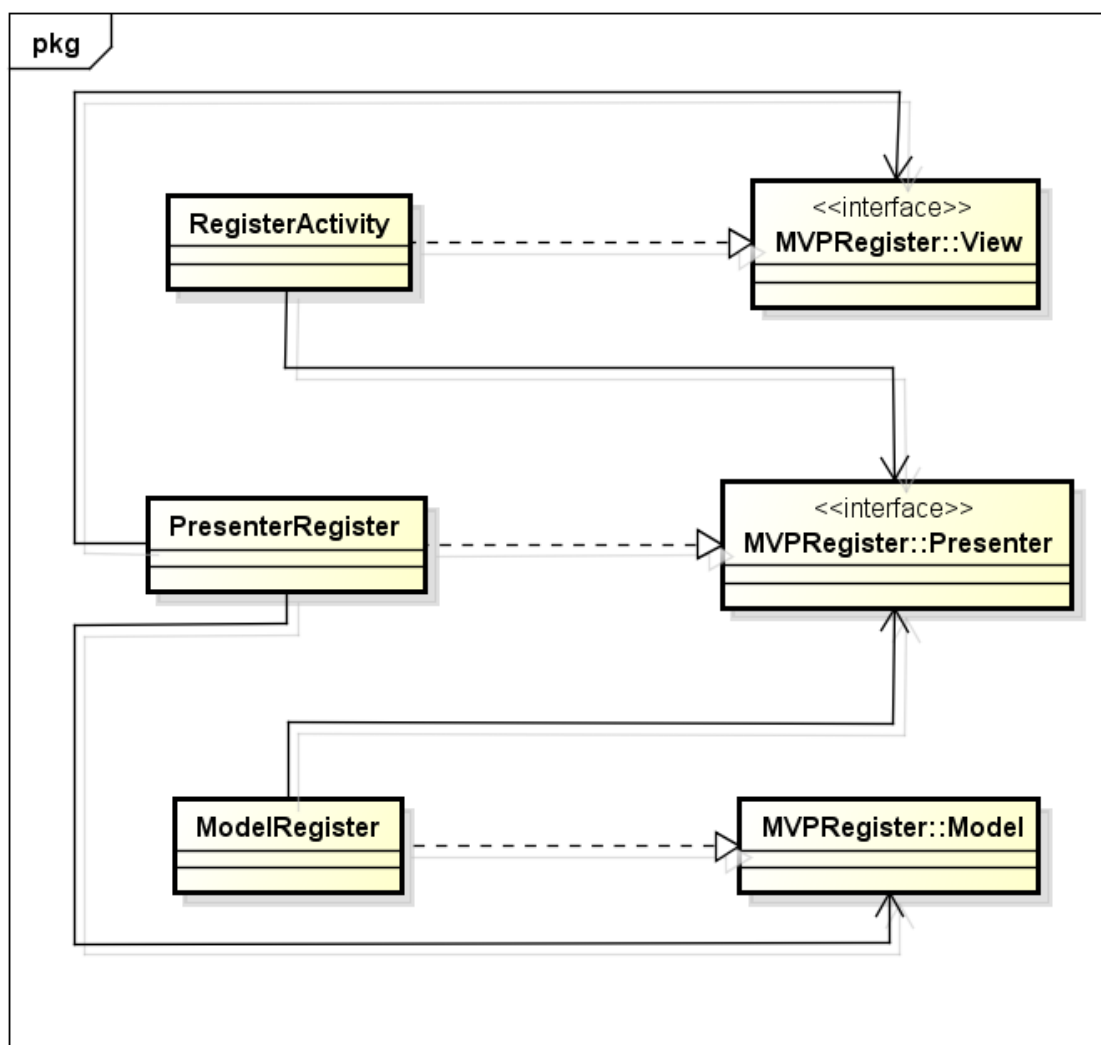
Tabela 43 - Métodos do Diagrama de classe de login.

Método	Descrição
<i>checkLogin</i>	Interface para o método <i>checkLogin</i>

Fonte: Elaborado pelo autor

A Figura 15 apresenta o diagrama de classe de Register.

Figura 15 - Diagrama de classe de Register.



Fonte: Elaborado pelo autor

RegisterActivity: Esta classe tem por objetivo receber os dados do usuário (Figura 15). Os métodos identificados para esta classe são apresentados na Tabela 44:

Tabela 44 – Métodos do Diagrama de classe de Register.

Método	Descrição
<i>cadastrar</i>	valida os campos
<i>cadastro_realizado</i>	abre a tela de <i>Splash</i>
<i>entrar</i>	abre a tela de Login
<i>erro</i>	emite uma janela de diálogo para informar de erro
<i>onCreate</i>	estabelece contato com o Layout da <i>Activity</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

PresenterRegister: Esta classe representa a camada *Presenter* da arquitetura MVP (Figura 15). Os métodos identificados para esta classe são apresentados na Tabela 45:

Tabela 45 - Métodos do Diagrama de classe de Register.

Método	Descrição
<i>PresenterRegister</i>	construtor do <i>presenter</i>
<i>Cadastrar</i>	<i>presenter</i> do método da classe <i>RegisterActivity</i>
<i>getContext</i>	<i>presenter</i> do método que retorna o contexto para a classe que o chamar
<i>setView</i>	<i>presenter</i> para construção da janela de dialogo
<i>erro</i>	<i>presenter</i> do método da classe <i>LoginActivity</i>
<i>cadastro_realizado</i>	<i>presenter</i> do método da classe <i>RegisterActivity</i>

Fonte: Elaborado pelo autor

ModelRegister: Esta classe de modelagem (regra de negócio) (Figura 15). Os métodos identificados para esta classe são apresentados na Tabela 46:

Tabela 46- Métodos do Diagrama de classe de Register.

Método	Descrição
<i>Cadastrar</i>	realiza requisição API para realizar Cadastro
<i>computeHash</i>	gerador de criptografia SHA-256
<i>data_sql</i>	converte formato da data para o formato aceito no SQL
<i>ModelRegister</i>	construtor da classe

Fonte: Elaborado pelo autor

Interface MVPRegister::View. Esta classe tem por objetivo realizar a interface para a view (Figura 15). Os métodos identificados para esta classe são apresentados na Tabela 47:

Tabela 47 - Métodos do Diagrama de classe de Register.

Método	Descrição
<i>Erro</i>	Interface para o método <i>error</i>
<i>cadastro_realizado</i>	Interface para o método <i>error</i>

Fonte: Elaborado pelo autor

Interface MVPRegister::Presenter. Esta classe tem por objetivo realizar a interface para o *presenter* (Figura 15). Os métodos identificados para esta classe são apresentados na Tabela 48:

Tabela 48 - Métodos do Diagrama de classe de Register.

Método	Descrição
<i>Cadastrar</i>	Interface para o método <i>cadastro_realizado</i>
<i>cadastro_realizado</i>	Interface para o método <i>cadastro_realizado</i>
<i>erro</i>	Interface para o método <i>error</i>
<i>getContext</i>	Interface para o método <i>getContext</i>
<i>setView</i>	Interface para o método <i>setView</i>

Fonte: Elaborado pelo autor

Interface MVPRegister:: Model: Esta classe tem por objetivo realizar a interface para o *model*(Figura 15). Os métodos identificados para esta classe são apresentados na Tabela 49:

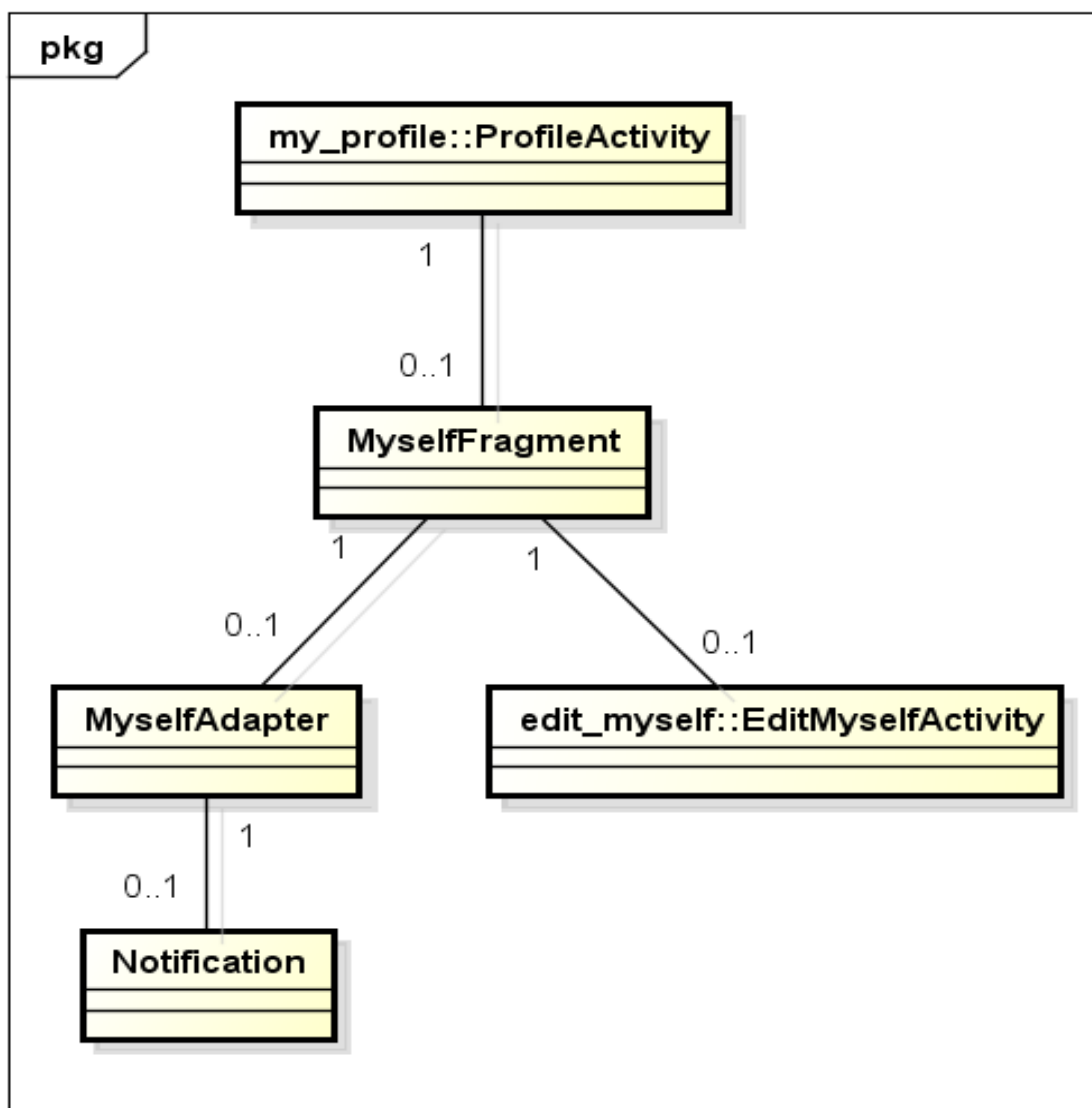
Tabela 49 - Métodos do Diagrama de classe de Register.

Método	Descrição
<i>cadastrar</i>	Interface para o método cadastrar

Fonte: Elaborado pelo autor

A Figura 16 apresenta o diagrama de classe de perfil de usuário.

Figura 16 - Diagrama de classe de perfil do usuário.



Fonte: Elaborado pelo autor

ProfileActivity: Esta classe tem por objetivo carregar os *fragments* do perfil do usuário (Figura 16). Os métodos identificados para esta classe são apresentados na Tabela 50:

Tabela 50 – Métodos do Diagrama de classe de perfil do usuário.

Método	Descrição
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onCreateOptionsMenu</i>	estabelece contato com o <i>Layout</i> do Menu, inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onOptionsItemSelected</i>	abre a tela de adicionar Time
<i>onPrepareOptionsMenu</i>	prepara a tela para mostrar as opções de menu
<i>onActivityResult</i>	abre a tela de times

Fonte: Elaborado pelo autor

EditMyselfActivity: Esta classe tem por objetivo realizar carregamento dos dados da tela de editar usuário (Figura 16). Os métodos identificados para esta classe são apresentados na Tabela 51:

Tabela 51 - Métodos do Diagrama de classe de perfil do usuário.

Método	Descrição
<i>setImagePath</i>	carrega imagem de um diretório
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

MyselfFragment: Esta classe tem por objetivo realizar carregamento dos dados pessoais do perfil (Figura 16). Os métodos identificados para esta classe são apresentados na Tabela 52:

Tabela 52- Métodos do Diagrama de classe de perfil do usuário.

Método	Descrição
<i>MyselfFragment</i>	construtor da classe
<i>getUserinfo</i>	requisita a API os dados do usuário
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

MyselfAdapter: Esta classe infla o conteúdo nos *cards* de notificação (Figura 16). Os métodos identificados para esta classe são apresentados na Tabela 53:

Tabela 53 - Métodos do Diagrama de classe de perfil do usuário.

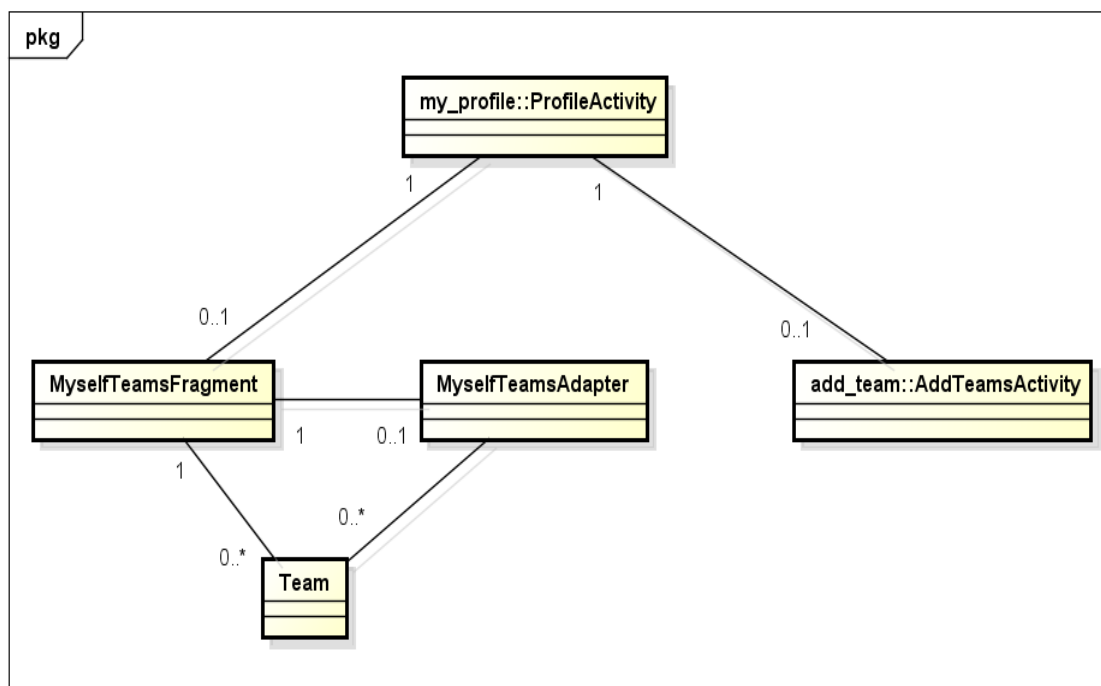
Método	Descrição
<i>MyselfAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de <i>Notifications</i>
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Notification: Esta classe tem por objetivo fornecer a classe *MyselfAdapter* os atributos necessários para as regras de negócio (Figura 16). Não há métodos nesta classe.

A Figura 17 apresenta o diagrama de classe de times de usuário.

Figura 17 - Diagrama de classe de times do usuário.



Fonte: Elaborado pelo autor

ProfileActivity: Esta classe tem por objetivo carregar os *fragments* do perfil do usuário (Figura 17). Os métodos identificados para esta classe são apresentados na Tabela 54:

Tabela 54 – Métodos do Diagrama de classe de times do usuário.

Método	Descrição
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onCreateOptionsMenu</i>	estabelece contato com o <i>Layout</i> do Menu, inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onOptionsItemSelected</i>	abre a tela de adicionar Time
<i>onPrepareOptionsMenu</i>	prepara a tela para mostrar as opções de menu
<i>onActivityResult</i>	abre a tela de times

Fonte: Elaborado pelo autor

AddTeamsActivity: Esta classe tem por objetivo realizar carregamento da tela de adicionar um time (Figura 17). Os métodos identificados para esta classe são apresentados na Tabela 55:

Tabela 55 - Métodos do Diagrama de classe de times do usuário.

Método	Descrição
<i>base64</i>	converte imagem em base64
<i>createTeam</i>	valida os campos
<i>sendData</i>	requisita da API a lista de times do usuário
<i>setImagePath</i>	carrega a imagem e o seu diretório
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

MyselfTeamsFragment: Esta classe tem por objetivo realizar carregamento da lista de times do usuário (Figura 17). Os métodos identificados para esta classe são apresentados na Tabela 56:

Tabela 56- Métodos do Diagrama de classe de times do usuário.

Método	Descrição
<i>getMyselfTeams</i>	requisita da API a lista de times do usuário
<i>MyselfTeamsFragment</i>	construtor da classe
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

MyselfTeamsAdapter: Esta classe infla os *cards* dos times do usuário com conteúdo (Figura 17). Os métodos identificados para esta classe são apresentados na Tabela 57:

Tabela 57 - Métodos do Diagrama de classe de times do usuário.

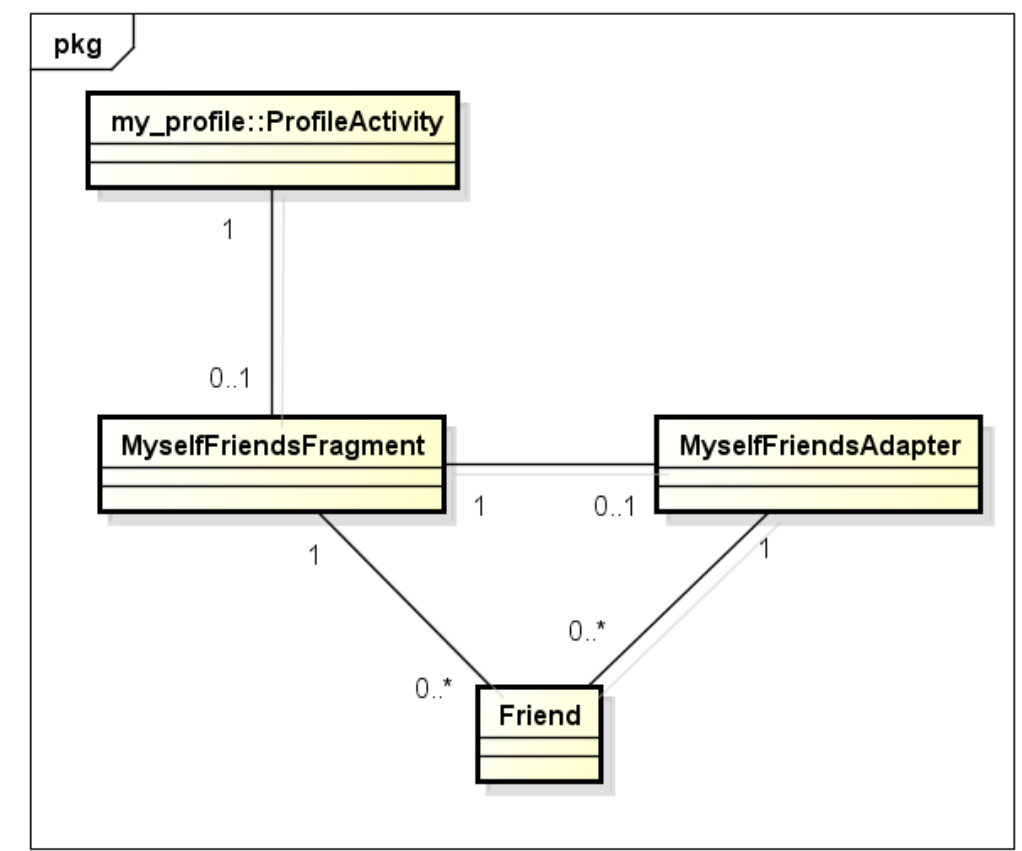
Método	Descrição
<i>MyselfFriendsAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Times
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Time: Esta classe tem por objetivo fornecer a classe *MyselfTeamsAdapter* e *MyselfTeamsFragment* os atributos necessários para as regras de negócio (Figura 17). Não há métodos nesta classe.

A Figura 18 apresenta o diagrama de amizade do usuário.

Figura 18 - Diagrama de classe de amizade do usuário.



Fonte: Elaborado pelo autor

ProfileActivity: Esta classe tem por objetivo carregar os *fragments* do perfil do usuário (Figura 18). Os métodos identificados para esta classe são apresentados na Tabela 58:

Tabela 58 – Métodos do Diagrama de classe de amizade do usuário.

Método	Descrição
--------	-----------

<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onCreateOptionsMenu</i>	estabelece contato com o <i>Layout</i> do Menu, inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onOptionsItemSelected</i>	abre a tela de adicionar Time
<i>onPrepareOptionsMenu</i>	prepara a tela para mostrar as opções de menu
<i>onActivityResult</i>	abre a tela de times

Fonte: Elaborado pelo autor

MyselfFriendsFragment: Esta classe tem por objetivo realizar carregamento e exibição da lista de seguidos/seguindo (Figura 18). Os métodos identificados para esta classe são apresentados na Tabela 59:

Tabela 59 - Métodos do Diagrama de classe de amizades do usuário.

Método	Descrição
<i>getFriendship</i>	requisita da API a lista de amizades
<i>MyselfFriendsFragment</i>	construtor da classe
<i>seguidoresClicked</i>	prepara requisição por filtro de seguidores
<i>aeguindoClicked</i>	prepara requisição por filtro de seguindo
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

MyselfFriendsAdapter: Esta classe infla o *card* de amizades com o conteúdo (Figura 18). Os métodos identificados para esta classe são apresentados na Tabela 60:

Tabela 60- Métodos do Diagrama de classe de amizades do usuário.

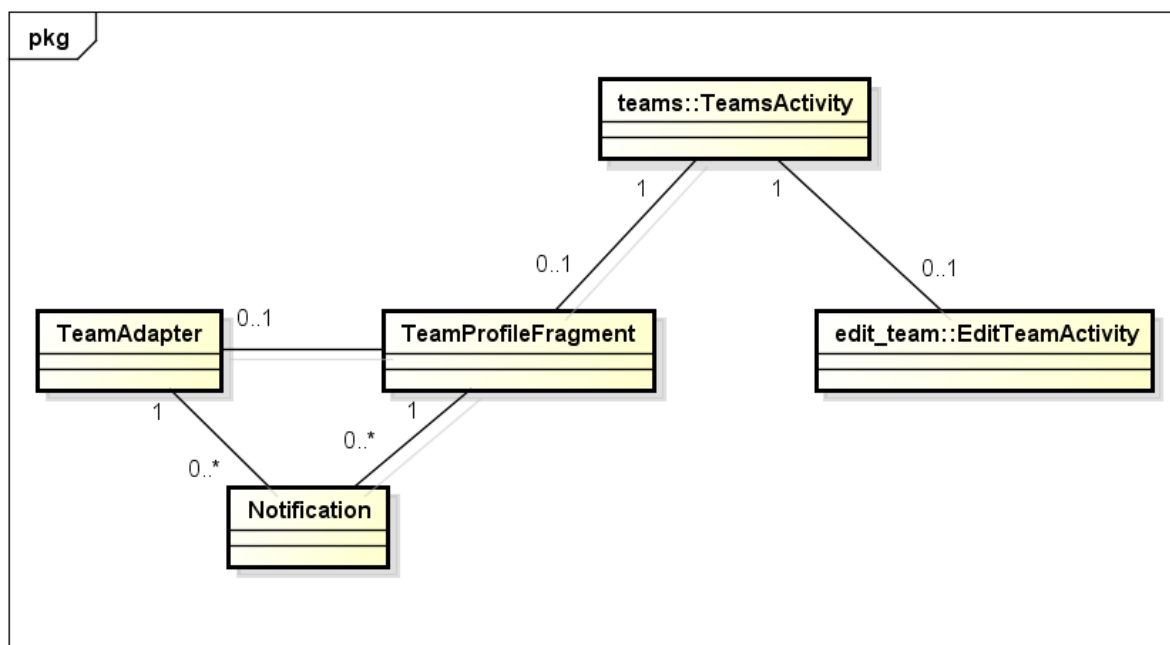
Método	Descrição
<i>MyselfFriendsAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Amizades
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o Layout do Fragment, inflando componente XML para transforma-lo em objeto do tipo Views

Fonte: Elaborado pelo autor

Friend: Esta classe tem por objetivo fornecer a classe *MyselfFriendAdapter* e *MyselfFriendFragment* atributos necessários para as regras de negócio (Figura 18). Não há métodos nesta classe.

A Figura 19 apresenta o diagrama de classe de perfil do time.

Figura 19 - Diagrama de classe de perfil do time.



Fonte: Elaborado pelo autor

TeamsActivity: Esta classe tem por objetivo carregar os *fragments* do perfil do time (Figura 19). Os métodos identificados para esta classe são apresentados na Tabela 61:

Tabela 61 – Métodos do Diagrama de classe de perfil do time.

Método	Descrição
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

<i>onCreateOptionsMenu</i>	estabelece contato com o <i>Layout</i> do Menu, inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>
<i>onOptionsItemSelected</i>	abre a tela de adicionar Time
<i>onPrepareOptionsMenu</i>	prepara a tela para mostrar as opções de menu
<i>onActivityResult</i>	abre a tela de jogadores

Fonte: Elaborado pelo autor

EditTeamActivity: Esta classe tem por objetivo realizar carregamento da tela de editar perfil do time (Figura 19). Os métodos identificados para esta classe são apresentados na Tabela 62:

Tabela 62 - Métodos do Diagrama de classe de perfil do time.

Método	Descrição
<i>setImagePath</i>	carrega imagem de um diretório
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

TeamProfileFragment: Esta classe tem por objetivo realizar carregamento dos dados do time (Figura 19). Os métodos identificados para esta classe são apresentados na Tabela 63:

Tabela 63- Métodos do Diagrama de classe de perfil do time.

Método	Descrição
<i>TeamProfileFragment</i>	construtor da classe
<i>getTeamDetails</i>	requisita da API os dados do Time
<i>changeButtonState</i>	torna botão de editar perfil do Time visível somente ao capitão do time
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

TeamAdapter. Esta classe infla os *cards* de notificações do perfil de time com conteúdo (Figura 19). Os métodos identificados para esta classe são apresentados na Tabela 64:

Tabela 64- Métodos do Diagrama de classe de perfil do time.

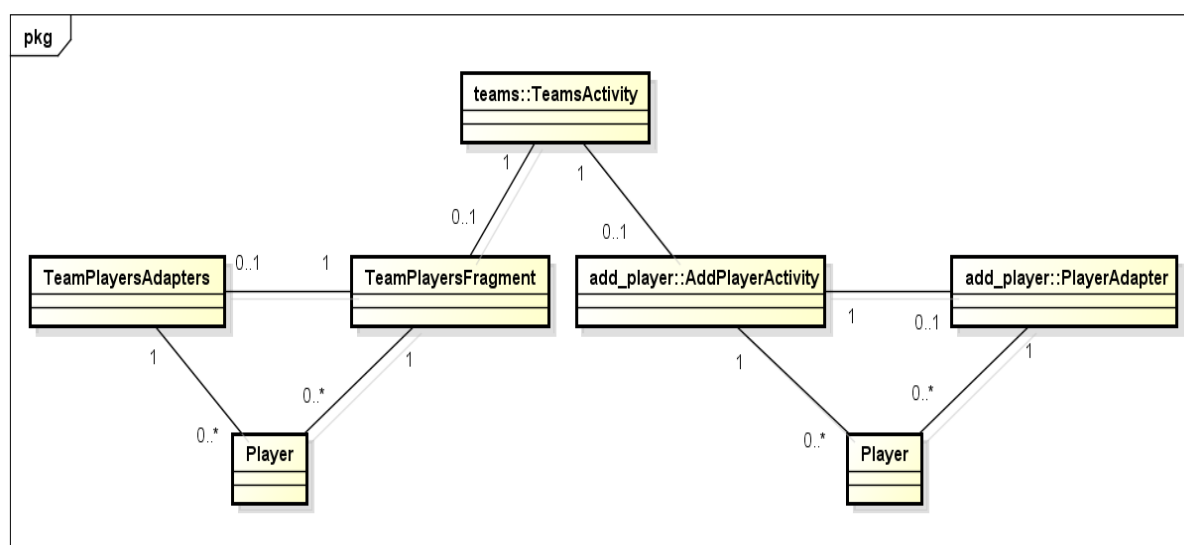
Método	Descrição
<i>TeamAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de <i>Notifications</i>
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Notification: Esta classe tem por objetivo fornecer a classe *TeamAdapter* e *TeamProfileFragment* os atributos necessários para as regras de negócio (Figura 19). Não há métodos nesta classe.

A Figura 20 apresenta o diagrama de classe de jogadores do time.

Figura 20 - Diagrama de classe de jogadores do time.



Fonte: Elaborado pelo autor

TeamsActivity: Esta classe tem por objetivo carregar os *fragments* do perfil do time (Figura 20). Os métodos identificados para esta classe são apresentados na Tabela 65:

Tabela 65 – Métodos do Diagrama de classe de jogadores do time.

Método	Descrição
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onCreateOptionsMenu</i>	estabelece contato com o <i>Layout</i> do Menu, inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onOptionsItemSelected</i>	abre a tela de adicionar Time
<i>onPrepareOptionsMenu</i>	prepara a tela para mostrar as opções de menu
<i>onActivityResult</i>	abre a tela de Jogadores

Fonte: Elaborado pelo autor

AddPlayerActivity: Esta classe tem por objetivo realizar carregamento da lista de jogadores para adicionar (Figura 20). Os métodos identificados para esta classe são apresentados na Tabela 66:

Tabela 66 - Métodos do Diagrama de classe de jogadores do time.

Método	Descrição
<i>addPlayertoTeam</i>	faz requisição na API para adicionar um novo jogador ao time
<i>getPlayers</i>	busca jogadores do time
<i>firstFilterClicked</i>	configura requisição para filtrar jogadores que estiverem na cidade
<i>secondFilterClicked</i>	configura requisição para filtrar jogadores que estiverem na região
<i>thirdFilterClicked</i>	configura requisição para filtrar jogadores que estiverem no estado
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

PlayersAdapter: Esta classe infla o *card* de jogadores a adicionar com conteúdo (Figura 20). Os métodos identificados para esta classe são apresentados na Tabela 67:

Tabela 67- Métodos do Diagrama de classe de jogadores do time.

Método	Descrição
<i>PlayersAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de usuários
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

TeamPlayersFragment. Esta classe tem por objetivo realizar carregamento da lista de jogadores do time (Figura 20). Os métodos identificados para esta classe são apresentados na Tabela 68:

Tabela 68- Métodos do Diagrama de classe de jogadores do time.

Método	Descrição
<i>TeamPlayersFragment</i>	construtor da classe
<i>getPlayers</i>	requisita da API os jogadores do time
<i>removePlayers</i>	torna botão de editar perfil do Time visível somente ao capitão do time
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

TeamAdapter. Esta classe infla os cards de jogadores do time com conteúdo (Figura 20). Os métodos identificados para esta classe são apresentados na Tabela 69:

Tabela 69- Métodos do Diagrama de classe de jogadores do time.

Método	Descrição
<i>TeamAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Jogadores
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo

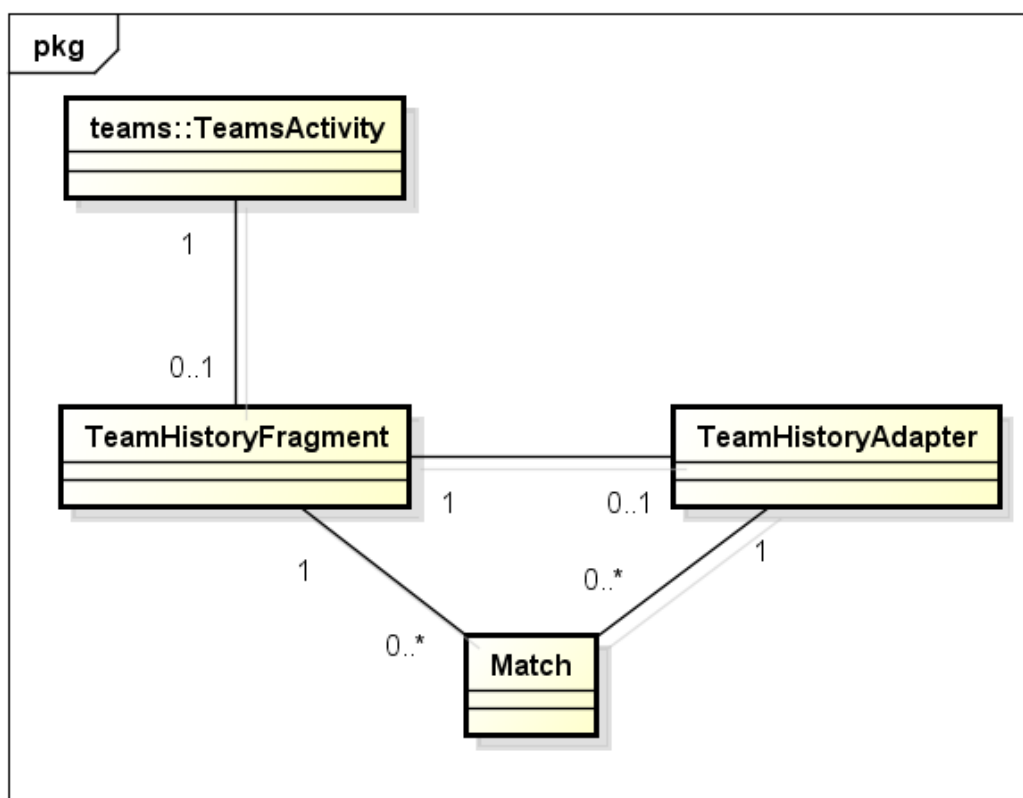
onCreateViewHolder estabelece contato com o *Layout* do *Fragment*, inflando componente XML para transformá-lo em objeto do tipo *Views*

Fonte: Elaborado pelo autor

Jogador. Esta classe tem por objetivo fornecer a classe *TeamPlayerAdapters*, *TeamPlayerFragment* e *AddPlayerActivity*, *PlayerAdapter* os atributos necessários para as regras de negócio (Figura 20). Não há métodos nesta classe.

A Figura 21 apresenta o diagrama de classe de histórico de time.

Figura 21 - Diagrama de classe de histórico do time.



Fonte: Elaborado pelo autor

TeamsActivity: Esta classe tem por objetivo carregar os *fragments* do perfil do time (Figura 21). Os métodos identificados para esta classe são apresentados na Tabela 70:

Tabela 70 – Métodos do Diagrama de classe de histórico do time.

Método	Descrição
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onCreateOptionsMenu</i>	estabelece contato com o <i>Layout</i> do Menu, inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>onOptionsItemSelected</i>	abre a tela de adicionar Time
<i>onPrepareOptionsMenu</i>	prepara a tela para mostrar as opções de menu
<i>onActivityResult</i>	abre a tela de Jogadores

Fonte: Elaborado pelo autor

TeamHistoryFragment: Esta classe tem por objetivo realizar carregamento da lista de histórico de partidas (Figura 21). Os métodos identificados para esta classe são apresentados na Tabela 71:

Tabela 71 - Métodos do Diagrama de classe de histórico do time.

Método	Descrição
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>
<i>TeamHistoryFragment</i>	construtor da classe <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

TeamHistoryAdapter: Esta classe infla os *cards* de histórico com conteúdo (Figura 21). Os métodos identificados para esta classe são apresentados na Tabela 72:

Tabela 72- Métodos do Diagrama de classe de histórico do time.

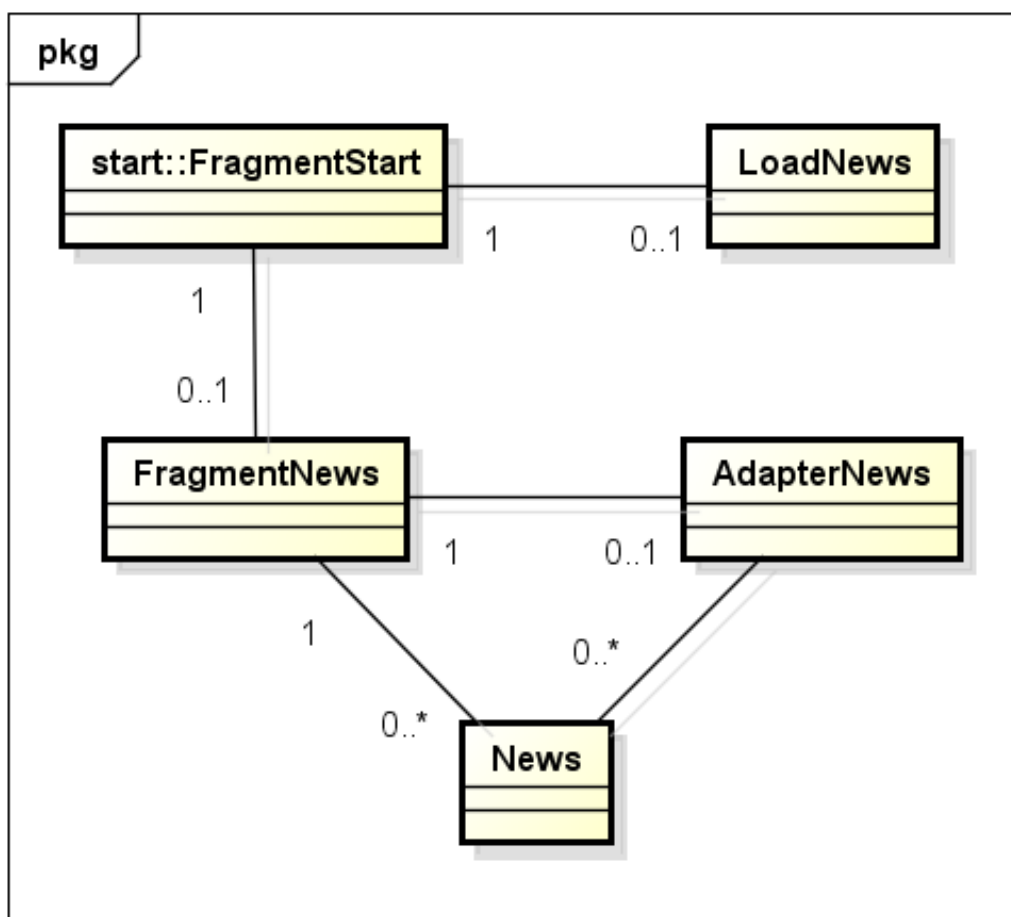
Método	Descrição
<i>TeamHistoryAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de <i>TeamHistorys</i>
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Partida: Esta classe tem por objetivo fornecer a classe *TeamHistoryAdapters* e *TeamHistoyFragment* os atributos necessários para as regras de negócio (Figura 21). Não há métodos nesta classe.

A Figura 22 apresenta o diagrama de classe de notícias.

Figura 22 - Diagrama de classe de notícias.



Fonte: Elaborado pelo autor

FragmentStart: Esta classe tem por objetivo gerenciar a exibição dos fragments notícias e desafio (Figura 22). Os métodos identificados para esta classe são apresentados na Tabela 73:

Tabela 73 – Métodos do Diagrama de classe de news.

Método	Descrição
<i>challenge_click</i>	abre a tela de Desafio
<i>FragmentInicio</i>	construtor da classe
<i>newsClicked</i>	abre a tela de notícias
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

LoadNews: Esta classe tem por objetivo realizara exibição da página de notícia (Figura 22). Os métodos identificados para esta classe são apresentados na Tabela 74:

Tabela 74 - Métodos do Diagrama de classe de notícias.

Método	Descrição
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> da <i>Activity</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

FragmentNews: Esta classe tem por objetivo realizar o carregamento da lista de notícias (Figura 22). Os métodos identificados para esta classe são apresentados na Tabela 75:

Tabela 75- Métodos do Diagrama de classe de News.

Método	Descrição
---------------	------------------

<i>FragmentNews</i>	construtor da classe
<i>getNewsList</i>	requisita da API a lista de notícias
<i>onCreateView</i>	carrega dados nos atributos
<i>populateNewsData</i>	chama o <i>adapter</i> para carregar no <i>Layout</i> os dados obtidos

Fonte: Elaborado pelo autor

AdapterNews: Esta classe infla os *cards* com o conteúdo (Figura 22). Os métodos identificados para esta classe são apresentados na Tabela 76:

Tabela 76- Métodos do Diagrama de classe de News.

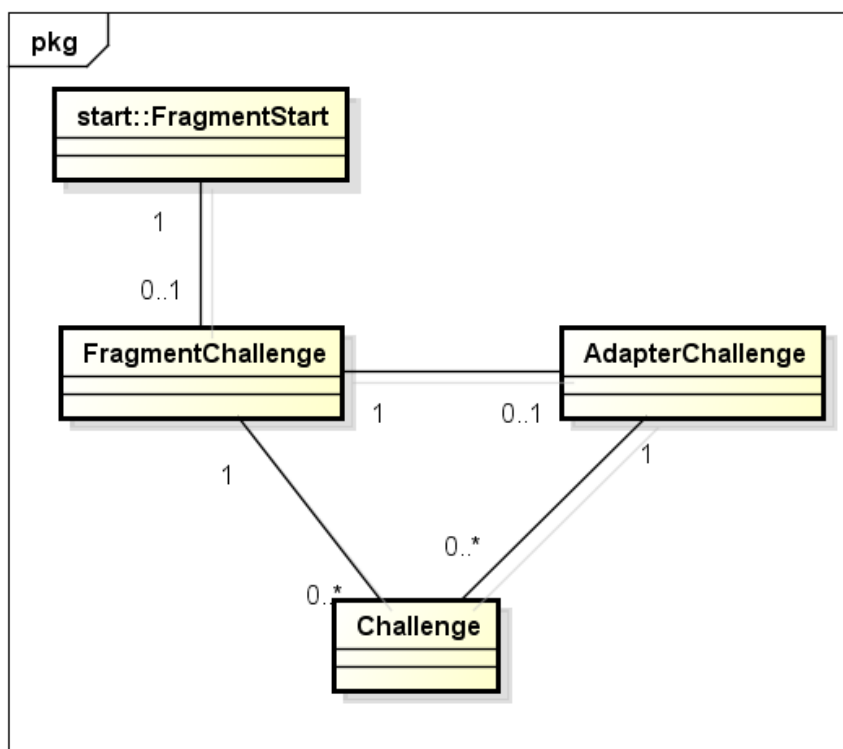
Método	Descrição
<i>AdapterNews</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de notícias
<i>onBindViewHolder</i>	retorna o tamanho da lista de notícias
<i>onCreateViewHolder</i>	carrega um <i>layout</i> que será inflado, ou seja, receberá os dados

Fonte: Elaborado pelo autor

Noticia: Esta classe tem por objetivo fornecer a classe *AdapterNews* e *FragmentNews* os atributos necessários para as regras de negócio (Figura 22). Não há métodos nesta classe.

A Figura 23 apresenta o diagrama de classe de desafio.

Figura 23 - Diagrama de classe de desafio.



Fonte: Elaborado pelo autor

FragmentStart: Esta classe tem por objetivo gerenciar a exibição dos fragments News e challenge (Figura 23). Os métodos identificados para esta classe são apresentados na Tabela 77:

Tabela 77 – Método do Diagrama de classe de Challenge.

Método	Descrição
<i>challenge_click</i>	abre a tela de Desafio
<i>FragmentInicio</i>	construtor da classe
<i>newsClicked</i>	abre a tela de notícias
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

FragmentChallenge: Esta classe tem por objetivo realizar o carregamento da lista de Challenges (Figura 23). Os métodos identificados para esta classe são apresentados na Tabela 78:

Tabela 78 - Método do Diagrama de classe de Challenge.

Método	Descrição
<i>FragmentChallenge</i>	construtor da classe
<i>getChallengeList</i>	requisita da API a lista de desafios
<i>populateNewsData</i>	chama o <i>adapter</i> para carregar no <i>Layout</i> os dados obtidos
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterChallenge: Esta classe infla os *cards* com o conteúdo (Figura 23). Os métodos identificados para esta classe são apresentados na Tabela 79:

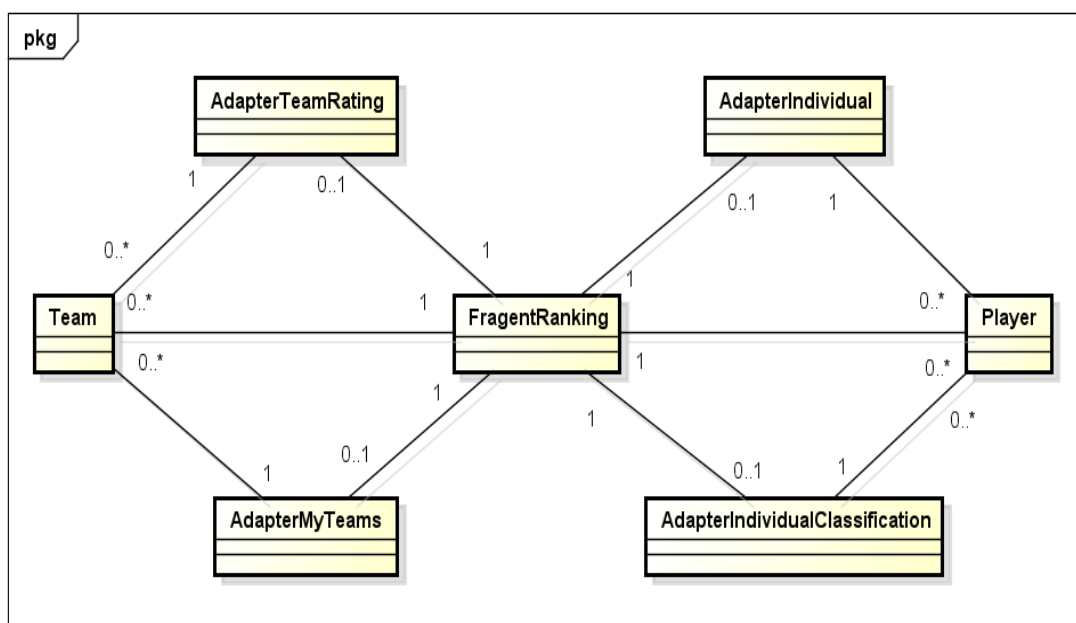
Tabela 79- Método do Diagrama de classe de Challenge.

Método	Descrição
<i>AdapterChallenge</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Desafios
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Challenge: Esta classe tem por objetivo fornecer a classe *AdapterChallenge* e *FragmentChallenge* os atributos necessários para as regras de negócio (Figura 23). Não há métodos nesta classe.

A Figura 24 apresenta o diagrama de classe de *Ranking*.

Figura 24 - Diagrama de classe de *ranking*.

Fonte: Elaborado pelo autor

FragmentRanking: Esta classe tem por objetivo de realizar o carregamento e exibição das listas de times do usuário, *ranking* de times, *card* dos dados pessoais e *ranking* de usuários (Figura 24). Os métodos identificados para esta classe são apresentados na Tabela 80:

Tabela 80 – Método do Diagrama de classe de *ranking*.

Método	Descrição
<i>FragmentRanking</i>	construtor da classe
<i>firstFilterClicked</i>	configura requisição para filtrar usuários da cidade ou time campo
<i>secondFilterClicked</i>	configura requisição para filtrar usuários da região ou time <i>society</i>
<i>thirdFilterClicked</i>	configura requisição para filtrar usuários do estado ou time Salão
<i>getTeams</i>	requisita da API a lista de times do usuário
<i>getTeamsRanking</i>	requisita da API a lista de times
<i>getUser</i>	requisita da API os dados do usuário
<i>getUserRanking</i>	requisita da API a lista de usuários
<i>populateMyTeamCardData</i>	chama o <i>adapter</i> para carregar no Layout os dados obtidos
<i>populateTeamRankingData</i>	chama o <i>adapter</i> para carregar no Layout os dados obtidos
<i>populatePersonalCardData</i>	chama o <i>adapter</i> para carregar no Layout os dados obtidos

<i>populateUserRankingData</i>	chama o <i>adapter</i> para carregar no Layout os dados obtidos
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterMyTeams: Esta classe infla os *cards* de times do usuário com conteúdo (Figura 24). Os métodos identificados para esta classe são apresentados na Tabela 81:

Tabela 81 - Método do Diagrama de classe de *ranking*.

Método	Descrição
<i>AdapterMyTeams</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Times de um usuário
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterTeamRating: Esta classe infla os *cards* do *ranking* de times com conteúdo (Figura 24). Os métodos identificados para esta classe são apresentados na Tabela 82:

Tabela 82- Método do Diagrama de classe de *ranking*.

Método	Descrição
<i>AdapterTteamRating</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Times
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterIndividualClassification: Esta classe infla o *card* com as informações do usuário (Figura 24). Os métodos identificados para esta classe são apresentados na Tabela 83:

Tabela 83- Método do Diagrama de classe de *ranking*.

Método	Descrição
<i>AdapterIndividualClassification</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de usuários do <i>ranking</i>
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterIndividual: Esta classe infla o *card* do *ranking* de usuários com conteúdo (Figura 24). Os métodos identificados para esta classe são apresentados na Tabela 84:

Tabela 84 - Método do Diagrama de classe de *ranking*.

Método	Descrição
<i>AdapterIndividual</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Usuários
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

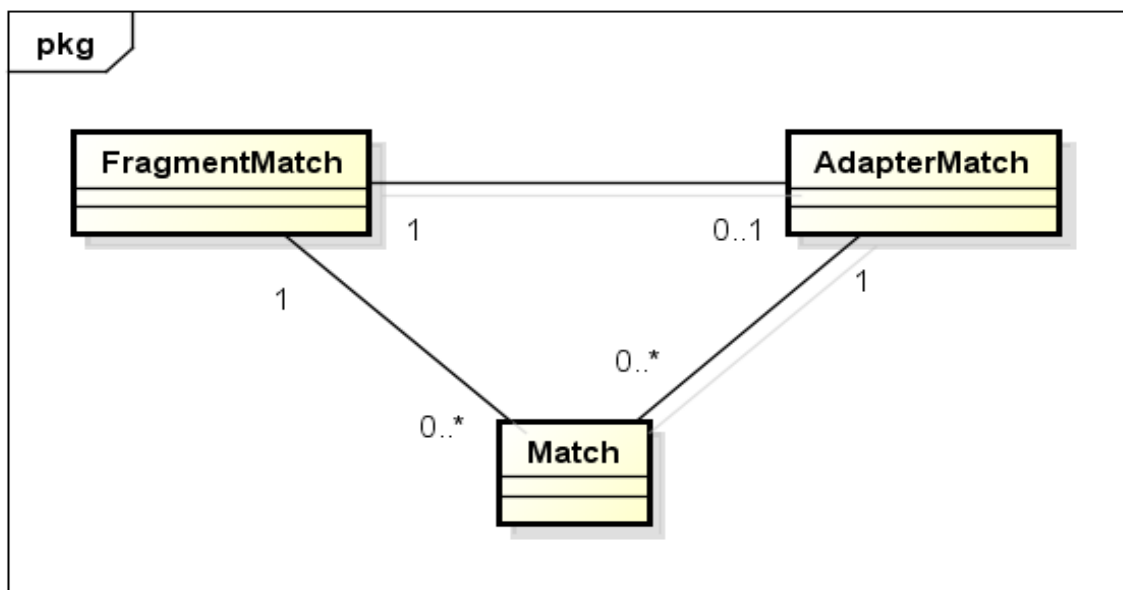
Fonte: Elaborado pelo autor

Time: Esta classe tem o objetivo fornecer a classe *AdapterTeamRating*, *FragmentRanking* e *AdapterMyTeams* os atributos necessários para as regras de negócio (Figura 24). Não há métodos nesta classe.

Jogador. Esta classe tem por objetivo fornecer a classe *AdapterIndividual*, *FragmentRanking* e *AdapterIndividualClassification* os atributos necessários para as regras de negócio (Figura 24). Não há métodos nesta classe.

A Figura 25 apresenta o diagrama de classe de *Match*.

Figura 25 - Diagrama de classe de *Match*



Fonte: Elaborado pelo autor

FragmentMatch: Esta classe tem por objetivo de realizar o carregamento e exibição das listas de partidas (Figura 25). Os métodos identificados para esta classe são apresentados na Tabela 85:

Tabela 85 – Métodos do Diagrama de classe de *Match*.

Método	Descrição
<i>FragmentMatch</i>	construtor da classe
<i>getChallengeList</i>	requisita da API a lista de desafios
<i>optionChallenge</i>	requisita API fazer um <i>update</i> do desafio
<i>populateNewsData</i>	chama o <i>adapter</i> para carregar no Layout os dados obtidos
<i>onCreateView</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterChallenge: Esta classe infla os *cards* de Challenge com conteúdo (Figura 25). Os métodos identificados para esta classe são apresentados na Tabela 86:

Tabela 86 - Métodos do Diagrama de classe de Match.

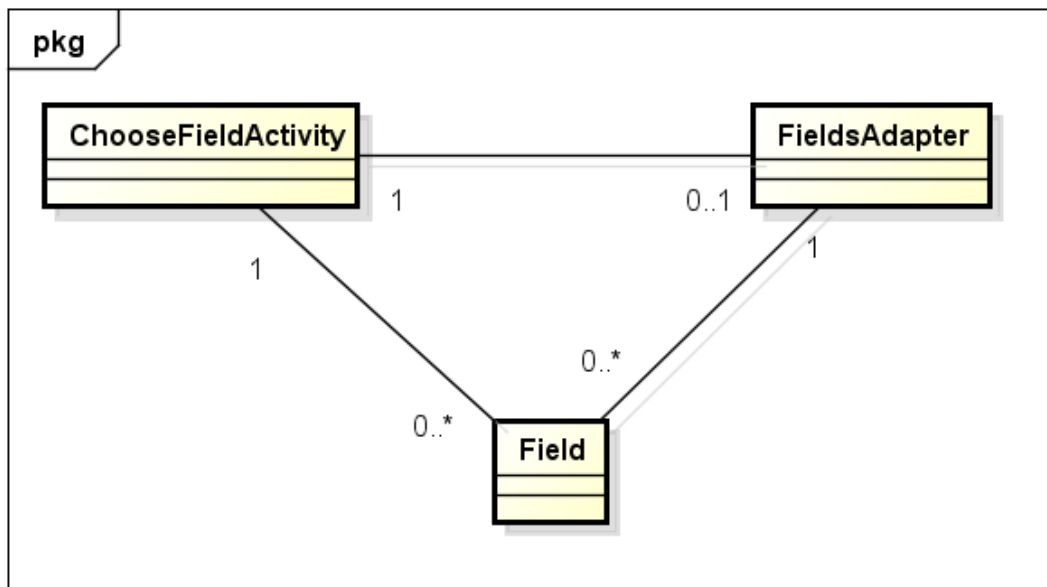
Método	Descrição
<i>AdapterChallenge</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Desafios
<i>onBindViewHolder</i>	atribui ao layout os valores para cada campo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transformá-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Partida: Esta classe tem por objetivo fornecer a classe *FragmentMatch* e *AdapterMatch* os atributos necessários para as regras de negócio (Figura 25). Não há métodos nesta classe.

A Figura 26 apresenta o diagrama de classe de marcar partida/escolher campo.

Figura 26 - Diagrama de classe de marcar partida/escolher campo.



Fonte: Elaborado pelo autor

ChooseFieldActivity: Esta classe tem por objetivo de realizar o carregamento da lista dos *cards* (Figura 26). Os métodos identificados para esta classe são apresentados na Tabela 87:

Tabela 87 – Método do Diagrama de classe de marcar partida/escolher campo.

Método	Descrição
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

FieldsAdapter: Esta classe infla os *cards* com o conteúdo (Figura 26). Os métodos identificados para esta classe são apresentados na Tabela 88:

Tabela 88 - Método do Diagrama de classe de marcar partida/escolher campo.

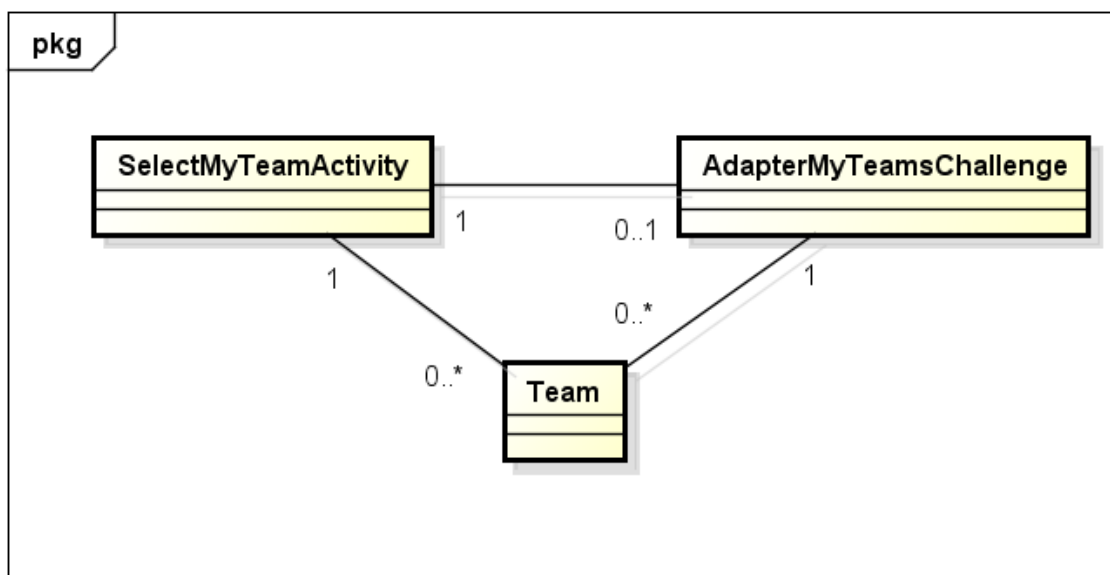
Método	Descrição
<i>FieldsAdapter</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de Campos
<i>onBindViewHolder</i>	atribui ao layout os valores para cada atributo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Campo: Esta classe tem por objetivo fornecer a classe *ChooseFieldActivity* e *FieldsAdapter* os atributos necessários para as regras de negócio (Figura 26). Não há métodos nesta classe.

A Figura 27 apresenta o diagrama de classe de marcar partida/selecionar meu time.

Figura 27 - Diagrama de classe de marcar partida/selecionar meu time.



Fonte: Elaborado pelo autor

SelectMyTeamActivity: Esta classe tem por objetivo de realizar a busca e exibição da lista de times (Figura 27). Os métodos identificados para esta classe são apresentados na Tabela 89:

Tabela 89 – Métodos do Diagrama de classe de marcar partida/selecionar meu time.

Método	Descrição
<i>getTeams</i>	requisita a API uma lista de times
<i>PopulateMyTeamCadData</i>	chama o <i>adapter</i> para carregar no Layout os dados obtidos
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterMyTeamChallenge: Esta classe infla cada *card* com os dados do time (Figura 27). Os métodos identificados para esta classe são apresentados na Tabela 90:

Tabela 90 - Métodos do Diagrama de classe de marcar partida/selecionar meu time.

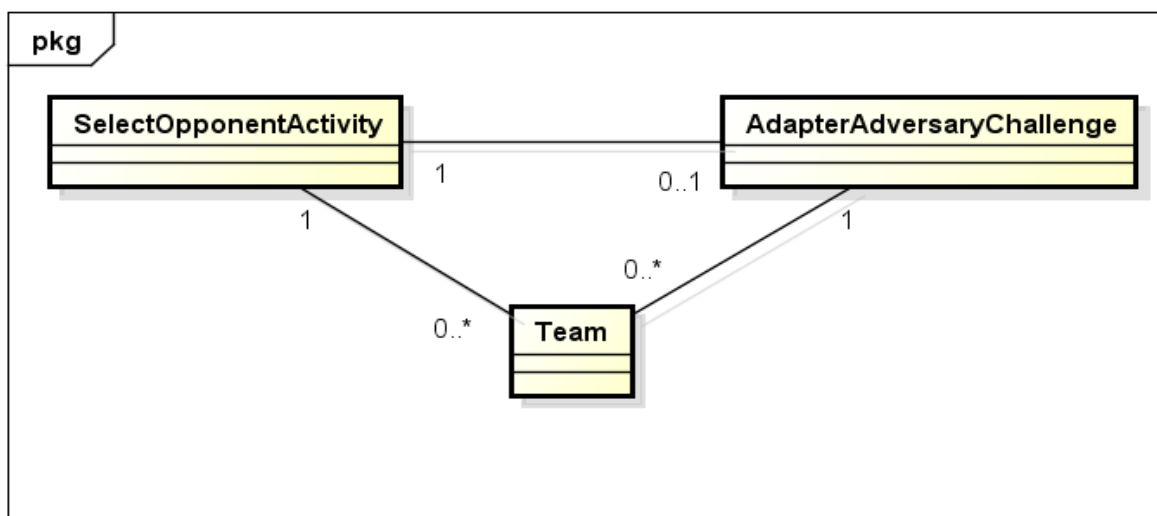
Método	Descrição
<i>AdapterMyTeamChallenge</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de times
<i>onBindViewHolder</i>	atribui ao layout os valores para cada atributo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Time: Esta classe tem por objetivo fornecer a classe *SelectMyTeamActivity* e *AdapterMyTeamChallenge* os atributos necessários para as regras de negócio (Figura 27). Não há métodos nesta classe.

A Figura 28 apresenta o diagrama de classe de marcar partida/selecionar time adversário.

Figura 28 - Diagrama de classe de marcar partida/selecionar time adversário.



Fonte: Elaborado pelo autor

SelectOpponentActivity: Esta classe tem por objetivo de realizar o carregamento e exibição da lista de times adversário (Figura 28). Os métodos identificados para esta classe são apresentados na Tabela 91:

Tabela 91 – Método do Diagrama de classe de marcar partida/selecionar time adversário.

Método	Descrição
<i>firstFilterClicked</i>	configura requisição para filtrar times da cidade
<i>getAversaryTeam</i>	requisita a API uma lista de times
<i>PopulateMyTeamCadData</i>	chama o <i>adapter</i> para carregar no Layout os dados obtidos
<i>secondFilterClicked</i>	configura requisição para filtrar times da região
<i>thirdFilterClicked</i>	configura requisição para filtrar times da região
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

AdapterAdversaryChallenge: Esta classe infla os *cards* dos adversários com conteúdo (Figura 28). Os métodos identificados para esta classe são apresentados na Tabela 92:

Tabela 92 - Método do Diagrama de classe de marcar partida/selecionar time adversário.

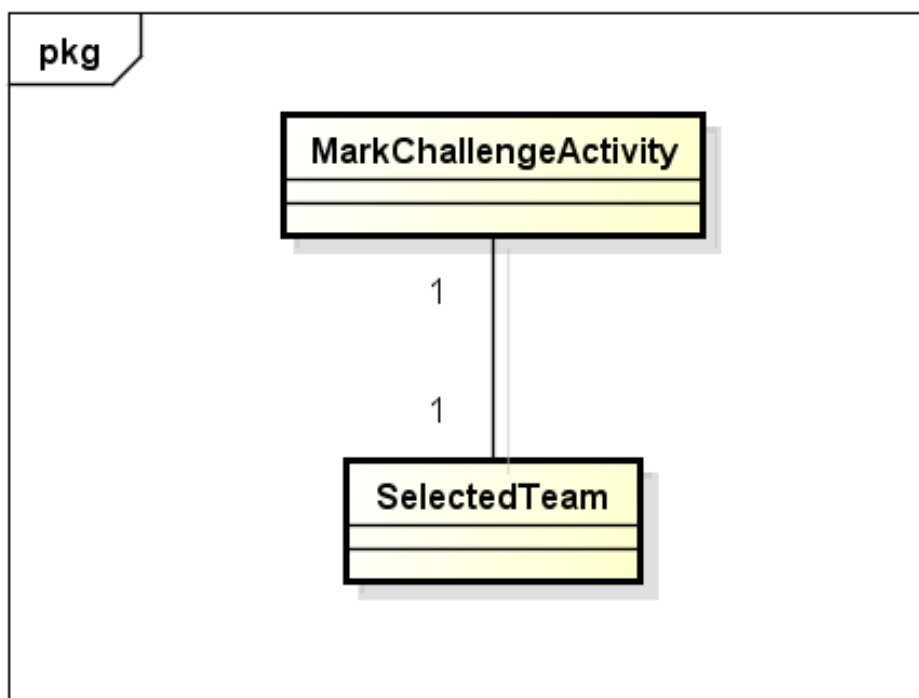
Método	Descrição
<i>AdapterAdversaryChallenge</i>	construtor da classe
<i>getItemCount</i>	retorna o tamanho da lista de times
<i>onBindViewHolder</i>	atribui ao layout os valores para cada atributo
<i>onCreateViewHolder</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

Time: Esta classe tem por objetivo fornecer a classe *SelectOpponentActivity* e *AdapterAdversaryChallenge* os atributos necessários para as regras de negócio (Figura 28). Não há métodos nesta classe.

A Figura 29 apresenta o diagrama de classe de marcar partida/marcar desafio.

Figura 29 - Diagrama de classe de marcar partida/marcar desafio.



Fonte: Elaborado pelo autor

MarkChallengeActivity: Esta classe tem por objetivo de realizar o carregamento do conteúdo e fazer a requisição para a realização do desafio (Figura 29). Os métodos identificados para esta classe são apresentados na Tabela 93:

Tabela 93 – Métodos do Diagrama de classe de marcar partida/escolher campo.

Método	Descrição
<i>desafiar</i>	requisita a API o cadastro de um desafio
<i>onCreate</i>	estabelece contato com o <i>Layout</i> do <i>Fragment</i> , inflando componente XML para transforma-lo em objeto do tipo <i>Views</i>

Fonte: Elaborado pelo autor

SelectedTeam: Esta classe tem por objetivo fornecer a classe *MarkChallengeActivity* os atributos necessários para as regras de negócio (Figura 29). Não há métodos nesta classe.

4 Desenvolvimento

Por se tratar de um software complexo, a metodologia utilizada para o desenvolvimento foi a ágil *Scrum*, pois esta é um *framework* que permite as pessoas tratarem e resolverem problemas herméticos e adaptativos (SCHWABER e SUTHERLAND, 2013), dado que se apoia sobre três pilares de sua implementação, sendo estes:

- **Transparência:** Onde os responsáveis pelos resultados enxergam claramente os aspectos do processo.
- **Inspeção:** Em que, de forma diligente, inspetores especializados analisam constantemente os progressos e variações do processo.
- **Adaptação:** Ajustes realizados de forma a minimizar desvios no processo.

Os três pilares citados aplicam-se sobre as *Sprints*, que são módulos ou versões incrementais potencialmente utilizáveis do produto, sendo este criado em um período que varia em torno de 15 dias, sendo este definido e gerenciado através dos eventos formais: Reunião de planejamento da *Sprint*, Reunião diária, reunião da revisão da *Sprint* e Retrospectiva da *Sprint*.

A metodologia *Scrum* também define a organização da equipe/time *Scrum*, sendo ela composta pela seguinte hierarquia:

- *Product Owner*: Responsável por expressar claramente e ordenar os itens do produto para a equipe.
- *Scrum Master*: Responsável por garantir que o *Scrum*, baseado nos itens definidos pelo *Product Owner*, seja aplicado pela equipe.
- **Time de Desenvolvimento:** profissionais responsáveis em entregar o módulo ou incremento potencialmente usável ao final de cada *Sprint*.

4.1 Etapas de Desenvolvimento

O desenvolvimento deste projeto aconteceu em 5 *sprints* com cerca de 15 dias cada. Seguindo o consenso da equipe de produzir inicialmente o layout do aplicativo e posteriormente aplicar as integrações com as APIs, as entregas resumiram-se,

portanto, em incrementos contendo o *Design* e as funcionalidades internas entre as *Sprint 1* à *Sprint 3* e as integrações com as APIs nas *Sprints 4* e *5*.

4.1.1 *Sprint 1*

No dia 15 de abril de 2018 o grupo se reuniu para realizar o planejamento do primeiro entregável (15 dias de desenvolvimento – prazo 29 de abril de 2018). Nesta reunião, os membros definiram as atividades e seus níveis de dificuldades – representados por pontos. De modo geral, essas atividades estão relacionadas ao desenvolvimento do *layout* e suas funcionalidades no aplicativo. A Tabela 94 apresenta detalhadamente as atividades, seu tempo de realização em dias e sua respectiva pontuação.

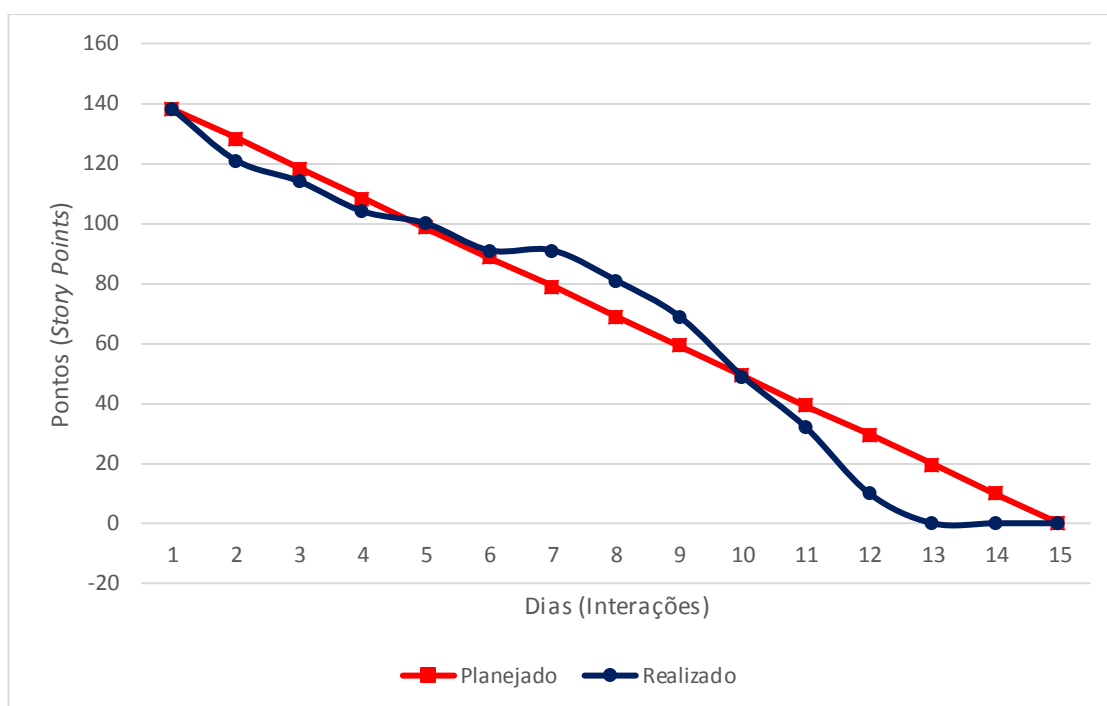
Tabela 94 – Planejamento da *Sprint 1*

Atividade	Tempo (em dias)	Pontos
<i>Design</i> da tela de Login	0,2	7
<i>Design</i> da tela de Cadastro	0,25	7
<i>Design</i> da tela de <i>Splash</i>	0,05	2
Função da tela de <i>Login</i>	0,2	10
Função da tela de Cadastro	0,25	10
Função da tela de <i>Splash</i>	0,05	2
<i>Review</i>	0,5	9
<i>Bug fix</i> e consertos	0,5	10
<i>Design</i> do <i>Float Button</i>	0,3	5
<i>Design</i> da tela de notícias	0,5	5
<i>Design</i> da tela de <i>Ranking times</i>	0,5	7
<i>Design</i> da tela de <i>Ranking individual</i>	0,5	7
Função do <i>Float Button</i>	0,2	10
Função da tela de notícias	0,5	10
Função da tela de <i>Ranking times</i>	0,5	10
Função da tela de <i>Ranking individual</i>	0,5	10
<i>Review</i>	0,5	7
<i>Bug fix</i> e consertos	1	10
Total	7	138

Fonte: Elaborado pelo autor

Durante os 15 dias de desenvolvimento, a equipe realizou baixas na pontuação conforme as atividades foram sendo concluídas. A Figura 30 apresenta o gráfico de *Burndown* da entrega 1, destacando o planejamento de baixas nos pontos e a baixas realizadas pela equipe.

Figura 30 - *Burndown* 1



Fonte: Elaborado pelo autor

No dia 29 de abril de 2018 a equipe se reuniu para realizar a revisão do entregável. Durante a reunião, a equipe fez uma auto avaliação procurando detectar os principais pontos de facilidade e dificuldades, e juntos, propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, a equipe fez o seguinte resumo sobre algumas questões:

- O que deu certo: Houve uma boa comunicação entre os integrantes, isso garantiu uma boa execução das tarefas propostas;
- O que deu errado: Falta do mesmo nível de conhecimento técnico em toda equipe a fim de possibilitar no auxílio de uma codificação padronizada e que siga convenções de desenvolvimento;

- Ações de melhorias: Explicação clara sobre as tarefas realizadas de forma que a equipe por completo possa compreender as tecnologias utilizadas;
- Observações: Um dos integrantes não tinha disponibilidade para reunião para os próximos meses.

4.1.2 *Sprint 2*

No dia 29 de abril de 2018 o grupo se reuniu para realizar o planejamento do primeiro entregável (15 dias de desenvolvimento – prazo 13 de maio de 2018). Nesta reunião, os membros definiram as atividades e seus níveis de dificuldades – representados por pontos. De modo geral, essas atividades estão relacionadas ao desenvolvimento do *layout* e suas funcionalidades no aplicativo. A Tabela 95 apresenta detalhadamente as atividades, seu tempo de realização em dias e sua respectiva pontuação.

Tabela 95 - Planejamento da *Sprint 2*

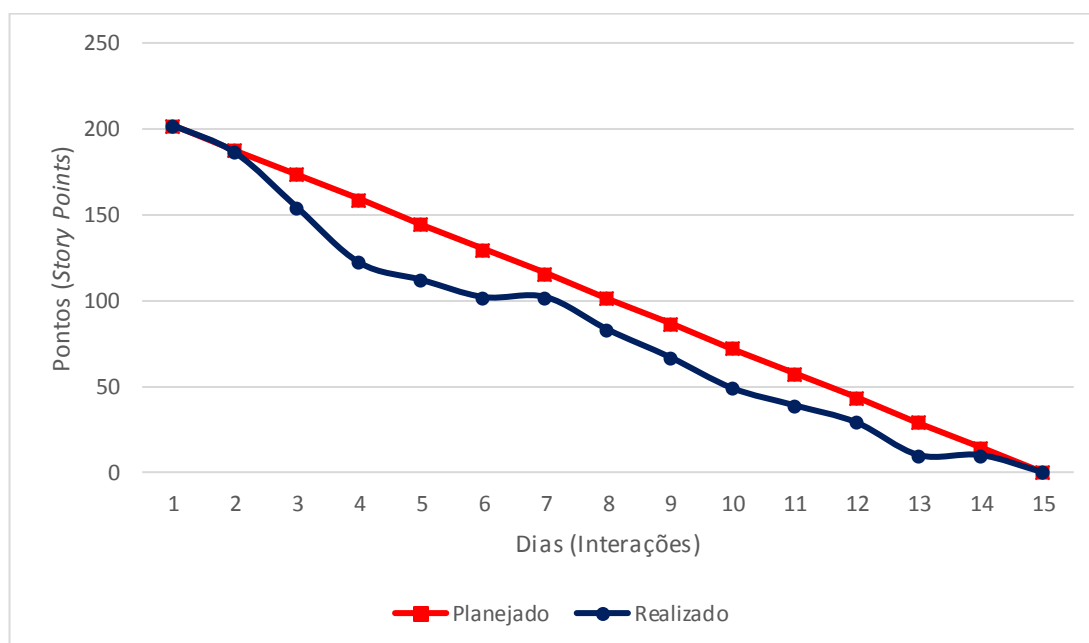
Atividade	Tempo (em dias)	Pontos
<i>Design</i> de Partidas	0,5	7
<i>Design</i> de Marcar Partida - escolher Campo	0,25	7
<i>Design</i> de Marcar Partida - escolher time	0,25	7
<i>Design</i> de Marcar Partida - escolher adversário	0,25	7
<i>Design</i> de Marcar Partida - definir informações	0,25	7
Função do <i>Float Button</i>	0,5	9
Função do Marcar Partida - escolher Campo	0,25	9
Função do Marcar Partida - escolher time	0,25	9
Função de Marcar Partida - escolher adversário	0,25	9
Função de Marcar Partida - definir informações	0,25	9
Conexão com API <i>login</i>	1	10
Conexão com API Cadastro	1	10
<i>Review</i>	0,5	9
<i>Bug fix</i> e consertos	0,5	10
<i>Design</i> do Perfil do Usuário - <i>feed</i>	0,5	8
<i>Design</i> do Perfil do Usuário - amizades	0,5	8
<i>Design</i> do Perfil do Usuário - times	0,5	8

Atividade	Tempo (em dias)	Pontos
Função do <i>Float Button</i> - Perfil	0,5	10
Função do Perfil do Usuário - <i>feed</i>	0,5	10
Função do Perfil do Usuário - amizades	1	10
Função do Perfil do Usuário - times	1	10
<i>Review</i>	0,5	9
<i>Bug fix</i> e concertos	1	10
Total	12	202

Fonte: Elaborado pelo autor

Durante os 15 dias de desenvolvimento, a equipe realizou baixas na pontuação conforme as atividades foram sendo concluídas. A Figura 31 apresenta o gráfico de *Burndown* da entrega 2, destacando o planejamento de baixas nos pontos e a baixas realizadas pela equipe.

Figura 31 - *Burndown* 2



Fonte: Elaborado pelo autor

No dia 13 de maio de 2018 a equipe se reuniu para realizar a revisão do entregável. Lembrando que a ausência de um integrante, obrigou a equipe se reunir com um dos integrantes do projeto, e outro que auxiliou em todo desenvolvimento. Durante a reunião, a equipe fez uma auto avaliação procurando detectar os principais pontos de facilidade e dificuldades, e juntos, propor ações de melhorias para o

desempenho da equipe nos próximos entregáveis. Ao final, a equipe fez o seguinte resumo sobre algumas questões:

- **O que deu certo:** Houve uma boa comunicação entre os integrantes, isso garantiu uma boa execução das tarefas propostas;
- **O que deu errado:** Falta do mesmo nível de conhecimento técnico em toda equipe a fim de possibilitar no auxílio de uma codificação padronizada e que siga convenções de desenvolvimento;
- **Ações de melhorias:** Explicação clara sobre as tarefas realizadas de forma que a equipe por completo possa compreender as tecnologias utilizadas;
- **Observações:** A produção das imagens e logos encontrou-se sob a responsabilidade de um dos integrantes.

4.1.3 *Sprint 3*

No dia 12 de agosto de 2018 o grupo se reuniu para realizar o planejamento do terceiro entregável (15 dias de desenvolvimento – prazo 26 de agosto de 2018). Nesta reunião os membros definiram as atividades e seus níveis de dificuldades – representados por pontos. De modo geral, essas atividades estão relacionadas ao desenvolvimento do *layout* e suas funcionalidades no aplicativo. A Tabela 96 apresenta detalhadamente as atividades, seu tempo de realização em dias e sua respectiva pontuação.

Tabela 96 - Planejamento da *Sprint 3*

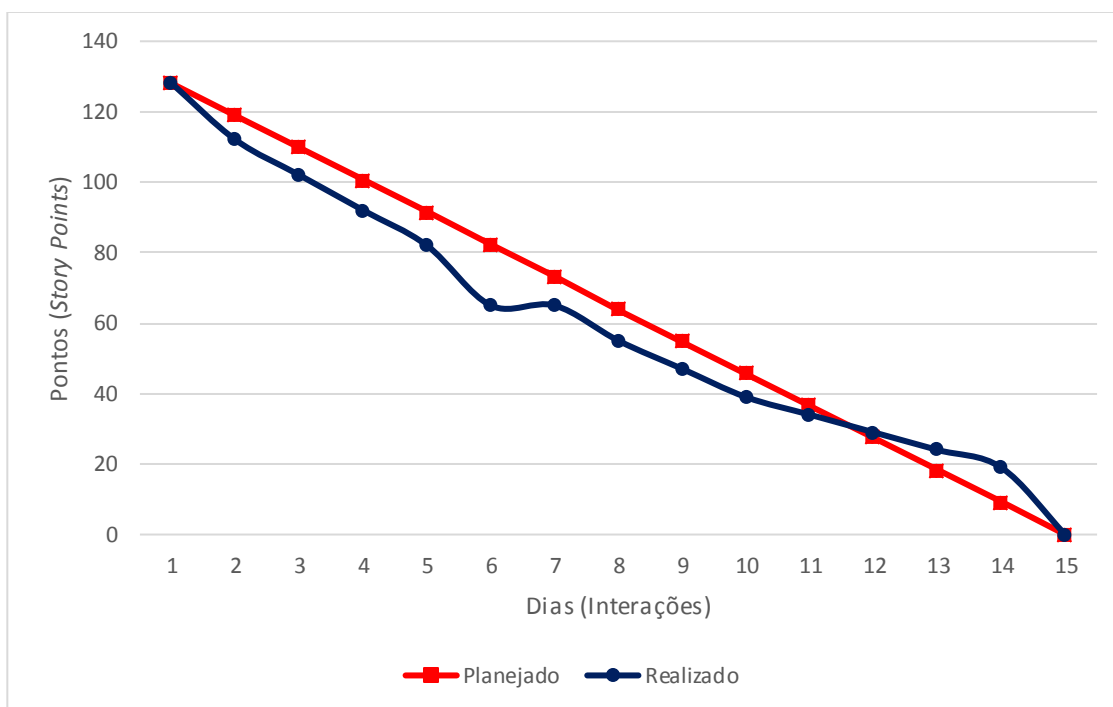
Atividade	Tempo (em dias)	Pontos
<i>Design</i> do Perfil do Time - <i>feed</i>	0,5	8
<i>Design</i> do Perfil do Time - jogadores	0,5	8
<i>Design</i> do Perfil do Time - histórico	0,5	8
Função do Perfil do Time - <i>feed</i>	1	10
Função do Perfil do Time - jogadores	1	10
Função do Perfil do Time - histórico	1	10
<i>Review</i>	0,5	9
<i>Bug fix</i> e consertos	0,5	10
<i>Design</i> do Perfil do Time - Adiciona jogador	1	8

Atividade	Tempo (em dias)	Pontos
Design do Perfil do Usuário - Cria Time	1	8
Função do Perfil do Time - Adiciona jogador	2	10
Função do Perfil do Usuário - Cria Time	2	10
Review	0,5	9
Bug fix e consertos	0,5	10
Total	12,5	128

Fonte: Elaborado pelo autor

Durante os 15 dias de desenvolvimento, a equipe realizou baixas na pontuação conforme as atividades foram sendo concluídas. A Figura 32 apresenta o gráfico de *Burndown* da entrega 3, destacando o planejamento de baixas nos pontos e a baixas realizadas pela equipe.

Figura 32 - *Burndown* 3



Fonte: Elaborado pelo autor

No dia 26 de agosto de 2018 a equipe se reuniu para realizar a revisão do entregável. Durante a reunião, a equipe fez uma auto avaliação procurando detectar os principais pontos de facilidade e dificuldades, e juntos, propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, a equipe fez o seguinte resumo sobre algumas questões:

- **O que deu certo:** Houve uma boa comunicação entre os integrantes, isso garantiu uma boa execução das tarefas propostas;
- **O que deu errado:** Falta de conhecimento técnico mais aprofundado aliado à escassez de tempo disponível para estudo e desenvolvimento resultou em tarefas realizadas de formas que fogem às convenções de desenvolvimento;
- **Ações de melhorias:** Aplicação de um membro para revisão e otimização de *Designs* e funções já desenvolvidas;
- **Observações:** Tarefas realizadas por 2 desenvolvedores a fim de cumprir com o prazo de entrega.

4.1.4 Sprint 4

No dia 26 de agosto de 2018 o grupo se reuniu para realizar o planejamento do quarto entregável (15 dias de desenvolvimento – prazo 09 de setembro de 2018). Nesta reunião os membros definiram as atividades e seus níveis de dificuldades – representados por pontos. De modo geral, essas atividades estão relacionadas a Desenvolvimento e integração de API's. A Tabela 97 apresenta detalhadamente as atividades, seu tempo de realização em dias e sua respectiva pontuação.

Tabela 97 - Planejamento da Sprint 4

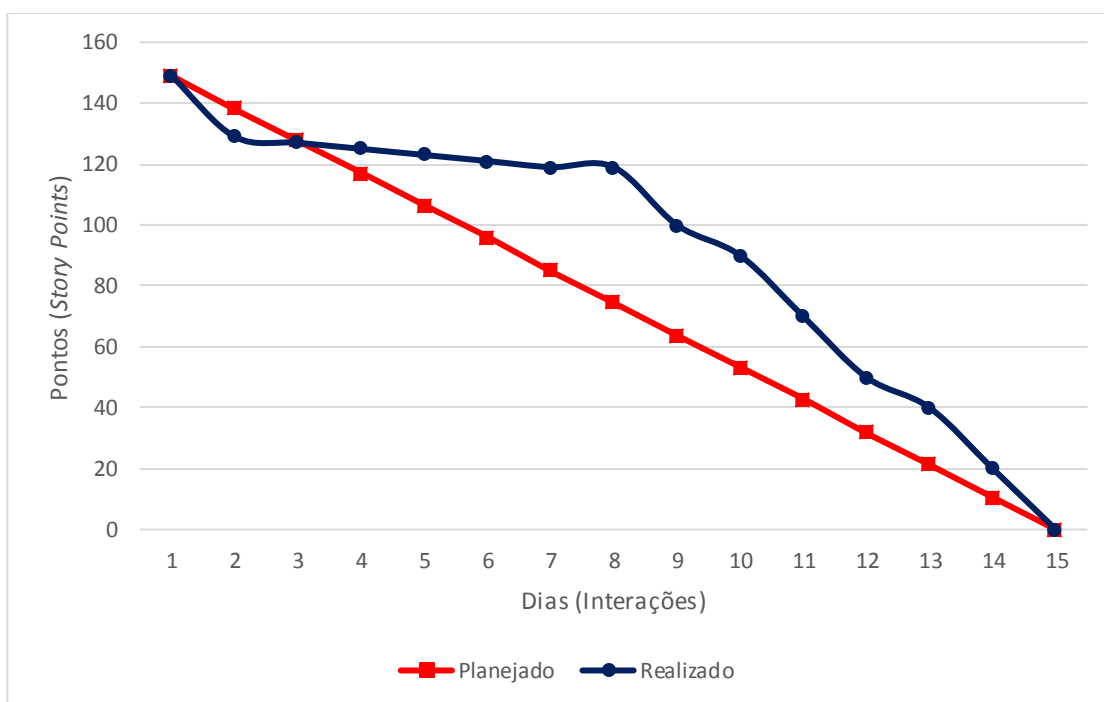
Atividade	Tempo (em dias)	Pontos
Integrar API - <i>ranking</i>	5	10
Otimização na integração de API - login	0,5	10
Otimização na integração de API - cadastro	0,5	10
<i>Review</i>	0,5	9
<i>Bug fix</i> e concertos	0,5	10
Integrar API - Marcar Partida	1	10
Integrar API - Notícias	0,5	10
Integrar API - Desafios	0,5	10
Integrar API - Partidas	0,5	10
Integrar API - Perfil do Usuário	0,5	10
Integrar API - seguidores/seguindo	1	10
Integrar API - times	0,5	10
Integrar API - criar times	0,5	10

Atividade	Tempo (em dias)	Pontos
Review	0,5	10
Bug fix e consertos	0,5	10
Total	13	149

Fonte: Elaborado pelo autor

Durante os 15 dias de desenvolvimento, a equipe realizou baixas na pontuação conforme as atividades foram sendo concluídas. A Figura 33 apresenta o gráfico de *Burndown* da entrega 4, destacando o planejamento de baixas nos pontos e a baixas realizadas pela equipe.

Figura 33 - *Burndown* 4



Fonte: Elaborado pelo autor

No dia 09 de setembro de 2018 a equipe se reuniu para realizar a revisão do entregável. Durante a reunião, a equipe fez uma auto avaliação procurando detectar os principais pontos de facilidade e dificuldades, e juntos, propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, a equipe fez o seguinte resumo sobre algumas questões:

- **O que deu certo:** Houve uma boa comunicação entre os integrantes, isso garantiu uma boa execução das tarefas propostas;

- **O que deu errado:** A complexidade de algumas das telas exigiu um maior tempo para ser realizada, isso resultou no atraso de quase toda a *Sprint*;
- **Ações de melhorias:** Aplicação de codificação em pares de forma que possa garantir a realização da tarefa mais rapidamente;
- **Observações:** Divisão de tarefas entre desenvolvedores afim de garantir o cumprimento do prazo.

4.1.5 *Sprint 5*

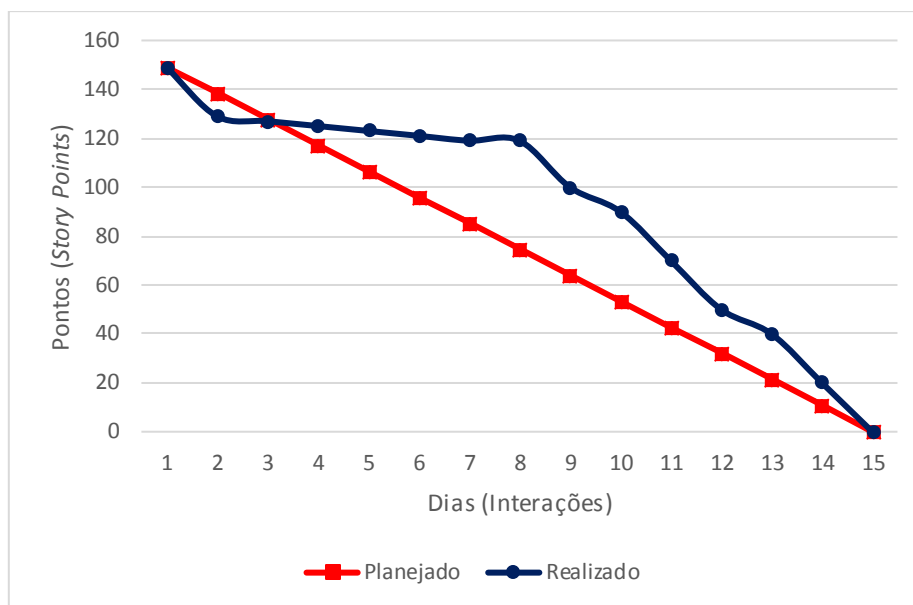
No dia 09 de setembro de 2018 o grupo se reuniu para realizar o planejamento do quinto entregável (15 dias de desenvolvimento – prazo 23 de setembro de 2018). Nesta reunião os membros definiram as atividades e seus níveis de dificuldades – representados por pontos. De modo geral, essas atividades estão relacionadas ao Desenvolvimento e Integração de APIs. A Tabela 98 apresenta detalhadamente as atividades, seu tempo de realização em dias e sua respectiva pontuação.

Tabela 98 - Planejamento da *Sprint 5*

Atividade	Tempo (em dias)	Pontos
Integrar API - Perfil do time	1	10
Integrar API - jogadores do time	1	10
Integrar API - adicionar jogadores ao time	1	10
Log de API	5	10
Review	1	10
Bug fix e consertos	2	10
Total	11	60

Fonte: Elaborado pelo autor

Durante os 15 dias de desenvolvimento, a equipe realizou baixas na pontuação conforme as atividades foram sendo concluídas. A Figura 34 apresenta o gráfico de *Burndown* da entrega 5, destacando o planejamento de baixas nos pontos e a baixas realizadas pela equipe.

Figura 34 - *Burndown 5*

Fonte: Elaborado pelo autor

No dia 23 de setembro de 2018 a equipe se reuniu para realizar a revisão do entregável. Durante a reunião, a equipe fez uma auto avaliação procurando detectar os principais pontos de facilidade e dificuldades, e juntos, propor ações de melhorias para o desempenho da equipe nos próximos entregáveis. Ao final, a equipe fez o seguinte resumo sobre algumas questões:

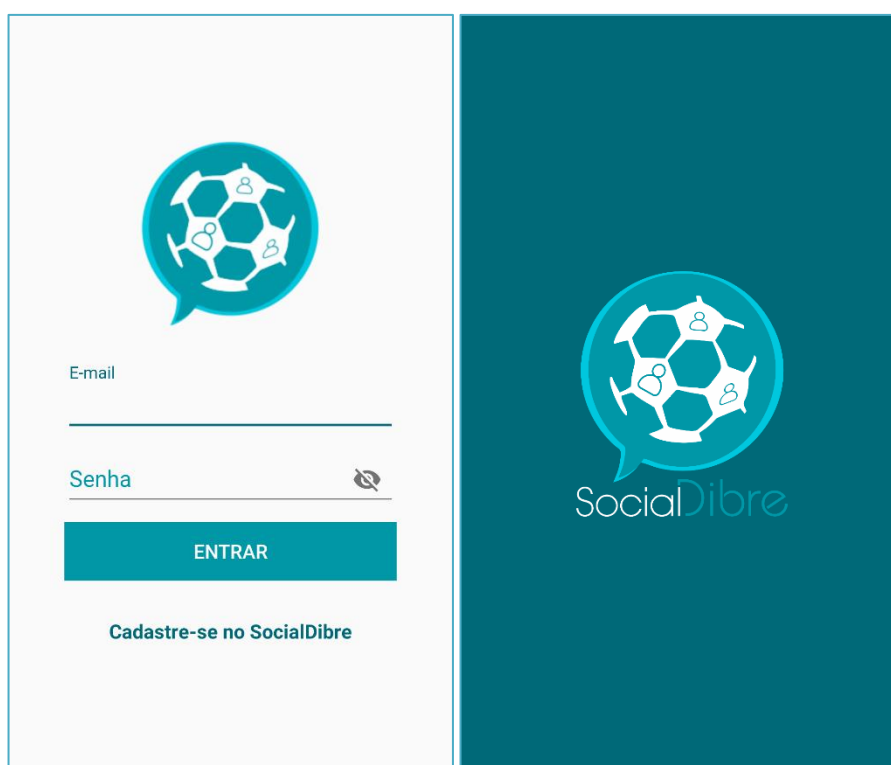
- **O que deu certo:** Houve uma boa comunicação entre os integrantes, isso garantiu uma boa execução das tarefas propostas;
- **O que deu errado:** a falta de disponibilidade da hospedagem online do webservice, resultando na interrupção do desenvolvimento do aplicativo;
- **Ações de melhorias:** Migrar webservice para um ambiente flexível;
- **Observações:** Desenvolvimento interrompido até a conclusão da migração do *WebService*;

4.2 Interfaces de Usuário

A necessidade da construção de uma interface amigável ao usuário é fundamental em um sistema. A interface faz parte do sistema computacional e determina como as pessoas operam e controlam o sistema. Quando uma interface é bem projetada, ela é compreensível, agradável e controlável. Neste contexto, estes protótipos têm como objetivo apresentar a aplicativo e os recursos da tela.

A Figura 35 apresenta a tela de Login, onde se consegue entrar no aplicativo, e ter privilégios de usuário, pode desfrutar do aplicativo, colocando o e-mail e senha pré cadastrados.

Figura 35 – Captura da tela de login e da tela de apresentação (*splash screen*).



Fonte: Elaborado pelo autor

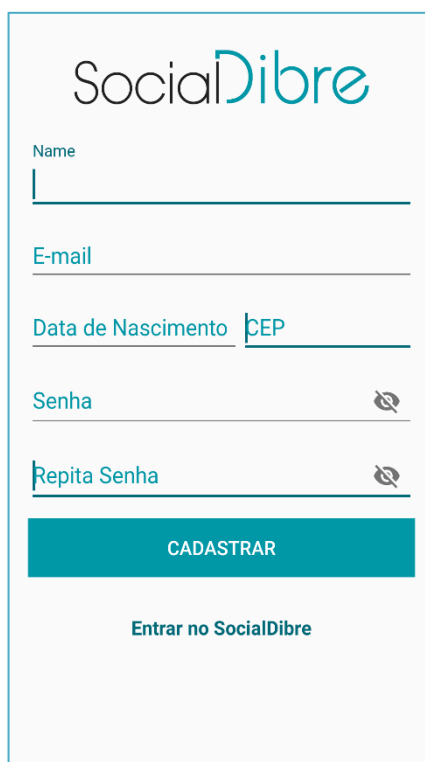
A tela de login apresentada na Figura 35 é composta por:

- **Campo E-mail:** Para colocar seu e-mail antes cadastrado.
- **Campo Senha:** Para colocar sua senha antes cadastrado.
- **Botão Entrar:** Para conseguir entrar no sistema.

- **Link “Cadastrar no SocialDibre”:** Que permite que eu possa ir direto à tela de Cadastro para me cadastrar.
- **Tela de *Splash*:** Tela de carregamento entre *login* e *home*.

A Figura 36 apresenta a tela de cadastro, onde se completa essas informações para ter acesso ao sistema com e-mail e senha, sendo todas informações obrigatórias.

Figura 36 - Captura da Tela de cadastro.



Fonte: Elaborado pelo autor

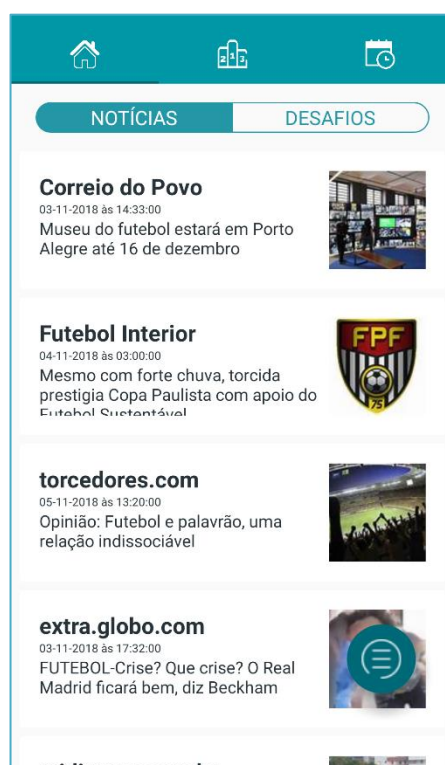
A tela de cadastro apresentada na Figura 36 é composta por:

- **Campo E-mail:** Para colocar seu e-mail.
- **Campo Data de nascimento:** Para colocar sua data de nascimento.
- **Campo CEP:** Para colocar seu CEP.
- **Campo senha e redigite senha:** Para colocar uma senha nos dois.
- **Botão Cadastrar:** Ao clicar nesse botão, salva suas informações declaradas, e podendo ter a aprovação de um futuro login para entrar no sistema.

- **Link “Entrar no SocialDibre”:** Que permite que eu possa ir direto à tela de login se já tiver um cadastro.

A Figura 37 apresenta a tela de Notícias, onde se consegue visualizar tudo o que está acontecendo à sua volta, em relação ao futebol, sendo possível sobre times e jogos também.

Figura 37 - Captura da Tela de Notícias.



Fonte: Elaborado pelo autor

A tela de Notícias apresentada na Figura 37 é composta por:

- **Botão *Home*:** Para direcionar à tela de Notícias, onde estarão disponíveis as notícias e partidas
- **Botão *Ranking*:** Para direcionar a tela onde vai conter o *ranking* dos jogadores, inclusive o preferido.
- **Botão *Acessar Partidas*:** Para direcionar à tela dos “Meus jogos agendados”.
- ***Switch Desafio*:** Para Alternar entre “Notícias” e “Meus Desafios”.

- **Float Button:** Ao clicar nesse botão, gera duas opções (outros botões), que você pode estar sendo direcionado ao perfil ou para agendar uma partida.

A Figura 38 apresenta a tela de Desafio, onde se consegue visualizar todos desafios feitos ao usuário, e pode aceita-los ou rejeita-los.

Figura 38 - Captura da Tela de Desafios.



Fonte: Elaborado pelo autor

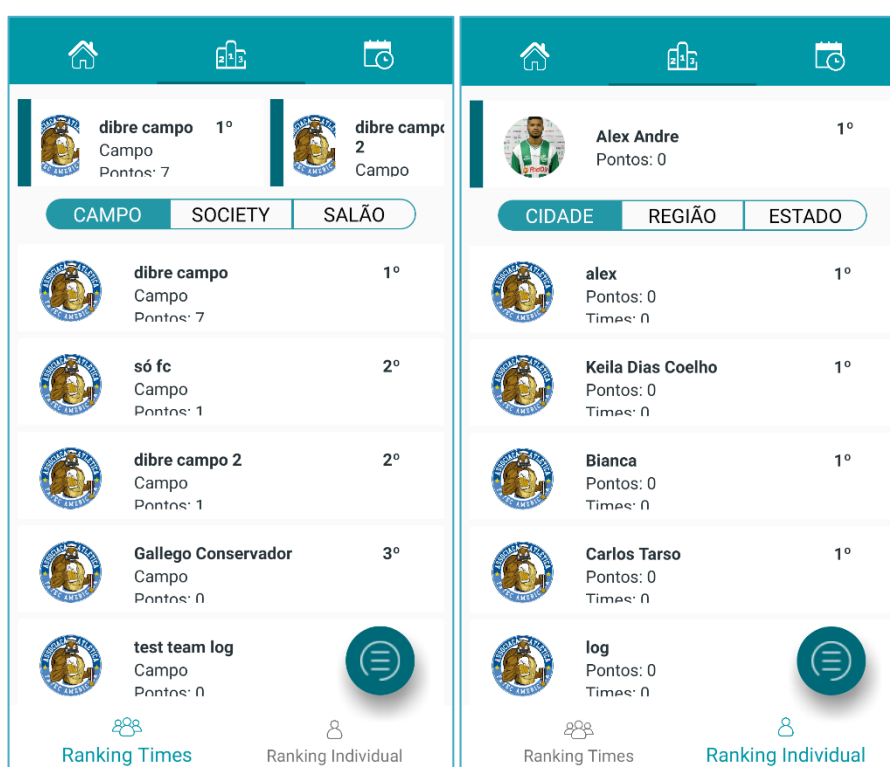
A tela de Desafios apresentada na Figura 38 é composta por:

- **Botão Home:** Para direcionar à tela de Notícias, onde estarão disponíveis as notícias e partidas
- **Botão Ranking:** Para direcionar a tela onde vai conter o *ranking* dos jogadores, inclusive o preferido.
- **Botão Acessar Partidas:** Para direcionar à tela “Meus jogos agendados”.
- **Switch Notícias:** Para Alternar entre Desafios, e Notícias.
- **Botão Rejeitar:** Para rejeitar o desafio proposto ao usuário

- **Botão Aceitar:** Para aceitar o desafio proposto ao usuário
- **Float Button:** Ao clicar nesse botão, gera duas opções (outros botões), direcionando o usuário ao perfil ou para agendar uma partida.

A Figura 39 apresenta a tela de *Ranking*, onde o usuário consegue visualizar, de acordo com a matemática de pontos, onde o usuário ou seu time estão em relação a outro em classificação.

Figura 39 - Captura da Tela de *Ranking*.



Fonte: Elaborado pelo autor

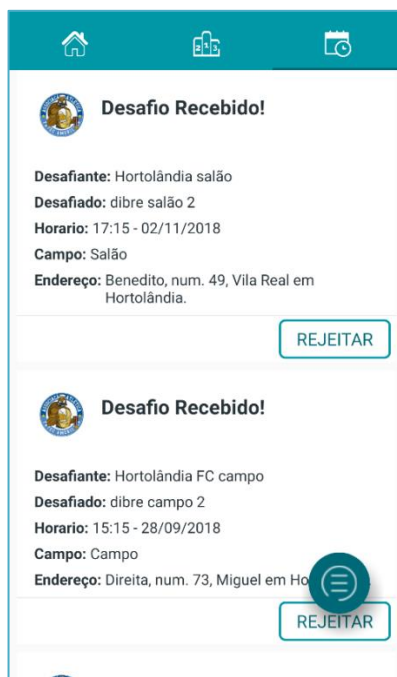
A tela de *Ranking* apresentada na Figura 39 é composta por:

- **Botão Home:** Para direcionar à tela de Notícias, onde estarão disponíveis as notícias e desafios
- **Botão Ranking:** Para direcionar a tela onde vai conter o *ranking* dos jogadores, inclusive o preferido.
- **Botão Acessar Partidas:** Para direcionar à tela dos Meus jogos agendados.

- **Lista de Meus Times:** É uma lista horizontal contendo os times que o usuário tem e/ou faz parte.
- **Lista de Usuários/Times:** É uma lista de vários usuários ou times, que estão em ordem crescente de acordo com quantidade de pontos na liga.
- **Filtro Campo:** Filtro para listar times que jogam no Campo.
- **Filtro Society:** Filtro para listar times que jogam no *Society*.
- **Filtro Salão:** Filtro para listar times que jogam no Salão.
- **Botão *Ranking Individual*:** Clicando nele, visualiza-se somente a sua classificação individual de acordo com as classificações individuais dos seus amigos/adversários.
- **Botão *Ranking Time*:** Clicando nele, o usuário visualiza somente a classificação do time em que está inserido, de acordo com as classificações de times dos seus amigos/adversários.
- **Filtro Cidade:** Filtro para listar jogadores da sua cidade.
- **Filtro Região:** Filtro para listar jogadores da sua região.
- **Filtro Estado:** Filtro para listar jogadores do seu estado.
- ***Float Button*:** Ao clicar nesse botão, gera duas opções (outros botões), direcionando o usuário ao perfil ou para agendar uma partida.

A Figura 40 apresenta a tela de Acessar Partidas, onde o usuário visualiza, não somente suas partidas agendadas/desafiadas, como também informações básicas como: local, data, hora, etc.

Figura 40 - Captura da Tela de Acessar Partidas.



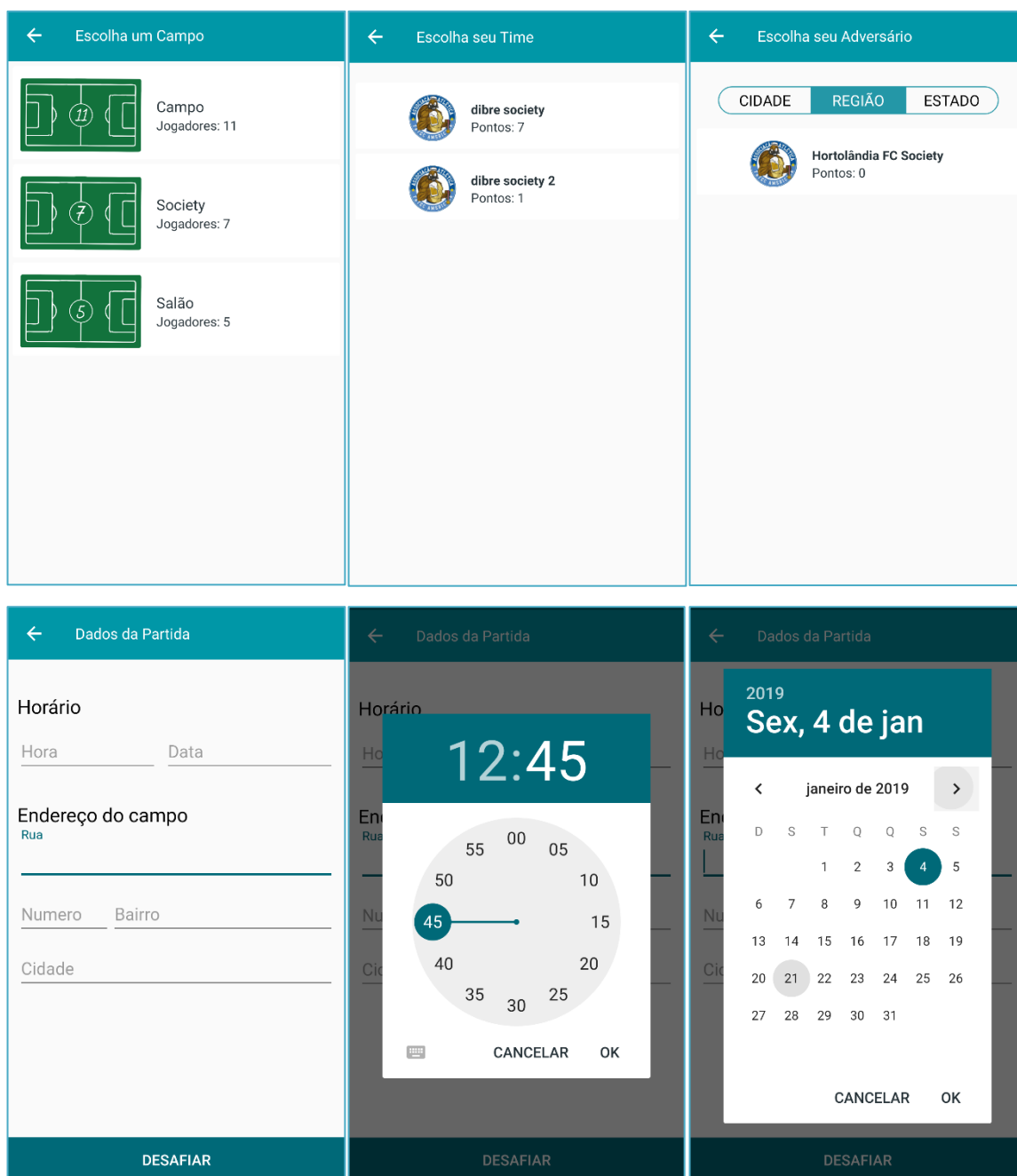
Fonte: Elaborado pelo autor

A tela de Acessar Partidas apresentada na Figura 40 é composta por:

- **Botão *Home*:** Para direcionar à tela de Notícias, onde estará disposto as notícias e desafios
- **Botão *Ranking*:** Para direcionar a tela onde vai conter o *ranking* dos jogadores, inclusive o preferido.
- **Botão *Acessar Partidas*:** Para direcionar à tela dos “Meus jogos agendados”.
- **Listas de Partidas:** Há uma lista de partidas/desafios agendados, que pode visualizar informações da partida.
- **Botão *Rejeitar*:** Para rejeitar o desafio marcado com o usuário.
- ***Float Button*:** Ao clicar nesse botão, gera duas opções (outros botões), direcionando o usuário ao perfil ou para agendar uma partida.

A Figura 41 apresenta a tela de Criar Partida, onde o usuário marca uma partida, escolhendo (Clicando) no tipo de campo, e no adversário que se quer desafiar.

Figura 41 - Captura da Tela Criar Partidas.



Fonte: Elaborado pelo autor

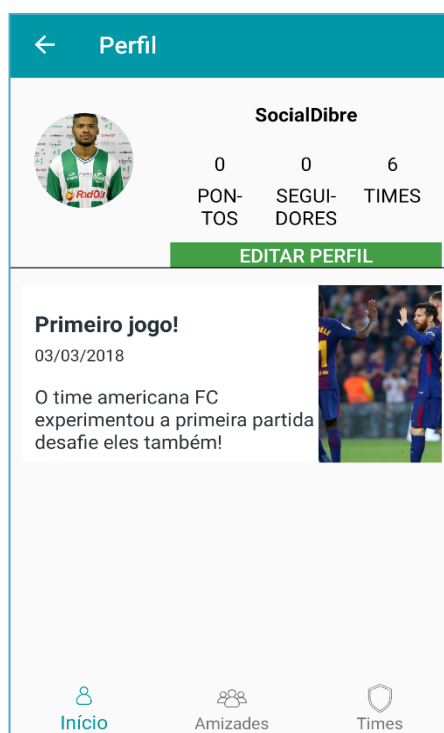
A tela de Criar Partidas apresentada na Figura 41 é composta por

- **Tipo de Campo:** Mostra os tipos de campo possíveis para marcar a partida, e ao clicar seja direcionado para escolha do meu time.
- **Lista de meus times:** Para escolher com qual time o usuário vai querer jogar, e ao clicar, seja direcionado para escolha do adversário.
- **Lista de times adversários:** Para escolher o time que quer desafiar e, ao clicar, seja direcionado para a tela de marcar a hora, e dia da partida.

- **Filtro Cidade:** Filtro para listar times da sua cidade.
- **Filtro Região:** Filtro para listar times da sua região.
- **Filtro Estado:** Filtro para listar times do seu estado.
- **Campo Horário:** Para colocar o horário da partida.
- **Campo Data:** Para escolher a data da partida
- **Campos com Endereço do local:** Que são divididos por rua, número, bairro e cidade, que informa onde ocorrerá essa partida.
- **Botão Desafiar Time:** Ao clicar, salva essa partida, e manda uma notificação a todos os envolvidos do time adversário e do próprio time, que fica anotado na aba agenda de partidas no app de cada um.

A Figura 42 apresenta a tela de Perfil Pessoal. Nessa tela o usuário visualiza coisas relacionadas a ele, desde conquistas e pontos, até de times que ele participa.

Figura 42 - Captura da Tela Perfil Pessoal.



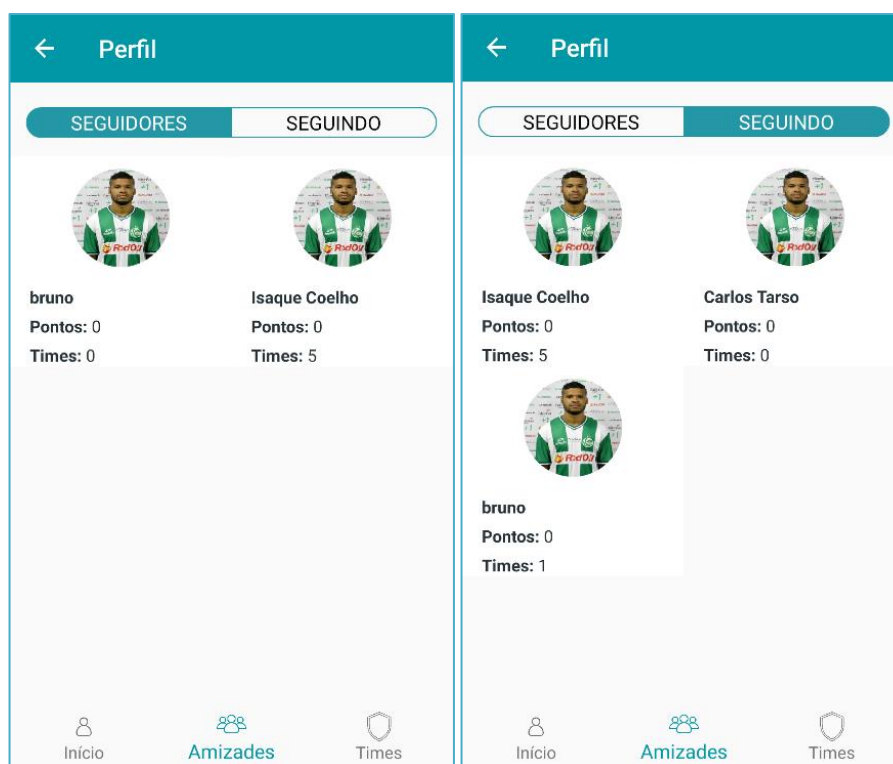
Fonte: Elaborado pelo autor

A tela de Perfil Pessoal apresentada na Figura 42 é composta por:

- **Panel Superior:** Nesse espaço fica a informações relacionadas ao usuário. Como nome, pontos quantos amigos e times. Também é possível acrescentar uma foto sua.
- **Panel Central:** É onde fica todas a notícias referente ao usuário somente, onde contém conquistas ou *posts* clicáveis.
 - **Botão Editar Perfil:** é a área em que o usuário pode alterar suas informações de nome e de foto do perfil
- **Botão Perfil:** É onde fica toda informação pessoal, desde fotos, pontos, até conquista em jogos com respectivos times.
- **Botão Amizades:** Mostra todos os seus seguidores, e as pessoas que o usuário segue.
- **Botão Times:** Lista de times dos quais o usuário está participando.

A Figura 43 apresenta a tela de Amizades Pessoais, é formada por uma lista de amigos, que o usuário pode ter ao seguir e ao ser seguido.

Figura 43 - Captura da Tela Amizades.



Fonte: Elaborado pelo autor.

A tela de Amizades Pessoais apresentada na Figura 43 é composta por:

- **Botão Seguidores:** Formada por todos seus seguidores, que, ao clicar, o usuário consegue ver seus perfis.
- **Botão Seguindo:** Formada por todos o que o usuário segue, que ao clicar você consegue ver seus perfis.
- **Botão Perfil:** É onde fica toda informação pessoal, desde fotos, pontos, até conquista em jogos com respectivos times.
- **Botão Amizades:** Mostra todos os seus seguidores, e as pessoas que o usuário segue.
- **Botão Times:** Lista de times dos quais o usuário está participando

A Figura 44 apresenta a tela de Meus Times, nessa tela o usuário visualiza todos os times em que está inserido, com possibilidade de entrar em mais, ou excluir os seus times.

Figura 44 - Captura da Tela Meus Times.



Fonte: Elaborado pelo autor

A tela de Times apresentada na Figura 44 é composta por:

- **Botão adicionar time:** É o botão que quando clicado, dá ao usuário a possibilidade de criar um time, e ser o capitão automaticamente, ou ser o administrador daquele time.
- **Lista de times:** É uma lista de times nos quais o usuário está ativos, ou seja, está participando/jogando.
- **Botão Perfil:** É onde fica toda informação pessoal, desde fotos, pontos, até conquista em jogos com respectivos times.
- **Botão Amizades:** Mostra todos os seus seguidores, e as pessoas que o usuário segue.
- **Botão Times:** Lista de times dos quais o usuário está participando

A Figura 45 apresenta a tela de Criar Times. Nessa tela o usuário cria seus times, e torna-se o capitão deles

Figura 45 - Tela Criar Time.

← Criar Time

Dados do time
Nome do Time

Tipo de campo

Campo Society Salão

CRIAR TIME

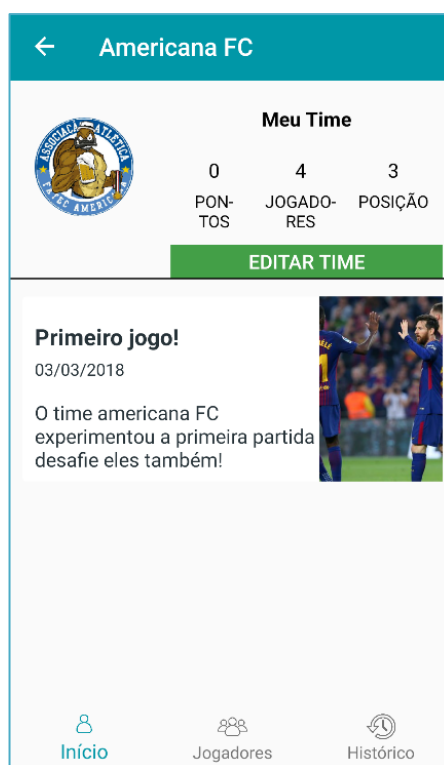
Fonte: Elaborado pelo autor

A tela de Criar Times apresentada na Figura 45 é composta por:

- **Adicionar Foto:** ao clicar, o usuário tem a oportunidade de adicionar uma foto do seu time.
- **Campo Nome do time:** nele o usuário insere o nome que seu time terá.
- **Radio Button tipo de campo:** Seleciona em que modalidade aquele time vai jogar (Campo, *Society* ou Salão).
- **Botão Criar Time:** Salva-se as informações de time.

A Figura 46 apresenta a tela de Perfil de Time. Nessa tela o usuário visualiza todas as coisas relacionadas a seu time, desde conquistas e pontos, até histórico de partidas.

Figura 46 - Tela Perfil do Time.



Fonte: Elaborado pelo autor

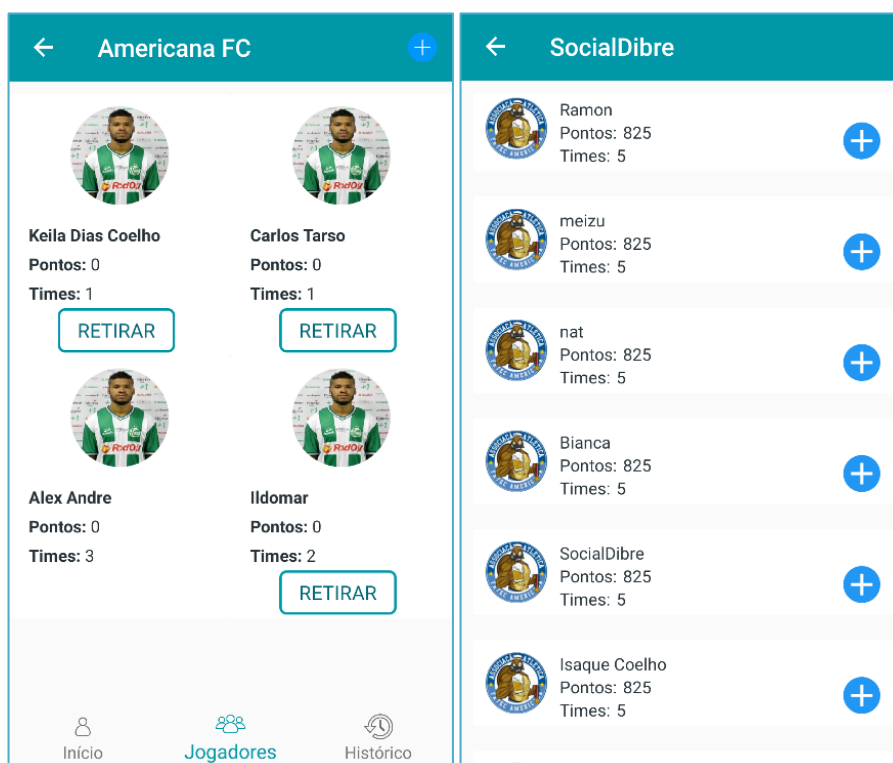
A tela de Perfil de Time apresentada na Figura 46 é composta por:

- **Panel Superior:** Nesse espaço ficam as coisas relacionadas ao time do usuário, como nome, pontos e partidas. Também é possível acrescentar uma foto do time.

- **Panel Central:** É onde ficam todas as notícias referentes a seu time somente, onde contem conquistas ou *posts* clicáveis.
 - **Botão Editar Perfil:** é a área em que você pode alterar suas informações de nome e de foto do time.
- **Botão Perfil:** É onde fica toda informação de time, desde fotos, pontos, até conquista em jogos com os times.
- **Botão Jogadores:** Mostra todos os seus jogadores daquele time.
- **Botão Histórico:** Lista de partidas em que o usuário já jogou, com informações básicas do jogo.

A Figura 47 apresenta a tela de Jogadores do Time. Nessa tela o usuário tem uma lista de jogadores do respectivo time onde se está situado, podendo removê-lo (se no caso for capitão) ou adicionar outro jogador.

Figura 47 - Tela Jogadores do Time.



Fonte: Elaborado pelo autor

A tela de Jogadores do Time apresentada na Figura 47 é composta por:

- **Adicionar Jogador:** Leva a uma página para a Procura um jogador e o adiciona a esse respectivo time cujo estamos navegando.
- **Retirar Jogador:** Consegue-se retirar Jogador desse respectivo time (só se for capitão, ou seja, criador do time).
- **Botão Perfil:** É onde fica toda informação de time, desde fotos, pontos, até conquista em jogos com os times.
- **Botão Jogadores:** Mostra todos os seus jogadores daquele time.
- **Botão Histórico:** Lista de partidas em que o usuário jogou, com informações básicas do jogo.

A Figura 48 apresenta a tela de Histórico de partidas, onde é possível ver partidas em que já jogou, e mais alguns detalhes como lugar, hora e data.

Figura 48 - Tela Histórico de Partidas.



Fonte: Elaborado pelo autor

A tela de Histórico de Partidas apresentada na Figura 48 é composta por:

- **Lista do Histórico de Partidas:** Tem-se uma lista de partidas/desafios já jogados, que pode visualizar informações
- **Botão Perfil:** É onde fica toda informação de time, desde fotos, pontos, até conquista em jogos com os times.
- **Botão Jogadores:** Mostra todos os jogadores daquele time.
- **Botão Histórico:** Lista de partidas em que o usuário já jogou, com informações básicas do jogo.

5 Considerações Finais

Este trabalho teve como objetivo final o desenvolvimento de uma rede social que permita a jogadores amadores e/ou profissionais de futebol uma fácil conexão a este esporte e a outros integrantes da rede, garantindo ao usuário uma experiência intuitiva, contínua e interativa através de tecnologias atuais.

O trabalho inicialmente demonstrou-se desafiador pela complexidade do seu desenvolvimento, garantindo uma parte significativa do tempo em delinear quais seriam os requisitos funcionais e não funcionais que seriam essenciais em um primeiro protótipo, requisitos coletados através de análise das necessidades observadas e também obtidos diretamente por sugestões de praticantes do futebol amador através de exposições da ideia ao público e também pesquisa de campo.

Na sequência, apresentou-se todo o processo de desenvolvimento do aplicativo, nesta etapa tomou-se por base todo projeto, metodologia e escopo definidos anteriormente, mas com importantes melhorias, obtidas através de novas sugestões de tecnologias e funcionalidades que não fugissem demasiadamente do escopo e do cronograma previamente definido.

No desenvolvimento do aplicativo, houve algumas dificuldades dada a falta de experiência e de domínio das tecnologias utilizadas quanto com o porte do projeto proposto, que se demonstrou elevado para o cronograma previamente definido.

A interface do usuário foi construída a fim de manter a experiência do mesmo concisa. Seguindo os padrões da plataforma Android, em especial, procurou-se aderir aos padrões do *Design*, que foi desenvolvido pela Google.

O aplicativo cumpre o que promete, de forma geral, porém pelo limite do cronograma algumas funcionalidades não foram completadas, no entanto, em uma apresentação do protótipo em um evento aberto, o público no geral demonstrou apreciar muito bem a proposta e o aplicativo demonstrado, embora não estivesse totalmente completo.

Como possíveis trabalhos futuros, pode-se apontar a melhoria geral do sistema, a princípio do *design* a fim de que se encaixe melhor nos padrões utilizados no

mercado, também a codificação mais otimizada do aplicativo e do *WebService* de forma a torná-lo mais eficiente e rápido, e por fim melhorias no banco de dados e suas consultas, de forma que seja mais consistente e rápido e garanta ao aplicativo uma boa flexibilidade independente da quantidade de usuários que estiverem ativos.

Referências

COMPOSER. **Introdução ao Gerente de Dependência para PHP**. Disponível em: <<https://getcomposer.org/doc/00-intro.md>>. Acesso em 19 nov 2018.

CORELDRAW. **Visão geral do Produto**. Disponível em: <https://www.coreldraw.com/static/cdgs/product_content/cdgs/2018/cdgs2018-reviewers-guide-br.pdf>. Acesso em 20 nov 2018.

DEV MEDIA. **Introdução ao Visual Studio**. Disponível em: <<https://www.devmedia.com.br/introducao-ao-visual-studio-code/34418>>. Acesso em 19 nov 2018.

DUCROHET, Xavier. **Android Studio**: um IDE criado para Android. Disponível em: <<https://android-developers.googleblog.com/2013/05/android-studio-ide-built-for-android.html>>. Acesso em 19 nov. 2018.

ESPN. **Estudo aponta que 40% dos brasileiros têm interesse em futebol**. Disponível em: <http://www.espn.com.br/noticia/711689_estudo-aponta-que-40-dos-brasileiros-tem-interesse-em-futebol-um-quarto-frequenta-estadios>. Acesso em 11 abr 2018.

FERREIRA, Cleber de F; MOTA, Roberto D. COMPARANDO APLICAÇÃO WEB SERVICE REST E SOAP. **Unipar**, 2014. Disponível em: <[http://web.unipar.br/~seipar/2014/artigos/pos/Cleber_de_F_Ferreira_Roberto_Dias_Mota%20\(1\).pdf](http://web.unipar.br/~seipar/2014/artigos/pos/Cleber_de_F_Ferreira_Roberto_Dias_Mota%20(1).pdf)>. Acesso em 14 de jun. 2018.

GITHUB, Features. Disponível em: <<https://github.com/features>>. Acessado em 20 nov. 2018.

GOERG, Marcelo. **Futebol na Várzea**: Uma investigação sobre os valores presentes no cotidiano da prática. Dissertação (Bacharelado em Educação Física) – Faculdade de Educação Física, Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, 25 p. 2010.

GOMES, Rafael C.; FERNANDES, Jean Alves R.; FERREIRA, Vinicius C. **Sistema Operacional Android**. 2012. 31 f. TG (Trabalho de Graduação do Curso Superior de Engenharia de Telecomunicações) - Faculdade de Engenharia de Telecomunicações, Universidade Federal Fluminense, Rio de Janeiro. 2012.

HOJE EM DIA. **Futebol Amador**: Uma paixão comunitária. Disponível em: <<http://hojeemdia.com.br/opini%C3%A3o/colunas/professor-wendel-1.542133/futebol-amador-uma-paix%C3%A3o-comunit%C3%A1ria-1.554533>>. Acesso em 11 abr. 2018.

HOPSON, K. C.; INGRAM, Stephen E. **Developing professional Java applets**. Indianapolis, 1996. p. Disponível em: <<http://www.webbasedprogramming.com/Developing-Professional-Java-Applets/>> Acesso em 14 jun. 2018.

Introdução à UML. In: BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário**. 2006. 13-38 p. Disponível em <<https://www.passeidireto.com/arquivo/23916916/uml-guia-do-usuario---grady-booch-james-rumbaugh-e-ivar-jacobson> > Acesso em 19 nov. 2018.

IPSOS. **Futebol**: Quem disse que é apenas um esporte?. Disponível em: <<https://www.ipsos.com/pt-br/futebol-quem-disse-que-e-apenas-um-esporte> >. Acesso em 19 nov. 2018.

MICROSOFT AZURE. **Documentação da Pesquisa do Bing News**. Disponível em <<https://docs.microsoft.com/en-us/azure/cognitive-services/bing-news-search/#5-minute-quickstarts> > Acesso em 19 nov. 2018.

MILANI, André. **MySQL**: guia do programador. 1. ed. São Paulo: Novatec, 2006. 397 p. v. 1.

MYSQL. **MySQL Workbench**. Disponível em: <<https://www.mysql.com/products/workbench/>>. Acessado em 20 nov. 2018.

NETO, Moacyr Franco. **Tutorial da ferramenta de modelagem ASTAH**. Santa Catarina, 2017. 13 p.

NIEDERAUER, Juliano. **Desenvolvendo Websites com PHP**. 2. ed. São Paulo: Novatec, 2011. 320 p. v. 1.

OLIVEIRA, Celso Henrique P. **SQL: Curso Prático**. 9. ed. São Paulo: Novatec, 2013. 272 p. v. 1.

ORIGUELA, Milena Avelaneda. **Clube, futebol e lazer: as dinâmicas cultural e educativa de um campeonato de futebol amador na cidade de Piracicaba/SP**. 2015. 152 p. Dissertação (Pós-Graduação em Ciências do Movimento Humano) – Faculdade de Ciências da Saúde, Universidade Metodista de Piracicaba, São Paulo, p.1-59, 2015.

SANTOS, Claudemir José dos. **Futebol se aprende na escola: Novas práticas de sociabilidade esportiva no contexto urbano**. 2007. 131 p. Dissertação (Pós-Graduação em Ciências Sociais) - Ciências Humanas, Universidade Federal de São Carlos, São Carlos, p.1-23, 2007.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do SCRUM: Um guia definitivo para o Scrum: As regras do jogo**. 2013. 19p. Disponível em: <<https://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>>. Acesso em 13 jun. 2018.

SLIM. **Documentação**. Disponível em: <<https://www.slimframework.com/docs/>>. Acesso em 20 junho de 2018.

SOARES, André. Futebol Amador. **Fadebate**, 2012. Disponível em: <<http://fadebate.com.br/futebol-amador/>>. Acesso em 11 abr 2018.

SOMMERVILLE, Ian. **Engenharia de Software**. São Paulo, 2011. 544 p. Disponível em:

<ftp://ftpaluno.umc.br/Aluno/Daisy/Engenharia%20de%20Software/Engenharia_Software_3Edicao%20sommerville.pdf> Acesso em: 11 set. 2018.

SOUZA, Margarida M. da S. **Plataforma Web de Gestão e Partilha de Citações Científicas**. 2013. 86 p. Dissertação (Mestrado em Multimédia) – Faculdade de Multimédia, Universidade do Porto, Porto, 39-40 p. 2013.

SPARX SYSTEM. **Enterprise Architect**. Disponível em: <<https://sparxsystems.com/products/ea/>>. Acessado em 20 nov. 2018

W3C. **Web Services Architecture**. Disponível em: <<http://www.w3.org/TR/wsdl/>> Acesso em: 14 de jun. 2018.