Universidade da Beira Interior
Departamento de Informática

Nº 51 - 2019: *Hojábola*

Elaborado por:

José Nuno Rocha Lamarão

Orientador:

Professor Doutor Pedro Ricardo Morais Inácio

Co-Orientador:

Professor Doutor João Manuel Messias Canavilhas

22 de Junho de 2019

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Excertos de Código

Acrónimos

API *Application Programming Interface*

AWS *Amazon Web Services*

EC2 *Elastic Compute Cloud*

FIFA *Fédération Internationale de Football Association*

HTML *HyperText Markup Language*

HTTP *Hypertext Transfer Protocol*

HTTPS *Hyper Text Transfer Protocol Secure*

IDE *Integrated Development Environment*

JSON *JavaScript Object Notation*

OSI *Open System Interconnection*

PHP *Hypertext Preprocessor*

PBKDF2 *Password-Based Key Derivation Function 2*

SCP *Secure Copy Protocol*

SDK *Android Software Development Kit*

SO Sistema Operativo

SQL *Structured Query Language*

SSH *Secure Shell*

TLS *Transport Layer Security*

VM *Virtual Machine*

XAMPP *X (Cross-Platform), Apache, MySQL, PHP and Perl*

Capítulo 1

Introdução

Serve o presente documento como relatório referente à Unidade Curricular "Projeto", incluida na Licenciatura em Engenharia Informática da Universidade da Beira Interior. É nele que está inserido uma explicação teórico-prática de todo o trabalho desenvolvido ao longo do semestre. Desta feita, o capítulo que se segue aborda o enquadramento do tema a desenvolver, assim como a motivação para a escolha do mesmo, os objetivos, e por fim delimita a organização do documento.

1.1 Enquadramento

O futebol é o desporto rei e isso prova-se através do número de pessoas que o praticam ou até pelo número de espetadores e telespetadores que assistem à modalidade. Segundo a última contagem [1] realizada pela *Fédération Internationale de Football Association* (FIFA), obtida no ano de 2006, haveria cerca de 265 milhões de futebolistas registados mundialmente. De salientar que esta já foi realizada há algum tempo e seguramente os números serão mais elevados atualmente. Apesar de já serem números significativos, não se comparam aos números dos espetadores e telespetadores. Na final do mundial de clubes de 2018, uma songagem [2] indicava que 3,5 mil milhões assistiram à partida. Portugal não foge à regra e, como tal, na Primeira Liga portuguesa houve um total de cerca de 3 milhões e 630 mil espetadores na temporada 2017/2018, o que dá uma média de 12 mil espetadores por jogo[3]. Com isto, surge um público alvo para o desenvolvimento de um sistema capaz de facilitar o acesso a informações essenciais aquando uma ida a um jogo de futebol. Assim, o projeto incluí o desenvolvimento de uma aplicação móvel para apresentação de informação acerca de jogos de futebol (e de muitos detalhes que os envolvem) e de uma interface *web* que medeia um *backoffice*, que serve de gestor de alguns dados.

Posto isto, este trabalho é realizado no âmbito da unidade curricular `Projeto`,

incluída no segundo semestre do terceiro ano de Engenharia Informática e lida com temáticas de programação, sistemas operativos, segurança informática, redes e base de dados. Nele encontra-se inserido um pouco engenharia de software, programação de dispositivos móveis, programação web, assim como gestão de base de dados, mecanismos de segurança e comunicação em rede.

1.2 Motivação

Nos dias de hoje, a tecnologia e a humanidade andam de mãos dadas em perfeita simbiose e como tal através do surgimento e inovação desta coligação, há de facto um impacto significativo nas vidas do ser humano. Um estudo [4] no qual participaram 53 mil pessoas, espalhadas por 31 países, refere que numa idade compreendia entre os 18 e os 75 anos, em média 80% são donos de um *smartphone*. É de facto, um número elevado e atesta que à priori um indivíduo escolhido ao acaso tem acesso a aplicações móveis. Assim sendo, torna-se completamente plausível explorar esta vertente através da elaboração de um projeto aliciante, de forma a consolidar e a obter novos conhecimentos a nível de desenvolvimento de software. Para além disso, há também a possibilidade de causar impacto positivo no dia-a-dia, sabendo que a criação deste sistema serve de *All-in-one* visando um utilizador assíduo ou esporádico das infraestruturas de um clube de futebol, para que possa saber como chegar, onde estacionar, onde comer, entre outras. Já em termos pessoais, dado ser necessário a realização deste projeto para a aprovação à unidade curricular e consequentemente a conclusão da licenciatura, torna-se um fator importante o término bem sucedido deste trabalho.

1.3 Objetivos e Planificação do Projeto

O objetivo principal passa por desenvolver uma aplicação móvel e o seu respetivo *Backend*. Este sistema tem como propósito colecionar, gerir e mostrar informações sobre jogos e estádios das equipas da Primeira Liga Futebol Portuguesa, bem como ajudar os adeptos a escolher as melhores formas de chegar, de estacionar, de encontrar locais para comer, beber, entre outros. Como tal será necessário estudar as soluções existentes, de forma a enquadrar ideias que possam ser reaproveitadas ao longo da elaboração deste trabalho e tem que se reunir e planificar a estrutura do sistema. Para atingir os objetivos pretendidos foi sugerida a seguinte planificação e cronologia:

T1 Contextualização com os objetivos propostos e preparação do ambiente de trabalho;

T2 Análise detalhada dos requisitos; desenho da base de dados e do sistema;

T3 Implementação da parte do servidor, incluindo a definição do esquema da base de dados com dados de teste para esta temporada e a implementação da interface web;

T4 Implementação da aplicação móvel e integração com o servidor;

T5 Testes e aprimoramento;

T6 Escrita do relatório.

A figura 1.1 tem uma ilustração do cronograma, para melhor entendimento da planificação proposta e que foi seguida de forma bastante aproximada durante o semestre.

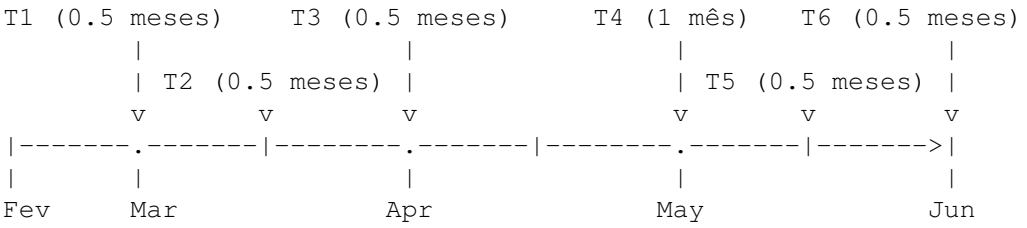

Figura 1.1: Cronograma de tarefas a realizar.

1.4 Organização do Documento

Primeiramente, para a elaboração do relatório foi utilizado o sistema de preparação de documentos LaTeX, através do editor *online OverLeaf*. De modo a refletir o trabalho realizado, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, dando o enquadramento para o mesmo, a motivação para a sua escolha, os seus objetivos e a respetiva organização do documento;

2. O segundo capítulo – **Tecnologias Utilizadas e Trabalhos Relacionados** – descreve algumas soluções semelhantes no âmbito deste projeto, bem como as tecnologias utilizadas durante do desenvolvimento do sistema;

3. O terceiro capítulo – **Engenharia de Software** – enumera os requisitos funcionais e não funcionais, bem como os casos de uso do sistema. Para além disso são apresentados diagramas de atividades, o modelo físico da base

de dados que suporta o sistema e por fim é demonstrada a arquitetura do mesmo;

4. O quarto capítulo – **Implementação** – expõe os passos mais importantes tomados ao longo do trabalho referentes à implementação do sistema tanto a nível funcional como cosmético;

5. O quinto capítulo – **Testes e Modelo de Negócios** – detalha todos os testes realizados durante e no final do desenvolvimento, bem como um modelo de negócios no que toca à expansão e publicação da aplicação desenvolvida no âmbito deste trabalho;

6. O sexto capítulo – **Conclusões e Trabalho Futuro** – apresenta as conclusões alcançadas na realização do projeto e para terminar indica possíveis alterações ou melhoramentos para o futuro do sistema.

Capítulo 2

Tecnologias Utilizadas e Trabalhos Relacionados

2.1 Introdução

Neste capítulo é feita uma explicação sucinta das tecnologias utilizadas, bem como um estudo de trabalhos relacionados, de forma a executar um levantamento das soluções existentes, que serviram de inspiração para resolver problemas relativos à execução do projeto. Assim sendo, este capítulo encontra-se dividido da seguinte forma:

- a secção 2.2 – Tecnologias Utilizadas – expõe as tecnologias empregues no projeto, tal como a razão do uso das mesmas;
- a secção 2.3 – Soluções Existentes – descreve as soluções consoante os trabalhos relacionados, identificando as suas vantagens e desvantagens.

2.2 Tecnologias Utilizadas

De seguida, vão ser enumeradas as diversas tecnologias, que ajudaram no desenvolvimento do projeto.

2.2.1 Aplicação móvel

Android

Baseado no *kernell* do Linux, o *Android* é um Sistema Operativo (SO) para dispositivos móveis. Uma vez que é disponibilizado pela *Google* gratuitamente, através

de código aberto, e está associado a uma relativa simplicidade a nível do desenvolvimento de aplicações, faz com que seja o SO mais usado. Neste momento, conta com cerca de 80% do mercado global, o que justifica a escolha desta tecnologia.

Android Studio

O *Android Studio* é um ambiente de desenvolvimento integrado (*Integrated Development Environment* (IDE)), e neste caso o oficial para aplicações *Android*. Junto com o *Android Studio* vem tipicamente o *Android Software Development Kit* (SDK), o provedor de várias ferramentas que auxiliaram a execução do projeto, nomeadamente através do emulador onde foram efetuados vários testes. Foi neste programa que se deu a criação, assim como a compilação de todo o código referente à aplicação móvel.

SQLite

O *SQLite* É um sistema de gestão de base de dados, pequeno, rápido, de alta confiança e com todas as funcionalidades disponíveis de *Structured Query Language* (SQL). É neste momento o mais utilizado no mundo, já que vem embutido em todos os telemóveis, assim como em grande parte de computadores. Na aplicação móvel é utilizado para guardar informações vindas do servidor, servindo de *backup* caso não haja comunicação entre o software local e o remoto.

Retrofit2

Retrofit2 é uma versão melhorada da biblioteca *Retreofit*, e tem como função simplificar o acesso a um *webserviceRESTful*, ao traduzir a *Application Programming Interface* (API) em interfaces JAVA, nomeadamente objetos. Para isso aproveita-se de bibliotecas capazes de converter *JavaScript Object Notation* (JSON), que no caso deste projeto foi a *Gson*. Utiliza também a ferramenta *OkHttp* para lidar com pedidos *Hypertext Transfer Protocol* (HTTP).

2.2.2 Servidor Web

AWS

A *Amazon Web Services* (AWS) é um serviço de computação na nuvem, disponibilizado pela *Amazon*, onde apenas a parte do *Elastic Compute Cloud* (EC2) foi utilizada. Através dele foi criada uma instância, ou por outras palavras uma máquina virtual, que representa o servidor *web* ligado tanto à aplicação móvel, como à interface *web* designado para o efeito. Estes últimos serão abordados posteriormente. A instância de máquina virtual na EC2 referida antes é a que integra

a parte de servidor do sistema de software desenvolvido, com o qual a aplicação móvel comunica e para a qual também existe uma interface web de administração dedicada.

HTTP

O HTTP é um protocolo de comunicação, que se dá na camada de aplicação, segundo o Modelo *Open System Interconnection* (OSI). É através dele que a aplicação móvel se comunica com o servidor *web*, contudo, em junção com o protocolo de segurança, *Transport Layer Security* (TLS), explicado de seguida.

TLS

Como referido em cima, o TLS é um protocolo de segurança que visa manter a confidencialidade e a integridade dos dados transmitidos entre serviços que se comunicam. A junção deste com o HTTP cria o *Hyper Text Transfer Protocol Secure* (HTTPS), a base da comunicação entre o servidor e a aplicação móvel deste projeto. Desta forma, todas as comunicações seguem este protocolo para prevenir ataques do tipo *man in the middle* e *eavesdropping*.

MySQL

O *MySQL* é um sistema de gestão de base de dados, que utiliza SQL, uma linguagem usada para guardar, manipular e receber dados. No projeto procedeu-se ao desenho e respetiva criação da base de dados através desta ferramenta com o intuito de satisfazer as necessidades do sistema.

PHP

O *Hypertext Preprocessor* (PHP) é uma linguagem de programação, desenhada especialmente para o desenvolvimento *web*, através de *scripts* que são alocados no lado do servidor. Contudo, é graças à sua compatibilidade para correr em maior parte dos sistemas operativos e o suporte para bases de dados (e.g., *MySQL*), que faz com que seja uma tecnologia extremamente útil. No trabalho, usou-se para criar os *scripts* referentes a API da aplicação móvel. De salientar também o *phpMyAdmin*, um serviço *web* escrito em PHP, usado para gerar a base de dados do sistema, assim como para realizar alguma depuração e testes.

Apache Web Server

O *Apache Web Server*, ou coloquialmente conhecido como *Apache*, é um *software* gratuito e de código aberto desenvolvido com o objetivo de correr servidores

web. Conta com vários módulos implementados e disponibilizados gratuitamente, como por exemplo uma interligação para um interpretador de PHP, assim como o suporte para a utilização do protocolo TLS. Posto isto, é através do *Apache* que corre o servidor web referente ao sistema.

XAMPP

X (Cross-Platform), Apache, MySQL, PHP and Perl (XAMPP) é um pacote de instalação com um conjunto de *softwares*, nomeadamente os três últimos abordados anteriormente. O *X* significa *CROSS-platform*, isto é, corre em vários SOs, o *A* faz referência ao *Apache*, o *M* corresponde ao *MySQL*, o *P* refere-se ao PHP e por fim o último *P* condiz com *Perl*, uma linguagem de programação. Importa salientar que, do conjunto de tecnologias referidas neste parágrafo, apenas a última não foi utilizada, porém graças à facilidade da instalação, tal como a configuração do servidor, foi empregue esta tecnologia no projeto.

2.2.3 Sistema

API

API, traduzido para português, interface de programação de aplicações, é um conjunto de rotinas, de protocolos de comunicação e de ferramentas que visam facilitar a criação de *software*. Resumidamente, executa a comunicação entre serviços com funções pré-programadas, possibilitando a abstração aquando da concretização do programa. Neste projeto, criou-se uma API própria associada à aplicação móvel, e consumiu-se outra, para inserir informações requisitadas na base de dados. Mais tarde, será explicado com o devido detalhe.

JSON

JSON é um formato de troca de dados simples que assenta em duas estruturas universais, pares de nomes/valores (e.g objetos) e lista de valores(e.g *arrays*). É através dele que se dá a obtenção da informação proveniente da API do sistema.

2.3 Soluções Existentes

Antes de ser iniciado o desenvolvimento do sistema foi feito um levantamento das aplicações já existentes no mercado e que tivessem parecenças com as deste, de forma a procurar soluções e funcionalidades que servissem de inspiração. Aqui constatou-se que apesar de haver várias aplicações ligadas ao futebol, que fornecem notícias e resultados ao vivo, bem como a classificação da Primeira Liga

Portuguesa, nenhuma oferecia informações acerca de estacionamentos e estabelecimentos de restauração nas mediações dos estádios onde ocorrem os jogos. Assim sendo, o que se segue é uma análise detalhada das funcionalidades encontradas. Em primeiro lugar, na *Play Store* da *Google* na categoria do *Desporto*, encontra-se a aplicação `FlashScore` [5], que conta com resultados e classificações ao vivo, assim como detalhes dos jogos. De seguida, há o `SofaScore` [6] que apresenta as mesmas funcionalidades que a aplicação anterior, no entanto com uma vertente mais estatística do jogo.

Com o mesmo intuito surge a aplicação `LIGA PORTUGAL` [7], que acrescenta vídeos relativos à Liga Portuguesa e à Taça de Portugal. É importante referir que todas as aplicações até aqui abordadas, não apresentam notícias, ao contrário das aplicações `Maisfutebol` [8] e `Onefootbal` [9]. De notar que estas mantêm as funcionalidades das anteriores. Já a `Forza Football` [10], em termos de mais funcionalidades, permite que se comente acerca de um jogo, para além de avaliações de desempenho dos jogadores. Posto isto, na tabela 2.1 que se segue, são demonstradas as funcionalidades com auxílio visual, onde o vermelho indica a ausência da funcionalidade no sistema desenvolvido neste projeto, a amarelo a possível integração dessa funcionalidade a curto prazo e por fim a verde a integração dessa funcionalidade. Aqui é também feita uma comparação, com o sistema desenvolvido no projeto.

Funcionalidades \Aplicações	FlashScore	SofaScore	LIGA PORTUGAL	Maisfutebol	Onefootball	Forza Football	Hojabola
Resultados							
Classificações							
Onzes iniciais							
Informações em direto							
Calendarização dos jogos							
Comentários sobre os jogos							
Notícias							
Vídeos de golos e resumo do jogo							
Informações sobre estádios							
Informações sobre estacionamentos							
Informações sobre estabelecimentos de restauração							

Tabela 2.1: Levantamento das funcionalidades presentes e não presentes nas soluções apresentadas.

De forma a completar as informações, segue-se a figura 2.1 que ilustra 3 aplicações escolhidas ao acaso da lista que se estudou, sendo elas: a `FlashScore`, a `LIGA PORTUGAL` e a `MaisFutebol`.

2.4 Conclusões

Neste capítulo foram descritas as ferramentas e as tecnologias utilizadas bem como o motivo para o uso das mesmas. Dado serem um vasto leque de opções, permitiu o desenvolvimento de novas competências.

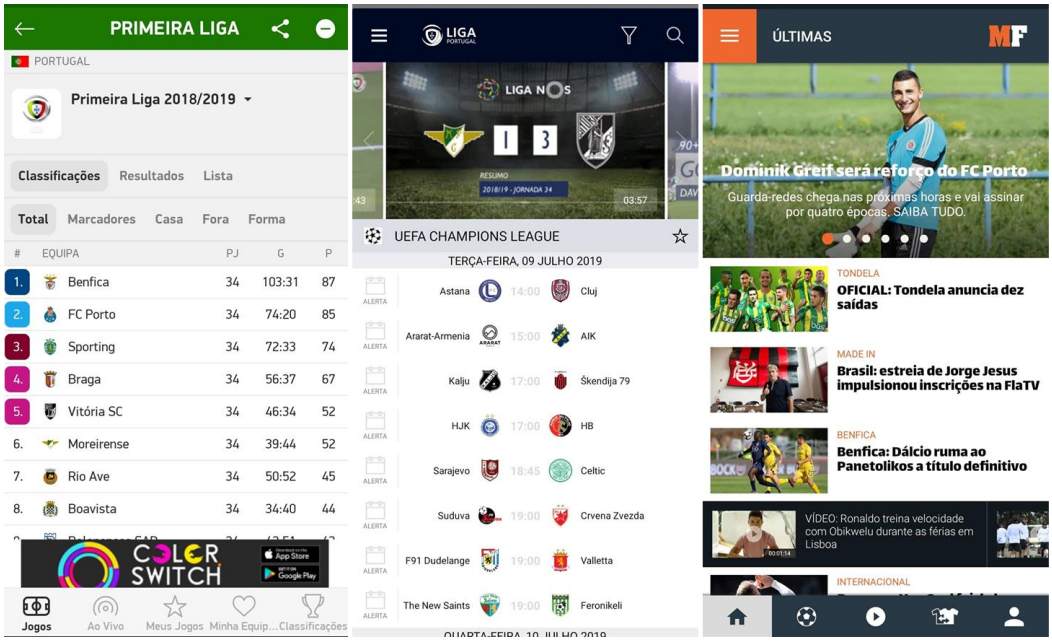

Figura 2.1: Capturas de ecrã referentes às aplicações: `FlashScore`, `LIGA PORTUGAL` e `MaisFutebol` por ordem de menção.

A nível de soluções existentes verificam-se algumas parecenças em termos das funcionalidades disponíveis em cada uma. Demonstra-se, no entanto, que o sistema desenvolvido se destaca por ser o único do mercado capaz de fornecer informações acerca dos estádios, dos estacionamentos nas mediações dos mesmo, bem como estabelecimentos de restauração.

Capítulo 3

Engenharia de Software

3.1 Introdução

Neste capítulo é feito o levantamento e respetiva análise dos requisitos funcionais e não funcionais, assim como dos casos de uso da aplicação, e alguma modelação do sistema a desenvolver, nomeadamente recorrendo a diagramas de atividade, classe e modelos de dados. Desta forma, o seguinte capítulo está estruturado em:

- a secção 3.2 – Análise dos Requisitos – identifica os requisitos funcionais e não funcionais da aplicação móvel, definindo o funcionamento expectável da mesma;
- a secção 3.3 – Casos de Uso – apresenta os casos de uso da aplicação móvel, assim como do sistema, onde são descritas possíveis interações;
- a secção 3.4 – Diagramas de Atividade – apresenta o fluxo de algumas atividades a partir de diagramas de atividade, especificando, por exemplo, o fluxo da utilização da aplicação móvel;
- a secção 3.5 – Modelo de Dados – demonstra como foi desenhada a base de dados e explica o relacionamento das diferentes entidades entre si;
- a secção 3.6 – Arquitetura do Sistema – representa a forma de como o sistema está estruturado e em que ambiente agem os diversos componentes constituintes.

3.2 Análise dos Requisitos

Como referido na introdução nesta secção são identificados os requisitos funcionais e não funcionais da aplicação, sendo os primeiros uma representação do que

é expetável em termos de funcionamento da aplicação móvel, e os seguintes uma definição dos requisitos e de alguns comportamentos pelos quais a aplicação se vai reger.

3.2.1 Requisitos Funcionais

Os requisitos funcionais da aplicação são:

RF1 Permitir o registo de um utilizador;

RF2 Permitir a escolha do clube afeto ao utilizador aquando do registo;

RF3 Permitir o *login* de um utilizador registado, quer seja pela rede social *Facebook* ou por um componente designado para o efeito;

RF4 Permitir a apresentação de todas as informações relativas ao estádio não só do clube afeto ao utilizador, mas também dos restantes clubes;

RF5 Permitir a consulta de todos os jogos referentes ao campeonato em que o clube afeto ao utilizador se encontra inserido, mostrando os seus resultados;

RF6 Apresentar as informações relativas ao próximo jogo do clube afeto ao utilizador, assim como os seguintes;

RF7 Permitir selecionar um jogo, e apresentar várias formas de interação ou obtenção de informação acerca do mesmo;

RF8 Permitir identificar e especificar o caminho para o estádio onde ocorre o jogo que o utilizador escolheu, com ajuda da aplicação *Google Maps*;

RF9 Permitir apresentar os estacionamentos perto do estádio do jogo selecionado;

RF10 Apresentar informação relativa a restaurantes e bares num raio de 15 a 20 Km junto do estádio, onde o jogo selecionado pelo o utilizador está a decorrer;

RF11 Permitir criticar os respetivos restaurantes e bares;

RF12 Permitir ao utilizador com nível de confiança 10 criar uma *meeting* referente ao clube afeto e ao jogo selecionado;

RF13 Permitir a um utilizador com nível de confiança 7 aderir aderir a uma *meeting* referente ao clube afeto e ao jogo selecionado;

RF14 Permitir convidar utilizadores a aderir ao *meeting*;

RF15 A aplicação deverá permitir a troca de mensagens entre utilizadores, redirecionamento para Whatsup ou Facebook messenger se essa informação for colocada no registo;

RF16 Um utilizador de nível 10 pode atribuir nível de confiança 6 ou 7 a outros utilizadores;

Os requisitos funcionais para o *backoffice* são:

RF17 O acesso ao *backoffice* é feito apenas pelo administrador (nome de utilizador `admin`);

RF18 Permitir adicionar informações e/ou fotos referentes ao estacionamento selecionado, assim como a criação de um novo bloco de informação acerca de um estacionamento não antes registado;

RF19 Permitir analisar e filtrar comentários referentes aos estacionamentos;

RF20 Permitir adicionar informações e/ou fotos referentes a restaurantes e bares;

RF21 Permitir analisar e filtrar comentários referentes aos restaurantes e bares;

RF22 O administrador pode alterar a sua palavra-passe de acesso ao *backoffice*;

RF23 O administrador pode atribuir ou retirar nível 10 de confiança aos utilizadores;

RF24 O administrador pode alterar os níveis de confiança dos utilizadores.

3.2.2 Requisitos Não Funcionais

Os requisitos não funcionais referentes à aplicação são:

RNF1 Deverá ter acesso à Internet para se poder utilizar como um todo;

RFN2 A API mínima é a 24.0 (Android 7.0);

RFN3 A API destinada é a 28.0 (Android 9.0);

RFN4 Deverá estar disponível para download na *Play Store*;

RFN5 Deverá usar o mínimo de bateria possível;

RFN6 Deverá pedir permissão para aceder à localização do usuário;

RFN7 Deverá pedir permissão para utilizar a câmara do dispositivo para que possa tirar fotos de um estacionamento;

RFN8 A aplicação deverá ter interfaces simples e minimalistas;

RFN9 A aplicação deverá ter um código portável para que seja facilmente adaptado para outras partes do globo.

Os requesitos não funcionais do *backoffice* são:

RFN10 A comunicação entre o sistema, isto é, entre a aplicação e o servidor *WEB* deve ser sempre segura (HTTPS);

RFN11 O sistema deve estar sempre disponível;

RFN12 O design da interface *web* referente ao *backoffice* deverá ser o mais simples possível, e.g., adotando o w3css.

3.3 Casos de Uso

Nesta secção, como indicado na introdução a este capítulo, o intuito será demonstrar as possíveis interações que podem ocorrer no sistema.

3.3.1 Descrição dos Atores

Antes de se poder analisar os diferentes casos de uso, é necessário indicar e especificar os três atores do sistema. O primeiro é *um utilizador com o nível de confiança inferior a 10*, o que significa que apenas terá acesso a certas funcionalidades de menos importância, enquanto o *segundo utilizador, de confiança de nível 10* terá acesso a todas as utilidades da aplicação. Por fim o terceiro, que será *o administrador*, aquele que serve de curador das informações fornecidas pelos utilizadores.

3.3.2 Descrição dos Casos de Uso

Primeiramente, será analisado o diagrama dos casos de uso de um utilizador com um nível de confiança igual a 10; de seguida o de nível inferior a 10. Por fim, será abordado o utilizador *administrador*, este já num outro plano, o de *backoffice*. Assim sendo, será agora estudado o primeiro caso de uso.

Como se observa na figura 3.1, *o utilizador com confiança de nível 10* pode visualizar a informação relativa ao estádio afeto, assim como ver um jogo, ou até o resultado do mesmo, todavia isso só é possível após o processo de autenticação

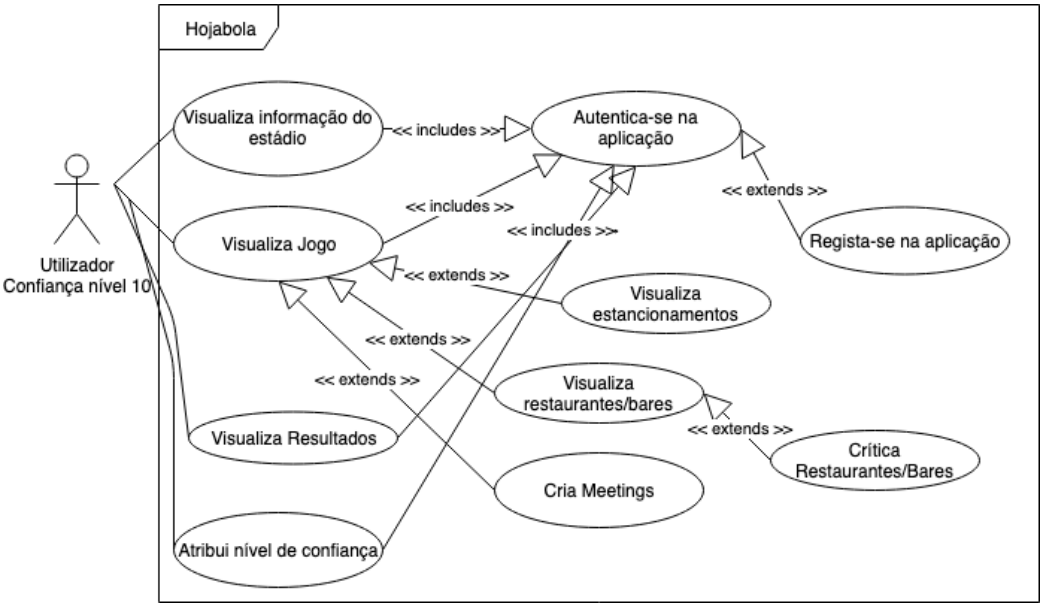

Figura 3.1: Caso de Uso para um *utilizador de confiança de nível 10*.

bem-sucedido, que requer uma registo prévio. Para além disso, através do caso onde o agente visualiza um jogo, é possível obter informações sobre os estacionamentos disponíveis, bem como os restaurantes e bares, tendo a hipótese de deixar uma crítica. Por fim, e dado ser o fator de distinção entre os dois tipos de utilizadores em termos de nível de confiança, *o utilizador de nível mais elevado* pode atribuir níveis a outros utilizadores, bem como criar um *meeting*, enquanto o de níveis menores que 10 apenas pode aderir a esta reunião, como se pode visualizar na figura 3.2.

Para terminar esta secção analisa-se o último caso de uso, onde o utilizador é o *Administrador*, e o plano de ação é o *backoffice*. O diagrama realça que o *Administrador* pode gerir, bem como atribuir o nível de confiança apropriado ao utilizador. Para além disso, ele serve de curador das críticas efetuadas à restauração e atesta os estacionamentos criados. Tudo isto após a autenticação na interface *web*, com uma palavra passe pré-definida.

3.4 Diagramas de Atividades

Na secção que se segue vão ser representados três diagramas de atividade, sendo dois deles referentes a um possível fluxo de utilização a partir da aplicação móvel, e o último já no plano de ação da interface *web*.

Como se pode visualizar na figura 3.4, um utilizador inicia aplicação, e logo

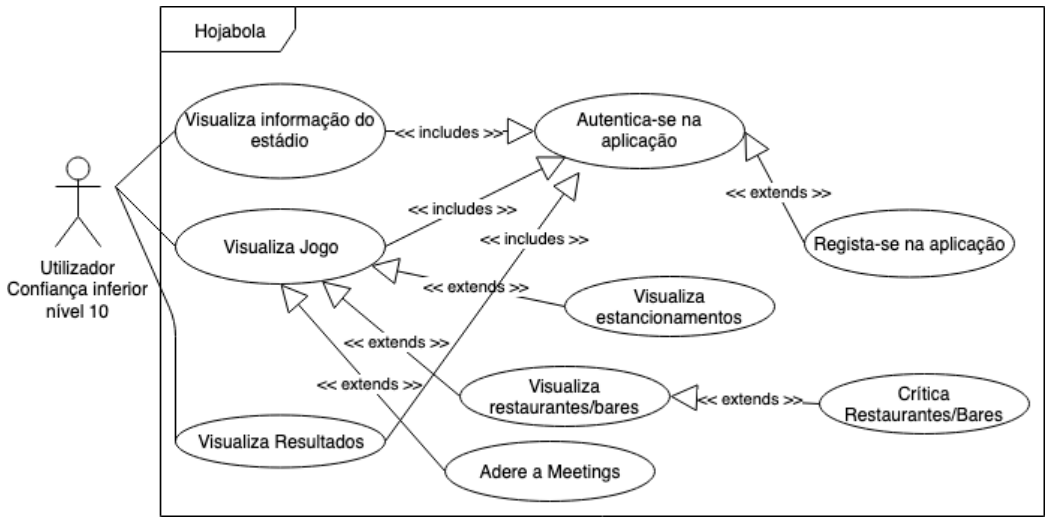

Figura 3.2: Caso de Uso para *um utilizador de confiança inferior ao nível 10*.

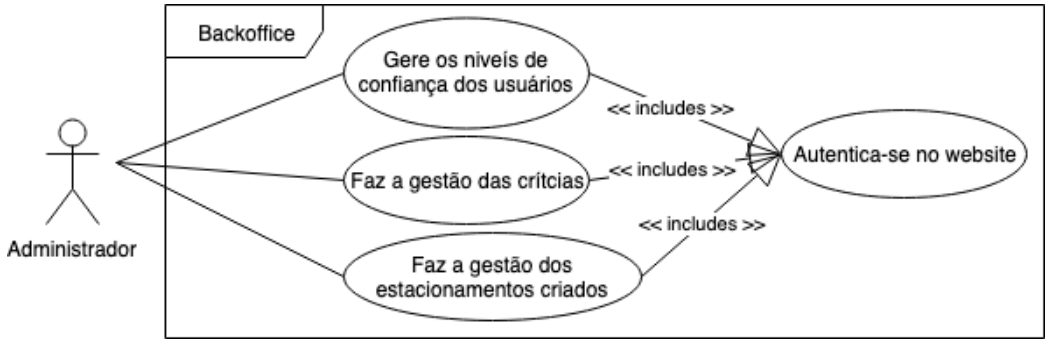

Figura 3.3: Caso de Uso para o *Administrador*.

de seguida aparece um *splash screen*, que verifica se este já se encontra autenticado. Em caso positivo, remete-o para o menu principal; caso contrário, mostra a atividade Inicial. É nela que o utilizador pode escolher fazer o *login*, inserindo os dados necessários, os quais são posteriormente verificados, e à semelhança do acontecimento anterior, ou entra no menu principal, ou é mostrada uma mensagem de erro, mantendo-se na atividade inicial. Se for um utilizador novo (que nunca se tenha registado), pode-o fazer, preenchendo os dados necessários, tendo estes que passar também por um processo de verificação. Aqui, mais uma vez, em caso de erro, apresenta uma mensagem ou volta para a atividade inicial, onde o utilizador devidamente registado, já pode executar o *login*.

A figura 3.5, alusiva a um possível fluxo de utilização da aplicação móvel, com o intuito de um utilizador realizar uma crítica a um restaurante ou bar, é agora analisada. Um utilizador autenticado inicia a aplicação e é enviado um

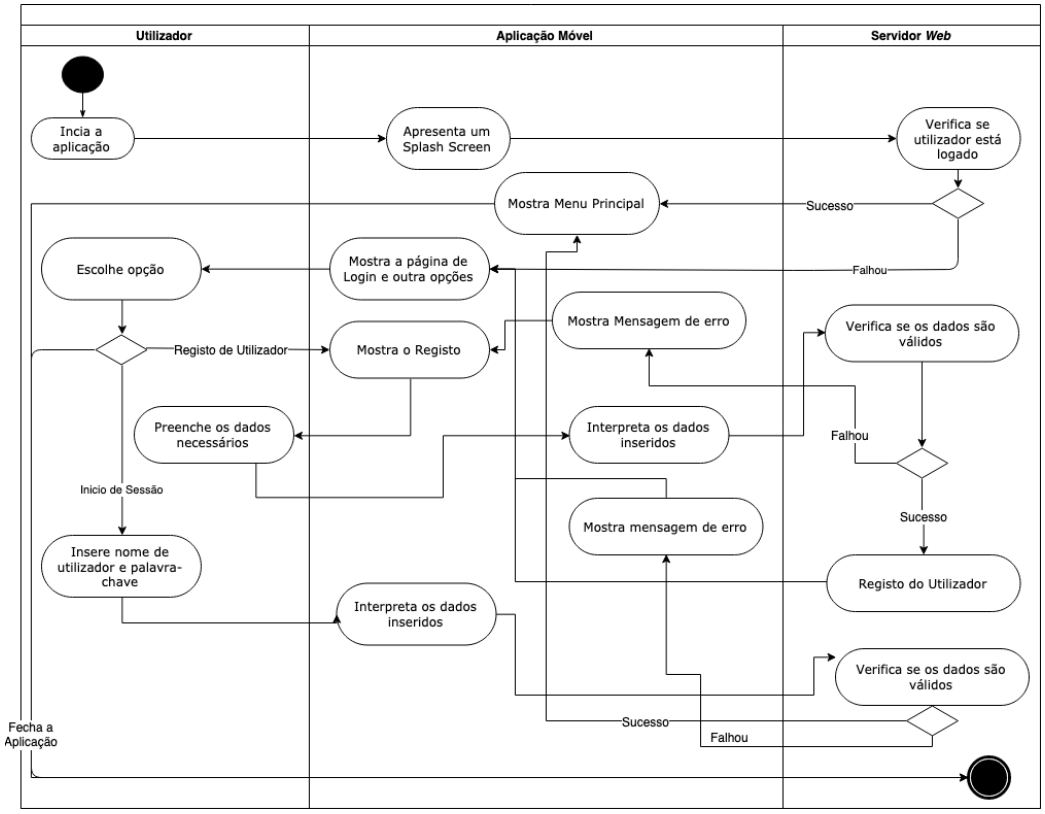

Figura 3.4: Diagrama de Atividade relativo à autenticação de um utilizador.

pedido ao servidor para retornar os próximos jogos do clube afeto. De seguida, eles são apresentados no ecrã, dando a hipótese para a escolha de um. Após a decisão do jogo pretendido, é feita uma busca no sistema, dado o raio perto do estádio onde irá decorrer a partida, para que sejam exibidos os restaurantes e bares resultantes da pesquisa. Mais uma vez, é dado o poder de eleição, ao utilizador, selecionando o estabelecimento que mais lhe agrada, podendo optar ou por visualizar mais informações acerca do mesmo, ou efetuar uma crítica. No caso da última opção, é remetido para uma nova atividade para escrever a sua apreciação. No ato da submissão, a crítica é guardada no servidor, para que seja verificada por um administrador e posteriormente publicada.

Para terminar a secção vai examinar-se o último digrama de atividades, incluído na figura 3.4, onde o plano de ação das atividades se dá na interface *web* referente ao sistema desenhado, ao contrário dos outros diagramas analisados. Inicialmente, o administrador insere o endereço no seu *web browser* e assim que é enviado o *request* HTTPS é retornado o ficheiro PHP com código *HyperText Markup Language* (HTML) que se encontra na raiz do servidor, posteriormente

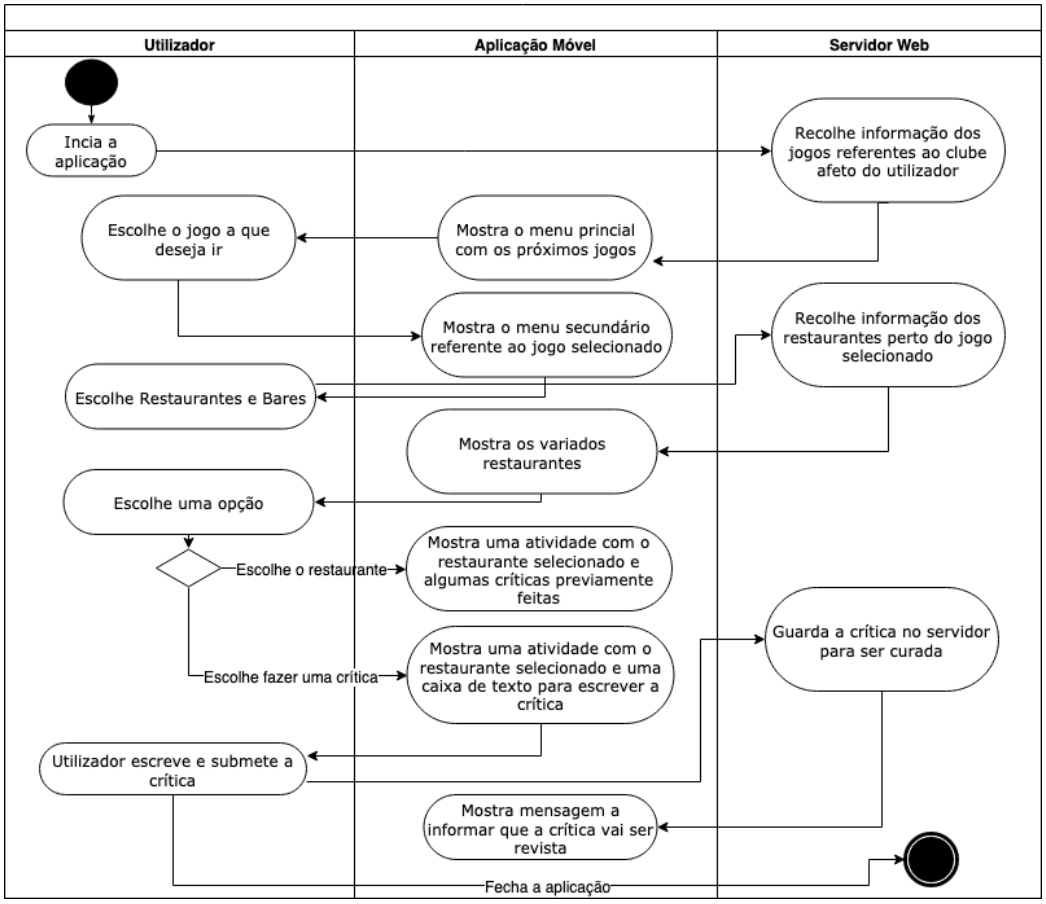

Figura 3.5: Diagrama de Atividade relativo à execução de uma crítica a um estabelecimento de restauração.

apresentado no ecrã. Ao preencher os dois parâmetros para se autenticar, isto é *username* e *password*, é feita a verificação dos dados. Se corresponderem ao esperado são apresentadas as três opções de ação do administrador, caso contrário apresenta uma mensagem de erro e volta à página inicial. Relativamente às opções, uma vez selecionadas, cada uma redireciona para uma página diferente. No primeiro caso, há a possibilidade de alterar o nível de confiança de um utilizador, preenchendo as informações, e no ato de submissão, estas são alteradas na respetiva base de dados. Já no segundo caso, em semelhança ao último, é apresentada uma tabela com as críticas e estacionamentos, onde é efetuada a depuração dos dados, que se estiverem errados, são eliminados os respetivos registos.

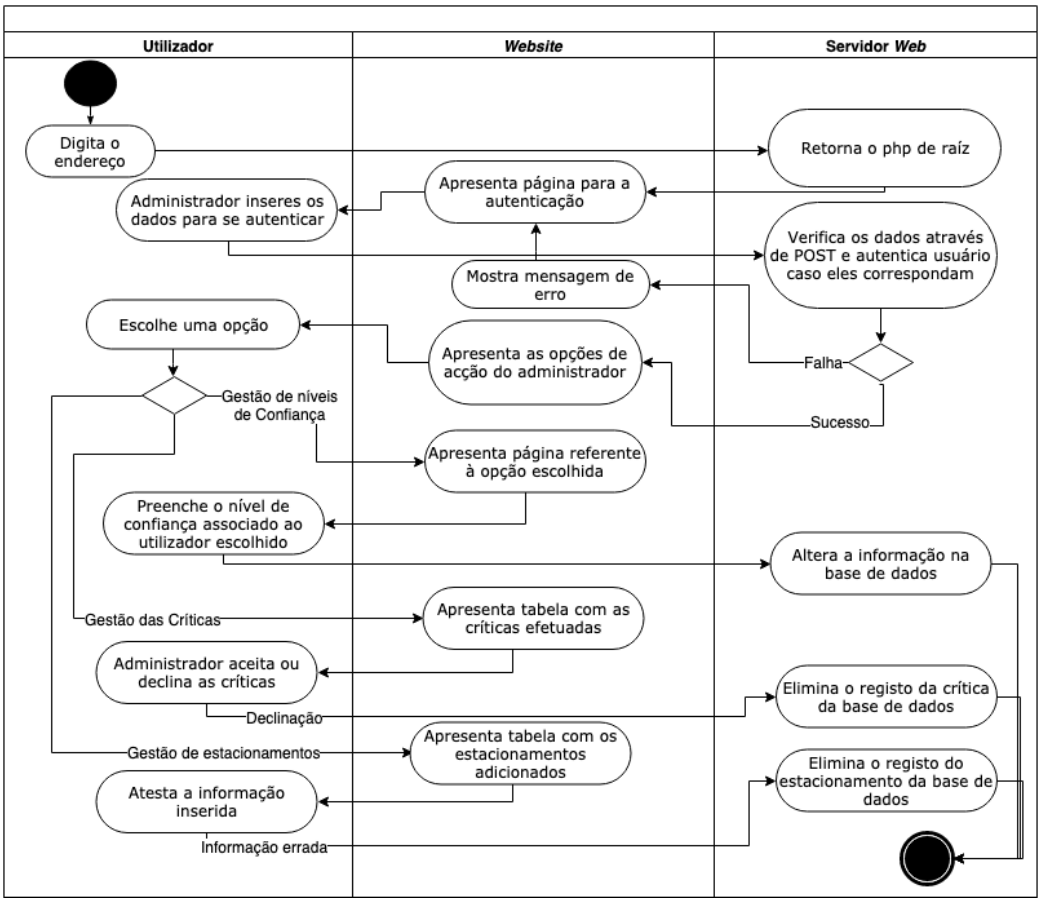

Figura 3.6: Diagrama de Atividades para diversas funcionalidades disponíveis na interface *web*.

3.5 Modelos de Dados

Numa primeira fase foi desenhado um modelo de dados que deu origem a sete tabelas numa base de dados relacional: uma com o nome `User` e mais outras seis, sendo que as primeiras três tinham o intuito de normalizar a relação estabelecida entre o utilizador e as restantes entidades, nomeadamente o estacionamento, o jogo e a restauração. Apesar do modelo ser apropriado, não atingia os objetivos delineados para um bom funcionamento do sistema pretendido. Em virtude disso, aproveitaram-se as relações geradas e foi ampliado o modelo, com a criação de novas tabelas, associando o clube ao utilizador, inserindo o clube em competições através de participações, e para terminar formar a tabela estádio e ligá-la ao jogo.

Após esta breve introdução vão ser analisadas com mais detalhe as tabelas, começando pela `User`. Nesta foi escolhido como chave primária um inteiro de

nome `idUser`, e estão inseridos nela o `uName`, `uEmail`, `uRep_Password` e o `uSalt` , sendo estes o nome de utilizador, o *e-mail*, a representação do *Password-Based Key Derivation Function 2* (PBKDF2) da *password* e o *salt* respetivamente, tal como o nome sugere. Para além disso, há também o `uTrust`, representante do nível de confiança do utilizador e o `isAdmin`, um inteiro 0 ou 1, que serve para distinguir o tipo de utilizador. Por fim, existe uma chave estrangeira ligada à tabela `has` para haver uma relação normalizada entre o `User` e o `Club`. Portanto, a tabela `has` contém a chave primária de identificação que auto incrementa e a chave secundária ligada ao número de identificação de um clube. Esta última chave está inserida e faz referência à tabela `Club`, sendo ela primária nesse contexto. De notar, que existem mais parâmetros presentes nomeadamente o nome ,`cName`, e a chave estrangeira que remete para o estádio. A tabela `participates`, serve como elemento de conexão entre o `Club` e a `Competition`, desempenhando o mesmo papel da `has`, e por isso, surgem naturalmente duas chaves estrangeiras, ambas ligadas ao número identificador da respetiva entidade. Para terminar, em cada participação há um inteiro associado, servindo de chave primária (`idParticipates`). Na tabela `Competition`, o mesmo acontece, no entanto através da `idCompetition`, onde se encontra também inserido o nome da prova (`coName`), o país no qual ocorre (`coCountry`) e por fim a `Season` que faz referência à temporada da competição. Por outro lado, criou-se o `Game`, com a chave primária referente à data e hora do jogo (`gDate`), assim como com o resultado (`gScore`), com a ligação entre dois clubes, isto é o visitante e o visitado, e por fim o estádio. Estes três últimos parâmetros são obtidos através de chaves estrangeiras. Como o utilizador vai a vários jogos, há a necessidade de criar uma correlação normalizada entre ambas as entidades, e por essa razão surge a tabela `goesToGame`, com duas chaves estrangeiras e uma primária, as primeiras referentes ao utilizador e ao jogo, já a última serve de *id* da conexão. Sendo o estádio uma entidade, este possuí uma tabela, com o seu número de identificação (`idStadium`), o nome (`sName`), o clube ao qual está associado (`sClub`), a morada (`sAdress`), as coordenadas do local onde se encontra (`sCoordinates`) e por último uma descrição histórica do mesmo(`sHistory`).

Posto isto, adicionou-se ao modelo uma associatividade entre os estacionamentos e os estádios, ou seja, um estádio pode possuir vários locais para deixar o veículo, que tem como efeito a criação de uma chave estrangeira alusiva ao estádio em questão. A tabela `Parking` possuí também uma chave primária de identificação (`idParking`), um nome (`pName`), e uma morada (`pAdress`), com as coordenadas do sítio (`pCoordinates`). Com isto agrega-se a informação dos lugares possíveis e disponíveis, tendo em conta a ótica do utilizador. Graças ao facto de um utilizador ter acesso ao estacionamento, é gerada a `goesToParking` correspondente-te à correlação gerada, em semelhança às outras referidas até aqui. Dado haver a possibilidade de realizar críticas, originou-se

a `pReviews` associada às últimas duas tabelas referidas, abrangendo assim duas chaves estrangeiras. Para além disso, tem duas chaves primárias: uma com a data e hora da submissão da crítica e outra com um inteiro identificador. Foi utilizada esta técnica para prevenir que, na eventualidade de dois utilizadores escreverem diferentes apreciações, uma seja negada, por restrições de equivalência na data e hora. Encontra-se vinculado à tabela o espaço para o comentário, bem como os lugares disponíveis. Uma vez que é necessário fazer uma verificação das críticas, criou-se uma coluna de nome `isChecked` com um booleano associado, que é posteriormente alterado após o administrador aceitar a crítica.

Com um subsistema relativamente parecido, aparece a restauração, já que contém também estabelecimentos e críticas, aos quais um utilizador tem acesso. De notar que não existem diferenças substantivas dignas de referência para que seja efetuada uma análise detalhada sobre a mesma.

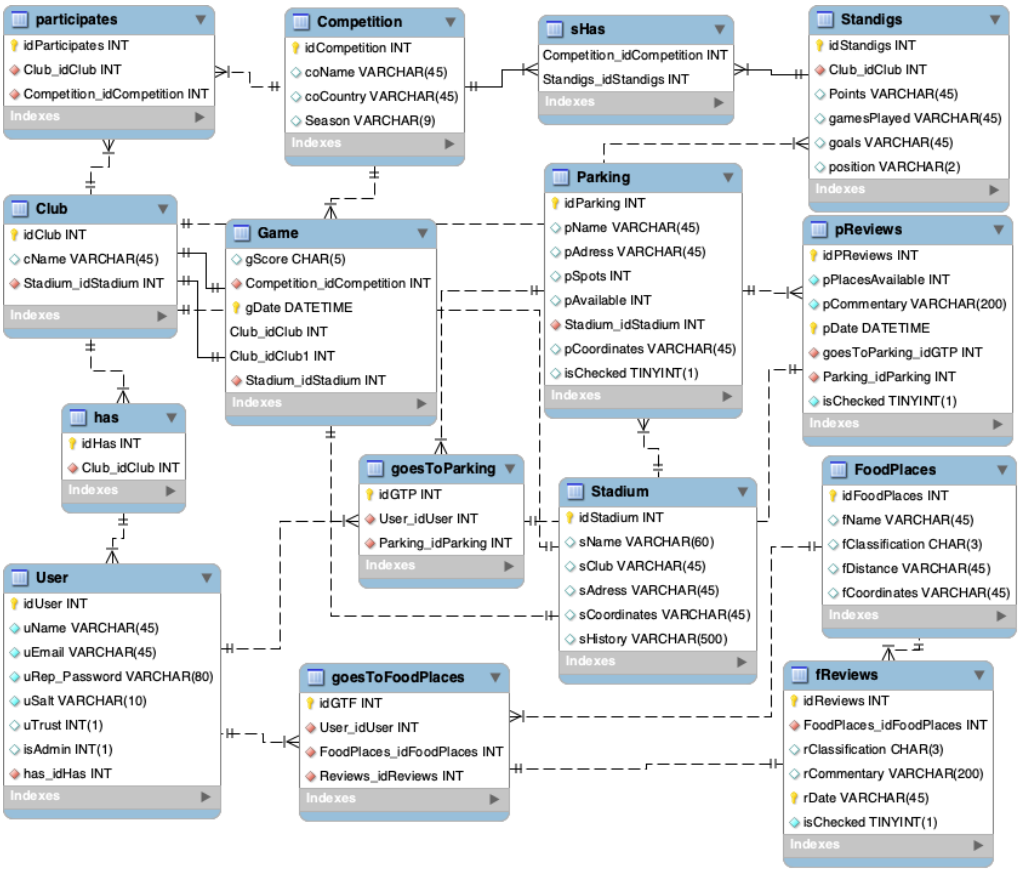

Figura 3.7: Modelo Físico referente à base de dados.

Recapitulando, através do modelo de base de dados é notório que o sistema vai

suportar vários utilizadores, e que cada um vai poder ter um clube associado. Os utilizadores podem ir aos jogos, inseridos numa competição, consequentemente visitando os estádios. Além disso, há a possibilidade estacionar o carro e visitar vários estabelecimento de restauração, deixando as suas críticas.

3.6 Arquitetura do Sistema

Recorrendo à figura 3.8 são apresentados os componentes que constituem o sistema. No servidor estão instalados os módulos *Apache*, *PHP*, *MySQL*, abordados na secção 2.2. Uma vez que é através do *Apache* que é criado o servidor, são aí feitas as ligações tanto do dispositivo móvel, como do navegador, sempre de forma segura através do protocolo HTTPS. Para terminar, o dispositivo móvel conta com o módulo *SQLite*, também mencionado na secção referida, onde estão guardadas as informações previamente recolhidas na base de dados, caso não haja comunicação com o servidor.

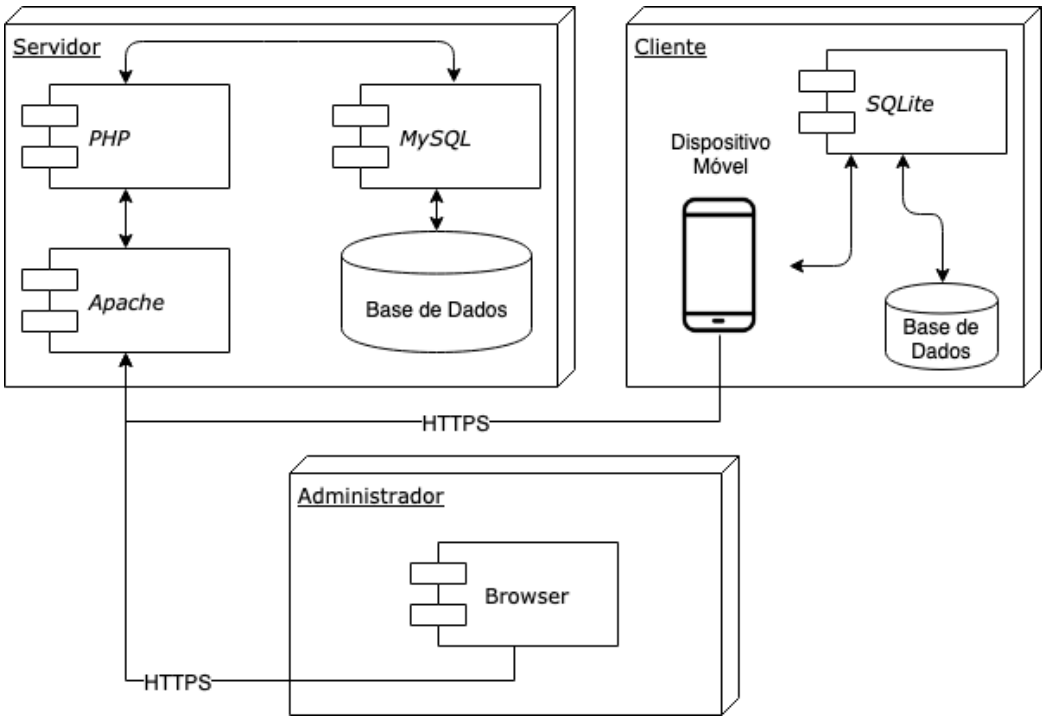

Figura 3.8: Diagrama dos componentes do Sistema.

3.7 Conclusões

Após o término de etapas como o levantamento de requisitos funcionais e não funcionais, o estudo dos casos de uso, a análise de diagramas de atividades, a criação do modelo de dados e por fim a delimitação da arquitetura do sistema, consegue-se avançar para a próxima fase do trabalho (implementação), já que foram dados passos importantes que servem de base para definir um sistema sólido e funcional.

Capítulo 4

Implementação

4.1 Introdução

Neste capítulo são descritos os passos de maior relevância que concretizaram o sistema, à imagem da análise realizada anteriormente. Assim sendo, ele encontra-se dividido da seguinte forma:

- a secção 4.2 – Servidor – descreve a configuração do mesmo para que possa responder às necessidades do sistema;
- a secção 4.3 – *Backoffice* – explica a criação da interface *web* onde se realiza toda a gestão de críticas e comentários, bem como os ficheiros que dão origem à API do sistema;
- a secção 4.4 – Aplicação Móvel – trata de todo o *layout* da aplicação móvel, assim como as atividades presentes nela.

4.2 Servidor

Numa primeira fase de desenvolvimento, criou-se um servidor no computador onde o próprio desenvolvimento da aplicação era feito com auxílio da instalação do XAMPP. Contudo graças às exigências do projeto, era necessário que o servidor estivesse na *Internet*. Desta forma, optou-se por se adquirir os serviços da Amazon, e através do seu programa AWS, obteve-se uma instância na *Cloud*. Também aqui se instalou o XAMPP de forma a obter as tecnologias necessárias para a configuração do servidor. Uma vez *online*, há que tomar medidas de segurança e, com esse objetivo particular em mente, criou-se um certificado auto-assinado através do comando no excerto de código 4.1.

```
openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes -
    keyout server.key -out server.crt
```

Excerto de Código 4.1: Comando para criar um *Self-signed Certificate*.

Associado ao facto de apenas a porta 433 se encontrar aberta para comunicações HTTPS garante uma comunicação segura entre o servidor e o dispositivo móvel ou até o computador. De notar que, idealmente, o certificado devia provir duma autoridade de certificação, mas para tal eram necessários gastos monetários, algo que não estava estipulado no âmbito do projeto.

De forma a comunicar com o servidor remoto deixa-se ainda aberta (para além da 433), a porta 22, que é a que é normalmente usada pelo protocolo *Secure Shell* (SSH). Para tornar a autenticação a esta máquina segura foi necessário fazer upload de uma chave pública que se associou à *Virtual Machine* (VM). A ligação é estabelecida através de um comando semelhante ao que está no excerto de código 4.2.

```
ssh -i webserverkeys.pem ubuntu@ec2-35-181-153-*.eu-west-3.
    compute.amazonaws.com
```

Excerto de Código 4.2: Comando para estabelecer a ligação com a máquina virtual.

O *upload* de ficheiros para efeitos de transmissão de código fonte referente ao *Backoffice* faz uso do protocolo *Secure Copy Protocol* (SCP) (também baseado no protocolo SSH), normalmente conseguido através de comandos semelhantes ao que se ilustra no excerto de código 4.3.

```
scp -i webserverkeys.pem /Users/JoseLamarao/server.key
    ubuntu@ec2-35-181-153-*.eu-west-3.compute.amazonaws.com:~
```

Excerto de Código 4.3: Exemplo de uma utilização do comando para enviar um ficheiro para a máquina virtual.

4.3 *Backoffice*

Esta secção trata de explicar a materialização do *backoffice*. Esta é atingida através de uma coletânea de ficheiros PHP, onde há três que têm a mesma função e nome, quer na interface *web*, quer na API do sistema. São eles o `config`, o `db_connect` e por último o `db_functions`. Os dois primeiros servem como conector à base de dados, onde a conexão gerada é posteriormente utilizado no último ficheiro mencionado. Neste está inserida uma classe com funções que executam *queries* e retornam o resultado das mesmas.

4.3.1 Interface *Web*

Antes de se avançar, faz sentido referir-se que todos os ficheiros foram protegidos com a requisição de um ficheiro nomeado `verifica_login`, já que contém código que certifica que há uma sessão iniciada através do parâmetro `username` e que, em caso negativo, redireciona o utilizador para a página inicial, tal como se pode ver pelo trecho de código seguinte:

```
if(!$_SESSION['username']) {
header('Location: index.php');
exit(); }
```

Excerto de Código 4.4: Comando para estabelecer a ligação com a máquina virtual.

Além disso, dada a importância dos dados que vão ser curados nesta parte do sistema, verificou-se a necessidade de criar um processo de autenticação, onde apenas o utilizador `administrador` tem acesso. Note-se que este utilizador é inserido na base de dados aquando a criação da mesma e a distinção entre um utilizador regular ou um administrador é feita através dum booleano, de acordo com o que foi estipulado na subsecção 3.5. Assim sendo, o *login* é conseguido com a cooperação de dois ficheiros: (i) o `index`, que é a página inicial onde é inserido o nome de utilizador e respetiva *password* e (ii), o `login` no qual é feita a verificação dos dados. Aqui, e em semelhança aos restantes ficheiros futuramente mencionados, os parâmetros são passados através de *POST*. Neste caso, é através de um botão que se despoleta a transmissão dos dados e com auxílio de uma função denominada `isAdmin`, presente no `db_functions`. Caso a função retorne *true*, é dado acesso ao `painel` de gestão. Caso contrário, é mostrada uma mensagem de erro, no `index`.

É no `painel` que o administrador tem acesso às opções previamente definidas no capítulo 3, sendo que todas elas redirecionam para uma nova página. À semelhança do processo anteriormente referido, também estas opções são conseguidas através de dois ficheiros. Um com a junção entre HTML e PHP, enquanto o outro conta apenas com PHP, uma vez que é nele que se realizam as alterações na base de dados, quer seja a nível de inserção, alteração ou até mesmo eliminação de registos. Com o intuito de evitar repetições e dado o procedimento ser algo semelhante em ambos os casos, abordar-se-á apenas a opção respeitante à análise de estacionamentos.

Definiu-se uma tabela que é preenchida com o auxílio da função `getParkingsWebsites`, que retorna todos os parques ainda não verificados, como está especificado no trecho de código 4.5. Na primeira coluna da tabela, cada linha é preenchida com uma *checkbox*, onde o seu valor é definido pela coluna `idParking`. Adicionalmente, todas as *checkboxes* criadas têm um formulário HTML associado, e um *array* com o intuito de transmitir os dados, bem como discernir os ids dos parque

selecionados, respetivamente. Também ao formulário estão ligados dois botões, o de aceitar e o de eliminar, que efetivam a transmissão *POST* dos identificadores dos parques, e especificam o tipo de atuação do ficheiro `reviewParking`. Quer seja a aceitação ou a eliminação dos registos, há sempre uma iteração sobre o *array* dos ids dos parques, sendo de seguida é chamada uma das duas funções seguintes: `acceptParking` ou `deleteParking`. No primeiro caso é alterado o valor da coluna `isChecked` para 1, com o intuito de ser mostrado na aplicação móvel, e no segundo caso, o registo é eliminado do parque. Importa realçar que antes de se proceder a qualquer ação é sempre emitido um alerta a confirmar a decisão do interveniente, da mesma forma que se dá outro alerta após a conclusão dos procedimentos discutidos. De forma a auxiliar a compreensão, está em apêndice o respetivo código que executa estas tarefas (ver excertos de código A.1 e A.2).

```
public function getParkingWebsite(){
    $stmt=$this->conn->prepare("SELECT * FROM Parking WHERE
        isChecked=0");
    $stmt->execute();
    return $stmt->get_result();
}
```

Excerto de Código 4.5: Comando para estabelecer a ligação com a máquina virtual.

4.3.2 API

De acordo com a menção à introdução da tecnologia, na subsecção 2.2.3, no âmbito deste projeto consumiu-se uma API gratuita e criou-se uma outra.

Sendo o produto do projeto uma possível substituição de outras aplicações já estabelecidas no mercado, graças à inovação a nível de informação dada referente a estacionamentos e restaurantes, há que manter um certo patamar no que diz respeito às informações do futebol. Desta forma, era importante obter um mecanismo automático que obtivesse informação de resultados para alimentar a base de dados, e como solução obteve-se acesso ao `footboll-data.org` [11], que disponibiliza uma API gratuita. Após o devido registo adquiriu-se uma chave de acesso, e posteriormente, com a devida configuração do pedido HTTP, conseguiu-se popular a base de dados do sistema, com as informações necessárias. No trecho de código A.3, está ilustrado o exemplo da obtenção dos jogos que dizem respeito à primeira liga, para posterior inserção na base de dados. Estes pedidos são feitos automaticamente pelo servidor remoto e de forma a não ultrapassar os limites impostos pela API a consumir.

A API desenvolvida propositadamente para o sistema pode ser dividida em duas: (i) a parte que fornece as informações, no*Backoffice* e (ii), a parte que as

consome, na aplicação móvel. A primeira é um conjunto de ficheiros que têm funções específicas e servem sempre como facilitadores da base de dados do sistema. O trecho de código 4.6, parte do ficheiro `getGames.php`, ajuda a entender esta parte da API. Neste ficheiro é invocada a função com o mesmo nome, através do auxílio do `db_functions` previamente abordado, e esta retorna todos os jogos referentes ao `id` do clube do utilizador. De seguida, é criado o *array* `Game`, que contém para cada jogo um objeto desse tipo, inserido-se ao tal vetor. Para terminar, é enviada, usando JSON, a lista de jogos para a aplicação que a requisitar.

```
$games = $db->getGames($_POST["club"]);
    if($games != null){
        for($i = 0;$i<sizeof($games);$i++){
            $game= (object)array();
        $game->score=$games[$i][0];
        $auxCompetition=$db->getCompetition($games[$i][1]);
            $game->competition=$auxCompetition;
        $game->timeanddate=$games[$i][2];
        $game->round=$games[$i][3];
            $auxHome=$db->getClubName($games[$i][4]);
            $game->homeTeam=$auxHome;
            $auxAway=$db->getClubName($games[$i][5]);
            $game->awayTeam=$auxAway;
        $auxCoordinates=$db->getStadiumCoordinates($games[$i
            ][6]);
            $game->coordinates=$auxCoordinates;
        $game->stadium=$games[$i][6];
    array_push($response["Game"],$game);
        }
    echo json_encode($response);
    }
```

Excerto de Código 4.6: Código referente à obtenção dos jogos associados a um certo clube posteriormente enviados através de JSON.

Para complementar este exemplo há a necessidade de entender como a aplicação móvel lida com esta resposta. Assim sendo e também com auxílio do trecho de código A.4, que se encontra em anexo por restrições de espaço, vai-se estudar este caso particular. É através da instanciação do cliente dado pela biblioteca *Retrofit2*, que são requisitadas as informações disponibilizadas pela API. Estando as respostas em JSON, para que estas sejam entendidas pelo *parser*, há a necessidade de se criarem duas classes, correspondentes ao objeto `jogo` e a uma lista destes objetos. Após a geração dos objetos, a aplicação está apta para lidar com as respostas do servidor, e posteriormente inserir na base de dados do dispositivo móvel os dados obtidos. Tomou-se esta decisão de forma a reduzir a quantidade de pedidos ao servidor e com especial intuito de a aplicação funcionar corretamente caso não tenha acesso à *Internet* em algumas ocasiões.

Contudo, há que realçar o facto da aplicação fazer um novo *request* de 5 em 5 dias afim de poder ter informações atualizadas. A data da última atualização guarda-se nas *Shared Preferences*, e após os 5 dias volta a fazer o pedido automaticamente.

4.4 Aplicação Móvel

A aplicação móvel conta com 17 atividades (o bloco princípal de aplicações Android). Neste caso, apenas algumas serão referidas nesta secção.

A primeira atividade é um *Splash Screen* onde é verificada a autenticação do utilizador. Esta autenticação pode ser feita tanto a nível local, como através do SDK do *Facebook*. Caso o utilizador esteja autenticado através do *facebook*, é feita uma verificação com o intuito de saber se ele se encontra registado na base de dados do sistema, já que não haveria outra forma de ter acesso acesso às informações personalizadas. Esta atividade encontra-se ilustrada através da figura 4.1.

```
if(!isLogged()){
        AccessToken accessToken = AccessToken.
            getCurrentAccessToken();
        boolean isLoggedIn = accessToken != null && !accessToken
            .isExpired();
        if(!isLoggedIn){
            finish();
            Intent iActivity = new Intent(SplashScreen.this,
                Login.class);
            startActivity(iActivity);
        } else { isRegistered(); }
} else {
        finish();
        Intent iActivity = new Intent(SplashScreen.this, Main.
            class);
        startActivity(iActivity);
}
```

Excerto de Código 4.7: Código referente à verificação da autenticação de um utilizador.

A atividade alusiva aos estacionamentos à volta de um estádio é constituída por uma barra de ferramentas, no topo do ecrã que permite voltar à atividade anterior, por um caixa de texto que remete o utilizador para uma outra atividade (nessa nova atividade, o utilizador pode submeter novos estacionamentos) e por um recurso designado por *RecyclerView*, que é uma forma estruturada de representar listas de dados, para apresentar os estacionamentos já adicionados e revistos. É interessante focar na forma como esta atividade carrega e mostra os dados, motivo pelo qual se inclui o trecho de código A.5. Este trecho de código mostra como é

Figura 4.1: Atividade `SplashScreen`.

populado o adaptador da *RecyclerView*, que tem associado ao seu construtor uma lista de parques, adaptada da API, onde para cada um deles é instanciado o *layout* `item_parking` e preenchidas as respetivas informações nos lugares destinados para o efeito. É importante frisar que as imagens associadas a este item são carregadas após a descodificação em `base64` duma *String*. A codificação de imagem para *String* é feita durante o processo de adicionar um estacionamento.

Antes de terminar esta secção, é feita uma breve análise da atividade que diz respeito aos estádios: `stadiumsActivity`. A atividade é semelhante à anterior, uma vez que também possuí um adaptador; todavia, para criar páginas deslizantes baseia-se na expansão de um item de *layout* pré definido, o `slide_stadium`. Os dados que vão integrar o conjunto de páginas e posteriormente servir como construtor do adaptador, são obtidos através de uma query à base de dados do dispositivo móvel, uma vez que ao um utilizador autenticar-se consomem-se da API as informações relativas aos estádios, e estas são armazenadas. No trecho de código 4.8 está ilustrado o que foi dito até aqui.

```
ArrayList<Stadium> stadiums= new ArrayList<>();
Cursor oCursor = oSQLiteDB.rawQuery("SELECT * FROM Stadium",new
    String[]{} );
Boolean bCarryOn=oCursor.moveToFirst();
while (bCarryOn){
```

```
        stadiums.add(new Stadium(oCursor.getString(1),oCursor.
            getString(2),oCursor.getString(3),oCursor.getString(4),
            oCursor.getString(5),oCursor.getString(6),oCursor.
            getString(7)));
        bCarryOn=oCursor.moveToNext();
    }
    for (int i=0;i<stadiums.size();i++){
        slideStadiumName.add(stadiums.get(i).getsName());
        slideStadiumClub.add(stadiums.get(i).getsClub());
        slideStadiumCapacity.add(stadiums.get(i).getsCapacity());
        slideStadiumYear.add(stadiums.get(i).getsYear());
        slideStadiumHistory.add(stadiums.get(i).getsHistory());
    }
    stadiumSliderAdapter = new stadiumSliderAdapter(StadiumsActivity
        .this,slideStadiumName,slideStadiumClub,slideStadiumCapacity,
        slideStadiumYear,slideStadiumHistory,drawableImages);
    viewPager.setAdapter(stadiumSliderAdapter);
```

Excerto de Código 4.8: Código referente à query com o intuito de obter as listas necessárias para criar o adaptor do *ViewPager*.

Para terminar são agora apresentadas três capturas de ecrã relativamente às atividades: `Main`, `ParkingActivty` e a `stadiumsActivity`, através da figura 4.2.

Figura 4.2: Atividades `Main`, `ParkingActivty` e `stadiumsActivity` por ordem de menção.

4.5 Conclusões

Este capítulo dá uma ideia geral no que diz respeito à implementação de todas as componentes integrantes do sistema desenvolvido. Houve sempre um cuidado especial relativamente à segurança do sistema, visto se tratar de um sistema baseado numa arquitetura cliente-servidor (que portanto comporta comunicações através da Internet) e que lida com alguns dados eventualmente privados. Usaram-se de estratégias de programação defensivas bem como implementação de protocolos seguros, nem todos mencionados neste capítulo. Para terminar, toda a implementação se regeu pelas especificações estipuladas no capítulo anterior.

Capítulo 5

Testes e Modelo de Negócio

5.1 Introdução

Neste capítulo faz-se uma descrição dos diferentes testes e respetivos resultados a que todos os componentes integrantes do sistema estiveram sujeitos. A ideia é por vezes causar cenários de erro, de forma a obter informações de como age o sistema quando presente neste tipo de situações. Quaisquer erros verificados, ajudam a melhorar a robustez da solução. Para além disso, é feito um modelo de negócios, onde vai assentar os alicerces da expansão da aplicação desenvolvida. Desta forma, a delimitação dos conteúdos é a seguinte:

1. a secção 5.2 – Testes – apresenta os diferentes testes a nível da interface web e aplicação móvel;

2. a secção 5.3 – Modelo de Negócios – transparece as ideias principais para a expansão da aplicação *Android*.

5.2 Testes

Como foi referido na introdução a este capítulo nesta secção vão ser ilustrados os diferentes testes a que o sistema foi sujeito. Desta forma, optou-se por dividir-se esta secção em três partes de modo a manter a coerência e estrutura. Sendo assim, a primeira parte refere-se a todos os testes a nível da interface web, de seguida os testes alusivos à aplicação móvel e por fim é efetuado um teste de usabilidade também este à aplicação móvel.

5.2.1 Interface Web

Nesta parte estão detalhados os testes efetuados à interface web com o auxílio da tabela 5.1 e de seguida os seus resultados na tabela 5.2. Aqui pode-se verificar o funcionamento expectável desta parte do sistema, já que os resultados obtidos, seguem o padrão do que era esperado.

ID	Descrição do Teste	Resultado Esperado
0	Tentativa de Login sem inserir pârametros	Redirecionamento para a página inicial, indicando erro
1	Tentativa de Login com `Administrador` e password errada	Redirecionamento para a página inicial, indicando erro
2	Tentativa de Login com um `Username` registado da aplicação	Redirecionamento para a página inicial, indicando erro
3	Tentativa de alterar o nível de confiança sem especificar o `Username`	Indicação de erro para preencher os campos em falta
4	Tentativa de alterar o nível de confiança de um `Administrador`	Indicação de erro, indicando a impossibilidade dessa ação
5	Tentativa de aceitar um comentário de um estacionamento sem selecioná-lo	Indicação de erro, indicando para selecionar pelo menos um comentário
6	Tentativa de eliminar um comentário de um estacionamento sem selecioná-lo	Indicação de erro, indicando para selecionar pelo menos um comentário
7	Tentativa de aceitar um estacionamento sem selecioná-lo	Indicação de erro, indicando para selecionar pelo menos um estacionamento
8	Tentativa de eliminar um estacionamento sem selecioná-lo	Indicação de erro, indicando para selecionar pelo menos um estacionamento
9	Tentativa de aceitar uma crítica a um estabelecimento sem selecioná-lo	Indicação de erro, indicando para selecionar pelo menos uma crítica
10	Tentativa de eliminar uma crítica a um estabelecimento sem selecioná-lo	Indicação de erro, indicando para selecionar pelo menos uma crítica

Tabela 5.1: Bateria de Testes referentes à interface web.

5.2.2 Aplicação Móvel

À semelhança do que foi feito anteriormente, aqui estão especificados os testes efetuados à aplicação móvel(adiante designada por *app*) com o auxílio da tabela 5.3 e o seus resultados retratados na tabela 5.4. Através da análise da última, pode-se concluir que os resultados estão de acordo com o que era esperado.

5.2.3 Usabilidade

De forma a obter uma ideia geral da usabilidade da aplicação *Android* desenvolvida, procedeu-se à recolha de opiniões através de um formulário simples. Neste abordou-se não só o *design*, como também a utilidade, a facilidade de utilização,

ID	Resultados Obtidos
0	Apresentação de erro na página inicial
1	Apresentação de erro na página inicial
2	Apresentação de erro na página inicial
3	Apresentação de erro a indicar para preencher os campos
4	Apresentação de erro a indicar a impossibilidade da ação
5	Apresentação de erro a indicar para selecionar pelo menos um comentário
6	Apresentação de erro a indicar para selecionar pelo menos um comentário
7	Apresentação de erro a indicar para selecionar pelo menos um estacionamento
8	Apresentação de erro a indicar para selecionar pelo menos um estacionamento
9	Apresentação de erro a indicar para selecionar pelo menos uma crítica
10	Apresentação de erro a indicar para selecionar pelo menos uma crítica

Tabela 5.2: Resultados da Bateria de Testes relativos à interface web.

ID	Descrição do Teste	Resultado Esperado
0	Tentativa de Registo com um nome de utilizador já em uso	Toast a indicar que nome já está em uso
1	Tentativa de registo com inserção de um e-mail que não segue as regras xxxx@xx.xx	Toast a indicar que e-mail não é válido
2	Tentativa de registo com uma password menor que 5 caracteres	Toast a indicar que a password tem que ser maior ou igual a 5 caracteres
3	Tentativa de registo com duas passwords diferentes	Toast a indicar que as passwords diferem
4	Tentativa de registo sem escolher um clube afeto	Toast a indicar que utilizador tem que escolher um clube
5	Tentativa de login sem inserir pârametros	Toast a indicar para preencher os campos
6	Tentativa de login com uma password errada	Toast a indicar que login se encontra errado
7	Tentativa de inserção de um parque de estacionamento, sem informações, ou com dados em falta	Toast a indicar para preencher os campos em falta
8	Tentativa de inserir um comentário a um estacionamento, sem informações, ou com dados em falta	Toast a indicar para preencher os campos em falta
9	Tentativa de inserir uma crítica relativa a um estabelecimento de restauração, sem informações, ou com dados em falta	Toast a indicar para preencher os campos em falta
10	Tentativa de inserir uma crítica onde não há restaurantes associados	Impossibilidade de ter acesso a essa opção

Tabela 5.3: Bateria de Testes à aplicação móvel *Android*

intuitividade e por último a interatividade da *app*. Assim, foram escolhidos ao acaso 5 utilizadores amigos do autor que tiveram acesso à aplicação móvel para poderem responder ao questionário em questão. Mostra-se a seguir com o auxílio da tabela 5.5 os resultados obtidos. Aqui é possível ter uma noção de alguns aspetos a melhorar nomeadamente o *design* da aplicação, algo que nunca foi tomado em conta, embora tenha havido sempre um esforço de tornar a *app* apelativa. De resto, a aplicação demonstrou utilidade, bem como facilidade de utilização, intuitividade e Interatividade.

ID	Resultado Obtido
0	Notificação de erro graças ao nome de utilizador já em uso
1	Notificação de erro devido à má formação do e-mail indicado
2	Notificação de erro porque a password é demasiado pequena
3	Notificação de erro dado as passwords diferirem
4	Notificação de erro a indicar a falta de escolha de clube
5	Notificação de erro a pedir para inserir o nome de utilizador
6	Notificação de erro graças à falha de autenticação provocada pela palavra-passe errada
7	Notificação de erro a pedir para inserir as informações necessárias
8	Notificação de erro a pedir para inserir as informações necessárias
9	Notificação de erro a pedir para inserir as informações necessárias
10	Notificação de erro a pedir para inserir as informações necessárias

Tabela 5.4: Resultados da Bateria de Testes à aplicação móvel *Android*

	0	1	2	3	4	5
Design	0	0	0	1	3	1
Utilidade	0	0	0	0	1	4
Facilidade de Utilização	0	0	0	0	0	5
Intuitividade	0	0	0	0	0	5
Interatividade da Aplicação	0	0	0	0	0	5

Tabela 5.5: Resultados dos questionários da usabilidade da aplicação móvel *Android*.

5.3 Modelo de Negócios

Para ser realizada esta etapa do relatório foi necessário efetuar um levantamento de informações disponíveis na *Internet* com o intuito de materializar um modelo de negócios, visando o sistema desenvolvido. Através desta pesquisa obteve-se acesso ao livro "Bussiness Model Generator: A Handbook for visionaries, game changers, and challengers"[12] onde os autores formularam uma lista de nove questões indispensáveis para a realização de um modelo de negócios, sendo elas:

1 - Proposição de valor: Porque é que a sua oferta é única no mercado?;

2 - Clientes: Quais os clientes alvo?;

3 - Atividades: Que produto ou serviço oferece?;

4 - Parcerias estratégicas: Que empresas vai auxiliar a melhoria da sua oferta?;

5 - Fontes de receita: Quais as fontes geradoras de lucro?;

6 - Fontes de gastos: Quais as fontes de custo?;

7 - Recursos: Com que meios vai atingir os objetivos da sua oferta?;

8 - Canais de comunicação e distribuição: Como é que a sua oferta vai chegar ao cliente?;

9 - Relação com o cliente: Como é que a empresa interage com o cliente?;

Posto isto, vai agora tentar-se responder a cada pergunta de forma a concretizar o modelo de negócios. Com a ajuda da tabela 2.1 é possível responder à primeira questão atentando ao facto de não haver aplicações no mercado do futebol que facultem as mesmas informações, nomeadamente saber como chegar, onde estacionar e onde comer. Já no que toca à segunda pergunta tem-se que os clientes alvo são todos e quaisquer espetadores assíduos ou esporádicos das infraestruturas de um clube de futebol. Relativamente à terceira questão, o produto oferecido vai ser um sistema de software concentrado numa aplicação móvel que é capaz de responder às informações requisitadas tal como foi discutido até agora. Em termos de parcerias possíveis e respondendo à quarta questão era proveitoso uma associação com a `Primeira Liga`, uma vez que é esta que trata de todo o processo em termos de os jogos em Portugal. Podia assim, expandir e publicitar a aplicação desenvolvida, facilitando o processo. Para responder à quinta questão em termos de fontes de receitas será através de publicidade adjacente na *app*, associado a um quota para a inserção de um estabelecimento de restauração na base de dados, paga pelo estabelecimento que assim o desejar. Agora respondendo à sexta pergunta no que diz respeito aos gastos será sempre a manutenção de servidores, bem como um custo associado à melhoria e manutenção do sistema através de uma equipa designada para o efeito. À sétima questão não é necessário responder uma vez que a oferta já se encontra disponível e concretizada. A aplicação vai estar disponível gratuitamente na *Play Store* da *Google*, sendo esta a resposta à oitava pergunta. E para terminar respondendo à última questão a empresa poderá interagir com o cliente através das redes sociais, de forma a promover sorteios entre outras coisas.

5.4 Conclusões

Através dos resultados obtidos, relativamente aos testes efetuados, consegue-se concluir que o sistema encontra-se bem implementado e vai de encontro ao que foi proposto. Todavia há espaço para melhorias, nomeadamente a nível do design da aplicação móvel. De salientar que qualquer detalhe ou situação pontual pode

vir a ser alvo de melhorias no futuro. Relativamente à secção 5.3, com as respostas dadas consegue-se obter um modelo de negócios capaz de iniciar o processo de publicação e expansão do sistema desenvolvido. Chegando ao fim deste capítulo está na altura de fazer uma retrospetiva do que foi alcançado, retirando as devidas conclusões, bem como olhar para o futuro de forma a melhorar a proposta elaborada.

Capítulo 6

Conclusões e Trabalho Futuro

Neste capítulo final é tempo de reflexão de forma a obter conclusões, estando a secção 6.1 destinada para o efeito. Para além disso, serão discutido possíveis melhoramentos no que diz respeito ao sistema desenvolvido na secção 6.2.

6.1 Conclusões Principais

Era objetivo deste projeto a criação dum sistema capaz de fornecer informações de como chegar, onde estacionar, onde comer que visasse um utilizador assíduo ou esporádico das infraestruturas de um clube de futebol. De forma a responder a estas necessidades, este trabalho assentou no desenvolvimento de um sistema constituído por duas componentes nomeadamente: uma interface web e uma aplicação móvel; com a ajuda de um servidor web configurado no âmbito deste projeto. Através do primeiro integrante mencionado é possível a um `Administrador` devidamente autenticado realizar uma análise das informações inserida pelos utilizadores ao usarem a aplicação móvel, de modo a servir de curador. Já o segundo componente é direcionado para dispositivos móveis, com o SO *Android*, onde só é possível o acesso a quem esteja devidamente registado. Tendo isto em consideração a aplicação dispõe de todas as funcionalidades previamente mencionadas como objetivos. Assim sendo, o sistema desenvolvido atinge, com sucesso, os objetivos delimitados no início do projeto e o protótipo do mesmo encontra-se *online*, totalmente testado e funcional.

Em termos pessoais é da ótica do autor que por intermédio da elaboração deste projeto, houve não só a solidificação de conhecimentos adquiridos como também obtenção de novos paradigmas que se demonstrarão úteis no futuro. Para além disso, este trabalho serviu para cimentar todas virtudes que um bom trabalhador deve ter.

6.2 Trabalho Futuro

Embora as principais funcionalidades tenham sido implementadas, há sempre espaço para melhorar a solução até aqui descrita e como tal vão ser apresentadas algumas sugestões para se ter em conta no futuro:

1. Implementação de uma autenticação baseada em *Tokens* (e.g *oAuth 2.0*);
2. Implementar toda a nuance dos níveis de confiança a que a aplicação móvel se encontra sujeita;
3. Implementar a capacidade de disponibilizar as informações em tempo real;
4. Implementar vídeos e notícias referentes ao clube afeto de um utilizador;
5. Implementar um sistema de troca de mensagens;
6. Implementar uma outra aplicação paga que disponibiliza *odds* e informações privilegiadas no que diz respeito a apostas desportivas;
7. Expandir a aplicação de forma a atingir outros desportos, bem como outros países, algo que é possível dada a escalabilidade de que é dotado o sistema desenvolvido.

Apêndice A

Excertos de códigos relevantes no âmbito do projeto

Neste apêndice estão representados alguns trechos de código relevantes mencionados ao longo deste documento. Cada um tem uma legenda associada indicando a sua utilização.

```
<?php
session_start();
include('verifica_login.php');
require_once 'db_function.php';
$db = new DB_Functions();

$result= $db->getParkingWebsite();
?>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Rever Estacionamentos</title>
        <link rel="stylesheet" href="css/StyleWebService.css"
            type="text/css">
    </head>
    <body>
      <table border="2"  class="content-table">
                <tr>
                <th></th>
                    <th>Nome do Estacionamento</th>
                    <th>Morada do Estacionamento</th>
                    <th>N mero de Lugares</th>
                    <th>N mero de Lugares Disponive s</th>
                    <th>Est dio</th>
        <th>Coordenadas</th>
                    <th>  pago</th>
                    <th>Imagem</th>
```

```
                </tr>
                <?php  WHILE($row= $result->fetch_assoc() ){ ?>
                <tr>
        <td style="text-align:center;"><input form="myForm" type
            ="checkbox" name="idParking[]" value=<?php echo $row
            ['idParking']?>  /> </td>
                    <td><?php echo $row['pName']; ?></td>
                    <td><?php echo $row['pAdress']; ?></td>
                    <td><?php echo $row['pSpots']; ?></td>
                    <td><?php echo $row['pAvailable']; ?></td>
                    <td><?php echo $db->getStadiumName($row['
                        Stadium_idStadium']); ?></td>
                    <td><?php echo $row['pCoordinates']; ?></td>
                    <td><?php if($row['isPayed']=="0") echo "
                        Gr tis"; else echo "Pago";  ?></td>
        <td><?php if($row['path']!="NULL") echo "<img src='
            projectapi/{$row['path']}' width='250'"   ?></td>
                </tr>
            <?php }  ?>
         </table>
    <form action="reviewParking.php"  method="post" id="myForm"
        onsubmit="return confirm('Deseja aceitar/apagar o(s)
        estacionamento(s)?');">
    <p><input type="submit" name="isAccepted" value="Aceitar"></
        p>
    <p><input type="submit" name="isNotAccepted" value="Eliminar
        "></p>
    </form>
  </body>
    </body>
</html>
```

Excerto de Código A.1: Código referente à materialização da página alusiva à verificação de estacionamentos.

```
<?php
session_start();
include('verifica_login.php');
require_once 'db_function.php';
$db = new DB_Functions();

if($_POST['isAccepted']=="Aceitar"){
if(isset($_POST['idParking'])){
$idParkings=$_POST['idParking'];

for($i=0;$i<sizeof($idParkings);$i++){
$db->acceptParking(intval($idParkings[$i]));
}
    $message="Estacionamento(s) aceite(s) com sucesso";
```

```
    echo "<script type='text/javascript'>alert('$message');</
        script>";
    header("refresh:0;url=reviewParkingHelper.php");
    exit();
} else {
    $message="Selecione pelo menos um estacionamento";
    echo "<script type='text/javascript'>alert('$message');</
        script>";
    header("refresh:0;url=reviewParkingHelper.php");
    exit();
}
} else {
if(isset($_POST['idParking'])){
$idParkings=$_POST['idParking'];

for($i=0;$i<sizeof($idParkings);$i++){
$db->deleteParking(intval($idParkings[$i]));
}
    $message="Estacionamento(s) eliminado(s) com sucesso";
    echo "<script type='text/javascript'>alert('$message');</
        script>";
    header("refresh:0;url=reviewParkingHelper.php");
    exit();
} else {
    $message="Selecione pelo menos um estacionamento";
    echo "<script type='text/javascript'>alert('$message');</
        script>";
    header("refresh:0;url=reviewParkingHelper.php");
    exit();
}
}
?>
```

Excerto de Código A.2: Código auxiliar à verificação de estacionamentos.

```
<?php
require_once 'db_function.php';
$db = new DB_Functions();

    //for($j=1;$j<=34;++$j){
    //$uri = 'http://api.football-data.org/v2/competitions/PPL/
        matches/?matchday='.$j;
    $uri = 'http://api.football-data.org/v2/competitions/PPL/
        matches?season=2018';
    $reqPrefs['http']['method'] = 'GET';
    $reqPrefs['http']['header'] = 'X-Auth-Token: 5982
        c11e47ab47029b64c6b5b456c5a3';
    $stream_context = stream_context_create($reqPrefs);
    $response = file_get_contents($uri, false, $stream_context);
    $matches = json_decode($response,true);
```

```
    for($i=0;$i<sizeof($matches["matches"]);++$i) {
  $round=intval($matches["matches"][$i]["matchday"]);
  //$round=intval($matches[""]["matchday"]);
  $homeTeam=$matches["matches"][$i]["homeTeam"]["name"];
  $awayTeam=$matches["matches"][$i]["awayTeam"]["name"];
  $date=str_replace('T',' ',$matches["matches"][$i]["utcDate"]);
  $date=str_replace('Z','',$date);
  $new_date= date("Y-m-d H:i:s", strtotime($date . " +1 hours"))
      ;
  $score=$matches["matches"][$i]["score"]["fullTime"]["homeTeam"
      ]." - ".$matches["matches"][$i]["score"]["fullTime"]["
      awayTeam"];
  $db->storeGames($score,1,$new_date,$round,$homeTeam,$awayTeam)
      ;
    }
//}
?>
```

Excerto de Código A.3: Pedido HTTP para posterior inserção na base de dados a informação obtida.

```
mService.getClubGames(team).enqueue(new Callback<GameAPI>() {
            @Override
            public void onResponse(Call<GameAPI> call, Response<
                GameAPI> response) {
                GameAPI result=response.body();
                ArrayList<Game> aux = result.getGame();

                for(int i=0;i< aux.size();i++){
                    ContentValues contentValues=new
                        ContentValues();
                    contentValues.put("gScore",aux.get(i).
                        getScore());
                    contentValues.put("Competition",aux.get(i).
                        getCompetition());
                    contentValues.put("gDate",aux.get(i).
                        getTimeanddate());
                    contentValues.put("gRound",aux.get(i).
                        getRound());
                    contentValues.put("homeClub",aux.get(i).
                        getHomeTeam());
                    contentValues.put("awayClub",aux.get(i).
                        getAwayTeam());
                    contentValues.put("Coordinates",aux.get(i).
                        getCoordinates());
                    contentValues.put("stadium",aux.get(i).
                        getStadium());
                    oSQLiteDB.insert("Game",null,contentValues);
                }
```

```
                sp.edit().putString("date",String.valueOf(System
                    .currentTimeMillis())).apply();
                populateRV(team);
            }

            @Override
            public void onFailure(Call<GameAPI> call, Throwable
                t) {
                Toast.makeText(Main.this, "Ocorreu um Erro\
                    nVerifique a liga  o    Internet", Toast.
                    LENGTH_LONG).show();
            }
        });
```

Excerto de Código A.4: Código referente à inserção dos jogos na base de dados local após a receção da resposta da API.

```
public void onBindViewHolder(@NonNull ViewHolder holder, int
    position) {
        Parking currentParking=parkings.get(position);

        holder.parkingTitle.setText(currentParking.getpName());
        holder.availableSpots.setText("Lugares Dispon veis:"+
            String.valueOf(currentParking.getpAvailable()));
        String coordenadas=currentParking.getpCoordinates();
        if(coordenadas.equals("NULL"))
            holder.parkingCoordinate.setText(currentParking.
                getpAdress());
        else holder.parkingCoordinate.setText(currentParking.
            getpCoordinates());
        int gratis=currentParking.getIsPayed();
        if(gratis == 0)
            holder.isPayed.setText("Gr tis");
        else holder.isPayed.setText("Pago");
        String encodedString=currentParking.getImage();
        if(encodedString.equals("NULL")){
            holder.image.setBackgroundResource(R.drawable.
                no_image);
        }else {
            byte[] decodedString = Base64.decode(encodedString,
                Base64.DEFAULT);
            Bitmap decodedByte = BitmapFactory.decodeByteArray(
                decodedString, 0, decodedString.length);
            holder.image.setImageBitmap(decodedByte);
        }
    }
```

Excerto de Código A.5: Código referente à instanciação do textttitem_parking com os conteúdos provenientes em forma de lista da API.

Bibliografia

[1] Information Service FIFA Communications Division. FIFA Big Count 2006: 270 million people active in football, 2007. [Online] https://www.fifa.com/mm/document/fifafacts/bcoffsurv/bigcount.statspackage_7024.pdf. Último acesso a 18 de Maio de 2019.

[2] Publicis Media Sport & Entertainment. Global broadcast and audience summary, 2018. [Online] https://resources.fifa.com/image/upload/njqsntrvdvqv8ho1dag5.pdf. Último acesso a 18 de Maio de 2019.

[3] Liga Portugal. Estatísticas Liga NOS, 2018. [Online] http://ligaportugal.pt/pt/liga/estatisticas/espectadores/clube/20172018/liganos. Último acesso a 18 de Maio de 2019.

[4] Delloite. Global Mobile Consumer Survey, 2018. [Online] https://www2.deloitte.com/pt/pt/pages/technology-media-and-telecommunications/articles/global-mobile-consumer-survey.html. Último acesso a 21 de Maio de 2019.

[5] FlashScore. Google Play - Flashscore. [Online] https://play.google.com/store/apps/details?id=eu.livesport.FlashScore_com. Último acesso a 21 de Maio de 2019.

[6] SofaScore. Google Play - Sofascore. [Online] https://play.google.com/store/apps/details?id=com.sofascore.results. Último acesso a 21 de Maio de 2019.

[7] Sportinveste Multimédia SA. Google Play - LIGA PORTUGAL. [Online] https://play.google.com/store/apps/details?id=pt.vsports.client. Último acesso a 21 de Maio de 2019.

[8] Media Capital. Google Play - Maisfutebol. [Online] https://play.google.com/store/apps/details?id=pt.iol.maisfutebol.android. Último acesso a 21 de Maio de 2019.

[9] Onefootball GmbH. Google Play - Onefootball. [Online] https://play.google.com/store/apps/details?id=de.motain.iliga. Último acesso a 21 de Maio de 2019.

[10] Forza Football. Google Play - Forza Football. [Online] https://play.google.com/store/apps/details?id=se.footballaddicts.livescore. Último acesso a 21 de Maio de 2019.

[11] Football-Data. football-data.org. [Online] https://www.football-data.org/. Último acesso a 28 de Junho de 2019.

[12] Alexander Osterwalder, Yves Pigneur, Tim Clark, and Alan Smith. *Bussiness Model Generator: A Handbook for visionaries, game changers, and challengers*. John Wiley and Sons Ltd, 2010.