



Departamento
de Engenharia Informática e de Sistemas

Desenvolvimento de uma APP Android e Plataforma Web para comunicação com Beacons

Dissertação apresentada para a obtenção do grau de Mestre em
Informática e Sistemas

Autor
Nuno Alexandre Lemos de Paiva

Orientadora
Prof. Doutora Ana Rosa Borges
Instituto Superior de Engenharia de Coimbra

Coimbra, junho, 2016

Uma palavra especial para os meus pais,
a toda a minha família, namorada e amigos por todo o apoio e
motivação que me deram.

“The happiest people are those who are too busy to notice whether they are or not.”

William Feather

AGRADECIMENTOS

Agradeço ao Engenheiro Francisco Maia e ao Professor Doutor João Orvalho por todo o apoio, ensinamentos, sugestões e confiança que depositaram em mim desde o primeiro momento em que surgiu a proposta inovadora para a **Streamline**. Sempre dispostos a ajudar, sem ambos não seria possível a realização deste projeto.

À minha orientadora por parte do ISEC, a Professora Doutora Ana Rosa Borges por todo o apoio, conhecimento, paciência e tempo dispensado ao longo do desenvolvimento da dissertação.

Ao Instituto Superior de Engenharia de Coimbra (ISEC), mais concretamente ao Departamento de Engenharia Informática e de Sistemas (DEIS) e respetivos docentes, por, ao longo da licenciatura e mestrado, me fornecerem o conhecimento técnico necessário para o desenvolvimento do projeto, mas acima de tudo por me tornar na pessoa capaz que sou hoje.

De uma forma particular agradeço a toda a minha família, concretamente aos meus pais Luís Paiva e Ana Estela, por todo o apoio incondicional, e toda a motivação extra nas horas difíceis. Acreditando sempre em mim nunca deixaram de me apoiar em todos os sentidos.

À minha namorada por sempre acreditar em mim e estar ao meu lado em todos os momentos.

Aos meus amigos e principalmente os de infância, alguns mesmo colegas de curso, que sempre me deram forças, conselhos, motivação e alegria para nunca desistir.

A todos estes os mais sinceros agradecimentos.

RESUMO

O surgimento da tecnologia *iBeacon*, em 2014, veio dinamizar e inovar um conjunto de áreas de negócio, das quais se destaca o retalho.

Este trabalho tem como objetivo principal a implementação de uma solução assente na tecnologia *iBeacon* que permita trazer vantagens para o retalhista ao mesmo tempo que melhora e inova a experiência de compra para o cliente.

A solução desenvolvida é composta por três componentes.

A primeira, o *Content Management System* (CMS), consiste numa plataforma *Web*, destinada a ser utilizada pelo retalhista, a qual permite controlar o funcionamento da solução. Nesta, o retalhista pode gerir as suas lojas e mapear os seus *beacons* de forma interativa, tendo em conta a sua disposição no ambiente real. Consegue gerir todos os clientes, possibilitando o agrupamento dos mesmos após análise de resultados. Pode ainda efetuar o envio de campanhas direcionadas a uma determinada área, ou a um conjunto de clientes específicos, integrando o sistema *Passbook*. Por fim, o CMS disponibiliza um módulo de análise de dados, o qual é composto por seis secções: *Dashboard* geral; *Dashboard* por campanha; *Dashboard* por região; *Dashboard* por cliente; Percurso dos clientes e *Heatmap*. Nestas será possível recolher dados como: Afluência de clientes por local e data; TOP de campanhas enviadas, abertas, ignoradas; Locais mais frequentados, por cliente e data; Percurso efetuado por cliente numa loja, entre outros.

A segunda componente consiste numa aplicação móvel, desenvolvida para o sistema *Android*, destinada a ser usada pelos clientes. O foco do seu desenvolvimento consistiu em garantir a compatibilidade com quaisquer tipos de *beacons*, desde que respeitem a norma *iBeacon*, bem como interagir com o CMS, recebendo campanhas de acordo com os dados de identificação do cliente e *beacon* previamente enviados.

Os dispositivos *beacons* consistem na terceira componente, os quais, através do envio de sinais via *Bluetooth Low Energy* (BLE), permitem despoletar uma determinada ação na aplicação móvel do dispositivo do cliente.

Esta dissertação documenta o desenvolvimento de toda a solução.

Palavras Chave: *iBeacon*, Sistema de Gestão de Conteúdo, Aplicação móvel, Retalho.

ABSTRACT

The uprising of the *iBeacon* technology in 2014, creates several business opportunities among which is retail vending.

The scope of this work encompasses the implementation and documentation of an *iBeacon* based solution that leverages the retailer's business and its customer's experience.

The solution is composed by three components.

The first one is the *Content Management System* (CMS) which consists in a *Web* platform that is intended to be used for the vendor and provides the overall control of the solution. Moreover, it manages stores and map *beacons* interactively in accordance to their physical distribution. Furthermore, it enables customer management and clustering, allowing targeting campaigns to specific customers and incorporating the *Passbook* system. At last, the CMS provides a module for data analysis composed by 6 sections: General dashboard; Dashboard by campaign; Dashboard by region; Dashboard by customer; Customer pathway and *Heatmap*. These provide ways to analyze: Customer affluence by location and date; TOP sent, open and ignored campaigns; Most frequent customer locations by date; and, among others, the pathway of a customer within the store.

The second component consists in a mobile application developed for the *Android* system which is intended for customer use. Its primary concern is to ensure the compatibility with any kind of beacons that lies within the *iBeacon* standards. Moreover, it also concerns and addresses CMS interaction, getting campaigns regarding the customer and *beacon*'s data sent.

The third component consists in the *beacon* devices which, using *Bluetooth Low Energy* (BLE), triggers actions in the customer's mobile application.

This dissertation documents the development of the solution.

Keywords: *iBeacon*, Content Management System, Mobile App, Retail.

ÍNDICE

RESUMO	vi
ABSTRACT	vii
ÍNDICE	viii
ÍNDICE DE FIGURAS	xi
ÍNDICE DE TABELAS	xv
ABREVIATURAS	xvi
1. INTRODUÇÃO	1
1.1 Enquadramento	1
1.2 Apresentação da Streamline	2
1.3 Interesse e motivação	2
1.4 Objetivos	4
1.5 Estrutura da dissertação	5
2. A INTERNET DAS COISAS	7
2.1 O conceito e sua origem	7
2.2 Exemplos da sua aplicação	7
2.3 Vantagens e desvantagens / limitações	8
2.3.1 Vantagens	8
2.3.2 Desvantagens / limitações	8
2.4 Aplicações da IoT	9
2.5 Funcionamento da IoT	10
2.6 A IoT hoje	11
3. BEACON e iBEACON	13
3.1 O que é um Beacon?	13
3.2 O que é um iBeacon?	13
3.3 Aspetos técnicos	13
3.3.1 Funcionamento de uma solução assente na tecnologia iBeacon	13
3.3.2 Bluetooth Low Energy (BLE)	14
3.3.2.1 O que é?	14
3.3.2.2 Funcionamento de uma comunicação BLE	15
3.3.3 Pacotes de comunicação iBeacon	16
3.3.3.1 Visão detalhada de um pacote BLE	17
3.3.4 Elementos de identificação de um iBeacon	18
3.3.4.1 Grupo 1: iBeacon prefix	19
3.3.4.2 Grupo 2: Proximity UUID, Major e Minor	19
3.3.4.3 Grupo 3: Transmit Power	21
3.3.5 Precisão, atenuação e degradação de um sinal iBeacon	23
3.3.6 Monitoring e Ranging	25

3.3.6.1	Monitoring	26
3.3.6.2	Ranging	26
3.3.7	Constituição ao nível do hardware.....	28
3.3.8	Firmware de um beacon	29
3.3.9	Tempo médio de vida de um beacon	31
3.3.9.1	Variação dos parâmetros <i>Advertising Interval</i> e <i>Transmit Power</i>	31
3.3.10	Influência na descarga de bateria dos dispositivos recetores	31
3.4	Aspectos teóricos.....	33
3.4.1	Razões para o uso da tecnologia iBeacon no retalho	33
3.4.1.1	Casos de estudo na área do retalho e resultados obtidos	34
3.4.2	iBeacon vs Near Field Communication	36
3.4.3	Limitações da tecnologia iBeacon	38
3.4.3.1	Uso obrigatório do Bluetooth.....	38
3.4.3.2	Uso obrigatório de uma aplicação	40
3.4.4	Eddystone Beacon	40
3.4.4.1	Compatibilidade da solução com a tecnologia Eddystone	42
4.	DESENVOLVIMENTO DA SOLUÇÃO	43
4.1	Componentes da solução	43
4.2	Enquadramento técnico.....	44
4.2.1	Tecnologias utilizadas	44
4.2.1.1	HTML	45
4.2.1.2	CSS	45
4.2.1.3	Bootstrap.....	45
4.2.1.4	PHP	45
4.2.1.5	MySQL	46
4.2.1.6	JavaScript.....	46
4.2.1.7	jQuery UI	46
4.2.1.8	AJAX	46
4.2.2	Framework CakePHP	47
4.3	Metodologia de desenvolvimento	49
4.3.1	SCRUM	49
4.4	Arquitetura de desenvolvimento do projeto.....	50
4.4.1	Sistema de controlo de versões - GIT	50
4.4.2	Descrição da arquitetura de desenvolvimento	51
5.	SISTEMA DE GESTÃO DE CONTEÚDOS	53
5.1	Descrição da componente	53
5.2	Requisitos do CMS	53
5.3	Casos de utilização.....	54
5.4	Conceção do CMS	56

5.4.1	Criação da base de dados e integração da framework CakePHP	56
5.4.2	Desenvolvimento do aspetto	57
5.4.3	Explicação das principais funcionalidades.....	59
5.4.3.1	Lojas.....	59
5.4.3.2	Utilizadores	61
5.4.3.3	Beacons	62
5.4.3.4	Regiões de beacons	65
5.4.3.5	Plantas	66
5.4.3.6	Mapeamento de beacons	67
5.4.3.7	Clientes.....	69
5.4.3.8	Grupos de clientes	71
5.4.3.9	Campanhas	73
5.4.3.10	Análise de dados.....	89
5.4.3.11	Exportação de dados.....	112
5.4.3.12	Integração com sistemas externos	113
6.	APLICAÇÃO ANDROID	115
6.1	Descrição da componente	115
6.2	Requisitos da aplicação	115
6.3	Conceção da Aplicação	115
6.3.1	Verificações de serviços ativos	115
6.3.2	Identificação do cliente	116
6.3.3	Deteção de beacons.....	117
6.3.3.1	Como é feita a deteção?.....	117
6.3.3.2	Poupança de bateria.....	117
6.3.3.3	Cálculo da distância.....	118
6.4	Comunicação com o CMS e receção de campanhas	119
6.4.1	Tipo de campanha: Standard.....	121
6.4.1.1	Abertura de campanha.....	122
6.4.2	Tipo de campanha: Passbook.....	123
6.4.2.1	Download do Passe	123
6.4.2.2	Abertura do Passe.....	124
6.4.3	Tipo de campanha: Web	125
6.4.3.1	Abertura de campanhas Web.....	125
6.4.3.2	Sincronização de Eddystone URL.....	126
7.	CONCLUSÕES E TRABALHO FUTURO.....	129
REFERÊNCIAS BIBLIOGRÁFICAS	131	
APÊNDICES.....	139	

ÍNDICE DE FIGURAS

Figura 1.1 - Logotipo da empresa Streamline	2
Figura 1.2 – Dimensões do novo cliente	3
Figura 1.3 – Etiqueta Pinterest numa das lojas Nordstrom.....	3
Figura 2.1 - Data Information Knowledge Wisdom	8
Figura 2.2 – Categorias de IoT	10
Figura 2.3 – Número de dispositivos por pessoa ao longo dos anos	11
Figura 3.1 - Beacon	13
Figura 3.2 - iBeacon	13
Figura 3.3 – Funcionamento geral de uma solução assente na tecnologia iBeacon	14
Figura 3.4 – Pacote Bluetooth Low Energy.....	16
Figura 3.5 – PDU contendo apenas o campo Data	17
Figura 3.6 – Estrutura de um pacote iBeacon.....	17
Figura 3.7 – Pacote iBeacon decomposto segundo os cinco elementos	17
Figura 3.8 – Decomposição de um pacote BLE completo	18
Figura 3.9 – Elementos de identificação de um iBeacon	19
Figura 3.10 – Calibração de um beacon	21
Figura 3.11 – Alteração da precisão em função da atenuação do sinal	23
Figura 3.13 - Dispositivo perto do beacon.....	24
Figura 3.12 – Dispositivo longe do beacon	24
Figura 3.14 – Atenuação de sinal por objetos físicos	24
Figura 3.15 – Atenuação de sinal pelo corpo humano	24
Figura 3.16 – Monitoring region (In / Out)	26
Figura 3.17 – Ranging beacons	27
Figura 3.18 – Formatos existentes de beacons	28
Figura 3.19 – Conteúdo no interior de um beacon	29
Figura 3.20 – Forma de alimentação de um beacon	29
Figura 3.21 – Distância alcançada em função do parâmetro TX Power	30
Figura 3.22 – Consumo de bateria de dispositivos	32
Figura 3.23 – Uso da tecnologia iBeacon na área do Retalho	34
Figura 3.24 – Tecnologia iBeacon.....	37
Figura 3.25 – Tecnologia NFC	37
Figura 3.27 – Poster com informação de zona de Wifi grátis.....	39
Figura 3.26 – Permissão para a aplicação ligar o Bluetooth	39
Figura 3.28 – Dispositivos BLE	39
Figura 4.1 – Componentes da solução desenvolvida	43
Figura 4.2 – Arquitetura de funcionamento da framework CakePHP	48
Figura 4.3 – Funcionamento da metodologia Scrum	49
Figura 4.4 – Arquitetura do sistema GIT	50
Figura 4.5 – Arquitetura de desenvolvimento da solução	52

Figura 5.1 – Diagrama de caso de uso do CMS.....	54
Figura 5.2 – Todas as tabelas existentes na DB	57
Figura 5.4 – Página de login: Versão Smartphone.....	58
Figura 5.3 – Página de login: Versão PC	58
Figura 5.5 – Página de login: Versão Tablet (horizontal).....	58
Figura 5.6 – Página de login: Versão Tablet (vertical)	58
Figura 5.8 – Página principal: Versão Smartphone	58
Figura 5.7 – Página principal: Versão PC.....	58
Figura 5.9 – Página principal: Versão Tablet (vertical).....	59
Figura 5.10 – Página principal: Versão Tablet (horizontal).....	59
Figura 5.11 – API do Google Maps	60
Figura 5.12 – Listagem de lojas.....	60
Figura 5.13 – Relação entre as tabelas users, lojas e roles.....	61
Figura 5.14 – Escolha da norma a usar pelo beacon no CMS.....	63
Figura 5.15 – Parâmetros de difusão dos beacons no CMS	64
Figura 5.16 – Listagem de beacons no CMS	64
Figura 5.17 – Associação de beacons a regiões.....	65
Figura 5.18 – Adicionar uma planta no CMS	66
Figura 5.19 – Tabelas “beacons”, “lojas” e plants	66
Figura 5.20 – Mapeamento de beacons.....	67
Figura 5.21 – Tabela beacons	68
Figura 5.22 – Funcionamento do módulo draggable e resisable	68
Figura 5.23 – Anúncio da aplicação	69
Figura 5.24 – Não foi encontrado nenhum cliente	70
Figura 5.25 – Encontrado cliente	70
Figura 5.26 – Dados de identificação do dispositivo do cliente.....	70
Figura 5.27 – Associação de clientes a grupos	72
Figura 5.28 – Layouts para campanhas do tipo Standard	73
Figura 5.29 - Layouts para campanhas do tipo Web.....	75
Figura 5.30 – Formulário de criação de Conteúdos Web.....	76
Figura 5.31 – Listagem de conteúdos Web.....	77
Figura 5.32 – Listagem de traduções associadas a um conteúdo Web.....	77
Figura 5.33 – Criação de uma campanha: Passo 1	78
Figura 5.34 – Criação de uma campanha: Passo 2	78
Figura 5.35 – Tabela “campanhas_has_beacons”	78
Figura 5.36 – Zonas de proximidade	79
Figura 5.37 – Criação de uma campanha: Passo 3	79
Figura 5.38 – Criação de uma campanha: Passo 4 - Standard	80
Figura 5.39 – Criação de uma campanha: Passo 4 - Passbook	80
Figura 5.40 - Criação de uma campanha: Passo 4 - Web.....	81

Figura 5.41 – Criação de uma campanha: Passo 4 – tabela campanhas	81
Figura 5.42 – Criação de uma campanha: Passo 5 – conteúdo.....	82
Figura 5.43 - Criação de uma campanha: Passo 5 – códigos suportados.....	82
Figura 5.44 - Criação de uma campanha: Passo 5 – Escolha de cor.....	82
Figura 5.45 - Criação de uma campanha: Passo 5 – Notificação push	83
Figura 5.46 - Criação de uma campanha: Passo 5 – Número de envios	83
Figura 5.47 - Criação de uma campanha: Passo 5 – Escolha de conteúdo	83
Figura 5.48 - Criação de uma campanha: Passo 6	84
Figura 5.49 - Criação de uma campanha: Passo 7	85
Figura 5.50 – Layouts disponíveis em campanhas Web.....	89
Figura 5.51 – Registo de contato do cliente com um beacons.....	90
Figura 5.52 – Registo da receção de uma campanha por um cliente	90
Figura 5.53 – Número de clientes que frequentaram a loja	91
Figura 5.54 – Afluência de clientes no último semestre.....	91
Figura 5.55 – Afluência de clientes nos últimos sete dias	91
Figura 5.56 – Taxa de abertura de campanhas	92
Figura 5.57 – Fake URL	92
Figura 5.58 – TOP 10 campanhas abertas	93
Figura 5.59 – TOP 10 campanhas enviadas.....	93
Figura 5.60 – Tipos de campanhas mais clicadas.....	94
Figura 5.61 – Visão total do Dashboard Geral	95
Figura 5.62 – Intervalo de tempo a analisar	95
Figura 5.63 – Número de instâncias de campanhas enviadas, abertas e ignoradas	96
Figura 5.64 – Número de instâncias da campanha enviadas por hora	97
Figura 5.65 – Taxa de abertura de campanhas	97
Figura 5.66 – Alcance da campanha.....	98
Figura 5.67 – Dashboard por Campanha	98
Figura 5.68 – Campanhas recebidas, abertas e ignoradas por cliente.....	99
Figura 5.69 – Número de campanhas recebidas por hora.....	99
Figura 5.70 – Taxa de abertura de campanhas por cliente.....	100
Figura 5.71 – Tipos de campanhas preferidas pelo cliente	101
Figura 5.72 – Dashboard por Cliente.....	101
Figura 5.73 – Total de clientes por região de proximidade	102
Figura 5.74 – Taxa de clientes por proximidade	102
Figura 5.75 – Número de clientes por beacon e proximidade	103
Figura 5.76 – Taxa de clientes por género.....	103
Figura 5.77 – Afluência de clientes por hora.....	104
Figura 5.78 – Dashboard por Região.....	105
Figura 5.79 – Entrada na tabela “percursos_clientes”	106
Figura 5.80 – Entrada na tabela “beacons”	106

Figura 5.81 – Entrada na tabela “plantas”	106
Figura 5.82 – Legenda “Percurso dos clientes”.....	107
Figura 5.83 – Esquema do funcionamento da seção de percurso de utilizadores	108
Figura 5.84 – Exemplo de um percurso efetuado por um cliente	108
Figura 5.85 - Heatmap	110
Figura 5.86 – Número de clientes por beacon	110
Figura 5.87 – Heatmap: Seção completa	111
Figura 5.88 – Processo de exportação de ficheiro CSV	112
Figura 5.89 – Integração do CMS com sistemas externos	113
Figura 6.1 – Verificar se o Bluetooth se encontra ativo.....	116
Figura 6.2 – Identificação do cliente.....	117
Figura 6.3 – Recepção de notificação da campanha	120
Figura 6.4 – Campanha Standard: Layouts.....	121
Figura 6.5 – Esquema de abertura de uma campanha	122
Figura 6.6 – Esquema de download do ficheiro .pkpass.....	124
Figura 6.7 – Opções tendo em conta a instalação da aplicação Passwallet.....	125

ÍNDICE DE TABELAS

Tabela 3.1 – Exemplo de aplicação dos parâmetros iBeacon	20
Tabela 3.2 – Interferência de vários materiais	25
Tabela 3.3 – Estados da função de “Ranging”	27
Tabela 3.4 – Comparação entre as características Monitoring e Ranging	28
Tabela 3.5 – Influência dos parâmetros na duração da bateria de um beacon	31
Tabela 3.6 – iBeacon vs. NFC	38
Tabela 5.1 – Tipos de Passbook	74
Tabela 5.2 – Exemplo de uma entrada na tabela “clientes_campanhas”	96
Tabela II.1 - CMS: Requisito de interface: Página de login	149
Tabela II.2 – CMS: Requisitos de interface: Imagem do CMS	149
Tabela II.3 – CMS: Requisitos de interface: Layout do CMS	150
Tabela II.4 – CMS: Requisitos de Interface: Página inicial.....	150
Tabela II.5 – CMS: Requisitos de interface: Design responsivo	151
Tabela II.6 – CMS: Requisitos de Interface: Módulos	152
Tabela II.7 – CMS: Requisitos de interface: Funcionalidades acessíveis.....	152
Tabela II.8 – CMS: Requisitos de interface: Localização do utilizador	152
Tabela II.9 – CMS: Requisitos não funcionais: Usabilidade	153
Tabela II.10 – CMS: Requisitos não funcionais: Desempenho	153
Tabela II.11 – CMS: Requisitos não funcionais: Portabilidade.....	153
Tabela II.12 - CMS: Requisitos não funcionais: Fiabilidade.....	154
Tabela II.13 – CMS: Requisitos não funcionais: Segurança	154
Tabela II.14 – CMS: Requisitos funcionais: Utilizadores	154
Tabela II.15 - CMS: Requisitos funcionais: Lojas	156
Tabela II.16 – CMS: Requisitos funcionais: Plantas	157
Tabela II.17 – CMS: Requisitos funcionais: Beacons	157
Tabela II.18 – CMS: Requisitos funcionais: Clientes.....	157
Tabela II.19 – CMS: Requisitos funcionais: Grupos	158
Tabela II.20 – CMS: Requisitos funcionais: Campanhas	158
Tabela II.21 – CMS: Requisitos funcionais: Análise de dados	159
Tabela III.1 – Aplicação Android: Requisitos não funcionais: Usabilidade.....	161
Tabela III.2 – Aplicação Android: Requisitos não funcionais: Desempenho	161
Tabela III.3 - Aplicação Android: Requisitos não funcionais: Portabilidade	161
Tabela III.4 – Aplicação Android: Requisitos funcionais: Identificação do cliente	162
Tabela III.5 - Aplicação Android: Requisitos funcionais: Beacons	162
Tabela III.6 – Aplicação Android: Comunicação com o CMS.....	162
Tabela III.7 – Aplicação Android: Campanhas	163

ABREVIATURAS

API – *Application Programming Interface*

AJAX – *Asynchronous Javascript and XML*

BLE – *Bluetooth Low Energy*

CMS – *Content Management System* ou na língua portuguesa “Sistema de Gestão de Conteúdos”

CRC – *Cyclic Redundancy Check*

CSS – *Cascading Style Sheets*

CSV – *Comma-separated values*

DB – *Data Base* ou na língua portuguesa “Base de dados”

DIKW – *Data Information Knowledge Wisdom*

GPL – *General Public License*

GPS – *Global Positioning System*

HTML – *HyperText Markup Language*

IBSG – *Internet Business Solutions Group*

ID – Identificador

IDE – *Integrated Development Environment*

IoT – *Internet of Things* ou na língua portuguesa “Internet das Coisas”

IPv4 – *Internet Protocol version 4*

IPv6 – *Internet Protocol version 6*

JSON – *JavaScript Object Notation*

MVC – *Model-view-controller*

NFC – *Near Field Communication*

OSS – *Open-source Software*

PDU – *Protocol Data Unit*

PHP – *PHP: Hypertext Preprocessor*

RFID – *Radio Frequency Identification*

RSSI – *Received Signal Strength Indication*

SQL – *Structured Query Language*

SSH – *Secure Shell*

SVN – *Apache Subversion*

TCP/IP – *Transmission Control Protocol / Internet Protocol*

UUID – *Universally Unique Identifier*

URL – *Uniform Resource Locator*

WWW – *World Wide Web*

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

XML – *eXtensible Markup Language*

1. INTRODUÇÃO

Neste capítulo introdutório da dissertação especifica-se o enquadramento ao projeto, bem como, interesses e motivações que levaram ao seu desenvolvimento. Apresentam-se ainda os objetivos da solução, e estrutura da dissertação.

1.1. Enquadramento

Este projeto surge no contexto da unidade curricular de Estágio / Projeto Industrial do Mestrado em Informática e Sistemas – Ramo de Tecnologias da Informação e do Conhecimento lecionado pelo Instituto Superior de Engenharia de Coimbra (ISEC) [1] - Instituto Politécnico de Coimbra (IPC) [2], e foi desenvolvido pelo aluno Nuno Alexandre Lemos de Paiva na empresa **Streamline** [3].

A existência, na **Streamline**, de um projeto inovador e assente em tecnologias recentes foi o ponto de partida para o desenvolvimento da solução apresentada. Tendo em vista a implementação numa loja da cidade de Coimbra, a solução descrita nesta dissertação encontra-se em permanente desenvolvimento não sendo um produto finalizado.

Esta dissertação documenta o desenvolvimento da solução implementada, a qual é composta por três componentes.

A primeira componente, descrita no capítulo **5** e designada por Sistema de Gestão de Conteúdos, ou em inglês *Content Management System* (CMS), consiste numa plataforma *Web* responsável por controlar toda a solução. Destinada a ser usada pelo retalhista, possuirá módulos que lhe permite gerir as suas lojas, os seus *beacons* (mapeando-os de forma interativa numa área física) e os seus clientes. Possuirá ainda um módulo que permite ao retalhista o envio e gestão de campanhas com integração com o sistema *Passbook*. Por fim, através de um módulo de análise de dados e diversos *dashboards*, o retalhista poderá retirar conhecimento da interação dos seus clientes com a solução, como por exemplo: Afluência de clientes por local e data; TOP de campanhas enviadas, abertas, ignoradas; Locais mais frequentados, por cliente e data; Percurso efetuado por cliente numa loja.

A segunda componente, descrita no capítulo **6**, consiste numa aplicação móvel desenvolvida para o sistema *Android*. O objetivo do seu desenvolvimento consistiu em receber campanhas (guardando dados da sua interação) de acordo com os dados de identificação do cliente e *beacon* com o qual contactou, garantindo a compatibilidade com qualquer tipo de *beacon*.

Os dispositivos *beacons*, descritos no capítulo **3**, consistem na terceira componente. Estes, através do envio de sinais via *Bluetooth Low Energy* (BLE), permitem despoletar uma determinada ação na aplicação móvel do dispositivo do cliente com o qual contactaram.

1.2. Apresentação da Streamline

A Streamline é uma empresa sediada em Coimbra e fundada em 2009 [3] (logotipo na Figura 1.1). Consiste numa jovem empresa de Engenharia Informática com a missão de simplificar e tornar mais eficientes processos de sistemas e tecnologias de informação e também as áreas de gestão de redes, de comunicações ou de virtualização de ambientes críticos e heterogéneos.

No que diz respeito à área de desenvolvimento produz sistemas *Web* (*Design, Web Sites* e sistemas de informação associados) e *Mobile*, dos quais se destacam as seguintes áreas de aplicação: Portais Institucionais Dinâmicos; Portais de Colaborador; Sistemas Inovadores Multi-plataforma, em áreas emergentes; Sistema de Informação para Administração de Sistemas e Sistemas Conexos; Comércio Eletrónico; *Indoor Location* e Plataformas de Gestão Empresarial.



Figura 1.1 - Logotipo da empresa **Streamline**

1.3. Interesse e motivação

O panorama do retalho é atualmente diferente daquele que se encontrava há anos atrás. O cenário desfavorável macroeconómico trouxe novas exigências na gestão de recursos. No que diz respeito à oferta, os retalhistas foram obrigados a otimizar os seus investimentos de produção, logísticos e de marketing [4]. Relativamente à procura, com um poder de compra diminuído o cliente tornou-se mais racional, ponderando a suas decisões de compra.

O estudo “*Energizing Global Growth: Understanding the changing consumer*” apresentado pela Accenture [5] “conclui que o cliente se encontra a mudar”. Indica ainda que essas mudanças se dividem em quatro categorias: económicas, demográficas, tecnológicas e ideológicas. O cliente, consciente do poder que a tecnologia lhe dá, tem cada vez mais uma participação ativa e informada exigindo maior controlo, mais informação, maior comodidade e rapidez nos serviços. Por ser mais informado, assume cada vez menos compromissos a longo prazo, estando sempre disponível a mudar os seus hábitos. O mesmo estudo conclui ainda que todas estas alterações levam a um cliente social, independente e pró-ativo. A Figura 1.2 apresenta as 10 principais dimensões do novo cliente segundo [5].

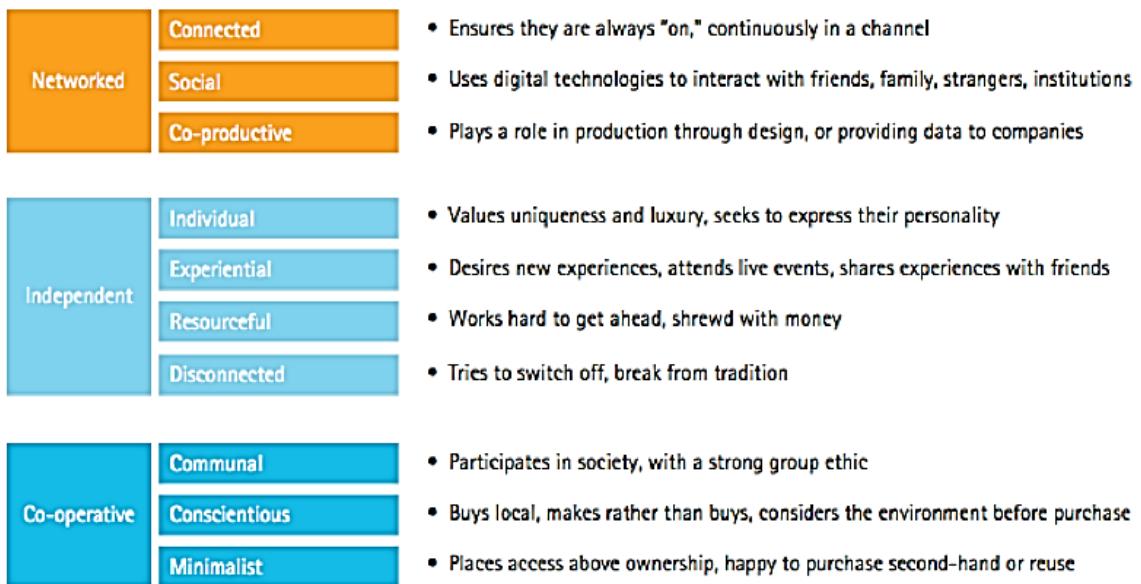


Figura 1.2 – Dimensões do novo cliente

O desafio atual da área do retalho consiste em contemplar estas mudanças e adaptá-las à sua estratégia de negócio.

Brian Walker [6] indica que o mundo do retalho está em constante evolução estabelecendo cinco desafios a ter em conta:

1. Foco no cliente;
2. Dar importância às redes sociais;
3. Conciliar *online* e *offline*;
4. Foco no processo de compra;
5. Customização da oferta.

O primeiro ponto indica que o cliente deve ser colocado no centro do negócio, tentando envolvê-lo, através dos mais diversos meios, de forma a garantir a presença em todas as fases do ciclo de compra.

O segundo ponto salienta a importância das redes sociais, pois é nestas onde o cliente expressa mais espontaneamente os seus interesses. A empresa *Nordstrom* integrou nas suas lojas a rede social *Pinterest*. Bryan Galipeau [7], gestor de marketing da *Nordstrom* afirma que: “Pinterest é a maior lista de desejos do mundo, por isso também se encaixa no nosso objetivo: fazer com que os nossos produtos apareçam na lista de desejos dos nossos clientes.” A primeira medida foi colocar nos artigos mais populares no *Pinterest* uma etiqueta informando o cliente que o produto assinalado na loja é bastante “desejado” (Figura 1.3).



Elsa Gomes [4] sustenta o terceiro ponto afirmando que “o mundo *offline*” deve integrar o “mundo *online*”, ou seja, a loja física deverá estar presente em diversos dispositivos e plataformas digitais aproveitando-as para criar experiências de comunicação e interatividade com o cliente.

Figura 1.3 – Etiqueta
Pinterest numa das
lojas Nordstrom

O quarto ponto relaciona-se com questões do tipo: “O que levou o cliente à loja?”, “Como chegou a uma determinada prateleira e porque comprou aquele produto?”, “Estará a fornecer ao cliente todas as informações relevantes no momento que decide procurar a sua loja?”.

Sridhar Ramaswamy, em “*Shopping Then and Now: Five Ways Retail Has Changed and How Businesses Can Adapt*” [8], fundamenta o quinto ponto. Afirma que a técnica de marketing “one size fits all” já não resulta. O cliente atual espera uma relação personalizada que o faça sentir compreendido e exclusivo. Assim, informações, campanhas de descontos ou outras formas de comunicação devem ter sempre em conta o cliente final sendo personalizadas para cada caso.

O surgimento da tecnologia *iBeacon* vem estimular em particular os pontos quatro (foco no processo de compra) e cinco (customização da oferta). No capítulo 3 (secção 3.4.1) encontram-se as razões que cimentam a tecnologia *iBeacon* na área do retalho, enunciando casos de estudo e resultados obtidos.

1.4. Objetivos

A solução desenvolvida tem como principais objetivos melhorar a experiência da comunicação em tempo real com o cliente, bem como trazer ao retalhista um conjunto de dados de forma a traçar o perfil de cada cliente (o que possibilitará adaptar mais eficazmente as suas campanhas), potenciando o seu crescimento económico (de acordo com as tendências de mercado definidas na secção 1.3).

Numa primeira fase, a **Streamline** pretende desenvolver uma solução adaptável a qualquer retalhista. Deseja-se que a solução possa ser utilizada por várias lojas em simultâneo, com dados completamente independentes entre si.

No que diz respeito à componente CMS, cada utilizador deve visualizar os dados respeitantes à sua loja associada, funcionando de forma independente para cada utilizador.

No que diz respeito à aplicação *Android*, pretende-se que seja criado um piloto que permita a comunicação com o CMS, com a respetiva receção de campanhas direcionadas, envio de dados de identificação do cliente e respetivas ações sobre as campanhas recebidas.

A adaptação da solução a cada loja passará pelo ajuste da aplicação móvel às suas necessidades, acrescentando os requisitos desejados. Toda a restante arquitetura funcionará de igual modo para todas as lojas, sem que seja necessário efetuar nenhuma alteração.

1.5. Estrutura da dissertação

A dissertação apresentada encontra-se dividida em sete capítulos.

Este primeiro capítulo tem como objetivo fazer um enquadramento teórico ao tema, bem como apresentar motivação ao uso da tecnologia *iBeacon* no retalho. Definem-se também um conjunto de objetivos gerais que o projeto deve alcançar, bem como resultados que devem ser obtidos.

O capítulo dois tem em vista abordar o tema: A Internet das Coisas, ou em inglês *Internet of Things* (IoT). Este destina-se a efetuar uma ligação ao tema e à tecnologia base, no qual assenta toda a solução.

O terceiro capítulo apresenta a tecnologia *iBeacon* explicitando o seu modo de funcionamento, limitações, tecnologias semelhantes e razões que sustentam o seu uso na área do retalho.

O capítulo quatro assenta na forma como todo o projeto foi desenvolvido. Destina-se a apresentar uma descrição do projeto, efetuando um enquadramento técnico (mostrando tecnologias, *frameworks* e ferramentas utilizadas). Apresentará por fim a metodologia de desenvolvimento de *software*, bem como a arquitetura de desenvolvimento usada.

O capítulo cinco contém uma explicação teórica e técnica das principais funcionalidades do CMS.

A aplicação móvel, desenvolvida para o sistema *Android*, será abordada no capítulo seis, incidindo sobre a forma como interage com os *beacons* e com o CMS.

O capítulo sete apresenta as conclusões ao projeto desenvolvido, bem como o trabalho que se sucede para a respetiva aplicação num ambiente real.

No final existem quadro apêndices: no apêndice I encontram-se outras áreas de atuação para além do retalho, e diversos casos de uso da tecnologia *iBeacon*; no apêndice II e III apresentam-se, de forma detalhada, os requisitos do CMS e da aplicação *Android* respetivamente; no apêndice IV mostra-se o diagrama de atividades da aplicação *Android* e no apêndice V encontra-se o diagrama da base de dados, ou em inglês *data base* (DB), do CMS.

2. A INTERNET DAS COISAS

Este capítulo tem como objetivo fornecer um conjunto de conceitos acerca da Internet das Coisas, ou em inglês *Internet of Things* (IoT), que sustenta a tecnologia usada. Incidirá sobre vantagens / desvantagens, aplicações e seu funcionamento.

2.1. O conceito e sua origem

A IoT é um termo usado para designar a conectividade entre vários objetos do dia-a-dia, desde eletrodomésticos (dentro de casa) a equipamentos dispersos por vários locais de uma cidade. Tecnicamente a IoT consiste numa rede de objetos físicos, criada por sensores, permitindo que esses objetos obtenham dados no local em que se encontram inseridos e efetuem a troca desses mesmos dados. Todos esses objetos podem ser acedidos e controlados de forma remota podendo alterar o estado ou modificar o seu comportamento na interação com o meio onde se encontram inseridos [9].

Em 1991 iniciou-se a discussão sobre a ligação de objetos, quando a rede TCP/IP e a *Internet* se começou a tornar acessível. Bill Joy, cofundador da *Sun Microsystems*, foi o cérebro por de trás da ideia de efetuar a ligação entre várias redes e dispositivos [10].

Em 1999, Kevin Ashton, pertencente ao MIT, propôs o termo “Internet das Coisas”. Após dez anos de estudo e projetos nos quais esteve envolvido, escreveu o artigo “A Coisa da Internet das Coisas” para o *RFID Journal* [11], tornando-se na rampa de lançamento até aos dias de hoje.

Segundo Ashton, a falta de tempo na rotina das pessoas fará com que necessitem de se ligar à *Internet* de várias formas. Com a tecnologia em constante evolução, será possível acumular diversos dados que poderão servir para otimizar e economizar recursos naturais ou energéticos (no caso do ambiente), ou mesmo melhorar as condições de vida em termos de saúde e bem-estar da população [11].

2.2. Exemplos da sua aplicação

Por exemplo, “*Numa Terça-Feira está a conduzir até casa, voltando do trabalho. Um sinal no ecrã do carro informa-o que deve passar no supermercado para comprar leite.*

O alerta anterior foi enviado pela central de gestão de casa que se encontra ligada ao frigorífico, detetando que não existem mais embalagens de leite. Esta central está por sua vez ligada ao GPS do carro, que localiza um supermercado a caminho de casa”.

O cenário anterior mostra uma rede de objetos interligados através da *Internet*, possibilitando a troca de informações entre si de modo a transformar dados em informação útil para o utilizador.

A IoT não obriga a que haja um conjunto de objetos ligados. Um único objeto ligado à *Internet* com determinados sensores, pode ser parte integrante da IoT. Um exemplo disso são as pulseiras usadas no desporto que permitem medir a pulsação, distância e velocidade, recorrendo a sensores, como por exemplo, acelerómetro. Usando uma ligação à *Internet* são

armazenados num servidor central onde posteriormente, após serem analisados, se transformam em conhecimento e informação útil. Toda a recolha, análise e amostragem de dados é feita de forma autónoma pelo objeto, fundamentando o conceito de IoT [12].

2.3. Vantagens e desvantagens / limitações

Esta mudança de paradigma no dia-a-dia das pessoas trás um conjunto de vantagens. Existem, no entanto, limitações em algumas tecnologias usadas na IoT.

2.3.1. Vantagens

A ligação de objetos permite uma enorme troca de dados possibilitando, através da sua análise, alcançar o topo da pirâmide (sabedoria) obtida pelo modelo *Data Information Knowledge Wisdom* (DIKW), apresentado na Figura 2.1.



Figura 2.1 - Data Information Knowledge Wisdom

Esta ligação produz benefícios para a sociedade, possibilitando um maior controlo e compreensão de como os sistemas interagem e, em última análise, proporcionar uma melhor qualidade de vida de toda a população [13].

Uma vasta forma de recolha de dados autónoma e posterior análise pode ainda resultar nos seguintes benefícios:

- Novas eficiências operacionais;
- Tomada de decisão mais rápida e melhor;
- Inteligência de controlo distribuída;
- Força de trabalho otimizada.

2.3.2. Desvantagens / limitações

Como principais desvantagens têm-se questões como a privacidade, segurança e confidencialidade dos dados transmitidos entre as diversas componentes [9].

Como limitações existem as seguintes:

- Limitação de endereçamento de objetos, pois apenas um pequeno conjunto de objetos faz uso do *Internet Protocol version 6* (IPv6), mantendo-se maioritariamente todos os restantes a usar o endereçamento *Internet Protocol version 4* (IPv4) que já atingiu o seu limite;
- Tempo de bateria dos sensores, uma vez que efetuar a troca de baterias em milhares de sensores, distribuídos por diversas zonas geográficas não é viável. É necessário que a tecnologia evolua, possibilitando que os objetos / sensores se tornem autossustentáveis. Uma das tecnologias mais promissoras neste campo são os nanogeradores;
- O facto de ser um conceito recente leva à falta de *standards*. É da responsabilidade dos governos e entidades reguladoras a criação dos mesmos, de forma a uniformizar todas as tecnologias possibilitando uma maior integração e uniformidade entre todos os objetos pertencentes à IoT.

2.4. Aplicações da IoT

A IoT pode ser aplicada em diversas áreas das quais se destacam as seguintes [9]:

Retalho

Por exemplo, pode ser substituído o código e barras por *Tags*, possibilitando a localização exata de cada produto. Com o surgimento da tecnologia *iBeacon* é possível, por exemplo, identificar o percurso efetuado por um cliente na loja.

Logística

A reposição de *stock* poderá usar este paradigma evitando que a verificação seja feita manualmente. Através da IoT, com rotinas informáticas desencadeadas por determinados sensores, é possível enviar notificações com uma previsão de rutura de *stock* e solicitar automaticamente ao fornecedor uma encomenda de um número específico de produtos, sem que haja intervenção humana.

Industria Farmacêutica

Fazendo uso de *Tags* no interior das embalagens dos medicamentos é possível, por exemplo, indicar ao paciente os efeitos secundários e a dose recomendada do medicamento a tomar.

Domótica

Fazendo uso de sensores é possível gerir uma casa através da *Internet*. Exemplo de operações: ligar / desligar o aquecimento, ligar / desligar o alarme, aquecer a comida 10 minutos antes de chegar a casa, entre outros.

Transportes

Poderá vir a ser possível que os automóveis obtenham informações baseadas na sua localização, permitindo uma maior eficiência nas viagens.

Design e Marketing de Produtos

Os sensores podem reportar exatamente onde, quando, e como um determinado produto é usado, sendo uma mais-valia em processos de *design* e marketing. O processo de recolha de dados em tempo real pode ter um custo menor, ser mais rápido, e mais preciso em comparação com pesquisas de mercado.

2.5. Funcionamento da IoT

O funcionamento da IoT é categorizado em oito tópicos [12], descritos seguidamente e ilustrados na Figura 2.2:

- **Comunicação:** troca dados entre os dispositivos;
- **Sensores:** utilizados na captura e representação do mundo físico no mundo digital;
- **Atuadores:** executam ações no mundo físico desencadeadas no mundo digital;
- **Armazenamento:** permite guardar os dados recebidos de todos os dispositivos para análise futura;
- **Dispositivos:** usados pelo utilizador na interação com o meio físico;
- **Processamento:** análise e transformação de dados recolhidos pelos sensores e armazenados em repositório central. São usadas técnicas de *data mining* para transformação de dados em conhecimento;
- **Localização:** Localização no mundo físico do sensor ou objeto de interação com o utilizador;
- **Identificação:** Uso de um ID único no mundo digital que permite identificar uma determinada ação no mundo físico.

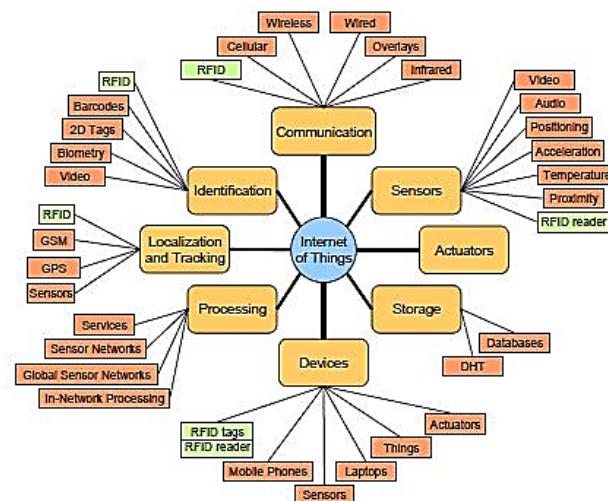


Figura 2.2 – Categorias de IoT

2.6. A IoT hoje

De acordo com o *Cisco Internet Business Solutions Group* (Cisco IBSG) [14], a IoT inicia-se oficialmente no momento em que foram ligados à *Internet* mais "coisas" do que pessoas.

Em 2003, havia aproximadamente 6,3 bilhões de pessoas no planeta e 500 milhões de dispositivos ligados à *Internet*. Ao dividir o número de dispositivos pela população mundial, obtém-se menos de um (0,08) objeto por pessoa. Tendo por base a definição apresentada pela *Cisco IBSG*, a IoT ainda não existia em 2003, pois o número de dispositivos era relativamente pequeno face à população mundial.

Com o crescimento exponencial de *smartphones* e *tablets* o número de dispositivos ligados à *Internet* em 2010 era cerca de 12,5 bilhões sendo a população mundial cerca de 6,8 bilhões. Assim, o número de dispositivos ligados por pessoa tornou-se superior a 1 (exatamente 1,84) pela primeira vez na história [14] (Figura 2.3).

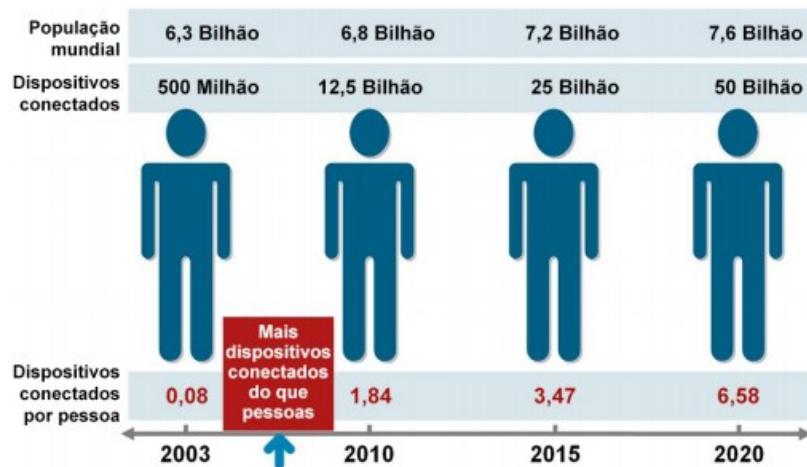


Figura 2.3 – Número de dispositivos por pessoa ao longo dos anos

O *Cisco IBSG* prevê que haja 25 bilhões de dispositivos ligados à *Internet* até 2015 e cerca de 50 bilhões até 2020. Ao efetuar uma redução da amostra da população tendo em conta apenas pessoas ligadas à *Internet* (cerca de 2 bilhões segundo o *Cisco IBSG*), o número de dispositivos ligados por pessoa aumenta para 6,25 em 2010, em vez de 1,84.

A *International Data Corporation*, na análise sobre "Universo Digital em 2020", informa que: "entre 2005 e 2020, o universo digital vai crescer por um fator de 300 vezes, de 130 exabytes a 40 trilhões de gigabytes (mais de 5.200 gigabytes por pessoa em 2020). A partir de agora até 2020, o universo digital será cerca de duas vezes maior a cada dois anos" [15].

Prevê-se assim um crescimento exponencial da IoT a todos os níveis, com enorme impacto não só na população, mas também em toda a economia.

3. BEACON e iBEACON

Este capítulo aborda o conceito de *Beacon* e *iBeacon* usado na solução desenvolvida. Descreve o seu funcionamento, tecnologia, elementos de identificação e sua constituição ao nível do *hardware*. Enumera ainda razões para o seu uso na área do retalho, limitações e tecnologias equivalentes.

3.1. O que é um Beacon?

Um *Beacon* consiste num dispositivo (Figura 3.1) que permite a emissão de dados com uma determinada frequência através da tecnologia *Bluetooth Low Energy* (BLE). Os dados emitidos são apenas do tipo: “Olá, estou aqui. Este é o meu ID” [16]. Através da tecnologia BLE outros dispositivos (por exemplo, *smartphones*) podem ler esses dados sem que seja necessário efetuar um emparelhamento entre os dois dispositivos.



Figura 3.1 - Beacon

3.2. O que é um iBeacon?

iBeacon não consiste num dispositivo físico. *iBeacon* consiste num protocolo (Figura 3.2) desenvolvido pela Apple e apresentado na *Apple Worldwide Developers Conference* em 2013 [16]. Este permite a todo o tipo de *smartphones* e *tablets* (*Android* e *iOS*) desencadear determinadas ações (secção 3.3.6) quando se encontrarem na proximidade de um *beacon*. Estabelece ainda um conjunto de dados que cada *beacon* deve enviar numa comunicação BLE (secção 3.3.4).

Ao apresentar o protocolo *iBeacon* a Apple continuou a designar *beacons* por *iBeacons* o que levou a que todo o mercado aderisse à nomenclatura ligando, desta forma, todos os dispositivos *beacons* à marca, desde que respeitem a norma *iBeacon*.



Figura 3.2 - iBeacon

3.3. Aspetos técnicos

Nesta secção será abordada a tecnologia *iBeacon* explicando o seu funcionamento, arquitetura, pacotes de comunicação e a constituição de um *beacon* ao nível do *hardware*.

3.3.1. Funcionamento de uma solução assente na tecnologia iBeacon

Quando um dispositivo móvel deteta um sinal enviado por um *beacon*, este interpreta os dados recebidos, com base nesses dados calcula a distância a que se encontra do *beacon* e desencadeia uma determinada ação. Essa ação pode tomar duas vertentes: ativa ou passiva.

A ação passiva consiste em armazenar (em memória local ou numa DB remota) que ocorreu uma comunicação com um determinado *beacon*. Por exemplo, o dispositivo com o identificador 1234 aproximou-se do *beacon* com o identificador 5678. Na prática, significa: um cliente passou às X horas por um determinado local.

Uma ação ativa acontece quando, o fato de existir uma comunicação entre um dispositivo e um *beacon*, desencadeia uma atividade no dispositivo do utilizador. Receber uma notificação, mudar o estado do sistema, receber um cupão de desconto, iniciar alguma ação numa aplicação do dispositivo, são exemplos de ações ativas que podem ocorrer da comunicação entre um dispositivo e um *beacon* [17]. Por exemplo, um cliente pode receber uma notificação indicando que “um determinado produto possui atualmente um desconto de 30%”.

De um modo geral, o funcionamento de uma solução assente na tecnologia *iBeacon* é representado em quatro passos enumerados seguidamente e ilustrados na Figura 3.3:

1. Um *beacon* transmite a sua identificação para o dispositivo ao seu alcance, por exemplo: 1234;
2. A aplicação que recebeu a identificação do *beacon*, envia-a para um servidor, traduzindo esse código em informação útil, tal como: 1234 = corredor de venda de produtos alimentares;
3. O servidor verifica se há alguma ação a ser tomada quando essa comunicação existe. Se houver, envia essa informação para a aplicação do dispositivo.
4. A aplicação desencadeia uma ação face à informação recebida do servidor. Por exemplo, “aproveite o desconto de 30% neste produto, só essa semana!”.

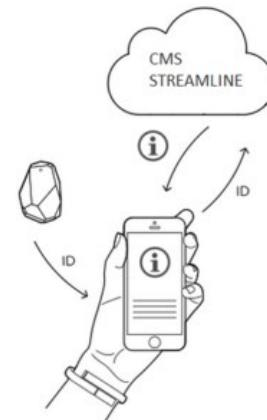


Figura 3.3 – Funcionamento geral de uma solução assente na tecnologia *iBeacon*

3.3.2. Bluetooth Low Energy (BLE)

Esta secção aborda a tecnologia BLE explicando a forma como os dados são transmitidos, dado a tecnologia *iBeacon* fazer uso desta para troca de informação.

3.3.2.1. O que é?

A tecnologia BLE, consiste numa parte da especificação da tecnologia *Bluetooth* 4.0 lançada em 2010. O seu desenvolvimento foi iniciado pela Nokia em 2006 com o nome *Wibree*, acabando a tecnologia *Bluetooth* e BLE por se fundir [18]. Apesar dessa fusão, dispositivos que suportem *Bluetooth* podem não suportar BLE existindo assim atualmente três tipos de dispositivos:

- ***Bluetooth*:** apenas suportam o modo clássico *Bluetooth*;

- **Bluetooth Smart Ready:** suportam tanto o modo clássico como o BLE;
- **Bluetooth Smart:** apenas suportam BLE.

Os novos *smartphones*, *tablets* são compatíveis com todo o protocolo *Bluetooth* 4.0 (fazendo parte dos dispositivos do tipo *Smart Ready*) conseguindo obter informações de dispositivos emitindo em modo clássico ou BLE. No que diz respeito à tecnologia *iBeacon* apenas suporta BLE fazendo parte dos tipos de dispositivos *Smart*.

O objetivo principal do BLE consiste na redução do consumo de energia através do envio de pequenos pacotes de dados [19] [20]. Por exemplo, alguns *beacons* têm capacidade de transmitir sinal durante mais de 2 anos com a mesma bateria. Este é um aspeto de enorme importância uma vez que grande parte dos *beacons* não permitem a substituição da bateria devido às suas condições de conceção e requisitos suportados (por exemplo, resistentes à água).

Tanto o *Bluetooth* como o BLE usam a mesma gama no espectro, (2.4 GHz – 2.4835 GHz) estando exposto a interferências e atenuação de sinal (abordado em detalhe na secção 3.3.5).

No que diz respeito ao alcance do sinal, tanto o BLE como o *Bluetooth* conseguem emitir até 100 metros dependendo do espaço evolvente.

3.3.2.2. Funcionamento de uma comunicação BLE

Uma comunicação BLE é composta por duas partes: anúncio (*advertising*) e ligação (*connecting*).

3.3.2.2.1. Anúncio (Advertising)

O anúncio consiste no envio de dados apenas do emissor para o receptor. Assim, os dispositivos que desejam ser descobertos por outros podem emitir pacotes de dados entre diversos intervalos de tempo cujos valores se situam entre os 20ms e os 10 segundos. Quanto mais curto for o intervalo de tempo de transmissão de pacotes menor será a duração da bateria. Por outro lado, intervalos mais curtos de anúncio permitem que o dispositivo seja descoberto mais rapidamente. Segundo a norma *iBeacon* o tempo ótimo de emissão de “anúncios” é de 100ms.

Cada pacote enviado na fase de “anúncio” pode ter um tamanho máximo de 47 bytes, sendo composto pelos seguintes dados [21]:

- 1 byte – *Preamble*;
- 4 bytes – *Access Address*;
- 2 a 39 bytes – *Protocol Data Unit* (PDU);
- 3 bytes – *Cyclic Redundancy Check* (CRC).

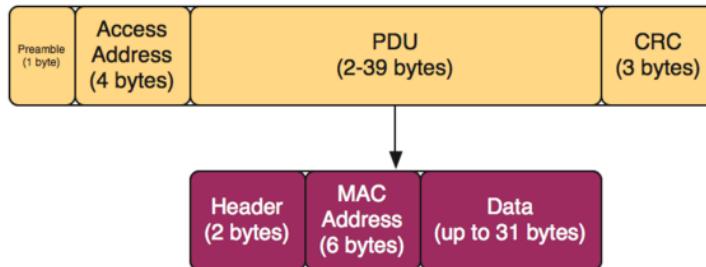


Figura 3.4 – Pacote Bluetooth Low Energy

O PDU, tal como apresentado na Figura 3.4, possui o seu próprio cabeçalho (*Header*) composto por 2 *bytes* contendo o tamanho de carga útil que se encontra no pacote, bem como, indicadores do dispositivo, por exemplo, se suporta a fase de ligação. É composto ainda pelo *Mac Address* do dispositivo (6 *bytes*) que se encontra a emitir o sinal. Por fim, o PDU reserva até 31 *bytes* (no campo *Data*) para armazenar todo o tipo de informação que se deseja, sendo a única com tamanho variável no pacote.

Se um dispositivo foi descoberto, significa que enviou um pacote do tipo “anúncio”. O dispositivo que o captou pode efetuar a leitura dos dados enviados pelo emissor, não necessitando de progredir para a fase de ligação. No caso de o dispositivo receptor desejar enviar dados para o emissor é necessário a segunda fase de ligação.

Por exemplo, se se desejar receber leituras de temperatura de um termostato apenas é necessário o uso da fase de “anúncio”. No caso de se desejar definir uma temperatura no termostato é necessário que haja a fase de “ligação”, envolvendo um processo de emparelhamento entre os dois dispositivos. Durante essa fase ocorre uma troca de chaves (privada e pública) de modo a garantir uma relação de confiança e saber em ligações futuras que o dispositivo com o qual emparelhou é um dispositivo fidedigno.

Clinton Huges aborda a fase de “ligação” de comunicações BLE [18]. Dado a tecnologia *iBeacon* não fazer uso dessa fase, não será abordada nesta dissertação.

3.3.3. Pacotes de comunicação iBeacon

De forma a entender o conteúdo existente num pacote do tipo *iBeacon*, foi intercetado um pacote BLE recorrendo aos seguintes utilitários:

- Um receptor *Bluetooth*;
- *Software scanner Bluetooth*:
 - **Windows**: Wireshark [22];
 - **MacOS**: Xcode [23].

Apresenta-se de seguida o conteúdo do pacote *iBeacon* capturado (Figura 3.5). Apenas será mostrado o campo *Data* não sendo considerados os campos *Header* e *Mac Address* do elemento PDU.

```
1 02 01 06 1A FF 4C 00 02 15 B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D 00 49 00 0A C5
```

Figura 3.5 – PDU contendo apenas o campo Data

O campo *Data* consiste no parâmetro responsável por conter a informação importante e variável numa comunicação BLE sendo, no caso de um pacote *iBeacon*, composto obrigatoriamente por cinco parâmetros: *iBeacon prefix*, *Proximity Universally Unique Identifier* (UUID), *Major*, *Minor* e *Transmit power*.

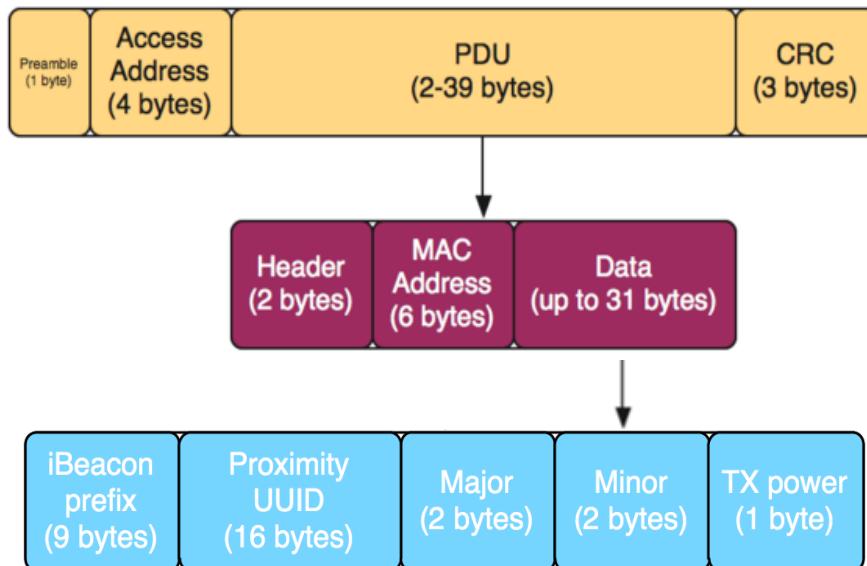


Figura 3.6 – Estrutura de um pacote iBeacon

Decompondo o campo *Data* do parâmetro PDU segundo os tamanhos em *bytes* apresentados na Figura 3.6 tem-se os valores representados na Figura 3.7.

```
1 02 01 06 1A FF 4C 00 02 15: iBeacon prefix (fixed except for 3rd byte - flags)
2 B9 40 7F 30 F5 F8 46 6E AF F9 25 55 6B 57 FE 6D: proximity UUID (here: Estimote's fixed UUID)
3 00 49: major
4 00 0A: minor
5 C5: 2's complement of measured TX power
```

Figura 3.7 – Pacote iBeacon decomposto segundo os cinco elementos

3.3.3.1. Visão detalhada de um pacote BLE

Importa analisar todo o pacote BLE de forma a visualizar qual a posição do pacote *iBeacon* no mesmo.

```
d6 be 89 8e 40 24 05 a2 17 6e 3d 71 02 01 1a 1a ff 4c 00 02 15 e2 c5 6d b5 df fb 48
d2 b0 60 d0 f5 a7 10 96 e0 00 00 00 00 c5 52 ab 8d 38 a5
```

Decompondo o anterior pacote segundo a estrutura de um pacote BLE, têm-se o apresentado no esquema da Figura 3.8.

Pacote BLE

```

d6 be 89 8e # Access address for advertising data (this is always the same fixed
value)
40 # Advertising Channel PDU Header byte 0. Contains: (type = 0), (tx add = 1),
(rx add = 0)
24 # Advertising Channel PDU Header byte 1. Contains: (length = total bytes of
the advertising payload + 6 bytes for the BLE mac address.)
05 a2 17 6e 3d 71 # Bluetooth Mac address
02 01 1a 1a ff 4c 00 02 15 e2 c5 6d b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 00 00
00 00 c5 # Bluetooth advertisement
52 ab 8d 38 a5 # checksum

```

1 → 02 # Number of bytes that follow in first AD structure
 01 # Flags AD type
 1A # Flags value 0x1A = 000011010
 bit 0 (OFF) LE Limited Discoverable Mode
 bit 1 (ON) LE General Discoverable Mode
 bit 2 (OFF) BR/EDR Not Supported
 bit 3 (ON) Simultaneous LE and BR/EDR to Same Device Capable (controller)
 bit 4 (ON) Simultaneous LE and BR/EDR to Same Device Capable (Host)
 1A # Number of bytes that follow in second (and last) AD structure
 FF # Manufacturer specific data AD type
 4C 00 # Company identifier code (0x004C == Apple)
 02 # Byte 0 of iBeacon advertisement indicator
 15 # Byte 1 of iBeacon advertisement indicator
 2 → e2 c5 6d b5 df fb 48 d2 b0 60 d0 f5 a7 10 96 e0 # iBeacon proximity uuid
 3 → 00 00 # major
 4 → 00 00 # minor
 5 → c5 # The 2's complement of the calibrated Tx Power

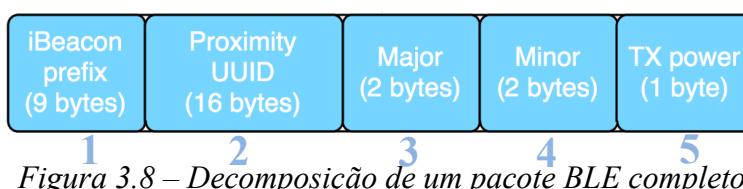


Figura 3.8 – Decomposição de um pacote BLE completo

3.3.4. Elementos de identificação de um iBeacon

Um pacote BLE do tipo “anúncio” é considerado *iBeacon* se no interior do campo *Data* do elemento PDU existirem os seguintes parâmetros [24] [25]:

- *iBeacon prefix*;
- *Proximity UUID*;
- *Major*;
- *Minor*;
- *TX Power*.

Dado a importância que cada elemento ocupa no modo de funcionamento de um *iBeacon*, é possível agrupá-los em três componentes distintas, destacadas na Figura 3.9 e abordados nas próximas secções.

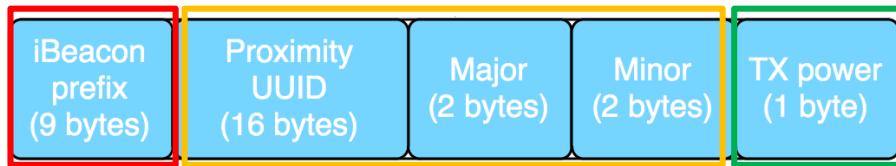


Figura 3.9 – Elementos de identificação de um *iBeacon*

3.3.4.1. Grupo 1: iBeacon prefix

O elemento *iBeacon prefix* consiste no parâmetro de menor importância contendo a seguinte informação:

- *Flags* de identificação do tipo de anúncio, por exemplo:
 - bit 0 (OFF) LE Limited Discoverable Mode
 - bit 1 (ON) LE General Discoverable Mode
- Número de *bytes* totais do pacote;
- Identificação do fabricante do *beacon*.

Este contém informações que permitem ao dispositivo receptor identificar o conteúdo que segue no restante pacote.

3.3.4.2. Grupo 2: Proximity UUID, Major e Minor

Os elementos *Proximity UUID*, *Major* e *Minor* permitem ao dispositivo receptor identificar o *beacon* que enviou o pacote. Estes encontram-se organizados hierarquicamente, possibilitando a identificação desde uma empresa que faz uso de determinados *beacons* (*Proximity UUID*) a um *beacon* específico (*Minor*). Dependendo da granularidade exigida em cada caso podem ser usados ou não todos os elementos.

Cada *beacon* contém um determinado valor por omissão para cada um dos elementos. A sua alteração é possível recorrendo a aplicações de gestão de *beacons*, disponíveis para *iOS* e *Android*, que fazendo uso da fase de ligação (de uma comunicação BLE) permitem o envio de novos dados.

Proximity UUID

Consiste num identificador de 16 *bytes* que pode ser usado para distinguir *beacons* entre diferentes organizações. Todos os *beacons* pertencentes à mesma organização devem possuir o mesmo UUID. No caso de uma cadeia de hipermercados possuir várias lojas, todos os *beacons*, independentemente da localização, devem possuir o mesmo UUID. Assim, independentemente

do local, a aplicação que receber os dados de um *beacon* consegue identificar a cadeia de hipermercados em que o cliente se encontra.

Major

Consiste num identificador de 2 *bytes* usado para agrupar um conjunto de *beacons*. Por exemplo, todos os *beacons* existentes dentro de uma determinada loja devem possuir o mesmo valor *Major*. Assim, a aplicação que comunicar com o *beacon* consegue identificar em que loja se encontra o cliente.

Minor

Consiste num identificador de 2 *bytes* usado para identificar cada *beacon*, devendo possuir um valor único em cada loja (para o mesmo valor de *Major*). Assim, é possível identificar exatamente em que local se encontra um determinado cliente.

Tabela 3.1 – Exemplo de aplicação dos parâmetros iBeacon

Localização		Coimbra	Lisboa	Porto	
UUID		D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C			
Major		1	2	3	
Minor	Roupa	10	10	10	
	Restauração	20	20	20	
	Limpeza	30	30	30	

A Tabela 3.1 [25] mostra um exemplo de uma cadeia de hipermercados com lojas em três cidades distintas, tendo todos os *beacons* o mesmo UUID independentemente da loja / localização. O valor *Major* é diferente para cada loja identificando assim a mesma em que um cliente se encontra. O valor *Minor* dentro da mesma loja possui valores diferentes identificando exatamente qual a secção em que o cliente se encontra. Note-se que o valor *Minor* é igual na mesma secção em lojas diferentes.

Exemplos de uso dos vários parâmetros

Por exemplo, se se desejar saber que um cliente frequentou a secção de roupa na loja existente em Coimbra pertencente à cadeia de hipermercados X, é necessário ter em conta os valores UUID, Major e Minor.

Se se pretender saber quais os clientes que frequentaram a loja existente em Coimbra, apenas é necessário ter em conta os parâmetros UUID e Major.

Por exemplo, se se desejar obter uma contagem do número de clientes que frequentou toda a cadeia de supermercados, apenas é necessário recorrer ao parâmetro UUID.

3.3.4.3. Grupo 3: Transmit Power

O parâmetro *Transmit Power* (no pacote anterior capturado: $0xC5 = 197$, complemento para 2 = $256-197 = 59$ dBm) (Figura 3.8) consiste na força do sinal que a aplicação recebeu vinda de um anúncio *iBeacon*. Consoante a força do sinal obtida pela aplicação no dispositivo do cliente, e sabendo a força do sinal que é suposto existir quando os dois dispositivos se encontram à distância de 1 metro, dado pelo parâmetro: *Received Signal Strength Indication* (RSSI), é possível calcular a distância a que o *beacon* se encontra do dispositivo que recebeu o seu sinal.

De modo a obter-se a melhor precisão na distância calculada todos os *beacons* devem ser calibrados registando o valor ótimo em dBm quando um dispositivo se encontra a 1 metro de distância. O processo de calibração permite registar o valor RSSI no *beacon* usado na fase posterior para obter a distância entre os dois dispositivos.

Calibração de um beacon

Segundo a norma *iBeacon*, o processo de calibração de um *beacon* deve respeitar os seguintes passos [25]:

1. Instalar o *beacon* no local onde deve transmitir o sinal;
2. Através do uso de um *iphone* com *iOS7+* (ou atualmente um *Android 4.3+*), e fazendo uso de aplicações de gestão *beacons* de cada fabricante, efetuar um emparelhamento entre o dispositivo e o *beacon* e obter uma leitura da potência de sinal recebida;
3. Colocar o *beacon* a uma distância de 1 metro;
4. Manter o dispositivo no mesmo local durante cerca de 10 segundos;
5. Após obter o primeiro valor de RSSI, deslocar o dispositivo lentamente numa distância de 30cm, tal como ilustrado na Figura 3.10, em linha reta, mantendo a orientação e garantindo que não se encontra nenhum obstáculo entre os dois dispositivos;

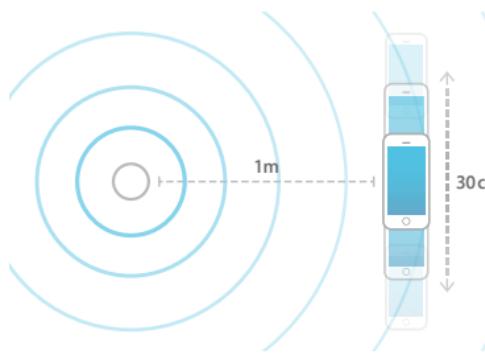


Figura 3.10 – Calibração de um beacon

6. Por fim, o valor de RSSI deve ser enviado para o *beacon* não devendo ser efetuada nenhuma alteração a este valor posteriormente.

Por omissão, e dependendo do fabricante, cada *beacon* já possui um valor RSSI. No entanto, este deverá ser sempre conferido e alterado caso se justifique.

Cálculo da distância entre os dois dispositivos

Para efetuar o cálculo da distância entre o dispositivo receptor e o *beacon* são necessários os valores:

- *Transmit Power* – Parâmetro em dBm recebido no pacote *iBeacon*;
- RSSI – Parâmetro em dBm recebido no pacote BLE.

A obtenção da distância ocorre através da divisão do valor RSSI pelo *Transmit Power*.

Apresenta-se a seguir o código responsável por esse cálculo na linguagem de programação JAVA.

```
protected static double calculateAccuracy(int txPower, double rssи) {
    if (rssи == 0) {
        return -1.0; // if we cannot determine accuracy, return -1.
    }
    double ratio = rssи / txPower;
    if (ratio < 1.0) {
        return Math.pow(ratio, 10);
    } else {
        double accuracy = (0.89976) * Math.pow(ratio, 7.7095) + 0.111;
        return accuracy;
    }
}
```

Para além dos valores mencionados torna-se necessário o uso das constantes 0.89976, 7.7095 e 0.111. Estas destinam-se a aproximações que devem ser alteradas consoante o dispositivo usado. No capítulo 6 (secção 6.3.3.3) será abordado este tópico, mostrando de que forma estes valores foram obtidos na aplicação móvel desenvolvida.

Resumo e problemas existentes

Um *beacon* consiste num dispositivo *wireless* que transmite um sinal rádio contínuo. A forma como o sinal é difundido não é direcional, mas sim sob um raio de uma esfera, sendo uniforme em todas as direções.

A difusão de um sinal *iBeacon* ocorre sobre a frequência de 2.4 GHz. Quando um dispositivo BLE deteta este tipo de sinal, usa a potência do sinal recebido (*Transmit Power*) para estimar a proximidade a que se encontra do *beacon*. Para além disso, usa também esse valor para calcular a precisão dessa estimativa. Assim, um sinal mais forte produz uma estimativa da distância mais precisa.

No entanto, os sinais na frequência 2.4 GHz estão sujeitos a atenuação derivada de diversos fatores físicos, tais como, paredes, portas, mas também por outros fatores como água, e corpos humanos.

Quando um sinal é atenuado ou degradado o dispositivo irá receber um sinal que não é tão forte quanto seria caso não possuísse objetos entre o emissor e o recetor. Isto levará a que haja uma redução da precisão e o cálculo da distância não seja ótimo.

Assim, o ambiente em que o *beacon* será implantado consiste num dos fatores essenciais e que maior alvo de estudo deve ter no momento de colocação de uma solução em produção.

Num ambiente ideal cada *beacon* instalado deveria estar o mais próximo possível dos dispositivos recetores não tendo nenhum objeto na sua direção, possibilitando estimativas mais precisas. No entanto, este ambiente raramente é possível especialmente em eventos com grandes quantidades de pessoas.

3.3.5. Precisão, atenuação e degradação de um sinal iBeacon

Para melhor entender o conceito de precisão efetua-se seguidamente uma analogia com a tecnologia *Global Positioning System* (GPS).

Nos sistemas GPS é possível alcançar uma maior precisão da localização de um determinado dispositivo quando este se encontra a receber sinais “limpos” de GPS. Para que isso aconteça é necessário que este se encontre fora de um edifício e não exista nenhum objeto atenuador de sinal entre o dispositivo recetor e o emissor.

Em todas as aplicações atuais para dispositivos *Android* e *iOS* é possível visualizar a precisão da localização através de um círculo em torno do indicador da localização. No caso de o dispositivo se encontrar dentro de um edifício, o sinal entre o emissor e recetor é obstruído aumentando o diâmetro do círculo. Este indica que o dispositivo se pode encontrar em qualquer região que o mesmo alcança, mostrando assim a perda de precisão na localização. Por outro lado, se o dispositivo se encontrar num espaço aberto a potência de sinal recebida é maior, reduzindo o diâmetro do círculo, aumentando assim a precisão da localização. Esta alteração da precisão é visível na Figura 3.11.

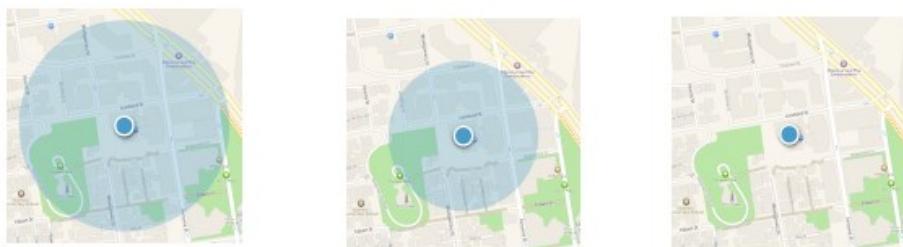


Figura 3.11 – Alteração da precisão em função da atenuação do sinal

Dispositivo dentro de um edifício. Baixa precisão. Círculo grande. O dispositivo pode-se encontrar em qualquer parte do círculo.

Dispositivo fora de um edifício, mas numa mala. Boa precisão. Círculo médio.

Dispositivo fora de um edifício, na mão com céu limpo. Nível de precisão ótimo.

Quando um sinal é difundido pela tecnologia *iBeacon*, a força do sinal está correlacionada com a distância a que o dispositivo se encontra do *beacon*. Numa situação ideal o dispositivo receptor deve conseguir “visualizar” o *beacon* em linha reta sem nenhum obstáculo entre os dois [25].

Tal como é apresentado na Figura 3.13, quando um dispositivo se encontra longe de um *beacon* a força do sinal é menor do que quando se encontra nas suas proximidades. Devido a esta diminuição de intensidade do sinal o dispositivo possui uma baixa precisão no cálculo da distância, assemelhando-se ao círculo grande na Figura 3.11.

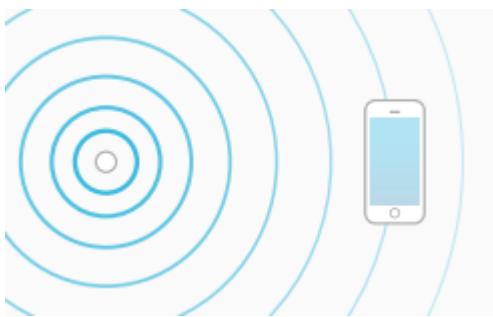


Figura 3.13 – Dispositivo longe do beacon

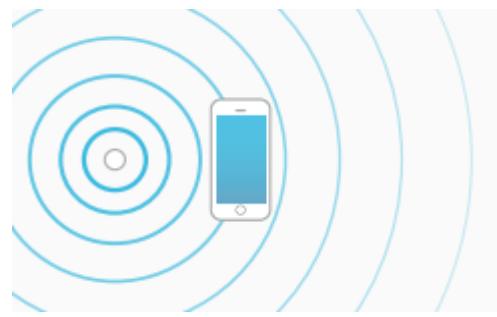


Figura 3.12 - Dispositivo perto do beacon

Quando o dispositivo se aproxima do *beacon* a potência do sinal aumenta e a precisão no cálculo da distância também (Figura 3.13).

No entanto, tal como a potência do sinal GPS é atenuada por objetos físicos, o mesmo acontece com a potência de sinal de um *beacon*. Tal como ilustrado na Figura 3.14 e 3.15 o facto de existirem objetos entre os dois dispositivos, faz com que o dispositivo receptor interprete o fraco sinal como se encontrando longe do *beacon* o que pode não corresponder à verdade, levando a uma falha de cálculo de distância (baixa precisão).

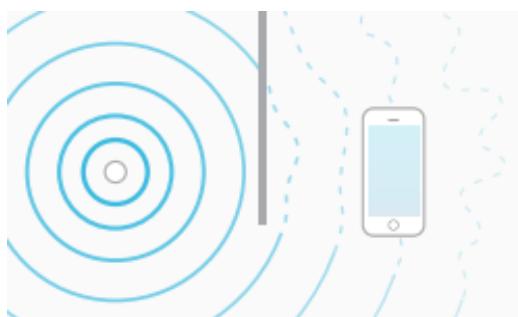


Figura 3.14 – Atenuação de sinal por objetos físicos

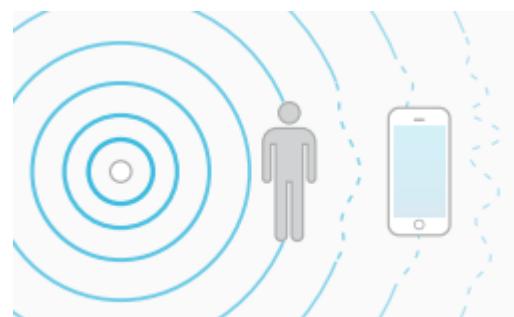


Figura 3.15 – Atenuação de sinal pelo corpo humano

A Tabela 3.2 mostra a categorização efetuada pela Apple para os diversos tipos de materiais tendo em conta a interferência, absorção e reflexão que cada um causa nos sinais radio (frequência 2.4 GHz).

Tabela 3.2 – Interferência de vários materiais

Tipo de material	Interferência
Madeira	Baixa
Material sintético	Baixa
Vidro	Baixa
Água	Média
Tijolo	Média
Mármore	Média
Gesso	Alta
Vidro à prova de bala	Alta
Metal	Muito alta

Köhne e Sieck [26] fazendo uso de um *beacon* (do fabricante Gimbal) e um iPhone 5 testaram a interferência dos diversos tipos de materiais entre os dois dispositivos. Para isso foram colocados os dois dispositivos à distância de 10 centímetros. Colocando uma camada de água de 4 centímetros entre os dois dispositivos foi obtida uma distância de 49 centímetros. Uma mão entre os dois levou a uma leitura de 1,5 metros e uma camada de metal (de 1 centímetro de espessura) levou a uma leitura de 2,21 metros de distância.

Tendo em conta estes fatores, a margem de erro no cálculo da distância consiste num fator importante e que deve ser ajustado a cada ambiente onde a solução sobre esta tecnologia é usada.

3.3.6. Monitoring e Ranging

Nesta seção será descrita a forma como a tecnologia *iBeacon* prevê que as aplicações existentes nos dispositivos receptores (por exemplo, *smartphones* ou *tablets*) interajam com *beacons*.

Esta estabelece duas formas de interação:

- **Monitoring:** permite despoletar ações quando o dispositivo entra / sai de uma região de *beacons*;
- **Ranging:** permite despoletar ações para um *beacon* específico com base na distância calculada.

3.3.6.1. Monitoring

A função *Monitoring* permite aos dispositivos receptores detetarem quando entram ou saem de uma determinada região de *beacons* (Figura 3.16). Por exemplo, se uma cadeia de supermercados deseja que um cliente seja notificado sempre que entra ou sai de todas as suas lojas, então deve ser definida uma região na aplicação tendo em conta o parâmetro UUID. Se se desejar que um cliente seja notificado apenas quando entra ou sai de uma determinada loja, então a região deve ser definida tendo em conta os parâmetros: UUID e *Major*. Se, por outro lado, se deseja que o cliente seja notificado apenas quando entra ou sai de uma determinada secção de uma loja, então deve ser considerada uma região com os parâmetros: UUID, *Major* e *Minor*.



Figura 3.16 – Monitoring region (In / Out)

Características da função Monitoring

- Apenas reconhece quando entra ou sai de uma região de *beacons*;
- Funciona quando a aplicação se encontra em *background*;
- Número máximo de regiões a monitorizar por aplicação: 20;
- Uma região pode ter um número ilimitado de *beacons*;
- O evento de entrada e saída de uma região possui um *delay* de 30 segundos de forma a evitar falsos positivos.

3.3.6.2. Ranging

Esta função permite ao dispositivo receptor obter uma lista de todos os *beacons* no seu raio de alcance, bem como uma distância estimada para cada um deles. A tecnologia *iBeacon* categoriza, com base na distância, os estados da Tabela 3.3 [26].

Tabela 3.3 – Estados da função de “Ranging”

Estado	Distância	Notas
Immediate	Poucos centímetros	Consiste no estado de maior nível de precisão para o cálculo de distância dado o sinal forte recebido pelo dispositivo.
Near	Entre 1 e 3 metros	A distância padrão diz respeito a um ambiente sem obstáculos entre o emissor e o receptor. No caso da existência de objetos pode existir uma atenuação de sinal podendo este estado não ser despoletado mesmo que fosse esse o caso real.
Far	Mais de 10 metros	Este estado indica que o <i>beacon</i> foi detetado mas a precisão do sinal é demasiado baixa para se encontrar no estado <i>Immediate</i> ou <i>Near</i> .
Unknown	----	Não é possível determinar a distância do <i>beacon</i> .

Com o estabelecimento das categorias anteriores é possível, para o mesmo *beacon*, oferecer diferentes tipos de interatividade com o utilizador baseado na distância a que este se encontra.

No caso da existência de obstáculos atenuadores de sinal, uma vez que o mesmo irá enfraquecer de uma forma mais rápida ao longo da distância, devem ser feitos testes e considerar as áreas *Immediate*, *Near* ou *Far* (Figura 3.17) consoante a potência de sinal recebida em cada localização.

Tal como a frequência com que o sinal de “anúncio” é transmitido pode ser ajustado, também a potência de sinal de cada *beacon* pode ser alterada de modo a colmatar ambientes menos propícios à propagação do sinal. A duração da bateria encontra-se no entanto correlacionada com estes fatores (detalhado em pormenor na secção 3.3.9.1).

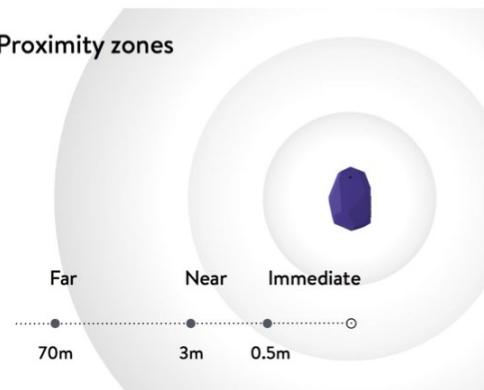


Figura 3.17 – Ranging beacons

Características da função Ranging

- Apenas funciona com a aplicação aberta (*foreground*);
- Apresenta os parâmetros de todos os *beacons* no seu alcance:
 - UUID, Major, Minor, TX Power;
- Obtenção de alteração de estado imediato (consoante frequência de emissão do sinal).

A Tabela 3.4 apresenta uma comparação entre as funções *Monitoring* e *Ranging*.

Tabela 3.4 – Comparação entre as características Monitoring e Ranging

	<i>Monitoring</i>	<i>Ranging</i>
Background	Sim	Não
Foreground	Não	Sim
Distância para os beacons	Não	Sim
Alteração de estado	30 segundos de atraso	Imediato

3.3.7. Constituição ao nível do hardware

O hardware existente no interior de um dispositivo *beacon* pode variar dependendo do fabricante. A Figura 3.18 mostra *beacons* de diferentes fabricantes com formatos distintos.



Figura 3.18 – Formatos existentes de beacons

Alguns dos fabricantes colocam no interior, por exemplo, sensores de temperatura e acelerómetro de forma a valorizar o seu produto.

No entanto, a constituição base de um *beacon* assenta nos seguintes elementos (ilustrados na Figura 3.19):

- Um microcontrolador;
- Um emissor / receptor BLE (*Chipset*);
 - Fabricado por duas empresas: *Texas Instruments* e *Nordic Semiconductor*;
- Uma bateria.



Figura 3.19 – Conteúdo no interior de um beacon

Bateria

As baterias mais comuns nos *beacons* são baterias de célula, compostas por íon de lítio capazes de fornecer energia até 1000 mAh:

- **CR2032** – Energia fornecida: 240 mAh (mais usadas):
 - Tamanho: 20mm de diâmetro e 3.2 mm de altura;
- **CR2450** – Energia fornecida: 620 mAh (tamanho médio);
- **CR2477** – Energia fornecida: 1000 mAh (tamanho grande).

Alguns *beacons* podem conter pilhas alcalinas AA podendo fornecer até cerca de 2000 mAh.

É possível ainda serem alimentados através de uma ligação mini-usb ou uma ligação à corrente elétrica. As diversas formas de alimentação são ilustradas na Figura 3.20.



Figura 3.20 – Forma de alimentação de um beacon

A energia fornecida pela bateria e os valores dos parâmetros *advertising interval* e *Transmit Power* influenciam o tempo de vida de um *beacon*.

A seção 3.3.9 apresentará um estudo efetuado pela aislelabs [27] onde mostra a variação do tempo médio de vida de um *beacon* em função da alteração dos parâmetros anteriores.

3.3.8. Firmware de um beacon

Cada *beacon* possui um *firmware* específico responsável pelo seu funcionamento. Este permite ainda configurar parâmetros que influenciam o tempo médio de bateira, nomeadamente:

- **Transmit Power**

- Beacons transmitem um sinal com um valor fixo de potência. À medida que o sinal se propaga no ar a força do sinal vai diminuindo. Assim, sinais mais fortes (valor de *Transmit Power* mais elevado) permitem alcançar longas distâncias (maior consumo de energia). Valor de *Transmit Power* menor permite menor consumo de energia, mas o alcance é menor.

Na Figura 3.21 é apresentada uma relação entre a distância alcançada pelo sinal, em função do valor *TX Power* configurado no beacon.

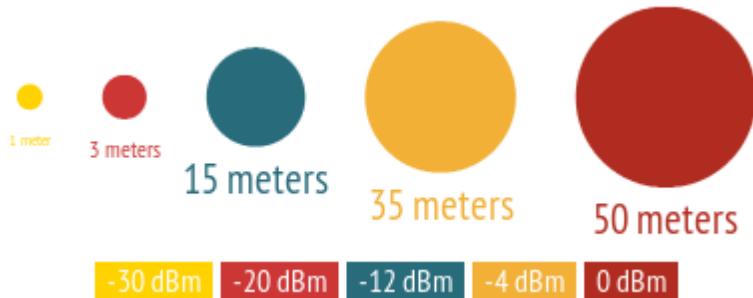


Figura 3.21 – Distância alcançada em função do parâmetro TX Power

- **Advertising Interval:**

- A frequência com que um beacon emite um sinal designa-se por *advertising interval*. Um intervalo de 100ms (estabelecido pela norma *iBeacon*) indica que um sinal é emitido a cada 100 milissegundos, ou seja, 10 vezes por segundo. Um intervalo de 500ms significa que um sinal é emitido duas vezes por segundo traduzindo-se num menor consumo de bateria. Com o aumento do *advertising interval* aumenta o tempo de vida da bateria de um beacon, mas diminui o tempo de resposta dos dispositivos receptores [27]. Assim, se se desejar uma resposta rápida dos dispositivos receptores o *advertising interval* deve ser diminuído. No caso de se desejar otimizar o tempo de bateria o *advertising interval* deve ser aumentado.

Alguns fabricantes de beacons, como Kontakt, Estimote, RadBeacon e BlueSense Networks fornecem o seu próprio *software* de forma a configurar os parâmetros referidos. Outros beacons, como Minew, podem ser configurados através de qualquer cliente GATT¹. A principal vantagem em suportar o método GATT consiste em poder configurar os parâmetros de qualquer beacon com uma única aplicação. Ambas as formas necessitam de prévia autenticação na API do fabricante de forma a garantir que as configurações não são alteradas por entidades não autorizadas.

¹ GATT – *Generic Attribute Profile*, estabelece operações comuns que permitem a troca de dados em comunicações BLE [75]

3.3.9. Tempo médio de vida de um beacon

Os resultados obtidos pela aislelabs [27] e apresentados na Tabela 3.5 foram efetuados com recurso aos *beacons* do fabricante “Estimote” tendo como bateria CR2477, fornecendo 1000 mAh de energia.

3.3.9.1. Variação dos parâmetros *Advertising Interval* e *Transmit Power*.

Na Tabela 3.5 é visível a relação entre o tempo (em meses) de bateria de um *beacon*, face aos parâmetros *advertising interval* e *Transmit Power*.

Tabela 3.5 – Influência dos parâmetros na duração da bateria de um beacon

		Advertising Interval		
		100ms	645ms	900ms
Transmit Power	0dBm	4.0	14.6	27.2
	-12dBm	5.0	21.4	32.3
	-20dBm	5.6	22.8	35.6

Verifique-se que o tempo de vida do mesmo *beacon* pode variar entre os 4 meses e os 36 meses (aproximadamente). Com a alteração dos parâmetros apresentados pode-se reduzir / aumentar o tempo de vida em 7 vezes.

No caso em que o *advertising interval* é de 645ms e o *Transmit Power* de -12dBm, com um alcance de cerca de 35 metros e um envio de sinal a cada 0.65 segundos (aproximadamente), o *beacon* consegue ter um tempo de vida de 21 meses (aproximadamente).

Tenha-se particular atenção na variação do tempo de vida em relação ao *advertising interval*. Para o mesmo *Transmit Power* (-12dBm) o tempo de vida do *beacon* pode variar entre os 5 meses e os 32 meses (aproximadamente). Apesar da norma *iBeacon* definir 100ms para o valor padrão do *advertising interval*, muitas das vezes valores superiores satisfazem os requisitos da solução e triplica o tempo de vida do *beacon*.

3.3.10. Influência na descarga de bateria dos dispositivos receptores

Segundo o estudo desenvolvido pela aislelabs [28], existem dois fatores que influenciam na descarga de bateria dos dispositivos receptores: o *advertising interval* e o número de *beacons* detetáveis em simultâneo.

No que diz respeito ao parâmetro *advertising interval*, tal como acontece na descarga da bateria de um *beacon*, quanto menor for este tempo, mais vezes o dispositivo receptor irá detetar o sinal e necessitar de efetuar o seu processamento.

Relativamente ao número de *beacons* detetáveis em simultâneo, o estudo indica que o consumo de bateria dos dispositivos recetores aumenta de forma exponencial com o aumento da deteção de vários *beacons* em simultâneo.

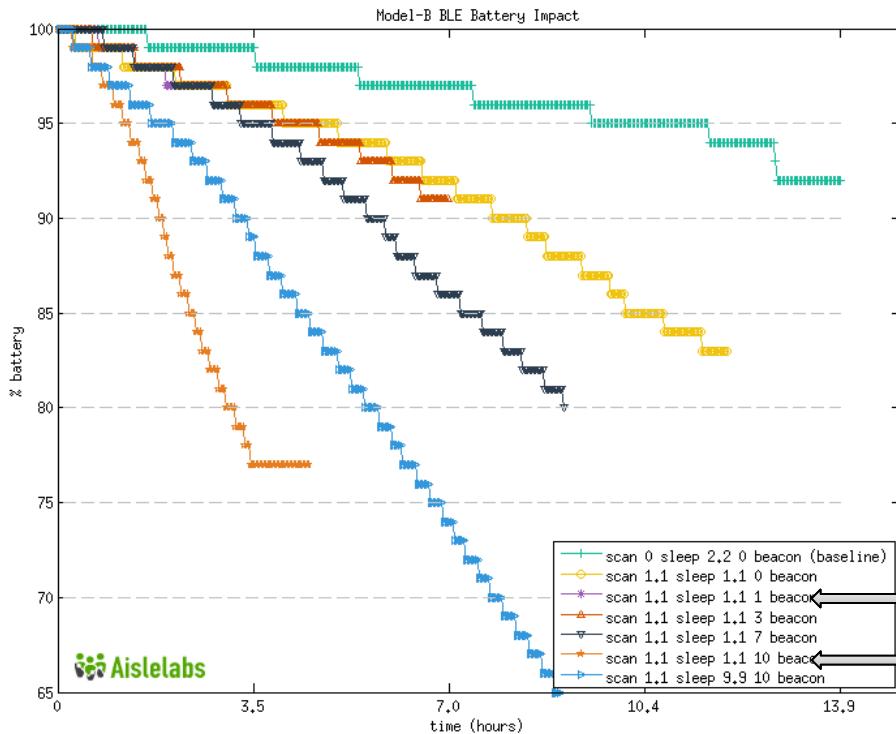


Figura 3.22 – Consumo de bateria de dispositivos

Compare-se a linha laranja com a linha roxa (assinaladas na Figura 3.22). No caso em que apenas é detetado um *beacon* (linha roxa), passadas 3 horas e 30 minutos, a bateria do dispositivo situava-se acima dos 95%. No caso de serem detetados 10 *beacons* em simultâneo (linha laranja), passado o mesmo período de tempo, a bateria do dispositivo situava-se abaixo dos 80%. Isto deve-se à quantidade de pacotes enviados e que o dispositivo recetor necessita de processar. A deteção de 10 *beacons* em simultâneo, significa, de uma forma simplista, o processamento de 10 vezes mais pacotes por parte dos dispositivos recetores.

Desta forma, o local onde os *beacons* serão posicionados deve ser extremamente ponderado, devendo ser efetuados testes de potência de sinal de forma a garantir o número mínimo de domínios de colisão (*beacons* detetáveis em simultâneo).

Aislelabs apresenta ainda outro estudo [29] onde são comparados os sistemas iOS e *Android* no que diz respeito à descarga da bateria em comunicação com *beacons*, concluindo que “o sistema *Android* é extremamente mais eficiente que o sistema iOS”.

3.4. Aspetos teóricos

Esta secção enuncia a forma como a tecnologia *iBeacon* pode melhorar a relação entre um cliente e um retalhista, fomentando as razões para o seu uso. Descreve ainda as limitações da tecnologia e de que forma podem ser ultrapassadas.

3.4.1. Razões para o uso da tecnologia *iBeacon* no retalho

Chuck Martin, no artigo “*Harvard Business Review*” [30] afirma que “*beacons* consistem na peça que faltava para completar o *puzzle* do retalho”. Apontando a capacidade que os *beacons* têm de enviar informação de forma “autónoma” e personalizada, o autor define a tecnologia *iBeacon* como a solução para ultrapassar um obstáculo no mundo do retalho: tratar cada cliente de forma única.

Paul Chaney [31] estabeleceu cinco razões que sustentam o seu uso:

- Conveniência do cliente – alguns clientes utilizam os seus *smartphones* enquanto se encontram a fazer as suas compras para comparar o preço, do produto que deseja noutras lojas. Receber um desconto personalizado no momento em que o cliente se encontra a ver o produto será apreciado e retira a atenção da procura noutras lojas;
- Melhorar a experiência de compra – Ron Johnson, antigo CEO da J.C. Penney indicou que “uma loja é muito mais do que um local para adquirir produtos”, “tem de enriquecer as suas vidas” [32]. Afirma ainda que deve ser dado ao cliente muito mais do que espera. A tecnologia *iBeacon* fundamenta essa ideia podendo criar experiências interativas, através de receção de informação personalizada, tendo em conta o local onde se encontra;
- Acessibilidade – O baixo custo de *beacons* permite a implementação da tecnologia sem grandes investimentos;
- Competitividade com grandes retalhistas – A capacidade de usar uma tecnologia inovadora aliada ao baixo custo permite manter o nível de competitividade entre pequenos e grandes retalhistas;
- Perfil do cliente – A tecnologia *iBeacon*, associada a aplicações móveis e plataformas de análise de dados, permite aumentar o conhecimento do perfil de cliente, obtendo dados que permitem uma maior personalização dos dados enviados.

De acordo com o relatório de “Consumo Móvel”, divulgado pela InMobi [33] o *m-commerce* (comércio eletrónico móvel) obteve um aumento de 15% no ano de 2014, afirmando que 83% dos consumidores assumem usar o comércio eletrónico em 2015. Com imensas marcas a investir em aplicações e recursos interativos, atrair clientes às suas lojas físicas tornou-se um desafio para a maioria dos hipermercados.

A tecnologia *iBeacon* consiste num forte aliado dos hipermercados permitindo melhorar e inovar a experiência de compra e assim combater uma regressão na deslocação dos clientes às lojas.

É possível melhorar a experiência de compra, através do uso do dispositivo móvel, recorrendo a várias ações, como por exemplo:

- Enviar ofertas de descontos em produtos quando um cliente se encontra perto de uma prateleira (Figura 3.23);
- Enviar uma notificação quando um cliente se encontra nas proximidades da loja (no exterior), e através do histórico do consumo do cliente, verificar se algum produto que comprou recentemente se encontra em promoção, enviando-lhe essa informação, incentivando-o a entrar;
- Oferecer cupões aos clientes após uma compra de forma a compartilhá-los com seus amigos, incentivando-os a visitar a loja;
- Devolver informação importante ao vendedor de forma a perceber a razão de, por exemplo, existir pouca afluência num determinado dia podendo desencadear ações que possam melhorar essa afluência;
- Oferecer recompensas a clientes que efetuem determinadas tarefas, por exemplo, tirar uma foto a um produto e partilhá-lo com um amigo;
- Oferecer vouchers apenas a clientes que frequentem determinadas áreas da loja, efetuarem determinadas compras, ou frequentem um evento promovido pela loja;
- Tirar partido do tempo de espera dos clientes. Enquanto um cliente se encontra, por exemplo, na fila para pagamento oferecer-lhe informação relevante de acordo com o seu的习惯 de consumo.



Figura 3.23 – Uso da tecnologia iBeacon na área do Retalho

3.4.1.1. Casos de estudo na área do retalho e resultados obtidos

Nesta seção serão apresentados casos de estudo na área do retalho que atualmente já se encontram a fazer uso da tecnologia *iBeacon*. O apêndice I apresenta outras áreas de atuação e respetivos casos de estudo.

McDonald's

A cadeia de franchising “McDonald's” adotou a tecnologia *iBeacon* em 26 lojas distribuídas pela Colômbia e Geórgia. Foi implementado um projeto piloto durante o mês de novembro 2014 enviando promoções relativas aos produtos “McChicken Sandwiches” e “Chicken McNuggets”.

O relatório apresentado pela “Piper” indica que, através da tecnologia *iBeacon*, a venda de “McChicken Sandwiches” aumentou cerca de 8% e de “Chicken McNuggets” cerca de 7,5% [34].

Jack Pezold, gestor de uma loja de *franchising* de “McDonald's” expressou que “Todos os clientes se encontram a olhar frequentemente para os seus telemóveis, especialmente a geração pós 1990. É através deste que devemos interagir com os nossos clientes” [35].

Regent Street

Regent Street, um destino de compras e turístico em Londres, faz uso da tecnologia *iBeacon* não para promover uma única marca, mas com o objetivo de privilegiar toda a extensão comercial da rua composta por 130 lojas [36]. A aplicação utiliza a tecnologia de forma a enviar promoções de novos produtos, descontos e outros alertas a todos os clientes que passem pela rua.

ANKAmall

O ANKAmall (Istambul) efetuou uma parceria com a Boni de forma a instalar 356 *beacons* num centro comercial. A aplicação “Walk & Win” permite aos clientes acumular pontos sem efetuar compras. Os pontos denominam-se “Bonis” e são acumulados através da visita do cliente ao centro comercial. Com os pontos recebidos, os clientes podem comprar produtos ou serviços em lojas do próprio centro comercial [37].

InMarket

A InMarket consiste numa empresa a qual já implementou mais de 200 *beacons* em diversas cadeias de supermercados nos Estados Unidos da América, entre os quais *Safeway* e *Giant Eagle* [38]. A sua forma de funcionamento assenta no envio de notificações através de aplicações de parceiros de publicidade como *Nast's Epicurious* e *Gannet's Key Ring*.

A InMarket indica que as interações dos clientes com os produtos anunciados tiveram um aumento de 19 vezes após o uso desta tecnologia. A utilização de aplicações foi 16,5 vezes superior para utilizadores que receberam uma notificação via *iBeacon* em comparação com os que não receberam. Para além disso, utilizadores que receberam uma notificação via *iBeacon* foram 6,4 vezes mais propensos em manter a aplicação no seu dispositivo.

Da parceria resultante entre a *InMarket* e *Hillshire Brands* resultou um aumento de 20 vezes na intenção de compra dos seus produtos, resultando num aumento de 500% em publicidade móvel.

A *American Craft* que adotou esta tecnologia incrementou também as suas vendas globais entre Abril e Junho 2014, fomentando a percepção da sua marca em 36% [39].

ShopKick

O *ShopKick* consiste numa ferramenta de compra de produtos, fundada em 2010, cuja tecnologia *iBeacon* foi implementada, em 2014, e se encontra a ser testada em diversas cadeias de supermercados nos EUA, tais como, *Macy's, Inc.*, *American Eagle*, *Target*, *Best Buy* e *Old Navy*.

De acordo com o relatório apresentado pela *ShopKick*, obteve um total de 1bilhão de dólares de receitas para os seus parceiros desde 2010, sendo mais de metade obtido desde a implementação da tecnologia *iBeacon* [40]. Em 12 meses, com recurso à tecnologia *iBeacon*, obteve um equivalente de receitas a quatro anos. Esse valor é decomposto em 50 milhões (através de notificações enviadas ao cliente que o fez entrar na loja) e 100 milhões de produtos em envio de promoções direcionadas, levando o cliente a efetuar a compra.

Foi efetuado um teste na cadeia de lojas *American Eagle Outfitters*, cujo objetivo consistia em guiar os clientes para uma determinada secção. Nos dias onde foi usada a tecnologia *iBeacon* e enviadas notificações *push* sobre vantagens em se dirigir à secção, o número de visitas duplicaram face aos dias em que a tecnologia não foi usada [40].

Apple

Sendo a tecnologia *iBeacon* desenvolvida pela Apple, esta foi a primeira a implementar nas suas lojas, iniciando apenas nas lojas existentes nos EUA. Atualmente encontra-se em expansão estando a ser implementado já nas lojas existentes em Portugal [41].

3.4.2. iBeacon vs Near Field Communication

Existem várias semelhanças entre a tecnologia *iBeacon* (assente na tecnologia BLE) e a tecnologia *Near Field Communication* (NFC) das quais se destacam as seguintes: ambas são tecnologias *wireless*; ambas necessitam de *hardware* especial; e ambas permitem a troca de informação entre dois dispositivos.

Existem, no entanto, diferenças que permitem destacar o uso da tecnologia *iBeacon*, particularmente, na área do retalho.

Uma das maiores diferenças consiste no alcance do sinal. De modo a que haja troca de informação, os dispositivos, no caso do NFC devem estar a uma distância máxima de 20 centímetros e no caso do *iBeacon* a uma distância até 70 metros. A emissão de dados no NFC apenas ocorre quando dois dispositivos se encontram próximos, enquanto que, na tecnologia *iBeacon* é efetuada uma emissão continua de dados. Verificando um exemplo prático, se um restaurante tivesse um cartaz com a informação “Toque com o seu *smartphone* para receber um desconto”, fazendo uso da tecnologia NFC, a decisão em receber o desconto estaria do lado do cliente. No caso de o restaurante usar a tecnologia *iBeacon* teria uma maior versatilidade na

comunicação podendo enviar o seu desconto diretamente aos seus clientes sem que estes tenham de efetuar uma determinada ação [43].

Outra diferença entre os dois é a velocidade de transmissão de dados. Enquanto a tecnologia *iBeacon* permite velocidades até 2 Mbps, o NFC apenas consegue alcançar um máximo de 400kbps. Isto permite que a primeira possibilite a troca de ficheiros, imagens e conteúdos multimédia ao contrário do NFC. Com uma taxa de transmissão de dados baixa e um reduzido alcance de sinal, não é necessário que dispositivos NFC possuam uma antena radio potente, traduzindo-se num menor consumo de bateria dos dispositivos.

Por outro lado, dispositivo capacitados com a tecnologia NFC conseguem interagir com objetos passivos (que não necessitam de energia), tais como *tags* RFID sendo a troca de informação iniciada com a aproximação dos dois dispositivos.

Face às vantagens enunciadas, a tecnologia *iBeacon*, vem solucionar as principais limitações que a tecnologia NFC apresenta particularmente na área do retalho. A Figura 3.24 mostra o alcance da tecnologia *iBeacon*, face à tecnologia NFC (Figura 3.25).



Figura 3.24 – Tecnologia *iBeacon*

Figura 3.25 – Tecnologia NFC

Na Tabela 3.6 encontram-se as principais diferenças entre a tecnologia *iBeacon* e NFC.

Tabela 3.6 – iBeacon vs. NFC

	iBeacon	NFC
Tecnologia	BLE	Near Field Communication
Suporte de SO	<i>Android, Windows Phone & iOS</i>	<i>Android & Windows Phone</i>
Top 10 de fabricantes de dispositivos móveis*	10 de 10	9 de 10
Velocidade	2 Mbps	400kbps
Distância de alcance	Até 70 metros	Até 20 centímetros
Intervalo de custo	Entre 30 e 50€	Menos de 1€

* Não suportado pela Apple

3.4.3. Limitações da tecnologia iBeacon

Existem duas regras de modo a que os dispositivos consigam receber e interpretar anúncios *iBeacon*, nomeadamente:

- Uso obrigatório do Bluetooth;
- Uso obrigatório de uma aplicação.

3.4.3.1. Uso obrigatório do Bluetooth

Segundo um estudo [44] desenvolvido no Reino Unido pela Empatika, a percentagem de dispositivos que circula diariamente com o *Bluetooth* ligado é de aproximadamente 33%. No entanto, essa percentagem pode variar tendo em conta outros países: entre os 25% no caso da Rússia e os 50% no caso dos Estados Unidos da América e Canadá.

Embora estes números possam ser reduzidos para o sucesso da tecnologia *iBeacon*, existem dois fatores que os permitem melhorar de forma significativa:

1. Informar o cliente da necessidade de ligar o *Bluetooth*;
2. Crescimento exponencial da IoT.

No que diz respeito ao primeiro ponto, o cliente deve ser informado que, para usufruir de uma determinada vantagem, o *Bluetooth* do dispositivo deve-se encontrar ligado. Essa informação deve ser apresentada diretamente na aplicação, Figura 3.26 (quando é descarregada ou aberta, direcionando o utilizador para a área de ligação de *Bluetooth* caso não esteja ligado), e sobretudo em espaços físicos, devendo ser afixados *placards* de forma a funcionar como um

lembrete para o cliente (à semelhança de como é apresentado atualmente para zonas *Wifi* gráts), Figura 3.27.

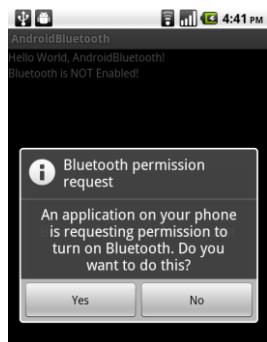


Figura 3.27 – Permissão para a aplicação ligar o Bluetooth



Figura 3.26 – Poster com informação de zona de WiFi gráts

Por outro lado, o crescimento exponencial da IoT leva a que o número de dispositivos com o *Bluetooth* ligado diariamente aumente de forma exponencial nos próximos anos. Segundo o estudo [45] apresentado pela *ABI Research*, o número de dispositivos capacitados com *Bluetooth* e que faziam uso deste diariamente em 2012 eram cerca de 3,5 bilhões. O mesmo estudo estima que (acompanhando o crescimento do número de dispositivos) o valor suba para os 10 bilhões no ano de 2018.

Como principal razão para este crescimento encontra-se o aumento de dispositivos *smart* que fazem uso da tecnologia BLE. Como exemplo de dispositivos tem-se: Próteses, Chapéus, Placas de trânsito, Chupetas, Relógios, Pulseiras, entre outros (Figura 3.28). Todos destes destinam-se a recolher dados e a enviá-los para o dispositivo do utilizador. Muitos, tal como os *smartwatches* exigem uma comunicação com o dispositivo em tempo real, necessitando deste modo que a tecnologia BLE do dispositivo se mantenha ligada sempre que é usado, ou seja, muitas vezes diariamente.

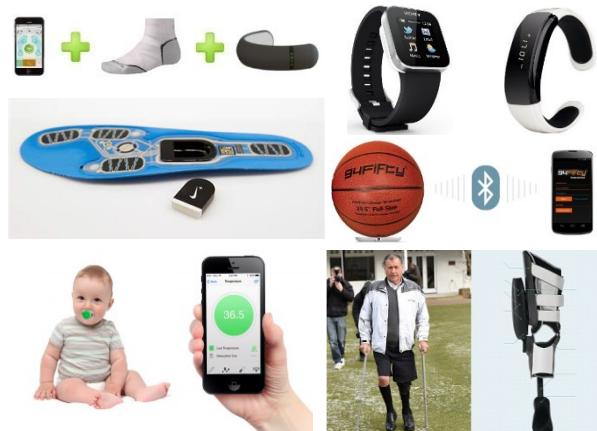


Figura 3.28 – Dispositivos BLE

3.4.3.2. Uso obrigatório de uma aplicação

De modo a que o cliente descarregue a aplicação e faça uso dela é necessário que o vendedor lhe introduza valor acrescentado e transmita ao cliente as vantagens que o seu uso lhe traz. Devem ser feitas campanhas de marketing e publicidade de modo a divulgar a aplicação e direcionar o cliente para o seu uso, pois só assim irá obter resultados da tecnologia *iBeacon*.

Se por um lado esta tecnologia pode trazer ao vendedor um enorme fator de crescimento e melhoria dos seus serviços, o facto de poucos utilizadores instalarem a aplicação faz com que seja um fracasso, pois, sem uma aplicação existente no lado do cliente torna-se impossível chegar até este com envio de promoções, e consequentemente efetuar toda a análise de dados resultante dessa interação. Assim, todo o cliente deve ser conhecedor que com a aplicação poderá obter vantagens exclusivas e personalizadas. Devem ser criadas ainda funcionalidades que, dependendo de cada caso, criem valor para além da receção de descontos, como por exemplo, (no retalho) conseguir definir uma lista de compras.

3.4.4. Eddystone Beacon

Eddystone consiste num protocolo de comunicação *open-source software* (OSS) [46], sobre a licença Apache, desenvolvido pela Google e apresentado em julho de 2015. O grande objetivo do seu desenvolvimento consistiu em colmatar uma das limitações da tecnologia *iBeacon*: uso obrigatório de uma aplicação, introduzindo o conceito de *Physical Web*. Teve ainda como objetivo melhorar a gestão de *beacons* à distância. Para isso, a norma *Eddystone* estabelece três tipos de pacotes: *Eddystone-UID*, *Eddystone-URL* e *Eddystone-TLM*.

Eddystone-UID

Um *beacon* configurado para enviar um pacote do tipo *Eddystone-UID* irá funcionar de forma semelhante a um *iBeacon*, enviando a sua identificação. Com base nesta a aplicação instalada no dispositivo receptor irá desencadear uma determinada ação. Enquanto que o identificador de um *iBeacon* é composto por três partes: UUID, Major e Minor (com um tamanho total de 20 bytes), um *Eddystone-UID* é composto por duas: *Namespace* (10 bytes) e *Instance* (6 bytes).

O parâmetro *Namespace* é semelhante ao parâmetro UUID da tecnologia *iBeacon*, sendo usado para distinguir *beacons* entre organizações. Para gerar um *Namespace*, a norma *Eddystone* recomenda obter os 10 primeiros bytes de uma chave SHA-1 baseada no nome da organização. Outra forma consiste em remover as três partes centrais de um UUID (versão 4) (útil no caso de já ter implementado uma solução baseada na tecnologia *iBeacon*).

***iBeacon* UUID** **B0CA750-E7A7-4E14-BD99-095477CB3E77**

Eddystone-UID **8B0CA750095477CB3E77**

O parâmetro *Instance*, à semelhança dos parâmetros *Major* e *Minor* da tecnologia *iBeacon*, serve para diferenciar cada *beacon*. Exemplo de um valor para o parâmetro *Instance*: 0BDB87539B67.

Eddystone-URL

O pacote do tipo *Eddystone-URL* é composto apenas por um único campo: URL. O tamanho do pacote é variável e está associado ao tamanho do URL. Este tipo de anúncio está relacionado com o conceito de *Physical Web*, ou seja, com o pacote *Eddystone-URL* os dados a mostrar ao cliente são enviados diretamente no anúncio BLE de cada *beacon*. No caso de pacotes do tipo *iBeacon* e *Eddystone-UID* existe a necessidade de uma aplicação do lado do cliente, que traduza o identificador recebido do *beacon* numa determinada ação.

O URL pode ser uma página Web, com informação relevante para o cliente, sendo o mesmo inserido no *beacon* através de uma API fornecida pelo fabricante.

Exemplo de um URL: http://my-restaurant.com/check-in?restaurant_id=6523

Para que o dispositivo interprete este tipo de pacotes, é necessário que o *browser* tenha a opção *Physical Web* ativada [47]. Atualmente apenas o *Chrome* para os sistemas *iOS* e *Android* e o *Opera* para o sistema *Android* têm a capacidade de interpretar este tipo de pacotes. Através de um destes *browsers*, o utilizador possui a capacidade de receber notificações sem que tenha uma aplicação específica instalada.

Eddystone-TLM

O tipo de pacote *Eddystone-TLM* tem como objetivo enviar dados que permitam identificar o “estado” do *beacon*, nomeadamente: voltagem da bateria (usado para estimar o nível de bateria); temperatura do *beacon*; número de pacotes enviados desde que foi ligado; tempo que o *beacon* se encontra ligado (desde o último arranque).

Estes dados são enviados apenas num único sentido, não podendo ser alterados no *beacon*. Apesar disso, através de uma aplicação no lado do cliente, é possível obter estes dados, enviar para a *Cloud* e, por exemplo, através de um CMS efetuar a gestão de *beacons* à distância. Por exemplo, fazendo uso de uma aplicação instalada nos dispositivos dos clientes, sempre que se aproximam de um *beacon* e recebem um pacote *Eddystone-TLM* para além de serem notificados com informação direcionada, de forma transparente são enviados dados acerca do estado do *beacon* para o CMS. Desta forma, permite que se tenha atualizada a informação acerca, por exemplo, da bateria de cada um dos *beacons* e se possa fazer a gestão destes à distância.

3.4.4.1. Compatibilidade da solução com a tecnologia Eddystone

Atualmente, apenas um único fabricante (Estimote) possui, de forma estável, *beacons* compatíveis com esta tecnologia. Este permite efetuar uma atualização do *firmware* dos seus *beacons* de forma a enviarem os três tipos de pacotes *Eddystone*. Por outro lado, o SDK fornecido pelo fabricante (a usar na aplicação móvel para comunicação com os *beacons*) ainda se encontra em fase de testes (*beta*).

Apesar da falta de maturidade da tecnologia, toda a solução foi desenvolvida tendo em vista a compatibilidade com os dois tipos de tecnologias: *iBeacon* e *Eddystone*.

Para efeitos de explicação será abordado em particular a tecnologia *iBeacon* dada a estabilidade, maturidade, abrangência de fabricantes e resultados obtidos em casos de aplicação prática.

4. DESENVOLVIMENTO DA SOLUÇÃO

Este capítulo tem como objetivo enunciar as tecnologias e *frameworks* utilizadas, bem como descrever a metodologia e arquitetura de desenvolvimento da solução.

4.1. Componentes da solução

O desenvolvimento da solução teve em vista a sua utilização de forma independente e simultânea por diversos retalhistas, visualizando e operando sobre dados independentes entre si. A adaptação a cada um passará pela implementação de requisitos adicionais na aplicação móvel do cliente.

A solução desenvolvida é composta por três componentes, subdividindo-se em quatro elementos, apresentados na Figura 4.1.



Figura 4.1 – Componentes da solução desenvolvida

Sistema de Gestão de Conteúdos

O CMS consiste numa plataforma *Web* cuja utilização será feita única e exclusivamente pelo retalhista. Será nesta plataforma onde o retalhista irá inserir os seus clientes, criar e enviar as suas campanhas e efetuar a análise de dados resultante da interação do cliente com as mesmas.

Base de dados

A DB consiste no elemento que permite guardar todos os dados de comunicação e configuração de toda a solução, funcionando de forma complementar ao CMS. Para interagir com esta componente é necessário recorrer ao CMS.

Beacons

Os *beacons* consistem na componente de *hardware* que a aplicação, existente no dispositivo do cliente, irá fazer uso de forma a comunicar com o CMS e devolver informação direcionada tendo em conta o identificador do *beacon* com o qual contatou.

Aplicação *Android*

A aplicação *Android* destina-se a ser utilizada pelos clientes e tem como objetivo detetar a proximidade de um *beacon*, recolher dados do cliente bem como o *beacon* que detetou, enviar essa informação ao servidor (composto pelo CMS e DB) e receber uma determinada informação mostrando-a ao cliente.

4.2. Enquadramento técnico

A situação atual em que se encontrava a solução anteriormente à realização do estágio era nula. Não existia na empresa quaisquer protótipos, ou projetos semelhantes na área, sendo da responsabilidade do estagiário inicializar, documentar, implementar e finalizar toda a solução.

Para a implementação da mesma foi necessário a aprendizagem de diversas linguagens de programação, tecnologias e normas em constante evolução. Toda a aprendizagem, implementação e documentação foi efetuada em horário pós-laboral, dada a atividade da empresa ir para além desta solução e o estagiário ter uma ligação contratual à empresa que abrange outras áreas de responsabilidade.

4.2.1. Tecnologias utilizadas

Esta secção apresenta as principais tecnologias usadas.

4.2.1.1. HTML

HyperText Markup Language (HTML) foi a linguagem base utilizada na construção do CMS. Criada por Tim Berners-Lee, consiste numa linguagem cujo funcionamento assenta em Tags. Interpretada e não compilada por *browsers*, soluções assentes nesta linguagem podem ser desenvolvidas com recurso a um simples editor de texto, ou um complexo *software*, por exemplo *NetBeans* [48].

4.2.1.2. CSS

De forma a definir um melhor aspeto de cada página *Web* foi usada a linguagem *Cascading Style Sheets* (CSS). Esta permite definir *layouts* e atribuir estilos a uma ou mais páginas *Web*. A sua grande vantagem reside em fazer a separação entre o estilo da página e o seu conteúdo [49].

4.2.1.3. Bootstrap

Bootstrap, inicialmente designado por *Twitter Blueprint*, foi criado por Mark Otto e Jacob Thornton consistindo numa biblioteca interna pertencente à rede social Twitter [50].

Consiste num conjunto de ferramentas OSS que, fazendo uso da linguagem CSS e *Javascript*, permite fornecer ao utilizador instrumentos (botões, tabelas, menus, caixas de dialogo, entre outros) que possibilitam a criação de páginas *Web* apelativas de forma fácil.

Para fazer uso desta basta fazer *download* dos ficheiros *bootstrap* e fazer a sua inclusão na página *Web*, tal como foi efetuado no CMS desenvolvido:

```
<head>
    <link rel="stylesheet" href="/css/bootstrap.min.css">
    <script src="/js/bootstrap.min.js"></script>
</head>
```

4.2.1.4. PHP

Criada em 1995 por Rasmus Lerdorf, consiste numa linguagem interpretada e executada do lado do servidor [51]. Esta permite tornar páginas *Web* dinâmicas sendo usada para ligação à DB [52].

Esta linguagem foi usada no desenvolvimento do CMS recorrendo à *framewrok CakePHP* detalhada na secção 4.2.2.

4.2.1.5. MySQL

MySQL consiste num sistema de gestão de DB criado por David Axmark, Allan Larsson e Michael Widenius. Este faz uso da linguagem *Structured Query Language* (SQL) responsável por criar *queries* que permitem obter os dados na formatação desejada. Assente na licença *General Public License* (GPL) [53] é compatível com qualquer tipo de plataforma.

4.2.1.6. JavaScript

Para melhorar as interações entre o CMS e o utilizador foi utilizada a linguagem *Javascript*. Esta foi usada em validação de formulários, antes de serem enviados ao servidor, alteração de estilos de forma dinâmica, alteração de conteúdo na página consoante a ação do utilizador, entre outros.

4.2.1.7. jQuery UI

jQuery consiste numa tecnologia que utiliza a linguagem *Javascript*. Lançada em 2006, é composta por uma biblioteca que deve ser incluída no projeto de modo a que as funções desta fiquem disponíveis em *Javascript*. Através desta pode-se atribuir eventos, definir efeitos, alterar ou criar elementos na página entre outras ações. Foi ainda utilizada a biblioteca *jQuery UI* que, através da utilização de recursos da *jQuery*, permite abstrair o programador da construção de interações, animações, efeitos avançados e *widgets*.

4.2.1.8. AJAX

Asynchronous Javascript and XML (AJAX) faz uso de tecnologias como *Javascript* e *eXtensible Markup Language* (XML) de modo a tornar páginas Web mais interativas com o utilizador através de atualizações assíncronas entre o cliente e o servidor. Atualmente em detrimento do XML pode ser utilizada a estrutura *JavaScript Object Notation* (JSON).

Apresenta-se de seguida a diferença da estrutura de dados em XML e JSON:

JSON	<pre>var filme = { "titulo" : "X-Men Origens: Wolverine", "genero" : "Aventura", "duracao" : "107", "ano" : "2009" }</pre>
XML	<pre><?xml version='1.0' encoding='utf-8'?> <filme> <titulo>X-Men Origens: Wolverine</titulo> <genero>Aventura</genero> <duracao>107</duracao> <ano>2009</ano> </filme></pre>

Esta tecnologia tem como objetivo dar responsabilidade ao cliente e não sobrecarregar o servidor com demasiada informação sempre que existe um pedido. Na primeira comunicação o servidor envia todas as informações que o cliente necessita e nas restantes o cliente apenas efetua o pedido de dados que deseja para aquele instante [54].

4.2.2. Framework CakePHP

CakePHP consiste numa *framework* escrita em PHP tendo como objetivo oferecer uma estrutura que possibilite aos programadores desenvolverem aplicações robustas, de forma rápida, sem perderem flexibilidade.

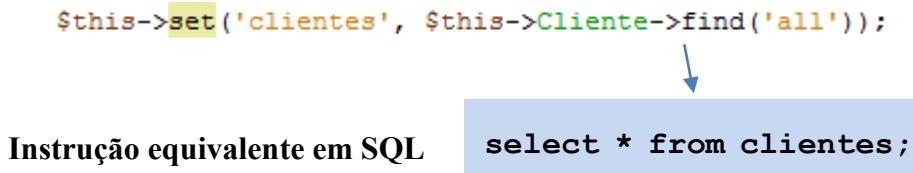
Criado em 2005 sob a licença MIT, todo o seu desenvolvimento assenta no padrão *Model, View Controller* (MVC), tendo respeitado este padrão todo o desenvolvimento do CMS.

A camada *Model* (modelo) destina-se a implementar a parte lógica da aplicação sendo responsável por obter os dados, bem como processar, validar e efetuar associações. No caso concreto da *framework* *CakePHP* é nesta camada onde se encontram definidas as regras de associação de tabelas, bem como regras de validação de formulários.

```
public $validate = array(
    'cliente_id' => array(
        'numeric' => array(
            'rule' => array('numeric'),
            'message' => 'Your custom message here',
            'allowEmpty' => false,
            'required' => false,
        ),
    ),
);
public $belongsTo = array(
    'Loja' => array(
        'className' => 'Loja',
        'foreignKey' => 'loja_id',
        'conditions' => '',
        'fields' => '',
        'order' => ''
    ),
);
...

```

A camada *Controller* é responsável por requisitar informações à camada *Model* e enviar dados para a camada *View*. Na *framework* *CakePHP* as *querys* SQL são executadas nesta camada, pedindo informação de associação de tabelas à camada *Model* e enviando o resultado da *query* através de uma variável para a camada *View*. O código seguinte mostra a instrução *select* em SQL de todos os clientes, encapsulada na função *find* dada pelo *CakePHP* bem como a passagem da variável “clientes” com os respetivos valores para a *View* através da função *Set*.



Uma vez que as *querys* à DB são efetuadas através de classes e funções específicas da framework CakePHP, com privilégios específicos, a plataforma encontra-se protegida a ataques, como por exemplo: *SQL injection*.²

A camada *View* tem como objetivo efetuar a amostragem dos dados previamente processados pela camada *Controller*. Por exemplo, no caso de se desejar apresentar todos os clientes registados na plataforma, basta fazer uso da variável “clientes” previamente instanciada no *Controller* e através de um ciclo percorrer o respetivo *array*, tal como é mostrado seguidamente.

```
<?php foreach ($clientes as $cliente): ?>
    <tr>
        <td><?php echo h($cliente['Cliente']['nome']); ?></td>
    </tr>
<?php endforeach; ?>
```

Na framework *CakePHP*, para cada tabela da DB é criado um *Model*, respetivo *Controller* e *View* sendo as operações sobre cada tabela efetuadas através da classe respetiva. A forma como essa criação é feita será enunciada no capítulo 5 (secção 5.4.1). A Figura 4.2 apresenta a arquitetura da framework *CakePHP*.

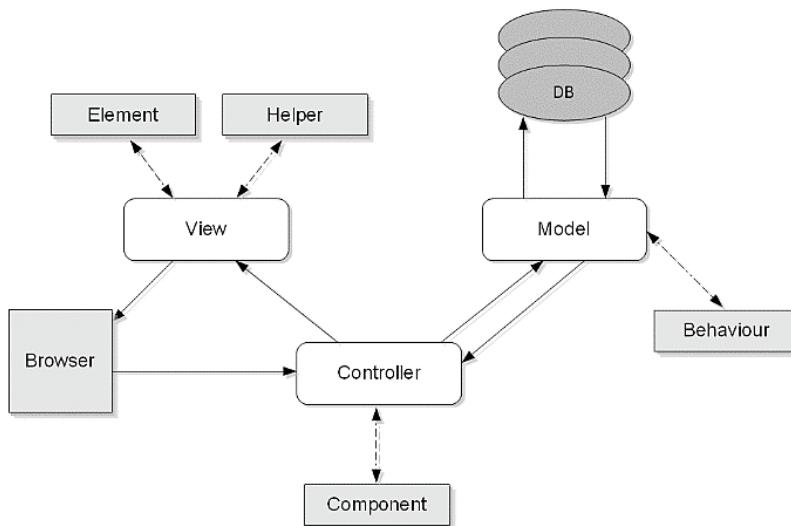


Figura 4.2 – Arquitetura de funcionamento da framework *CakePHP*

² SQL injection consiste na inserção de instruções SQL dentro de uma *query* à DB através da manipulação das entradas de dados de uma aplicação. Por exemplo, fazendo uso de *inputs* de formulários em HTML é possível introduzir uma *query* em SQL sendo executada normalmente caso o sistema não esteja preparado contra este tipo de ataques.

4.3. Metodologia de desenvolvimento

Foi adotada como metodologia de desenvolvimento de *software* uma versão adaptada de *Scrum*, abrangendo as componentes que faziam sentido tendo em conta o âmbito e situação do projeto.

4.3.1. SCRUM

Scrum consiste numa metodologia ágil de desenvolvimento de *software* tendo como objetivo estabelecer um conjunto de regras de forma a que durante toda a fase de desenvolvimento sejam cumpridas todas as metas estabelecidas [55].

Nesta metodologia, um projeto é dividido em ciclos, normalmente mensais tendo como nome *sprints*. Cada *sprint* possui um *deadline* cujos requisitos estabelecidos devem ser cumpridos até essa data.

Todos os dias é efetuada uma reunião, denominada *Daily Scrum Meeting*, tendo como objetivo indicar o que foi feito no dia anterior, identificar problemas e estabelecer prioridades para o dia que se inicia.

No final da cada *sprint* é efetuada uma *Sprint Retrospective* iniciando-se o planeamento da próxima *sprint*. A Figura 4.3 ilustra as diversas etapas da metodologia *Scrum*.

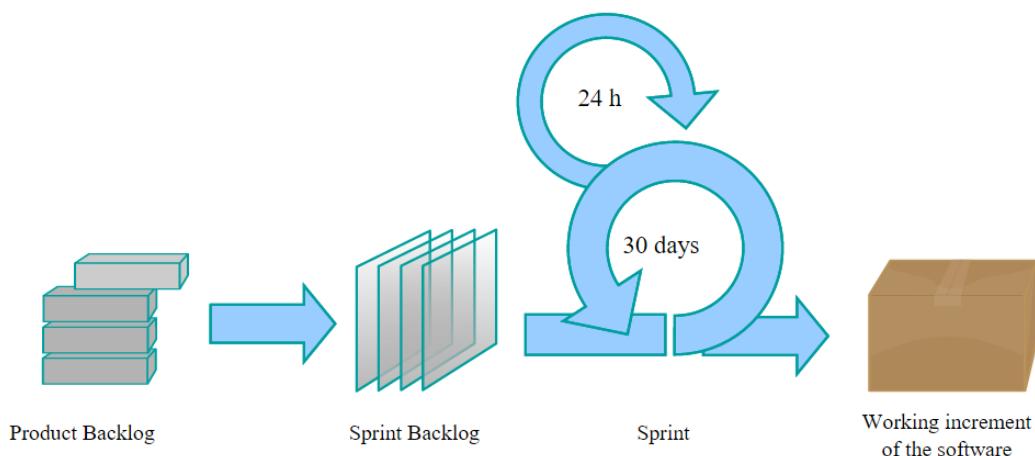


Figura 4.3 – Funcionamento da metodologia Scrum

Assim, durante o desenvolvimento do projeto, foram efetuadas reuniões diárias (*Daily Scrum Meeting*) entre o programador (Nuno Paiva) e o gestor de equipa (*Scrum Master* – Eng. Francisco Maia) tendo como objetivo identificar e resolver problemas, bem como planear o desenvolvimento de cada dia, em horário pós-laboral. Mensalmente foram efetuadas reuniões (*Sprint Review Meeting*) entre os mesmos elementos tendo em vista, por parte do programador, apresentar os requisitos implementados durante a respetiva *sprint*, bem como, definir os requisitos a implementar na que se inicia.

4.4. Arquitetura de desenvolvimento do projeto

O projeto iniciou-se com o desenvolvimento única e exclusivamente na máquina local. Com o avanço da solução em grandeza e complexidade, foi necessário o uso de um servidor com maior capacidade de processamento, tornando a solução acessível remotamente.

Como forma de agilizar o processo de desenvolvimento, e sincronismo de versões entre a máquina local e o servidor de produção, foi usado o sistema de controlo de versões GIT.

4.4.1. Sistema de controlo de versões - GIT

Criado por Linus Torvalds (7 de abril de 2005), no âmbito do desenvolvimento do *kernel* do Linux, o sistema GIT consiste num *software* de controlo de versões que permite armazenar código de um dado projeto de forma centralizada, mantendo um histórico de todas as alterações. Para além disso, permite que não ocorram conflitos resultantes de vários utilizadores manipularem o mesmo ficheiro em simultâneo, possibilitando recuar, a qualquer instante, para uma versão anterior [56].

Funcionamento base do sistema GIT

Para inicializar um repositório GIT, basta, na pasta do projeto usando a linha de comandos `git bash` executar o comando: **`git init`**.

Caso o projeto já exista remotamente, e se deseje obter o projeto no seu todo é necessário efetuar o comando:

`git clone user@servidor:/caminho/para/o/repositório.`

De forma a adicionar os ficheiros alterados ao repositório local é necessário efetuar os comandos:

- **Git add .**
 - Permite indicar quais os ficheiros que serão adicionados ao repositório local;
 - **Git commit -m "Corrigido bug na área de login"**
 - Permite criar uma nova versão do projeto, enviando os ficheiros alterados desde o último *commit*.

A arquitetura do sistema GIT, representada na Figura 4.4, permite que o utilizador possua o seu repositório local com diversas versões do projeto (vários *commits*), mas também tenha um repositório central onde são agregadas todas as versões.

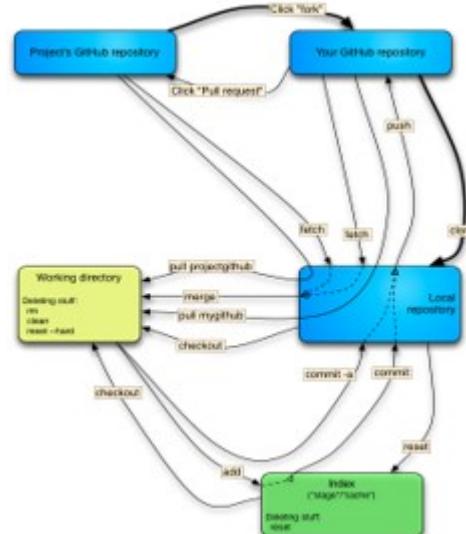


Figura 4.4 – Arquitetura do sistema GIT

Para enviar o código para o repositório remoto é necessário executar o comando: **git push origin master**.

De modo a conseguir enviar o código para um repositório remoto é necessário:

1. Possuir o URL do repositório GIT remoto, no repositório local.

a . Efetuado através do comando:

```
git remote add origin  
user@servidor:/caminho/para/o/repositório
```

2. Possuir dados de autenticação do servidor remoto, podendo usar:

a. Utilizador e password;

b. Chave pública SSH. Necessário gerar uma chave SSH composta por chave privada e chave pública, colocando a chave pública no servidor GIT remoto.

4.4.2. Descrição da arquitetura de desenvolvimento

Para a solução desenvolvida foram criados dois repositórios remotos:

1. CMS

a. URL: git@git.streamline.pt:s1/cms_beacons.git

2. Aplicação Móvel:

a. URL: git@git.streamline.pt:s1/app_beacons.git

A **Streamline** possui um servidor dedicado para o sistema GIT de nome: **git.streamline.pt**. Os URL's anteriores correspondem aos repositórios remotos de ambos os projetos nesse servidor.

Para a componente CMS, dado ser uma plataforma *Web* e de forma a estar acessível remotamente, foi usado outro servidor de nome: **beta.streamline.pt**. Este consistiu no servidor de produção encontrando-se nele a versão estável do CMS, incluindo a DB.

Assim, sempre que se pretendia enviar novas alterações para o repositório local na máquina eram executados os comandos:

- git add .
- git commit -m "Texto de identificação das alterações"

Para enviar as alterações para o repositório central, (para o servidor remoto: `git.streamline.pt`) era executado o comando:

- git push origin master

No caso da componente CMS, para obter no servidor de produção (`beta.streamline.pt`) a última versão do projeto, era necessário aceder a este remotamente via SSH (através do software **PUTTY**), e executar o comando:

- git pull origin master

A Figura 4.5 apresenta a arquitetura de desenvolvimento da solução mostrando o *workflow* e comunicação dos diversos servidores usados.

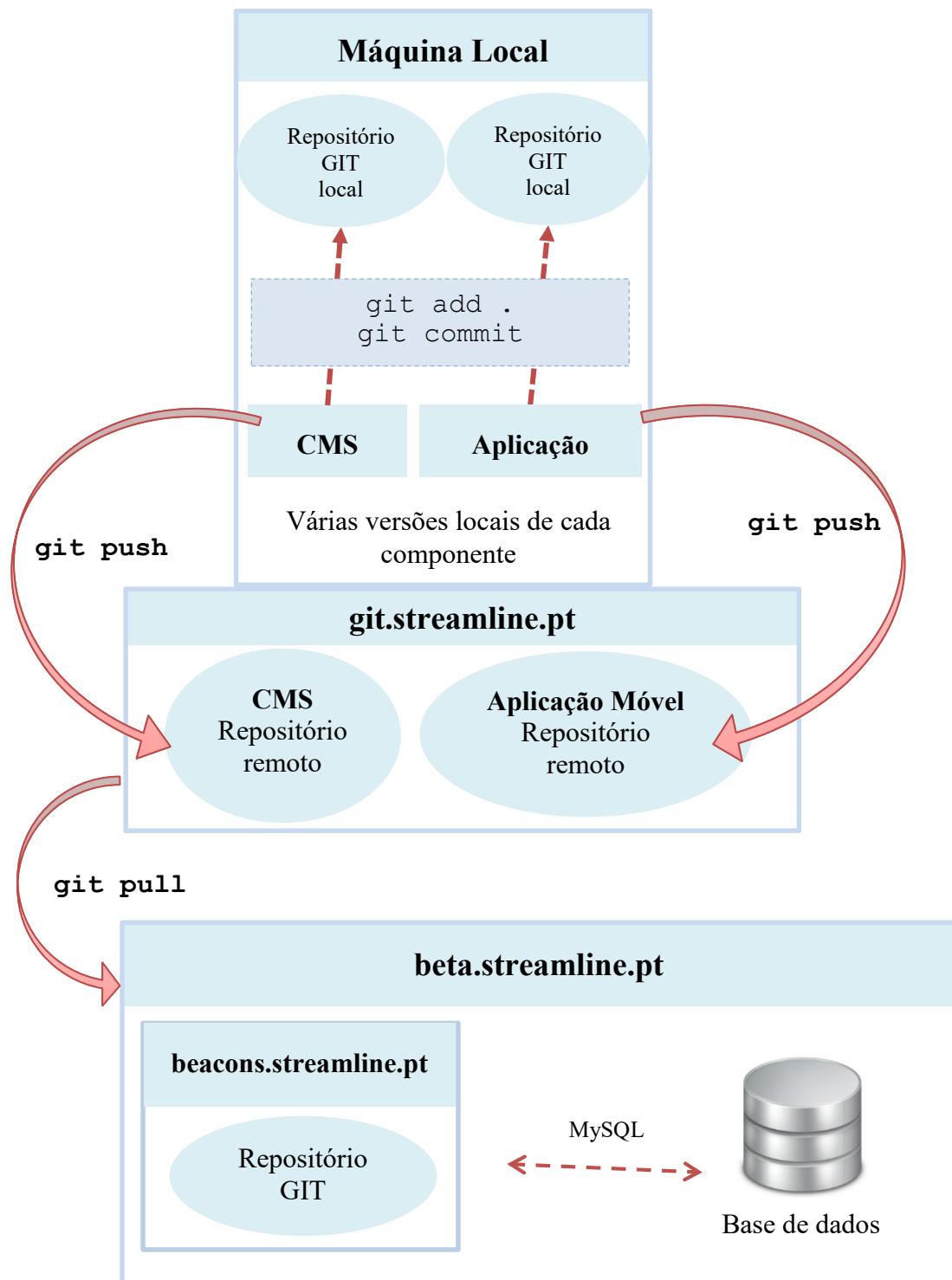


Figura 4.5 – Arquitetura de desenvolvimento da solução

5. SISTEMA DE GESTÃO DE CONTEÚDOS

Este capítulo foca-se na explicação do desenvolvimento do CMS, dos principais módulos implementados, e está organizado em 4 subcapítulos: Descrição da componente; Requisitos do CMS; Casos de utilização e Concepção do CMS.

5.1. Descrição da componente

Um CMS consiste “*uma plataforma usada para criar, editar, gerir e publicar conteúdo de forma consistente e organizada permitindo que o mesmo seja modificado, removido e adicionado com facilidade*”.

Sendo uma plataforma *Web*, destina-se a ser utilizada única e exclusivamente pelos gestores / lojistas da área do retalho.

Este consiste na componente principal de toda a solução, possibilitando ao retalhista inserir os seus clientes, criar e enviar as suas campanhas e efetuar a análise de dados resultante da interação do cliente com as mesmas.

Todo o funcionamento passa por esta componente, ou seja, sempre que o dispositivo de um cliente se aproxima de um *beacon*, este irá enviar essa informação ao CMS, recebendo seguidamente a informação que lhe diz respeito tendo em conta os dados enviados. Assim, o CMS consiste na componente que controla toda a solução efetuando funções de processamento e dando respostas às aplicações que entram em contato com este após aproximação de determinado *beacon*.

5.2. Requisitos do CMS

Os requisitos do CMS foram estabelecidos tendo por base os seguintes três tipos: requisitos de interface, não funcionais e funcionais.

No que diz respeito aos requisitos de interface, foi definido com a máxima prioridade a adaptação automática do *layout* do CMS a diversos dispositivos (*smartphones*, *tablets*, computadores e outros) implementando o conceito de *responsive layout*.

Como requisitos não funcionais, estabeleceu-se como prioridade elevada a compatibilidade do CMS com os *browsers* mais usados, assentando o seu desenvolvimento única e exclusivamente em tecnologias e linguagens *standard* e OSS.

Tendo em conta o tipo de requisitos funcionais definiu-se como prioridade máxima a implementação dos módulos de gestão de utilizadores, lojas, campanhas, *beacons* e análise de dados. Sendo uma solução com o objetivo de funcionar em paralelo e de forma independente em diversas lojas, a segmentação dos dados, partilha e visibilidade de módulos teve uma prioridade igualmente elevada tendo sido definidos três níveis de utilizadores: lojista, gestor e administrador. Igualmente importante consistiu a implementação do módulo de campanhas

permitindo o envio de campanhas *Standard*, *Web* e integração com o sistema *Passbook*. Definiu-se ainda que o módulo de análise de dados seria composto por seis componentes, nomeadamente: *Dashboard* geral; *Dashboard* por campanha; *Dashboard* por região; *Dashboard* por cliente; *Percorso dos clientes* e *Heatmap*;

A descrição detalhada de cada um dos requisitos, considerando separadamente requisitos de interface, não funcionais e funcionais, encontra-se no Apêndice II.

5.3. Casos de utilização

O CMS foi concebido com o objetivo de ser usado por diversos utilizadores. Cada utilizador destina-se a efetuar uma determinada ação e a visualizar um determinado conjunto de dados de acordo com os seguintes níveis de acesso: Lojista; Gestor; Administrador.

A Figura 5.1 mostra o diagrama de casos de uso do CMS.

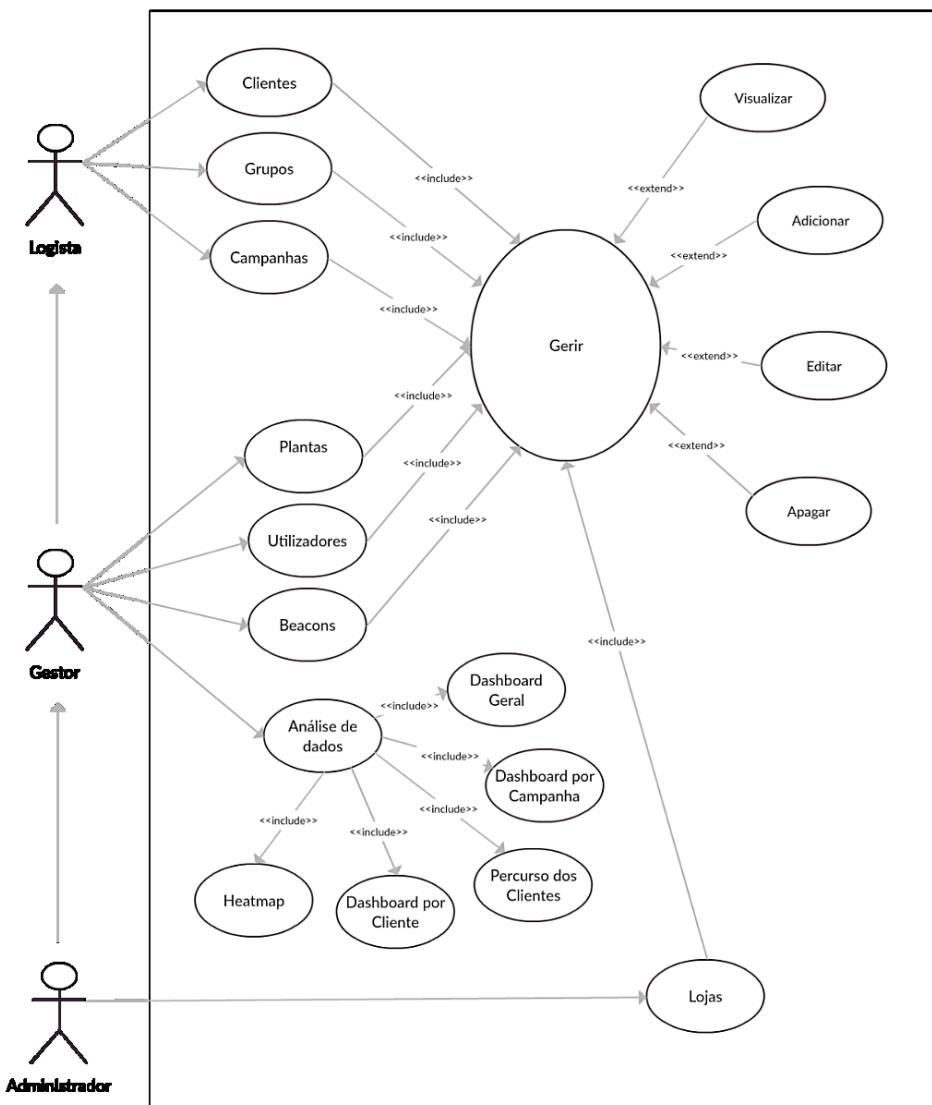


Figura 5.1 – Diagrama de caso de uso do CMS

Autor Lojista

Foi estipulado que um utilizador com perfil “lojista”, para além de ter acesso ao conteúdo relativamente à loja a que se encontra associado, apenas terá acesso à área de gestão de clientes, grupos e campanhas. Para cada uma dessas áreas o utilizador terá permissões de leitura e escrita, conseguindo listar, adicionar, editar e apagar todo o conteúdo relacionado.

Autor Gestor

Um utilizador com perfil “Gestor” destina-se a ser utilizado pelo gestor de loja e pela área de administração.

Para além de herdar todas as permissões do utilizador “lojista”, permite aceder à área de gestão, podendo gerir *beacons*, plantas, utilizadores, tendo ainda permissão para aceder à área de análise de dados.

A área de análise de dados permite dar acesso a seis componentes, nomeadamente: *dashboard* geral, *dashboard* por campanha, *dashboard* por região, *dashboard* por cliente, percursos dos clientes e *heatmap*. Na seção **5.4.3.10** serão detalhadas as potencialidades de cada módulo.

Este tipo de utilizador deve ser usado para efetuar a criação de contas do tipo “lojista”, deixando a cargo deste todas as tarefas relacionadas com clientes e campanhas.

Autor Administrador

O utilizador do tipo “Administrador” destina-se a ser utilizado única e exclusivamente por membros de desenvolvimento e gestão do CMS, ou seja, pela **Streamline**.

Para além de herdar todas as funcionalidades do utilizador “Gestor” permite visualizar o conteúdo de qualquer loja registada no CMS.

Apenas este utilizador possui acesso ao módulo “Lojas” onde é possível listar, adicionar, editar e remover lojas do CMS.

Permite ainda criar utilizadores do tipo “Gestor”.

5.4. Conceção do CMS

Esta seção descreve os principais passos ocorridos no desenvolvimento do CMS abordando as suas principais funcionalidades, configurações e uso.

5.4.1. Criação da base de dados e integração da framework CakePHP

Após o levantamento dos requisitos, o próximo passo consistiu na definição da DB, nomeadamente na construção do modelo de entidade e relacionamento, apresentado no Apêndice V.

Depois da criação do modelo, e com recurso ao *software MySQL Workbench*, foi criada a DB localmente na máquina de desenvolvimento. Seguidamente procedeu-se à integração da framework *CakePHP* com a DB.

Para isso, já com o serviço Apache e MySQL em execução na máquina (através do *software XAMPP*), foi necessário preencher a variável *default* da classe “*Database_config*” do *CakePHP* de forma a associar à DB que se deseja.

```
class DATABASE_CONFIG {  
  
    public $default = array(  
        'datasource' => 'Database/Mysql',  
        'persistent' => false,  
        'host' => 'localhost',  
        'login' => 'user',  
        'password' => '[password]',  
        'database' => 'beacons',  
        'prefix' => '',  
        'encoding' => 'utf8',  
    );  
  
    ...  
}
```

De modo a fazer uso do modelo MVC foi necessário criar os modelos, vistas e controladores necessários para o funcionamento da *framework*.

O *CakePHP* fornece um *script* (*cake.php*) que, através da receção dos argumentos *bake all* permite criar de forma automática um modelo, controlador e vista para a tabela da DB selecionada.

Assim, com recurso à linha de comandos, foi executado o *script* da seguinte forma: `php cake.php bake all`, devolvendo todas as tabelas existentes na DB, tal como mostra a Figura 5.2.

```
Welcome to CakePHP v2.4.6 Console
App : app
Path: C:\xampp\htdocs\projects\elupsa-bo\backoffice\app\
Bake All

Possible Models based on your current database:
 1. Beacon
 2. Campanha
 3. CampanhasHasBeacon
 4. CampanhasHasGrupo
 5. Cliente
 6. ClientesCampanha
 7. CodigosPostal
 8. Concelho
 9. Dispositivo
10. Distrito
11. Grupo
12. GruposHasCliente
13. Horario
14. Layout
15. Localidade
16. Loja
17. LojasHasCliente
18. PercursosCliente
19. Plant
20. Role
21. TiposCampanha
22. User
Enter a number from the list above,
type in the name of another model, or 'q' to exit
[q] >
```

Figura 5.2 – Todas as tabelas existentes na DB

Por exemplo, escolhendo a opção 1 o *script* iria criar um modelo, controlador e vista associado à tabela *Beacon*.

Concluído o processo para todas as tabelas, o CMS encontra-se a fazer uso do modelo MVC estabelecido pelo *CakePHP*. Todas as operações sobre tabelas são efetuadas através das suas classes associadas.

5.4.2. Desenvolvimento do aspeto

Para a criação do *layout* foi usada a *framework bootstrap* em conjunto com bibliotecas auxiliares, tais como, *jQuery UI*, *bootstrap dialogs*, *bootstrap buttons*, entre outros.

O *layout* foi desenvolvido tendo como objetivo a compatibilidade com qualquer tipo de dispositivo, implementando o conceito de *responsive*.

Foram adotadas técnicas em CSS 3 bem como adaptadas classes do *bootstrap* de forma a ajustar o *layout* consoante o tamanho do ecrã:

```
@media screen and (max-width: 1000px) {
    .nebula {
        width: 100%;
        overflow: hidden;
        margin: 0 -150px 0 -150px;
    }
}
```

O código acima permite atribuir características à classe “nebula” apenas se o ecrã tiver uma largura máxima de 1000 pixels.

As Figuras 5.3, 5.4, 5.5 e 5.6 apresentam o aspetto da página de login do CMS para os dispositivos: PC, *Smartphone* e *Tablet* (aspeto horizontal e vertical). As Figuras 5.7, 5.8, 5.9 e 5.10 dizem respeito à página principal para os mesmos dispositivos.

Página de login

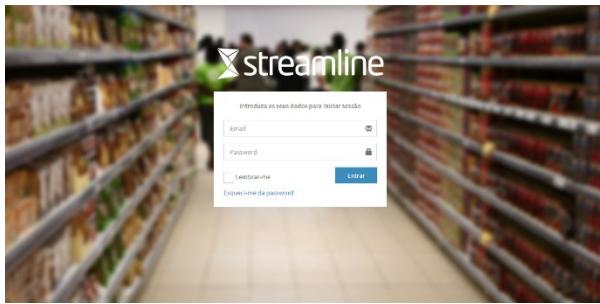


Figura 5.4 – Página de login: Versão PC



Figura 5.3 – Página de login:
Versão Smartphone

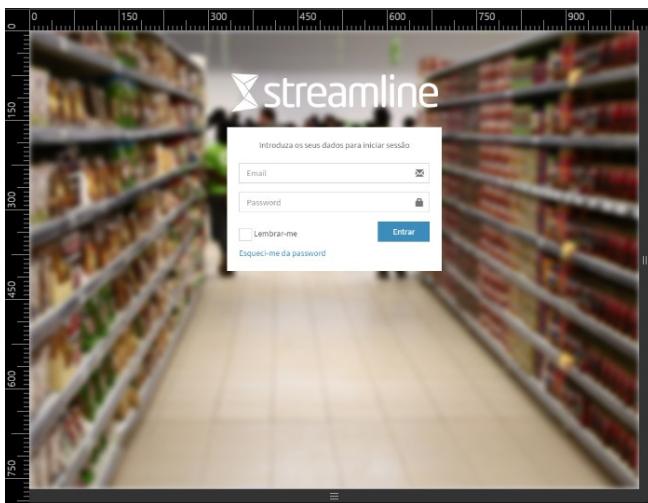


Figura 5.5 – Página de login: Versão Tablet
(horizontal)

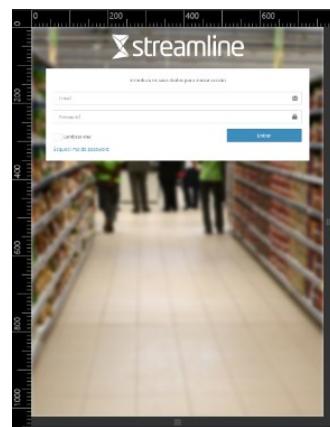


Figura 5.6 – Página de login:
Versão Tablet (vertical)

Página principal

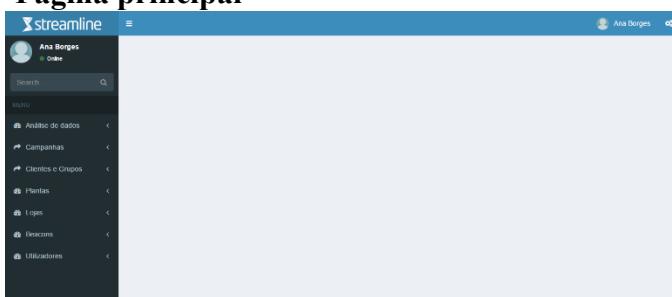
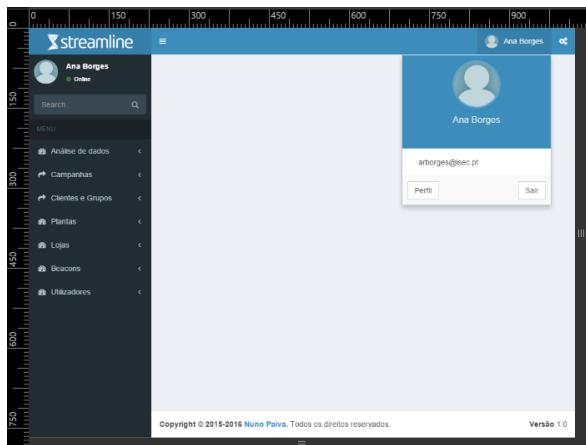


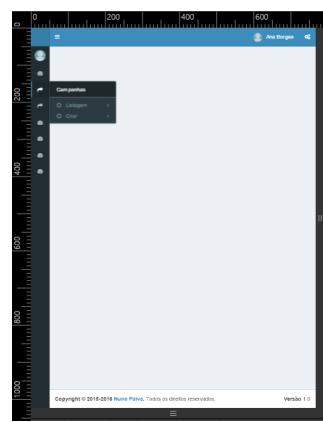
Figura 5.8 – Página principal: Versão PC



Figura 5.7 – Página principal:
Versão Smartphone



*Figura 5.10 – Página principal: Versão
Tablet (horizontal)*



*Figura 5.9 – Página principal:
Versão Tablet (vertical)*

5.4.3. Explicação das principais funcionalidades

Esta secção aborda as principais funcionalidades do CMS, descrevendo desde a sua implementação à sua utilização.

Identifica ainda de que forma a solução implementada se distingue de outras semelhantes no mercado.

5.4.3.1. Lojas

O CMS suporta a existência de várias lojas registadas em simultâneo, possuindo cada uma os seus dados.

As funcionalidades de criação, listagem, edição e remoção de lojas apenas estão disponíveis ao utilizador do tipo “Administrador”.

O uso da solução inicia-se com o registo de uma loja no CMS. Após contacto e compra da mesma, o registo é feito obrigatoriamente pelo administrador da plataforma, neste caso a **Streamline**.

Como dados de identificação de uma loja foram estabelecidos os seguintes: Nome, Descrição e Localização (composto por: País, Distrito, Concelho, Freguesia e Rua). Todos esses dados devem ser comunicados previamente à **Streamline** responsável pelo seu registo.

O registo da localização da loja é efetuado com recurso à API do *Google Maps*. Esta foi implementada (Figura 5.11) de forma a facilitar a localização exata da loja. Assim, através da inserção dos dados de localização no formulário, o mapa irá pesquisar automaticamente a localização devolvendo os valores de latitude e longitude, guardados posteriormente em DB.

Tecnicamente o registo de uma loja consiste no preenchimento de um formulário existente numa vista criada para o efeito (*add*). Os dados inseridos são enviados pelo método POST para o respetivo controlador *LojasController* e função associada *add()*. Caso a função seja chamada por POST então irá guardar os dados na tabela *lojas* com recurso ao modelo *Loja* e usando a função *save()* da framework *CakePHP*.

```
public function add() {
    if ($this->request->is('post')) {
        $this->Loja->create();
        if ($this->Loja->save ($this->request->data)) {
            $this->Session->setFlash(__('Registro inserido com sucesso'));
            return $this->redirect(array('action' => 'index'));
        } else {
            $this->Session->setFlash(__('Erro na inserção'));
        }
    }
}
```

Dados recebidos por POST

Função do *CakePHP* equivalente à seguinte instrução em SQL:
INSERT INTO lojas (Nome) VALUES ("Exemplo");

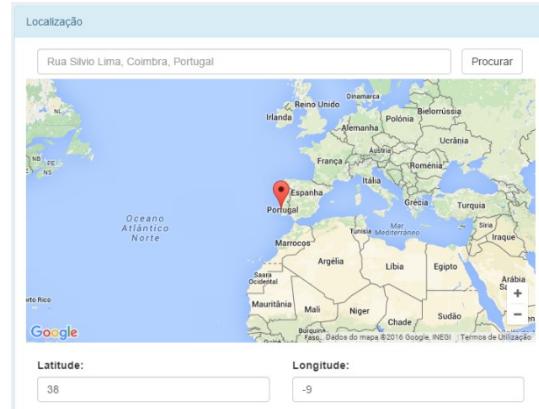


Figura 5.11 – API do Google Maps

De modo a criar a listagem de lojas apresentada na Figura 5.12, criou-se uma função *index()* pertencente ao controlador *LojasController* e respetiva vista com o mesmo nome. Na função *index()*, existente no controlador, é efetuada a *query* que permite listar todas as lojas registadas passando o valor obtido sob a forma de *array* para a vista. A vista *index* (como todas as outras) limita-se a mostrar as variáveis que recebe não interagindo em nenhum momento com a DB.

Listagem				Ações	
ID	Designação	Descrição	Ações		
1	Continente	Continente			
2	Elcorte Ingles	Elcorte Ingles			
3	Lidl	Lidl			
4	Jumbo	Jumbo			
5	Minipreço	Minipreço			
6	Zara	Zara			

Página 1 de 1. Apresentando 6 registo(s), de um total de 6.

Figura 5.12 – Listagem de lojas

Controlador LojasController

```
public function index() {
    $options = array('recursive' => -1);
    $lojas = $this->Loja->find('all', $options);
    $this->set('lojas', $lojas);
}
```

Permite enviar a variável *lojas* para a vista associada

Equivalente à seguinte instrução em SQL:

```
# SELECT * from lojas;
```

Vista Lojas/index

```
<table class="table table-striped table-condensed">
    <tr>
        <th><?php echo $this->Paginator->sort('id'); ?></th>
        <th><?php echo $this->Paginator->sort('designacao'); ?></th>
        <th><?php echo $this->Paginator->sort('descricao'); ?></th>
    </tr>
    <?php foreach ($lojas as $loja): ?>
        <tr>
            <td><?php echo h($loja['Loja']['id']); ?></td>
            <td><?php echo h($loja['Loja']['designacao']); ?></td>
            <td><?php echo h($loja['Loja']['descricao']); ?></td>
        </tr>
    <?php endforeach; ?>
</table>
```

Desta forma a camada de apresentação encontra-se separada da camada de dados trazendo maior segurança, robustez, organização e reutilização de código.

5.4.3.2. Utilizadores

Para que uma loja utilize o CMS é necessário que possua uma conta de utilizador.

É da responsabilidade da **Streamline** a criação de utilizadores do tipo *Gestor*, aquando da venda da solução, de forma a que possuam autonomia para usar o CMS.

Quando é registado um utilizador do tipo *Gestor*, é necessário indicar: a que loja pertence; Nome; *Email*, *Password* e Telemóvel.

Tecnicamente foi definido que um utilizador deve pertencer obrigatoriamente a uma loja e a um perfil, tal como apresentado pela relação entre as três tabelas na Figura 5.13.

Após a criação da conta de gestor, os dados são enviados para o *Email* inserido, conseguindo de imediato aceder ao CMS.

Também o gestor possui privilégios para criar utilizadores, mas apenas do tipo *lojista*. Para além disso, os utilizadores que ele criar apenas

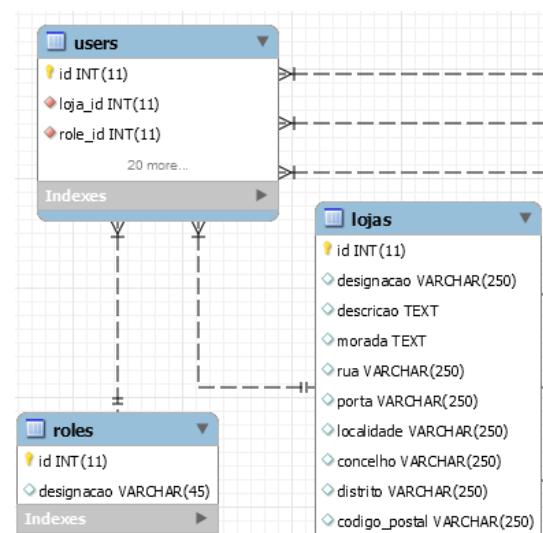


Figura 5.13 – Relação entre as tabelas *users*, *lojas* e *roles*

conseguirão visualizar o conteúdo da loja a que o gestor se encontra associado. Assim, garante-se total independência no funcionamento do CMS com várias lojas a fazer o seu uso em simultâneo.

Limitação de dados por tipo de utilizador

Quando um utilizador efetua login no CMS é guardado na variável de sessão os seus dados, sendo esse processo feito de forma automática através da ativação do componente *Auth* no *AppController* do *CakePHP*.

```
class AppController extends Controller {

    public $components = array(
        'Session',
        'Auth',
    );
    ...
}
```

De forma a restringir o acesso a determinados módulos, ou limitar a listagem aos dados pertencentes à loja do qual o utilizador faz parte, foi necessário aceder à variável de sessão *Auth*, verificar qual a loja e perfil que possui e, consoante esses dados, conceder permissão para aceder a determinado módulo, ou introduzir uma condição na *query* de forma a apresentar apenas os dados relativos à sua loja (tal como é exemplificado no seguinte código).

```
$session = $this->Session->read('Auth');

if ($session['User']['role_id'] == 1){ // Administrador
    // Visualiza todos os clientes
    $clientes = $this->Cliente->find('all');

}elseif ($session['User']['role_id'] == 2) { // Gestor
    // Mostra os clientes da sua loja.
    $options = array(
        'conditions' => array(
            'Cliente.loja_id' => $session['User']['loja_id']));
    $clientes = $this->Cliente->find('all',$options);

}elseif ($session['User']['role_id'] == 3) { // Logista
    // Mostra os clientes da sua loja
    $options = array(
        'conditions' => array(
            'Cliente.loja_id' => $session['User']['loja_id']));
    $clientes = $this->Cliente->find('all',$options);

...
}
```

5.4.3.3. Beacons

Acessível apenas ao utilizador *Gestor*, o CMS possui um módulo onde é possível registar os *beacons* de uma determinada loja.

De modo a que o CMS consiga identificar o *beacon*, nas comunicações que receber da aplicação móvel, é necessário que todos os *beacons* colocados em funcionamento sejam inseridos no CMS.

Assim, o primeiro passo a efetuar pelo utilizador *Gestor*, consiste na inserção dos dados de cada *beacon* no CMS. Ao selecionar o menu *Beacons* → *Adicionar*, deve ser escolhida a norma a usar pelo *beacon*, nomeadamente: *iBeacon* ou *Eddystone* (embora ainda em fase de testes o CMS encontra-se em constante adaptação à tecnologia recentemente lançada pela Google). No passo seguinte irão ser pedidos parâmetros de configuração de acordo com a norma selecionada. A Figura 5.14 apresenta os parâmetros pedidos em função da escolha da norma.

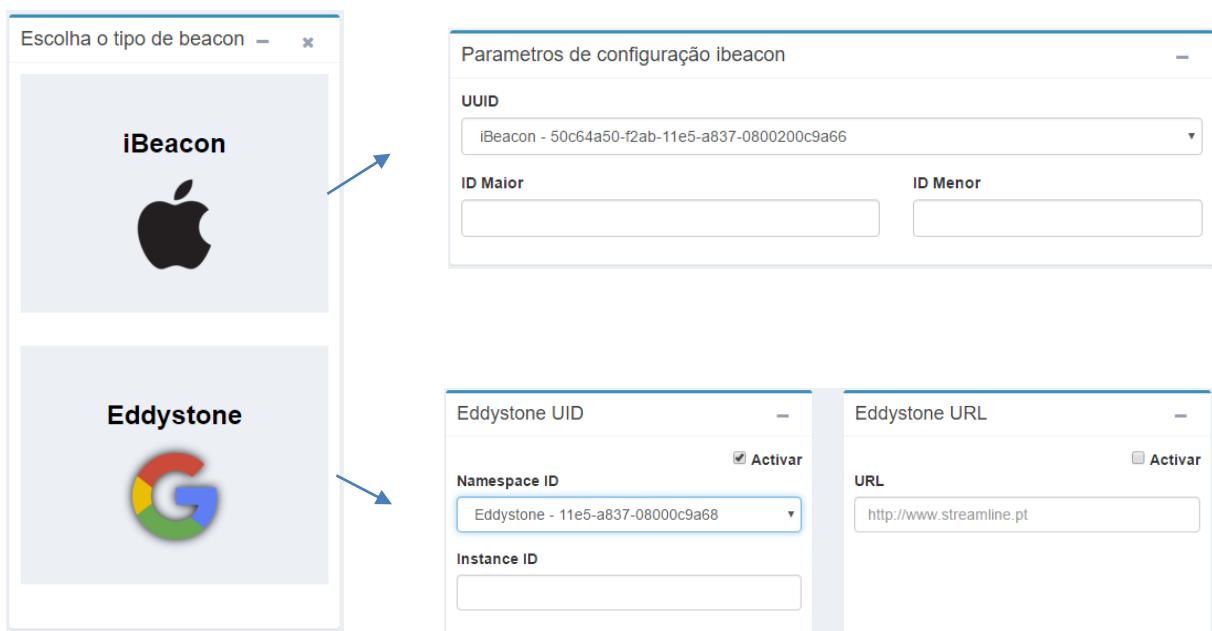


Figura 5.14 – Escolha da norma a usar pelo beacon no CMS

Independentemente da norma escolhida são ainda pedidos dados que permitam identificar o *beacon* no CMS, nomeadamente: fabricante, nome e imagem.

Por fim é possível a escolha dos valores para os parâmetros de difusão: *advertising interval* e *Transmit Power*. De forma a estarem de acordo com a norma, no capítulo 3 (secção 3.3.4), estabeleceu-se um conjunto de valores sendo escolhidos com recurso a um *slider*. É igualmente possível a escolha do nível de bateria, em percentagem, de forma a obter-se uma estimativa do tempo de vida do *beacon*, de acordo com os parâmetros inseridos.

A Figura 5.15 ilustra a forma como o utilizador escolhe os parâmetros de difusão para cada *beacon* no CMS.

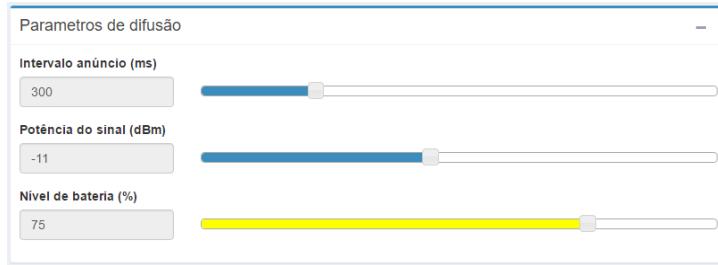


Figura 5.15 – Parâmetros de difusão dos beacons no CMS

Cada *beacon* é associado automaticamente à loja a que o utilizador pertence, sendo acrescentado o ID da loja ao *beacon* antes da inserção dos dados na DB.

Assim, quando um utilizador clica no botão *adicionar* é invocada a função *add()*, existente no controlador *BeaconsController*, sendo intercetados os dados enviados via POST e acrescentado ao *beacon* o ID da loja a que pertence o utilizador logado.

```
public function add() {
    if ($this->request->is('post')) {
        $session = $this->Session->read('Auth');
        $this->Beacon->create();
        $aux = $this->request->data;

        $aux['Beacon']['loja_id'] = $session['User']['loja_id'];

        if ($this->Beacon->save($aux)) {
            $this->Session->setFlash(__('Beacon inserido com sucesso'));
            return $this->redirect(array('action' => 'index'));
        }
    ...
}
```

A inserção dos *beacons* no CMS deve ocorrer após alteração dos valores *UUID*, *Major* e *Minor* em cada *beacon* com recurso a aplicações móveis de gestão de *beacons*, tendo em conta os conceitos apresentados no capítulo 3 (seção 3.3.4).

Após inserção de todos os *beacons* no CMS, o utilizador *Gestor*, para além de os conseguir editar e remover, consegue ainda listar, na forma de tabela, todos os *beacons* inseridos, existindo ainda a possibilidade de efetuar uma ordenação por Nome ou ID, tal como mostra a Figura 5.16.

Listagem					
Id	Loja	Designacão	Imagen	Acções	
22	Zara	Roxo			
23	Zara	Verde			
24	Zara	Ciano			
25	Zara	Onyx Beacon 1			
26	Zara	Onyx Beacon 2			
27	Zara	Onyx Beacon 3			

Página 1 de 1. Apresentando 6 registo(s), de um total de 6.

Figura 5.16 – Listagem de beacons no CMS

5.4.3.4. Regiões de beacons

No CMS foi desenvolvido o módulo *regiões* tendo em conta o conceito teórico apresentado no capítulo 3 (secção 3.3.6). Este destina-se a efetuar o agrupamento de *beacons*, de acordo com a sua localização física, possibilitando ao utilizador uma maior versatilidade aquando da criação e distribuição de campanhas, bem como na posterior análise de dados.

Acessível apenas ao utilizador *Gestor*, é possível a criação, edição, listagem e remoção de um número ilimitado de regiões.

Aplicação prática

No caso de se desejar enviar a mesma campanha a todos os utilizadores após entrada na loja, (composta por diversas entradas) a forma admissível seria selecionar todos os *beacons* associados a essa zona. No entanto, se se tratar de uma cadeia de supermercados com diversas lojas em locais físicos diferentes seria inviável a seleção de todos os *beacons* individualmente.

Neste caso concreto (e de acordo com a norma *iBeacon*) o correto será definir uma região de *beacons* de nome *entrada* e associar a esta todos os *beacons* que lhe digam respeito, tendo em conta os valores: *Minor* (igual em várias lojas) e *Major* (diferente em várias lojas).

5.4.3.4.1. Associação de *beacons* a regiões

A associação de *beacons* a regiões é possível através do menu *Beacons* → *Regiões* (Figura 5.17) sendo apresentada uma área onde é possível a escolha da região desejada. Após escolher a região são apresentadas duas áreas: na primeira encontra-se uma tabela onde são listados todos os *beacons* pertencentes à região; na segunda encontram-se os *beacons* que ainda não foram associados a quaisquer regiões. Através de uma *dropdown* de múltipla escolha é possível selecionar qual(ais) o(s) *beacon*(s) que se deseja(m) associar (o processo técnico será abordado na seção 5.4.3.8.1), sendo que um *beacon* apenas pode estar associado a uma região.

#	Nome	UUID	Maior	Menor
1	Verde	D7:4A:56:AA:23:00	8960	22186
2	Ciano	D2:86:81:B7:FE:35	65077	33207
3	Onyx Beacon 2	EC:24:B8:18:1E:49	0	0
4	Onyx Beacon 1	EC:24:B8:17:FD:61	0	0

Quais os beacons a associar à região?

Adicione um ou mais beacons

Nenhum beacon seleccionado

Pesquisar...

Selecionar todos

Roxo

Onyx Beacon 3

Figura 5.17 – Associação de *beacons* a regiões

5.4.3.5. Plantas

Para além de permitir o uso de várias lojas em simultâneo, o CMS permite que cada loja possua diversos pisos ou áreas dentro do mesmo edifício.

Embora o conceito de planta se destine a representar um piso (ou uma área de um edifício), este também pode ser usado para representar várias lojas em diferentes localizações. Assim, é possível que uma loja, não consista apenas num edifício com uma única localização, mas possa ser uma cadeia de lojas com vários edifícios em localizações distintas.

A área de carregamento de plantas é acessível através do menu *Plantas* → *Adicionar* sendo apresentado o formulário da Figura 5.18.

Uma planta é identificada por: Nome, Piso e Imagem (nos formatos *gif*, *jpeg* ou *png* e com um tamanho máximo de 2 *Megabytes* (MB)).

À semelhança do que acontece com os *beacons*, cada planta carregada é associada automaticamente à loja a que pertence o utilizador logado.

Figura 5.18 – Adicionar uma planta no CMS

Tecnicamente o CMS permite que uma loja possua vários *beacons* e várias plantas. Por sua vez, um *beacon* e uma planta apenas pode pertencer a uma única loja. As tabelas e relações existentes na DB que suportam este funcionamento são apresentadas na Figura 5.19.

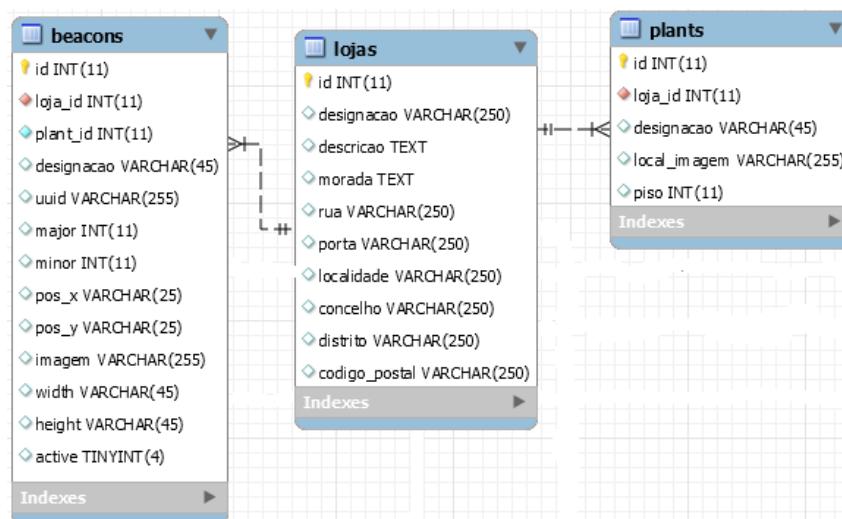


Figura 5.19 – Tabelas “beacons”, “lojas” e plants

5.4.3.6. Mapeamento de beacons

Após inserção dos *beacons* e plantas no CMS, o próximo passo consiste em associar cada *beacon* à planta carregada. Esse processo é possível através do menu *Plantas* → *Mapeamento de beacons*.

Esta ação apenas deve ser feita depois de efetuados estudos no local físico, verificando o alcance do sinal de cada *beacon* (tendo em conta objetos atenuadores), de modo a garantir o mínimo de domínios de colisão possível, tal como indicado no capítulo 3 (secção 3.3.10).

Este mapeamento consiste em informar ao CMS o local exato de cada *beacon* na loja física. Através deste será possível posteriormente efetuar análise de afluência de clientes por seção, bem como traçar um percurso (aproximado) efetuado por cada cliente na loja.

O módulo de mapeamento de *beacons* é composto por três áreas (Figura 5.20):

- ***Beacons disponíveis***: área onde são listados os *beacons* que não se encontram mapeados em nenhuma planta;
- **Planta**: local de escolha da planta onde serão mapeados os *beacons* disponíveis;
- **Remoção de *beacons***: área para onde devem ser arrastados os *beacons* de modo a desmapear-los de uma planta, tornando-os disponíveis.

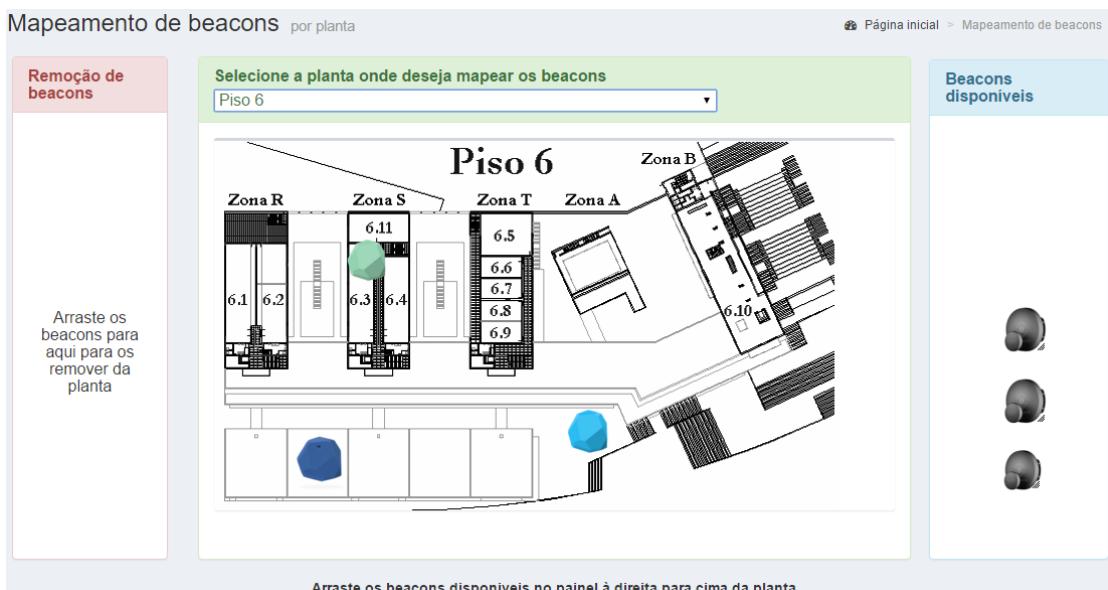


Figura 5.20 – Mapeamento de beacons

Funcionamento técnico

Cada *beacon* (na DB) para além de possuir os seus dados de identificação, inseridos pelo utilizador, possui uma determinada posição em relação ao eixo dos XX (campo *pos_x*) e posição em relação ao eixo dos YY (campo *pos_y*), tendo ainda uma relação com uma planta.

Assim, quando um *beacon* é mapeado numa planta, é guardada a sua posição XX e YY, bem como o ID da respetiva planta, na tabela *beacons* apresentada na Figura 5.21.

Todo o processo existente neste módulo é executado com recurso à tecnologia AJAX, não sendo necessário em nenhum momento o recarregamento da página.

De forma a ser possível mover cada *beacon*, usou-se o componente *draggable* da biblioteca jQuery UI. Quando o *beacon* é “largado” é despoletado o evento *stop*. Através do parâmetro *ui* é possível obter a posição XX e YY onde o *beacon* foi “largado”. Após ter o ID do *beacon*, a sua posição e a planta que foi escolhida, são enviados todos esses dados via AJAX (através de POST) guardando-os na DB. O código em *Javascript* disponível na Figura 5.22 representa este processo.

beacons	
id INT(11)	
loja_id INT(11)	
plant_id INT(11)	
designacao VARCHAR(45)	
uuid VARCHAR(255)	
major INT(11)	
minor INT(11)	
pos_x VARCHAR(25)	
pos_y VARCHAR(25)	
imagem VARCHAR(255)	
width VARCHAR(45)	
height VARCHAR(45)	
active TINYINT(4)	
Indexes	

Figura 5.21 – Tabela beacons

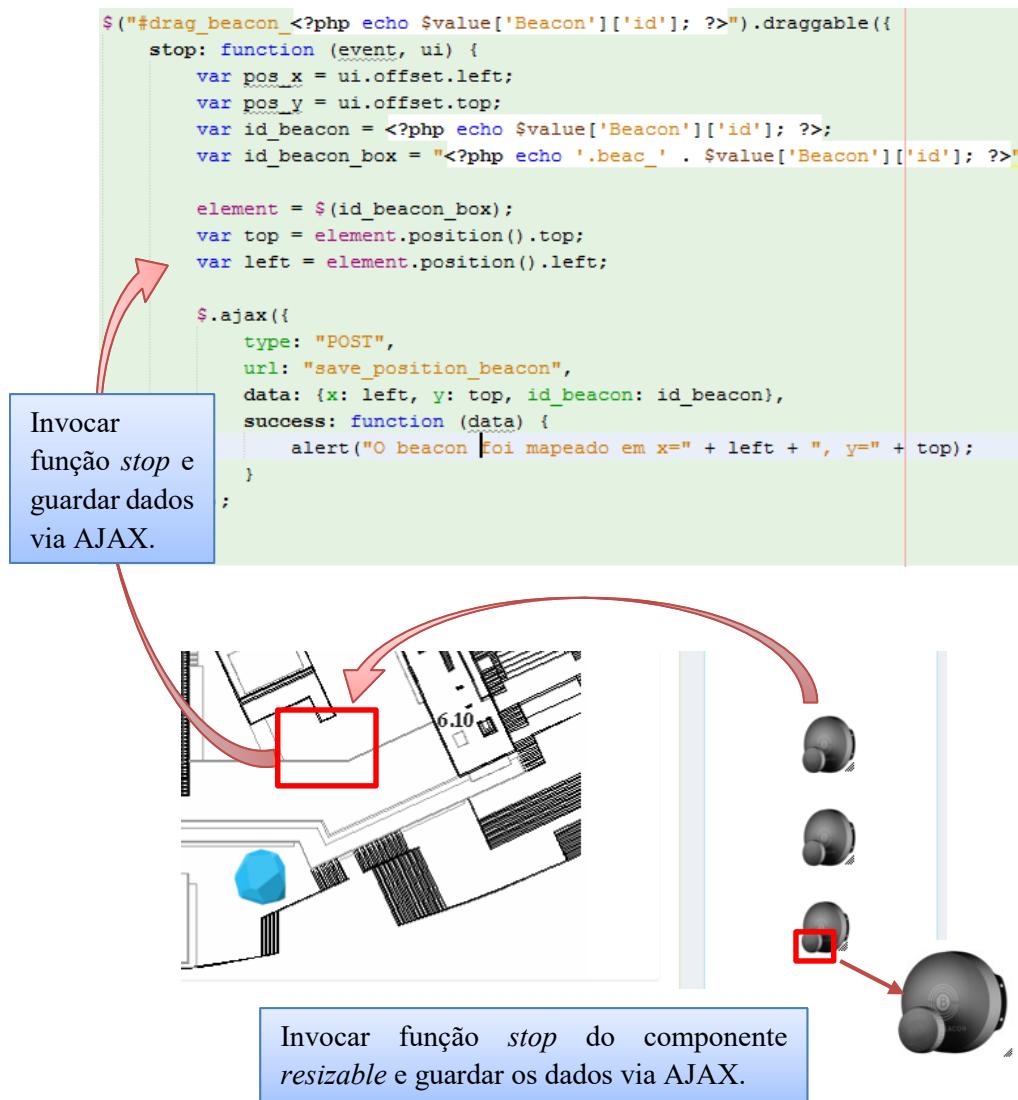


Figura 5.22 – Funcionamento do módulo draggable e resizable

De modo a ajustar o *beacon* de forma precisa a cada local, foi usado o componente *resizable* da biblioteca jQuery UI (destacado na Figura 5.22) possibilitando o seu redimensionamento.

O processo de remoção é igualmente semelhante, verificando apenas se o *beacon* foi “largado” nas coordenadas X e Y da área de remoção. Caso se verifique, então remove a associação do *beacon* com a planta e limpa as coordenadas X e Y do *beacon*.

5.4.3.7. Clientes

O utilizador do tipo *lojista* deve ser utilizado daqui em diante, com exceção do acesso ao módulo de análise de dados.

Quando um utilizador deseja criar um cartão de cliente os seus dados devem ser registados no CMS. Posteriormente, devem ser fornecidas todas as informações necessárias de forma a direcionar o cliente para o *download* da aplicação, por exemplo, a mostragem de um código QR que direcione o utilizador para a página de *download*, tal como exemplo da Figura 5.23.



Figura 5.23 – Anúncio da aplicação

5.4.3.7.1. Registo de um cliente

De forma a otimizar o tempo de inserção dos dados do cliente basta, em primeira instância, inserir o número de telemóvel. Caso o cliente já se encontre registrado noutra loja do CMS o formulário irá aparecer preenchido, bastando o lojista confirmar os dados ou alterá-los caso seja necessário.

Quando um cliente é encontrado noutra loja do CMS, é efetuada uma cópia dos dados do cliente (registando um novo), dado que os seus dados apenas pertencem a uma única loja. Esta função destina-se apenas a otimizar o processo de registo, uma vez que o cliente se encontra a aguardar enquanto este processo decorre.

Pretende-se, na adaptação do CMS a cada loja, a integração com sistemas internos de gestão de clientes de forma a automatizar o processo de registo do cliente (efetuando a importação dos mesmos) garantindo que todos os dados se encontram constantemente atualizados.

Tecnicamente, quando é pesquisado um cliente pelo número de telemóvel, é executado o seguinte código em *Javascript*:

```
$("button").click(function() {
    var telemovel = document.getElementById("telemovel").value;
    $.ajax({url: "/clientes/add_pesquisa_telefone_ajax/" + telemovel, success: function(result) {
        $("#dados_cliente").html(result);
    ...
}}
```

Este irá executar uma função do controlador *ClientesController* onde irá comparar todos os dispositivos de todos os clientes com o número de telemóvel inserido, devolvendo a informação apresentada na Figura 5.24 ou Figura 5.25 caso não encontre ou encontre um cliente respetivamente.

The screenshot shows a search interface with a header 'Pesquisa de cliente'. Below it is a search bar labeled 'Insira o telemóvel do cliente' containing the number '913500039'. A 'Pesquisar' (Search) button is located below the search bar. A green message box at the top states 'Foi encontrado o cliente Nuno Alexandre Paiva. Pode actualizar os seus dados' (Client Nuno Alexandre Paiva was found. You can update your data). The main area is titled 'Dados Gerais' (General Data) and contains fields for 'Nome do cliente' (Name) with 'Nuno Alexandre Paiva' and 'Email' with 'nunopava@streamline.pt'. Below these are fields for 'NIF' (NIF) with '000000' and 'Nif' (NIF) with '000000'.

Figura 5.25 – Encontrado cliente

The screenshot shows a search interface with a header 'Pesquisa de cliente'. Below it is a search bar labeled 'Insira o telemóvel do cliente' containing the number '91000000'. A 'Pesquisar' (Search) button is located below the search bar. A blue message box at the top states 'Não foi encontrado nenhum cliente. Crie um novo registo' (No client was found. Create a new registration). The main area is titled 'Dados Gerais' (General Data) and contains empty fields for 'Nome do cliente' (Name), 'Email', and 'Nif' (NIF).

Figura 5.24 – Não foi encontrado nenhum cliente

Posteriormente é necessário inserir (caso não seja encontrado um cliente) pelo menos uma das três informações pedidas: Número de telemóvel, Email registado no telemóvel, ou Mac Address do telemóvel (Figura 5.26).

The screenshot shows a form titled 'Marca e modelo' (Brand and Model). It has two input fields: 'Marca' (Brand) with 'LG' and 'Modelo' (Model) with 'Nexus 5'. Below this is a section titled 'Dados de contacto do cliente' (Customer contact data) with three input fields: 'Número de telemóvel' (Mobile number) with '913500039', 'Email registado no telemóvel' (Email registered on the mobile) with 'nunopava96@gmail.com', and 'Mac Address do telemóvel' (Mobile Mac Address) with 'bc:f5:ac:ff:c6:97'.

Figura 5.26 – Dados de identificação do dispositivo do cliente

Estas informações destinam-se a identificar o cliente nas comunicações que existirem com o CMS após aproximação de um determinado *beacon*.

Funcionamento técnico

A aplicação móvel irá obter essas três informações do dispositivo do cliente de forma transparente para o utilizador (explicado em detalhe no capítulo 6, secção 6.3.2). Assim, este não necessita de fazer login e preocupar-se com credenciais de acesso.

Quando o dispositivo móvel detetar um *beacon* nas suas proximidades irá enviar todas essas informações para o CMS que irá identificar o cliente com base num (ou mais) desses dados.

Tecnicamente irá efetuar um *OR* entre esses três parâmetros, pesquisando em todos os dispositivos registados no CMS.

```

$this->loadModel('Dispositivo');
$options = array('conditions' => array(
    'OR' => array(
        array('Dispositivo.numero_telemovel' => $this->request->data['tlm']),
        'OR' => array(
            array('Dispositivo.email_telemovel' => $this->request->data['email']),
            'OR' => array(
                array('Dispositivo.mac_address' => $this->request->data['mac_address'])
            )
        )
    )
);
$dispositivo = $this->Dispositivo->find('first', $options);
$cliente_id = $dispositivo['Dispositivo']['cliente_id'];

```

5.4.3.8. Grupos de clientes

Uma das vantagens da solução consiste na criação do perfil de cada cliente, conseguindo, à posteriori, a segmentação dos seus clientes, agrupando-os perante uma área comum.

De modo a que isso seja possível, o CMS contém um módulo de gestão de grupos de clientes, onde o lojista consegue efetuar a criação, listagem, edição e remoção desses grupos.

Por exemplo, uma cadeia de supermercados possui um determinado dia da semana em que a afluência de clientes é bastante menor que nos restantes dias. Como forma de atrair clientes nesse dia decide enviar cupões de desconto. No entanto, de modo a otimizar o seu lucro, tenciona enviar esses cupões apenas a clientes que efetuam compras, por exemplo, ao Domingo. Verificando os dias em que cada cliente frequenta a loja, criando um grupo e associando clientes a esse grupo, é possível o envio de cupões de desconto a um perfil de clientes específico.

5.4.3.8.1. Associação de clientes a grupos

Na fase de levantamento de requisitos definiu-se que um grupo pode conter vários clientes e um cliente pode pertencer a vários grupos, existindo uma relação M:N na DB.

O módulo de associação de clientes possui três áreas, apresentadas na Figura 5.27.

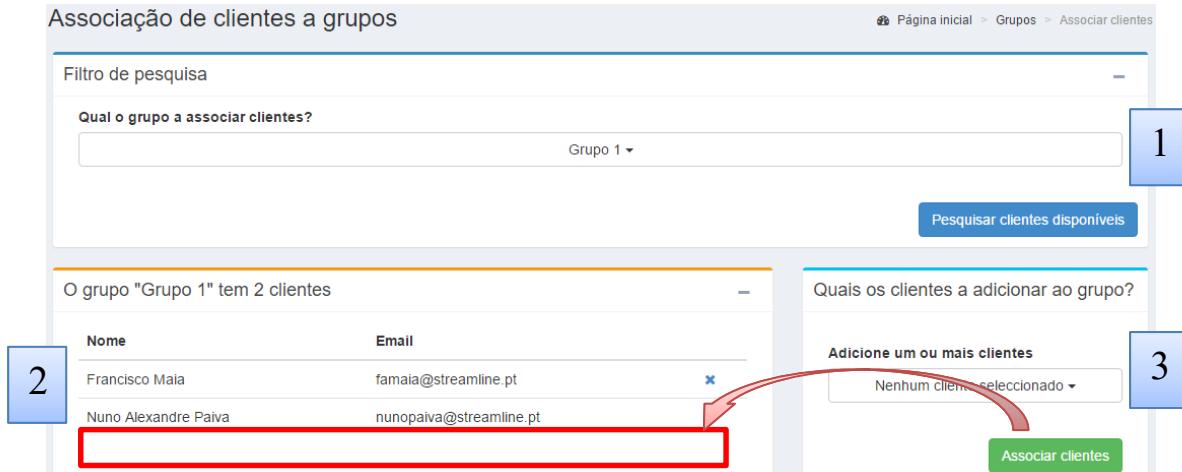


Figura 5.27 – Associação de clientes a grupos

1. **Pesquisa do grupo:** é apresentada uma *dropdown* com todos os grupos criados no CMS;
2. **Clientes existentes nesse grupo:** após clicar no botão “Pesquisar” são apresentados, nesta área e sob a forma de tabela, todos os clientes existentes no grupo selecionado. A remoção de clientes de cada grupo é feita em cada entrada na tabela através do *icon X*;
3. **Adição de clientes ao grupo:** nesta área é apresentada uma *dropdown* onde é possível a escolha de um ou mais clientes a adicionar ao grupo.

Funcionamento técnico

O processo de adição de um cliente a um grupo consiste na inserção de uma entrada na tabela *grupos_has_clientes*, relacionando o ID do cliente com o ID do grupo. Através de uma relação de M:N é possível que um cliente pertença a vários grupos e um grupo possua vários clientes.

Assim, quando o lojista pressiona o botão *Associar clientes* na área 3, são enviados pelo método POST, para a função *associar_clientes* do controlador *GruposController*, os ID's dos clientes selecionados, bem como o ID do grupo previamente escolhido.

Para cada cliente selecionado é inserido um registo na tabela *grupos_has_clientes* através da classe *GruposHasCliente*, tal como mostra o código abaixo.

```

for ($i = 0; $i < count($this->request->data['Cliente']['clientes']); $i++) {
    $array = [
        'grupo_id' => $grupo_id,
        'cliente_id' => $this->request->data['Cliente']['clientes'][$i],
    ];

    $this->loadModel('GruposHasCliente');
    $this->GruposHasCliente->create();
    if (!$this->GruposHasCliente->saveAll($array)) {
        $this->Session->setFlash(__('Erro na adição'), 'erro_back');
    }
}

```

5.4.3.9. Campanhas

Todas as comunicações existentes entre o CMS e o cliente denominam-se *campanhas*. Estas podem ser usadas para o envio de descontos (por exemplo: cupões, vouchers, bilhetes de viagem) ou para transmitir qualquer outro tipo de informações.

Uma campanha consiste num bloco de informação devolvido ao cliente quando este entra na proximidade de um *beacon*.

O CMS permite o envio de três tipos de campanhas: *Standard*, *Passbook* e *Web*.

5.4.3.9.1. Tipos de campanhas suportadas

Esta seção descreve os três tipos de campanhas suportadas.

5.4.3.9.1.1. Standard

Campanhas do tipo *Standard* destinam-se ao envio de informação ao cliente, podendo assumir os três *layouts* distintos apresentados na Figura 5.28.

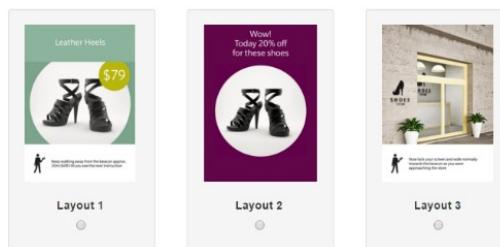


Figura 5.28 – Layouts para campanhas do tipo Standard

Por exemplo, o *layout 2* apenas possui um título, ao contrário do que acontece no *layout 1*. Consoante o *layout* escolhido na criação da campanha, a informação pedida ao lojista irá variar.

De forma a manter o rigor nas campanhas enviadas optou-se por definir os *layouts* internamente na **Streamline**, não tendo as restantes contas de utilizador permissões para os alterar. Essa possibilidade poderia levar a incoerências de cores ou formatos, podendo não ir ao encontro da imagem da loja. Deste modo, a criação de novos *layouts* para este tipo de campanhas implica a comunicação com a **Streamline**, de forma a que seja desenvolvido o *layout* não só no CMS como na aplicação móvel.

5.4.3.9.1.2. Passbook

Desenvolvido pela Apple e apresentado na *Apple Worldwide Developers Conference* (WWDC) a 11 de junho 2012, consiste num modo dos utilizadores conseguirem armazenar todos os seus passes de forma centralizada. Assim, foi desenvolvida uma aplicação, inicialmente para iOS de nome *Passbook*, e posteriormente para Android de nome *PassWallet*, onde é possível armazenar diversos tipos de passes com determinados formatos específicos [57].

Apresenta-se na Tabela 5.1 os cinco tipos de passes definidos pela Apple.

Tabela 5.1 – Tipos de Passbook

Tipo de Passe	Descrição	Estilo
Cartão de loja	Pode ser usado para armazenar o cartão de cliente de uma loja. Na solução pode ser enviado o valor acumulado que o cliente possui em cartão quando este entra na loja.	
Bilhete de evento	Bilhete de acesso a um evento. Na solução podem ser enviados bilhetes de um concerto, a clientes que se dirigem à loja num determinado dia da semana.	
Cupão de desconto	Tipo de passe que pode ser usado para oferecer desconto num determinado produto. Por exemplo, na solução pode ser enviado um desconto de 10% numa TV ao primeiro cliente que estiver a menos de 1 metro da mesma.	
Bilhete de viagem	Tipo de passe que representa um bilhete de avião, barco, autocarro ou comboio. Na solução pode ser disponibilizado um bilhete, por exemplo, ao primeiro cliente a entrar na loja num determinado dia.	
Genérico	Tipo de passe que pode ser usado para representar qualquer outra situação não abrangida nas categorias anteriores.	

Todos os tipos de passes apresentados são suportados pelo CMS, sendo pedidos os dados ao lojista de acordo com o tipo escolhido.

A Apple estabeleceu ainda três tipos de código usados na validação de cada passe:

- Código QR;
- Código PDF 417;
- Código AZTEC.

Todos pertencem ao tipo de código de barras 2D, sendo a sua escolha suportada também pelo CMS.

5.4.3.9.1.3. Web

Este tipo de campanhas surge no contexto da tecnologia *Eddystone* apresentada no capítulo 3 (secção 3.4.4), a qual faz uso do tipo de pacote *Eddystone URL* para a sua transmissão.

Assim, recorrendo a estas campanhas existe a possibilidade de enviar informações personalizadas aos clientes sem necessidade de instalarem e usarem uma aplicação.

De uma forma geral, este tipo de campanhas consiste numa página Web, com conteúdo e *layout* dinâmico, cujo URL ficará associado e será transmitido por cada *beacon* escolhido para a sua difusão.

A Figura 5.29 mostra os dois tipos de *layouts* existentes, nomeadamente: lista de conteúdos e *slider* de imagens.



Figura 5.29 - Layouts para campanhas do tipo Web

Cada página Web foi otimizada para uso mobile, ou seja, de forma a comportar-se o máximo possível como uma aplicação foram implementados eventos *touch* recorrendo à biblioteca *jquery mobile*. Por exemplo, para o *layout: slider* de imagens, foi implementado o evento *swipe* permitindo alternar entre imagens deslizando o dedo da direita para a esquerda ou vice-versa.

O conteúdo associado a este tipo de campanhas pode ser inserido em dois momentos: antes de criar uma campanha ou após a sua criação. Cada conteúdo funciona de forma independente a cada campanha, podendo ser utilizado em uma ou mais campanhas em simultâneo.

O menu principal “Conteúdos Web” disponibiliza uma área onde é possível listar, editar, adicionar ou remover conteúdos em qualquer instante.

A criação de um novo conteúdo Web é efetuada através do menu “Conteúdos Web” → “Novo Conteúdo” e recorrendo ao formulário apresentado na Figura 5.30.

The screenshot shows a multi-step creation form for web content. Step 1: 'Associar a uma campanha' (Associate with a campaign) has a checkbox 'Associar?' and a dropdown 'Restaurantes 1 -'. Step 2: 'Adicionar conteúdo web' (Add web content) includes a 'Título' field with 'Vinho Verde', a 'Resumo' field with 'Desconto em todas as garrafas de vinho verde.', and a 'Descrição' field with the same text. Step 3: 'Validade' (Validity) shows date inputs for 'Data de inicio' (2016-06-29 11:50:57) and 'Data de fim' (2016-06-30 11:55:57). Step 4: 'Detalhes e Imagem' (Details and Image) includes a 'URL para mais detalhes' field with 'www.exemplo.com' and an 'Imagen' section with a 'Escolher ficheiro' button and a message 'Nenhum ficheiro selecionado'.

Figura 5.30 – Formulário de criação de Conteúdos Web

Cada conteúdo Web é identificado por: Título, Resumo (opcional), Descrição (opcional), URL para site externo (opcional) e Imagem. Possui ainda uma determinada data e hora de validade, que servirá para definir o intervalo de tempo que o conteúdo será enviado (uma campanha pode apresentar conteúdos distintos de acordo com a hora que é recebida). Consoante o *layout* de cada campanha, onde o conteúdo será inserido, as diversas informações do conteúdo Web serão apresentadas de forma distinta.

Caso a campanha já tenha sido criada e se deseje associar o conteúdo Web de forma automática, aquando da sua criação, basta, no primeiro quadro “Associar a uma campanha”, escolher a campanha que se deseja associar.

É possível ainda distinguir conteúdos publicitários de conteúdos informativos através da caixa de seleção “Publicidade”. Assim, pode ser inserida publicidade numa determinada campanha potenciando o negócio do gestor.

Suporte Multi-idioma

Este tipo de campanhas permite o envio de conteúdos em vários idiomas. Assim, cada conteúdo Web pode possuir uma ou mais traduções associadas.

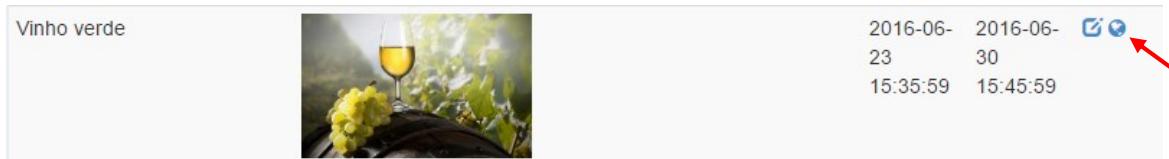


Figura 5.31 – Listagem de conteúdos Web

Através do ícone assinalado na Figura 5.31 e associado a cada conteúdo Web, é possível gerir as suas traduções. A Figura 5.32 mostra à esquerda uma listagem de todas as traduções associadas ao conteúdo Web, e à direita o respetivo conteúdo Web original.

Listagem				
Id	Idioma	Título	Imagem	Acções
11		Green Wine		Ver Editar Eliminar
12		Vin vert		Ver Editar Eliminar
13		Grüne Wein		Ver Editar Eliminar

Página 1 de 1. Apresentando 3 registo(s), de um total de 3.

Acções

+ Nova tradução

Conteúdo Original

Titulo
Vinho verde

Resumo
Vinho verde

Descrição
Vinho verde

Imagen

Figura 5.32 – Listagem de traduções associadas a um conteúdo Web

Para adicionar uma nova tradução basta clicar no botão “Nova tradução” (visível na Figura 5.32). Cada tradução é inserida com recurso a um formulário semelhante ao apresentado na Figura 5.30, sendo pedidos todos os dados registados no conteúdo Web original (inclusive imagem) e respetivo idioma associado.

Ao cliente, consoante as traduções inseridas para cada conteúdo Web, serão apresentadas as bandeiras de cada país associado, sendo obtida a tradução de cada conteúdo através do clique na respetiva bandeira.

5.4.3.9.2. Criação de uma campanha no CMS

O processo de criação de uma campanha é composto por seis passos. Independentemente do tipo de campanha escolhida (*Standard, Passbook ou Web*) os passos de criação são semelhantes, diferindo apenas na escolha do *layout* e nas informações pedidas.

Passo 1: Dados de identificação e validade

O primeiro passo consiste em inserir os dados que irão permitir identificar a campanha no CMS (Figura 5.33), sendo composto por: Título e Descrição.

Regra geral uma campanha destina-se a ser enviada durante um determinado período de tempo. Desta forma é solicitada também a data em que a campanha começará a ser enviada e a data em que a campanha ficará inativa.

Figura 5.33 – Criação de uma campanha: Passo 1

Passo 2: Localização da campanha e zona de proximidade

O CMS permite escolher qual(ais) o(s) *beacon*(s) onde a campanha será difundida. Pode-se pretender que uma campanha seja difundida em toda a loja, ou, no caso de uma cadeia de lojas, difundir em diversos edifícios. Por outro lado, pode-se desejar, por exemplo, enviar um cupão de desconto de um produto, sendo difundido apenas pelo *beacon* situado junto ao mesmo.

Assim, é possível indicar no CMS se a campanha irá ser difundida num (ou mais) *beacons*, ou numa (ou mais) região de *beacons*. Essa possibilidade é apresentada através de duas *tabs*, nomeadamente: *Beacons individuais* (Figura 5.34) e *Região de beacons*.

Para ambos os casos, através do uso da biblioteca *multiselect* da *framework bootstrap* é simples a seleção de vários *beacons* (ou regiões de *beacons*) sendo possível o filtro através da pesquisa direta na *dropdown* (útil no caso da existência de bastantes opções).

Figura 5.34 – Criação de uma campanha: Passo 2

Tecnicamente um *beacon* pode difundir várias campanhas e uma campanha pode ser difundida por vários *beacons*. Esta funcionalidade é conseguida através de uma relação M:N semelhante ao módulo de associação de clientes a grupos (secção 5.4.3.8). Assim, quando é clicado no botão *próximo passo* são enviados para a função *add_beacon* no controlador *CampanhasController* os ID's dos *beacons* (ou regiões) selecionados(as) bem como o ID da campanha, inserindo na tabela *campanhas_has_beacons* (Figura 5.35) um “*campanhas has beacons*” registo por cada *beacon* (região) escolhido(a).

Figura 5.35 – Tabela “*campanhas has beacons*”

Ainda neste passo é possível selecionar qual a zona de proximidade a que o cliente se deve encontrar do *beacon* de forma a receber a campanha.

Tendo em conta a norma *iBeacon*, foram definidas três zonas de proximidade (Figura 5.36), nomeadamente: Imediato (até 1 metro); Perto (até 3 metros); Longe (até 70 metros). Todos os valores são tratados de forma dinâmica sendo possível a adição ou alteração de cada distância, existindo uma área no CMS para esse efeito. Essa possibilidade deve-se à existência de uma tabela na DB de nome *zonas_proximidades*, responsável por armazenar os respetivos valores.

A possibilidade de definir a distância máxima a que o cliente se deve encontrar de forma a receber a campanha permite aumentar a dinâmica da interação dos clientes, bem como a análise pormenorizada da localização destes.

Por exemplo, atendendo às zonas de proximidade definidas, é possível criar campanhas do tipo: “Aproxime-se do produto e receba 10% de desconto na sua compra”.

Recorde-se a importância do valor de precisão enunciado no capítulo 3 (secção 3.3.5). Tendo em conta fatores de ruído e atenuação de sinal, a zona de proximidade escolhida deve ter em conta o meio físico onde o sinal se irá propagar.

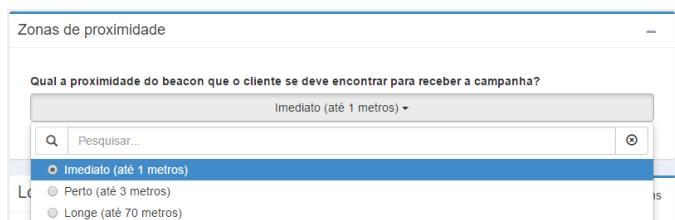


Figura 5.36 – Zonas de proximidade

Passo 3: Alcance da campanha

Este passo destina-se a indicar qual o alcance da campanha, ou seja, quais os clientes que poderão vir a recebê-la, através da seleção de um ou mais grupos (Figura 5.37). Consegue-se desta forma enviar campanhas e informação direcionada, tratando cada cliente de forma personalizada, e ir ao encontro da ideia de mercado de Chuck Martin, mencionada no capítulo 3 (secção 3.4.1).

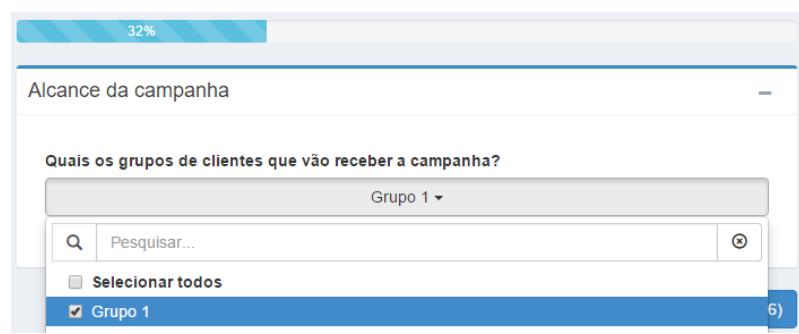


Figura 5.37 – Criação de uma campanha: Passo 3

Uma vez que a identificação do cliente é efetuada através da aplicação instalada no seu dispositivo, no caso de ser criada uma campanha Web este passo será omitido pois será enviada para todos os clientes (mesmo que não estejam registados no CMS), recebendo-a com recurso ao *browser* instalado no *smartphone*.

Passo 4: Layout da campanha

Consoante o tipo de campanha escolhida previamente (*Standard, Passbook ou Web*), serão apresentados, neste passo, *layouts* diferentes.

A Figura 5.38 mostra o conteúdo apresentado ao utilizador para o tipo de campanha *Standard*.

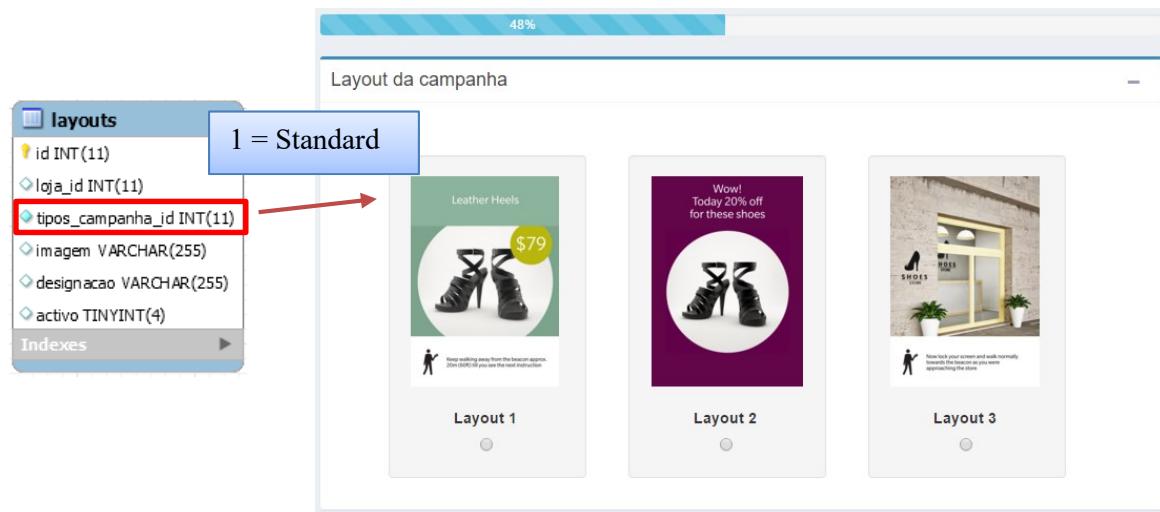


Figura 5.38 – Criação de uma campanha: Passo 4 - Standard

A Figura 5.39 mostra o conteúdo apresentado ao utilizador para o tipo de campanha *Passbook*.

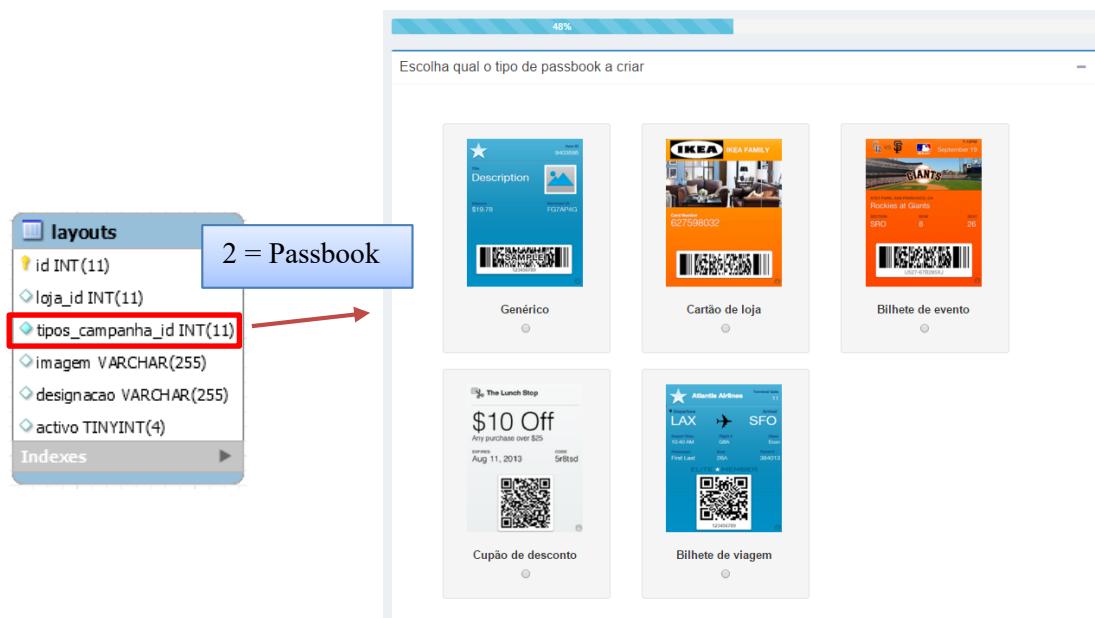


Figura 5.39 – Criação de uma campanha: Passo 4 - Passbook

A Figura 5.40 mostra o conteúdo apresentado ao utilizador para o tipo de campanha *Web*.

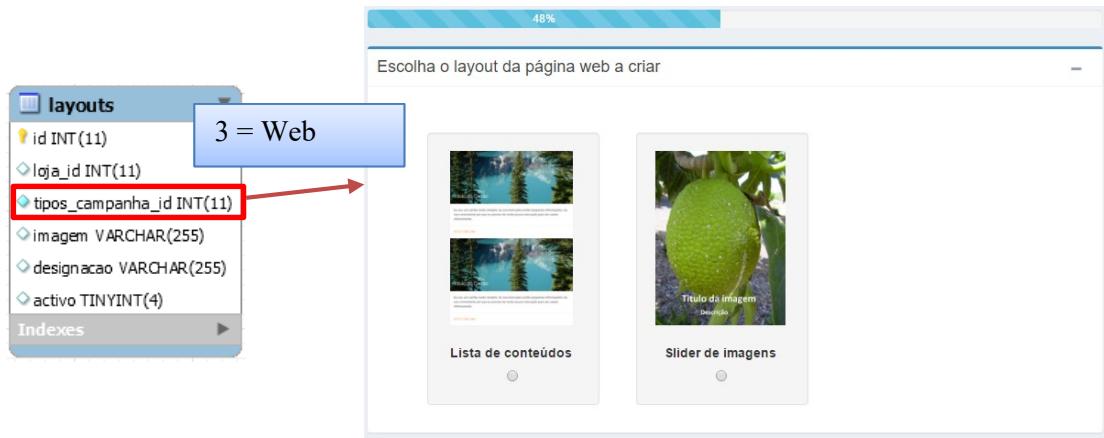


Figura 5.40 - Criação de uma campanha: Passo 4 - Web

Todos os *layouts* apresentados encontram-se criados dinamicamente sendo armazenados em DB (através da tabela *layouts*) e relacionados com o tipo de campanha.

O processo de associação do *layout* à campanha consiste em obter o ID do *layout* escolhido e inseri-lo no campo *layout_id* existente na tabela *campanhas* (Figura 5.41).

Através da existência das tabelas *layouts* e *tipos_campañas* o CMS encontra-se preparado para a implementação de um novo tipo de campanha de forma fácil e dinâmica.

campanhas	
id INT(11)	
tipos_campaña_id INT(11)	
loja_id INT(11)	
layout_id INT(11)	
42 more...	
Indexes	

Figura 5.41 – Criação de uma campanha: Passo 4 – tabela *campanhas*

Passo 5: Conteúdo a ser mostrado e número de envios

Neste passo é pedida a informação a conter na campanha de acordo com o *layout* escolhido no passo anterior. Para efeitos de explicação serão abordados o tipo de campanha *Passbook* e *Web*.

Tipo de campanha: Passbook

Para o tipo de campanha *Passbook* e o tipo de passe *Bilhete de evento*, será pedido ao utilizador para preencher os dados de acordo com a Figura 5.42.

Conteúdo da campanha		
Título	Localização	
Título da campanha. Máximo de 50 caracteres.		Coimbra. Ex: Estádio Municipal de Coimbra
Secção	Fila	Lugar
Secção do evento. Ex: A.	Fila onde irá ficar situado. Ex: B	Lugar. Ex 35
Data e hora	Escolha Código	Introduza o código do bilhete
Data e hora do evento	Código QR	Código que irá validar o bilhete.

Figura 5.42 – Criação de uma campanha: Passo 5 – conteúdo

É possível ainda a escolha do tipo de código a validar o bilhete, de acordo com os estabelecidos pela Apple, e apresentados na Figura 5.43.

Código QR



Código PDF 417



Código AZTEC



Figura 5.43 - Criação de uma campanha: Passo 5 – códigos suportados

O campo *introduza o código do bilhete* consiste naquele que irá permitir validar o bilhete, ficando, o valor inserido, encapsulado no tipo de código escolhido.

É ainda pedido o preenchimento do campo *detalhes* o qual será gravado na retaguarda do passe. Por norma, na retaguarda encontram-se informações úteis de uso, por exemplo, termos de uso.

A customização do passe é feita através da escolha da cor de fundo. De modo a facilitar a escolha da cor, recorreu-se à biblioteca *jscolor*, tal como visível na Figura 5.44.



Figura 5.44 - Criação de uma campanha: Passo 5 – Escolha de cor

Ainda neste passo é possível inserir o conteúdo (título e detalhes) que irá aparecer na notificação *push* do sistema Android, quando o cliente receber a campanha (Figura 5.45).



Figura 5.45 - Criação de uma campanha: Passo 5 – Notificação push

Por fim, é possível definir o número de vezes que a campanha será enviada. Esta consiste numa funcionalidade importante, no envio, por exemplo, de bilhetes de viagem ou eventos onde existe um limite de ocupações. Assim, após ter sido enviado o número de vezes estabelecido, a campanha passará automaticamente para o estado inativo.

Por outro lado, no caso de se desejar que uma campanha seja enviada de forma ilimitada, durante o seu período de validade, basta não preencher o campo apresentado na Figura 5.46.

Figura 5.46 - Criação de uma campanha: Passo 5 – Número de envios

Tipo de campanha: Web

No caso de ser criada uma campanha do tipo Web, neste passo deverão ser selecionados os conteúdos previamente criados no CMS (de acordo com o indicado na secção 5.4.3.9.1.3).

Recorrendo a uma *dropdown* de múltipla escolha, com filtragem por nome do conteúdo, é possível escolher um ou mais conteúdos Web que farão parte da campanha (Figura 5.47). É possível ainda selecionar conteúdos publicitários que serão automaticamente intercalados com os restantes conteúdos.

Figura 5.47 - Criação de uma campanha: Passo 5 – Escolha de conteúdo

Passo 6: Imagen de capa

Neste passo é possível efetuar o carregamento de uma imagem, de acordo com o *layout* e tipo de campanha previamente escolhida (passo omitido para campanhas do tipo Web dado o carregamento da imagem, associada a cada conteúdo, possuir módulo exclusivo para o efeito).

De modo a que a imagem fique ajustada à campanha, foi implementada a biblioteca *cropit*. Esta permite, através da indicação de um tamanho específico, ajustar a imagem (por exemplo, fazendo *zoom* – área destacada na Figura 5.48) a uma determinada área. Após o carregamento, a imagem é cortada exatamente com as medidas desejadas, adequando-se ao *layout* da campanha a que se destina.



Figura 5.48 - Criação de uma campanha: Passo 6

Ao clicar no botão *Guardar imagem de capa* será invocada a função em *Javascript* *onSubmitImg* responsável por obter a imagem da biblioteca *cropit*, o ID da campanha e, via AJAX, enviar os dados através do método POST para a função *upload_imagem* existente no controlador *CampanhasController* (representado no código abaixo).

```
function onSubmitImg() {
    var imageData = $('.image-editor').cropit("export");
    $('#loading').show();
    var URL = '/campanhas/upload_imagem';
    $.ajax({
        url: URL, // Url to which the request is send
        type: "POST", // Type of request to be send
        data: {
            id: <?php echo $campanha['Campanha']['id']; ?>,
            img: imageData
        }
    });
}
```

A função `upload_imagem` é responsável por efetuar o carregamento da imagem para uma pasta no CMS. Insere ainda o nome da imagem na tabela `campanhas` de forma a relacionar a imagem guardada no sistema de ficheiros com a campanha existente na DB.

De forma a evitar nomes de imagens repetidos, antes do carregamento da imagem é alterado o nome, usando a função `Standard uniqid()` do PHP (representado no código ao lado).

```
public function upload_imagem() {
    $this->autoRender = false;
    if ($this->request->is(['post', 'ajax'])) {

        $data = $_POST['img'];
        list($type, $data) = explode(';', $data);
        list($type, $data) = explode(',', $data);
        $data = base64_decode($data);
        $filename = uniqid() . '.png';
        $aux = array();
        $aux['Campanha']['id'] = $_POST['id'];
        $aux['Campanha']['imagem'] = $filename;
        $filename = 'img/img_campañas/' . $filename;
        file_put_contents($filename, $data);
        $this->Campanha->save($aux);
    }
}
```

Passo 7: Pré-visualização da campanha

No último passo, do processo de criação de campanhas, o lojista consegue obter uma pré-visualização da campanha antes de a publicar.

A página de pré-visualização (Figura 5.49) é dividida em duas partes: o que será mostrado ao cliente e os dados internos que permitirão identificar a campanha no CMS.

Neste passo é possível visualizar todos os dados inseridos, podendo voltar aos passos anteriores e efetuar alterações caso se justifique.

Todo o *layout* criado para pré-visualizar a campanha foi desenvolvido em HTML e CSS, adaptando-se de forma dinâmica a cada dispositivo.

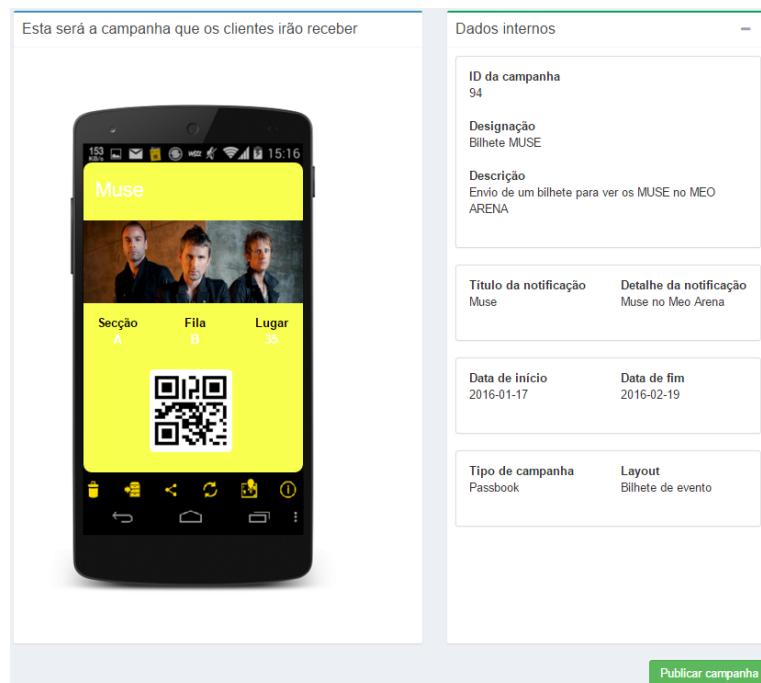


Figura 5.49 - Criação de uma campanha: Passo 7

O processo de publicação de uma campanha possui etapas distintas de acordo com o tipo de campanha criada.

Publicação de campanhas: Standard

No caso de ser uma campanha *Standard*, consiste apenas em atribuir o valor 1 à *flag activo* existente na tabela *campanhas* (correspondente à campanha em causa). Esta *flag* define uma campanha como estando ativa (*activo=1*) ou inativa (*activo=0*) sendo enviadas unicamente campanhas ativas.

O botão *publicar campanha* despoleta a execução da função abaixo, responsável por efetuar o enunciado.

```
public function add_preview_campagna($campagna_id) {
    $aux = array();
    $aux['Campanha']['id'] = $campagna_id;
    $aux['Campanha']['activo'] = 1;
    $this->Campanha->create();
    if ($this->Campanha->save($aux)) {
        $this->Session->setFlash(__('Campanha publicada com sucesso'))
    }
}
```

...

Publicação de campanhas: Passbook

No caso de ter sido criada uma campanha do tipo *Passbook* para além de efetuar o processo anterior é ainda despoletada uma função que permite a criação do ficheiro de extensão *pkpass*.

Para criar o ficheiro *pkpass* (*Passbook*), o CMS faz uso de uma biblioteca externa de nome PHP-PKPASS, tendo sido incluída através da seguinte instrução.

```
require_once APP . 'Vendor' . DS . 'PKPass/src/PKPass/PKPass.php';
```

De forma a criar um passe válido é necessário possuir os seguintes ficheiros:

- Certificado para criação de passes;
- *Apple Worldwide Developer Relations Certification Authority*.

```
$pass = new PKPass\PKPass();
$pass->setCertificate(WWW_ROOT . "files/passbook/" . 'Certificates_passtype_compwd.p12');
$pass->setCertificatePassword('xxxxxxxx');
$pass->setWWDRcertPath(WWW_ROOT . "files/passbook/" . 'Apple_Worldwide_Developer_Relations_Certification_Authority.pem');
```

Consoante o tipo de passe a gerar é necessário a criação de um conjunto de *arrays*, respeitando a estrutura imposta pela biblioteca, de forma a inserir o conteúdo conforme indicado pela norma. O código seguinte mostra, nas áreas destacadas, o *array secondaryFields* (responsável por inserir os dados na parte frontal do passe) e o *array backFields* (responsável por inserir os detalhes na retaguarda do passe).

```

/*
 * generic
 * boardingPass
 * storeCard
 * coupon
 * Event
 */

$styleKeys = array(
    'Event' => array(
        'secondaryFields' => array(
            array(
                'key' => 'seccao',
                'label' => "Secção",
                'value' => "{$campanha['Campanha']['evento_seccao']}"
            ),
            array(
                'key' => 'fila',
                'label' => "Fila",
                'value' => "{$campanha['Campanha']['evento_fila']}"
            )
        ),
        'backFields' => array(
            array(
                'key' => 'data',
                'label' => "Data do evento",
                'value' => "{$campanha['Campanha']['evento_data_hora']}"
            )
        )
    )
);
...

```

Valores a serem mostrados na frente do passe.

Valores a serem mostrados na retaguarda do passe.

O código seguinte permite implementar no passe, o tipo de código e conteúdo selecionado na criação da campanha, bem como a cor de fundo escolhida.

```

$visualAppearanceKeys = array(
    'barcode' => array(
        'format' => $format,
        'message' => $codigo,
        'messageEncoding' => 'iso-8859-1'
    ),
    'backgroundColor' => "{$campanha['Campanha']['cor_fundo']}",
    'logoText' => "{$campanha['Campanha']['titulo']}"
);

```

- PKBarcodeFormatPDF417
- PKBarcodeFormatQR
- PKBarcodeFormatAztec

No final do processo é criado uma pasta, tendo como nome o ID da campanha, na localização “/files/passes/[ID_CAMPANHA]” sendo, posteriormente, carregado o ficheiro de extensão *pk.pass* para essa localização.

Quando um cliente se aproximar de um *beacon* cuja campanha do tipo *Passbook* lhe esteja atribuída, irá receber no seu dispositivo este ficheiro.

Para que o cliente consiga abrir estes passes é necessário possuir a aplicação *PassWallet*, no caso do sistema *Android*. A forma como se comporta a aplicação móvel da solução face a existência ou não dessa aplicação, no dispositivo do cliente, é descrito no capítulo 6 (secção 6.4.2.2).

Publicação de campanhas: Web

No caso de ter sido criada uma campanha do tipo *Web*, para além de ser efetuado o descrito para campanhas do tipo *Standard*, é ainda gerado um *short URL*³.

O *short URL* será o URL que cada um dos *beacons* selecionados durante a criação da campanha irá transmitir para o cliente. Para efetuar a criação de um *short URL* recorreu-se a uma biblioteca externa de nome *GoogleUrlApi* que, fazendo uso da API do Google *urlshortener*, permite a devolução de um *short URL* com 21 caracteres. O uso desta API requer ainda o envio de uma chave, previamente criada, como forma de autenticação.

Antes de ser criado o *short URL* é verificado qual o *layout* que a campanha possui (lista ou *slider*). Cada *layout* da campanha corresponde a uma página Web com diferentes formatações. Assim, consoante o *layout* escolhido irá ser utilizada a respetiva página para a apresentação dos seus conteúdos e respetiva criação do *short URL* (apresentado no código abaixo).

```
$url_campanha_web = $_SERVER['SERVER_NAME'];
if ($campanha['Campanha']['layout_id'] == $lista_conteudos) {
    $url_campanha_web .= "/campanhas/campanha_web?c=";
} else if ($campanha['Campanha']['layout_id'] == $slider_imagens) {
    $url_campanha_web .= "/campanhas/campanha_web_slider?c=";
}
$key = 'XXXXXXXXXXXXXXXXXXXXXX';
$googer = new GoogleURLAPI($key);
$short_url = $googer->shorten($url_campanha_web . $campanha_id);
```

Após criação do *short URL* o mesmo é guardado na tabela *beacons*, existentes na DB, nomeadamente no campo *eddy_url* para todos os *beacons* escolhidos para a difusão da respetiva campanha. Note-se que, como o *short URL* é gerado com base no ID da campanha, no caso de edição, adição ou remoção de conteúdos posteriormente à sua publicação, o mesmo não necessita de ser regenerado.

Até este ponto o URL responsável por apresentar a campanha e respetivos conteúdos apenas se encontra associado ao *beacon* de forma “virtual”, ou seja, na DB do CMS. Surge neste momento a necessidade de “introduzir o URL no beacon”. A forma como é efetuado o sincronismo do URL em cada *beacon* é apresentado no capítulo 6 (secção 6.4.3.2).

A Figura 5.51 mostra a relação entre os dois *layouts* disponíveis, para o tipo de campanha *Web*, e respetivo *short URL*.

³ URL *shortening* é uma técnica na World Wide Web (WWW) que permite criar um URL substancialmente mais pequeno direcionando o utilizador para a mesma página. Isso é conseguido através de um redireccionalmento num domínio auxiliar (com nome mais pequeno) para o URL maior. Por exemplo, o URL “<http://example.com/about/index.html>” pode ser encurtado para “<https://goo.gl/aO3Ssc>”. Esta técnica é útil em casos onde existe limitação de carateres do URL, como por exemplo no pacote Eddystone URL.



Figura 5.51 – Layouts disponíveis em campanhas Web

5.4.3.10. Análise de dados

Para além de gerir todas as comunicações com a aplicação do cliente, o CMS permite a extração de dados através da interação entre os clientes e as respetivas campanhas.

Acessível apenas a utilizadores do tipo *Gestor*, estes, recorrendo a este módulo, poderão tomar decisões que possibilitarão otimizar o seu modelo de negócio bem como melhorar a satisfação de compra dos seus clientes.

Ao longo desta secção serão apresentadas as componentes constituintes do módulo de análise de dados existente no CMS.

5.4.3.10.1. Dashboard Geral

O CMS disponibiliza um *dashboard* geral onde é possível visualizar as seguintes informações:

- Afluência de clientes;
- Taxa de abertura de campanhas;
- TOP 10 de campanhas enviadas / abertas;
- Tipos de campanhas mais clicadas.

A extração destes dados apenas é possível se todas as comunicações existentes entre cada cliente e a solução forem registadas.

Registo de comunicações

Sempre que um cliente se aproxima de um *beacon* todos os dados dessa interação são registados em DB, nomeadamente:

- Data e hora de comunicação entre o cliente e o CMS;
- Identificação do cliente que comunicou com o CMS;
- Identificação do *beacon* com o qual o dispositivo do cliente comunicou;
- Identificação da campanha que o cliente recebeu.

Essa informação é armazenada nas tabelas das Figuras 5.52 e 5.53:

percursos_clientes	
!	<code>id INT(11)</code>
◆	<code>cliente_id INT(11)</code>
◆	<code>beacon_id INT(11)</code>
◆	<code>data_hora_contacto DATETIME</code>
Indexes	

Figura 5.52 – Registo de contato do cliente com um beacons

clientes_campañas	
!	<code>id INT(11)</code>
◆	<code>cliente_id INT(11)</code>
◆	<code>campanha_id INT(11)</code>
◆	<code>data_hora DATETIME</code>
◆	<code>clicked TINYINT(4)</code>
◆	<code>ignored TINYINT(4)</code>
Indexes	

Figura 5.53 – Registo da receção de uma campanha por um cliente

Afluência de clientes

Neste *dashboard* é possível obter uma análise de afluência de clientes, possibilitando visualizar os seguintes dados:

1. Número de clientes que visitaram a loja no dia atual;
2. Número de clientes que visitaram a loja no dia anterior face ao dia homólogo, com respetivo cálculo de taxa;
3. Número de clientes que visitaram a loja no mês atual face ao mês anterior, com respetivo cálculo de taxa;
4. Número de clientes que visitaram a loja no ano atual face ao ano anterior, com respetivo cálculo de taxa;

A apresentação destes dados é efetuada com base nos quadros da Figura 5.54:



Figura 5.54 – Número de clientes que frequentaram a loja

A cor dos quadros **2, 3 e 4** pode variar entre verde, laranja ou vermelho consoante o número de clientes seja superior, igual ou inferior face ao período homólogo comparado. Assim, pelo quadro **2** verifica-se que o número de clientes que frequentaram a loja na Quinta-Feira é menor quando comparado com o dia homólogo.

É ainda construído um gráfico do tipo *área* onde é possível visualizar:

1. Número de clientes por dia durante os últimos sete dias em comparação com a semana anterior (Figura 5.56);
2. Número de clientes por mês do último semestre em comparação com o penúltimo semestre (Figura 5.55).

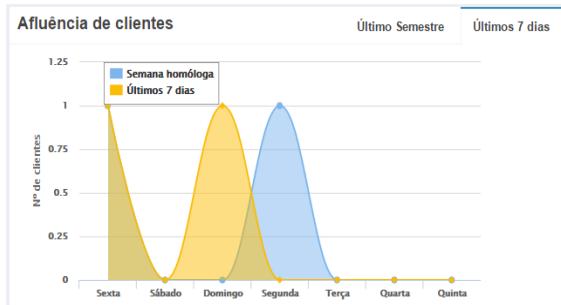


Figura 5.56 – Afluência de clientes nos últimos sete dias

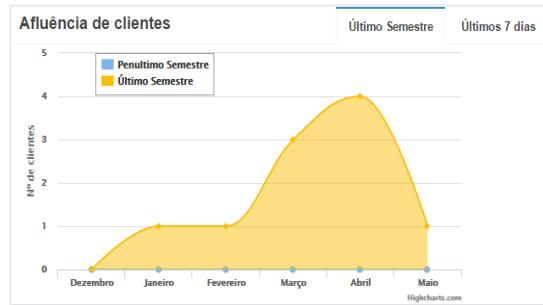


Figura 5.55 – Afluência de clientes no último semestre

Esta é uma análise de elevada importância pois possibilita ao *Gestor* perceber a tendência de afluência dos clientes na loja, identificando, por exemplo, momentos onde existe um decréscimo e tomar medidas de forma a contrariar essa tendência.

Taxa de abertura de campanhas

A taxa de abertura de campanhas, é representado por um gráfico *pie*, tal como apresentado na Figura 5.57.

Explicação técnica

Foi definido na fase inicial que uma campanha era considerada aberta após o cliente clicar na imagem da campanha. Assim, associado à imagem, encontra-se um URL (inserido pelo lojista durante a criação da campanha) cujo objetivo se destina a encaminhar o cliente para uma página *Web* com mais detalhes da campanha.

O URL, para o qual é direcionado o cliente, não consiste no URL de destino, mas num URL intermediário com informação que permite identificar o cliente e a campanha, sendo posteriormente redirecionado para o URL final, após o registo da comunicação na DB (ilustrado na Figura 5.58).

Tecnicamente, quando um cliente clica na campanha é enviado, para o CMS, o ID do cliente e da campanha via o método GET para a função *click_campagna* existente no controlador *ClientesCampanhasController*. Nessa função, é guardado o registo de que o cliente X abriu a campanha Y através da alteração da flag *clicked* existente na tabela *ClientesCampanha*. Por fim, é obtido o URL de destino da campanha, sendo redirecionado o utilizador para o URL originalmente inserido pelo lojista na campanha.

Todo este processo acontece de forma transparente para o cliente.



Figura 5.58 – Fake URL

TOP 10 de campanhas enviadas / abertas

O TOP 10 das campanhas com maior número de aberturas (Figura 5.60) e mais enviadas (Figura 5.59) é apresentado sob a forma de uma tabela.

Essa análise é possível com base na tabela *clientes_campañas*. Cada entrada nessa tabela permite identificar o envio de uma instância de uma campanha, registando o cliente para o qual foi enviada, a respetiva data e hora.

A obtenção do número de envios de uma campanha consiste em efetuar uma contagem do número de entradas, impondo como condição o ID da campanha desejada. Para obter as dez campanhas mais enviadas efetua-se uma contagem do número de entradas da tabela, agrupando pelo ID da campanha, ordenando decrescentemente pelo número de entradas e restringindo um limite de dez registos, tal como mostra o seguinte código:

```
$top_enviadas = $this->ClientesCampanha->find('all', array(
    'fields' => array(
        'COUNT(ClientesCampanha.clicked) AS enviadas',
        'Campanha.designacao',
        'Campanha.tipos_campanha_id',
        'Campanha.id',
    ),
    'group' => 'ClientesCampanha.campanha_id',
    'order' => array('enviadas DESC'),
    'limit' => 10));
...
...
```

Podem, por exemplo ser impostas outras condições, nomeadamente o ID do cliente (obtendo todas as campanhas enviadas para um determinado cliente), ou uma determinada data (obtendo, por exemplo todas as campanhas enviadas num fim-de-semana).

A obtenção das dez campanhas mais bem-sucedidas possibilita ao *Gestor* retirar informações destas, de modo a construir as próximas de acordo com bons exemplos de sucesso (tendo em vista o aumento da taxa de visualização e uso das mesmas).

Top 10 de campanhas		Abertas	Enviadas
#	Titulo	Nº aberturas	
73	Mais uma campanha	91	🕒
74	Campanha para o Onyxbeacon	22	🕒
71	Cupão	19	🕒
75	Teste	17	🕒
1	Teste	10	🕒
76	Teste	6	🕒
91	Teste	4	🕒
88	Campanha para o Nuno - Standard	3	🕒
78	Campanha para o beacon roxo	3	🕒
72	MUSE	3	🕒

Figura 5.60 – TOP 10 campanhas
abertas

Top 10 de campanhas		Abertas	Enviadas
#	Titulo	Nº Envios	
73	Mais uma campanha	870	🕒
71	Cupão	236	🕒
74	Campanha para o Onyxbeacon	232	🕒
75	Teste	224	🕒
78	Campanha para o beacon roxo	102	🕒
77	Campanha para o beacon verde	81	🕒
83	Passagem de Ano 2015	43	🕒
76	Teste	41	🕒
90	Apenas para o grupo 1	35	🕒
72	MUSE	33	🕒

Figura 5.59 – TOP 10 campanhas
enviadas

Tipos de campanhas mais clicadas

A visualização dos tipos de campanhas “mais clicadas” é apresentada na forma de um *podium*, gerada através de um gráfico de barras. Através desta análise é possível ao *Gestor* visualizar quais os tipos de campanhas mais bem-sucedidas, e adaptar conteúdo a enviar aos clientes de acordo com os tipos de campanhas do qual “gostam” mais.

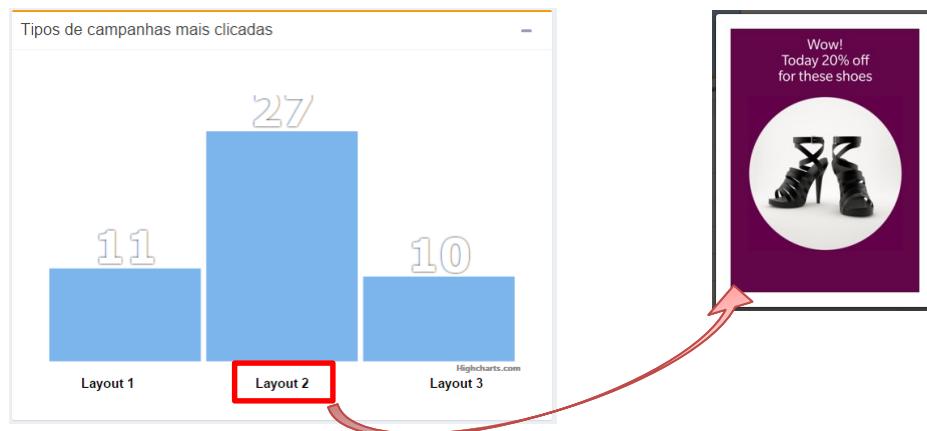


Figura 5.61 – Tipos de campanhas mais clicadas

Através do gráfico da Figura 5.61, conclui-se que campanhas enviadas com o *layout 2* possuem mais cliques do que campanhas enviadas com *layout* do tipo 1 ou 3.

Adaptar o conteúdo de acordo com o gosto do cliente permite não só aumentar a satisfação de compra do cliente como potenciar o crescimento económico, pois aumenta a possibilidade de este aderir a uma campanha.

Visualização global do Dashboard Geral

O *dashboard* geral consiste na página inicial do CMS e naquela que é apresentada ao *Gestor* após efetuar login. A Figura 5.62 apresenta uma visão global do *dashboard* geral.

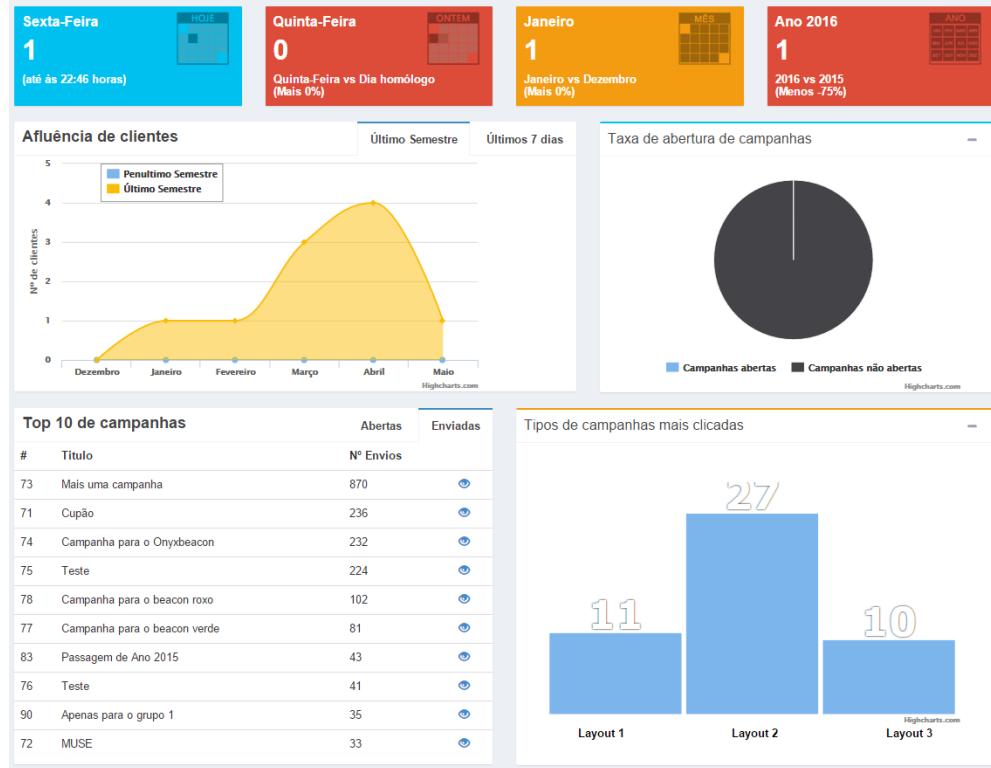


Figura 5.62 – Visão total do Dashboard Geral

5.4.3.10.2. Dashboard por Campanha

No *dashboard* por campanha é possível efetuar uma análise tendo em conta as instâncias de uma campanha enviada durante um determinado intervalo de tempo.

O seu uso assenta nos seguintes passos:

1. Escolher a campanha. Através do uso da biblioteca *multiselect* (restringido apenas a uma seleção) é possível a pesquisa pelo nome da campanha, sendo mostradas todas as correspondências de forma assíncrona;
2. Definir o intervalo de tempo. Fazendo uso da biblioteca *daterangepicker* é fácil e intuitivo indicar o intervalo de tempo pretendido. Foram pré-definidas seis opções: “Hoje”, “Ontem”, “Últimos 7 dias”, “Últimos 30 dias”, “Este mês” e “Último mês”. Se se desejar efetuar uma análise de outro intervalo basta selecionar a opção “Outro intervalo” indicando a data de início e de fim nos calendários, tal como apresentado na Figura 5.63.



Figura 5.63 – Intervalo de tempo a analisar

Este *dashboard* permite disponibilizar as seguintes informações:

- Número de instâncias da campanha enviadas;
- Número de instâncias da campanha abertas;
- Número de instâncias da campanha ignoradas;
- Número de instâncias da campanha enviadas por hora;
- Taxa de abertura da campanha;
- Alcance da campanha em número e taxa de clientes.

Número de instâncias da campanha enviadas, abertas e ignoradas

Os quadros da Figura 5.64 mostram os respetivos números.



Figura 5.64 – Número de instâncias de campanhas enviadas, abertas e ignoradas

O processo de ignorar uma campanha é registado de forma semelhante ao que acontece na sua abertura. Na aplicação móvel do cliente, existe um menu com a opção “Ignorar Campanha”. Ao selecionar essa opção irá ser invocada uma função no CMS, responsável por alterar a *flag ignored* do estado 0 para 1 na entrada da tabela *clientes_campañas*, correspondente ao envio da campanha para o respetivo cliente. Só posteriormente a campanha é removida do dispositivo do cliente.

Número de instâncias da campanha enviadas por hora

A obtenção deste valor é conseguida através de um gráfico do tipo *linha* onde no eixo dos XX se encontram representadas todas as horas de um dia e no eixo dos YY o número de campanhas enviadas.

Esta análise é conseguida através do registo, na tabela *clientes_campañas*, da data e hora sempre que é enviada uma instância de uma campanha para cada cliente.

Tabela 5.2 – Exemplo de uma entrada na tabela “*clientes_campañas*”

<i>id</i>	<i>cliente_id</i>	<i>campanha_id</i>	<i>data_hora</i>	<i>clicked</i>	<i>ignored</i>
2398	1	90	2016-01-23 10:36:10	0	0

A Tabela 5.2 mostra que a campanha com o ID 90 foi enviada para o cliente com ID 1 no dia 23 de janeiro de 2016 pelas 10 horas, 36 minutos e 10 segundos. O cliente ainda não abriu nem ignorou a campanha.

Efetuando uma contagem do número de entradas (na tabela) numa determinada data, para uma determinada campanha, e, agrupando por hora, é possível obter o gráfico seguinte.

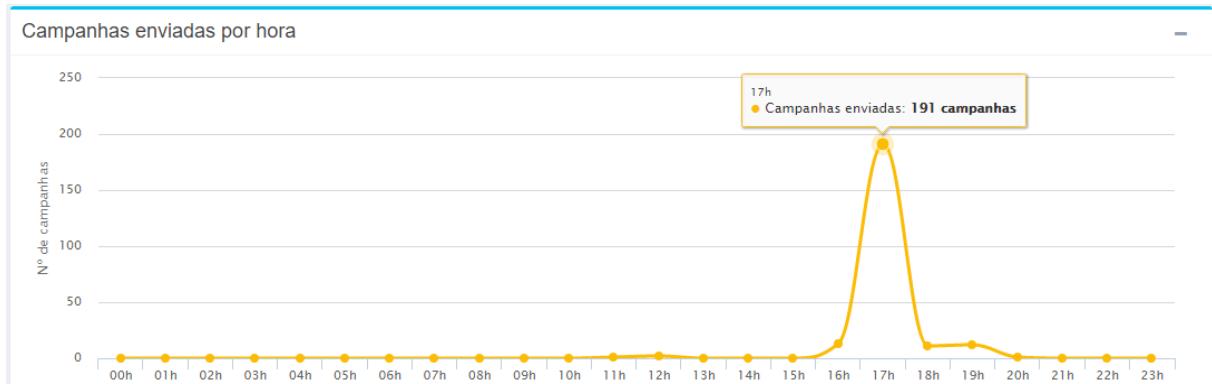


Figura 5.65 – Número de instâncias da campanha enviadas por hora

Por exemplo, segundo o gráfico da Figura 5.65 o número de instâncias enviadas para a campanha X possui o seu valor máximo às 17 horas. O *Gestor* deve tentar perceber a razão que levou a este pico, por exemplo, verificando a afluência de clientes nessa hora.

Taxa de abertura da campanha

A taxa de abertura da campanha é dada através da divisão do número de campanhas abertas pelo número de campanhas enviadas multiplicando por 100.

```
$enviadas = $this->ClientesCampanha->find('count', $options);
$abertas = $this->ClientesCampanha->find('count', $options_opened);
if ($enviadas > 0) $tax_opened = ($abertas / $enviadas);
else $tax_opened = 0;
$tax_opened = round(($tax_opened) * 100, 1);
$tax_notopened = 100 - $tax_opened;
```

A sua amostragem é apresentada através de um gráfico do tipo *pie* (Figura 5.66).

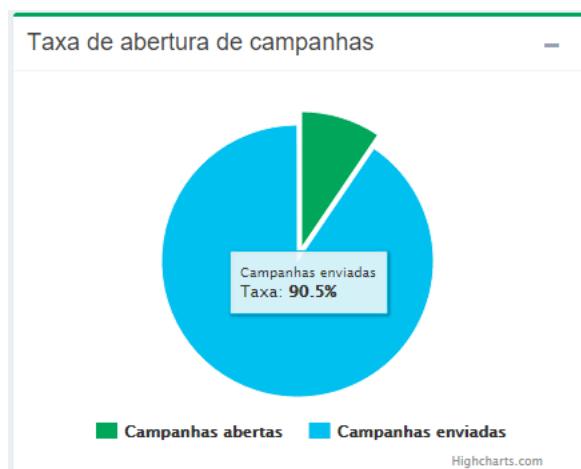


Figura 5.66 – Taxa de abertura de campanhas

Alcance da campanha em número e taxa de clientes

Através do cálculo deste valor é possível ter uma visão do número de clientes que receberam uma campanha, Figura 5.67. Assim, pode, no caso de possuir uma taxa de alcance baixa, promover novamente a campanha, uma vez que poucos a visualizaram, ou, em caso contrário, desativá-la caso já tenha atingido a taxa de clientes estipulada.



Figura 5.67 – Alcance da campanha

Visualização global do Dashboard por Campanha

A Figura 5.68 mostra a página completa correspondente a este *dashboard*.

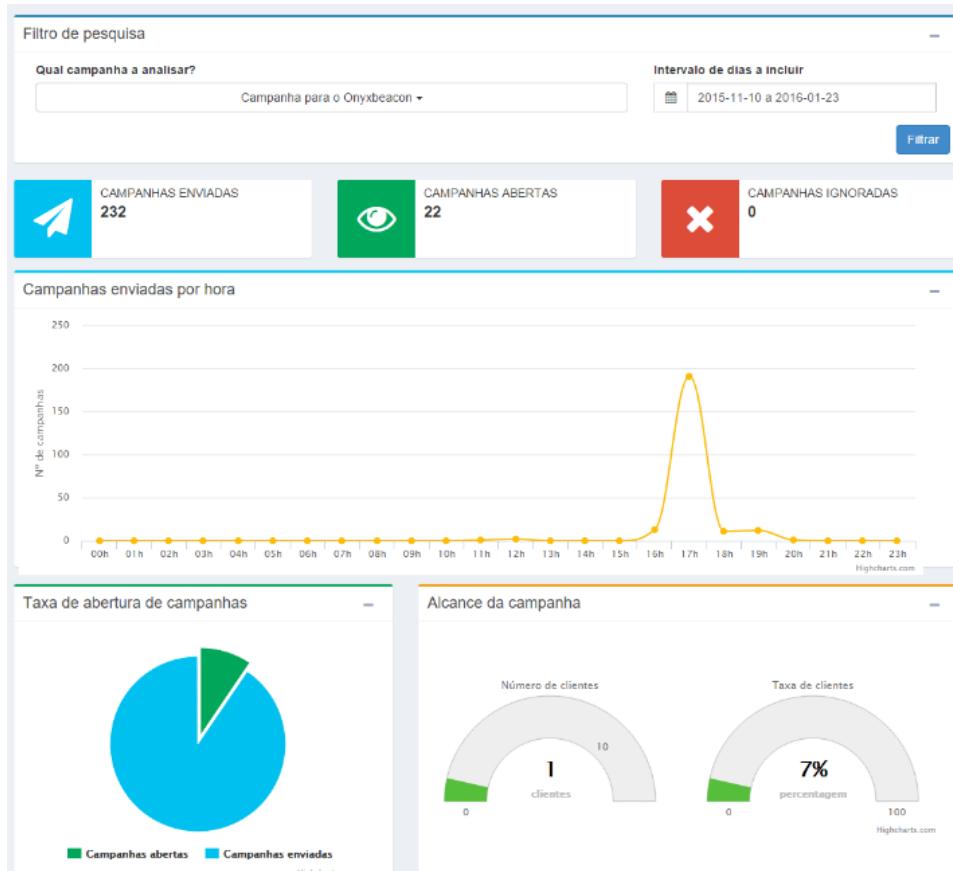


Figura 5.68 – Dashboard por Campanha

5.4.3.10.3. Dashboard por Cliente

O *dashboard* por cliente permite extrair as seguintes informações:

- Número de campanhas recebidas por cliente;
- Número de campanhas abertas por cliente;
- Número de campanhas ignoradas por cliente;
- Número de campanhas recebidas por hora;
- Taxa de campanhas abertas por cliente;
- TOP 3 dos tipos de campanhas preferidas por cliente.

O seu funcionamento assenta no *dashboard* anteriormente apresentado, distinguindo-se no facto de ser escolhido o cliente que se deseja analisar.

Número de campanhas recebidas, abertas e ignoradas por cliente

O número de campanhas recebidas, abertas ou ignoradas por cliente é apresentado através dos quadros da Figura 5.69.



Figura 5.69 – Campanhas recebidas, abertas e ignoradas por cliente

Número de campanhas recebidas por hora

Através do gráfico do tipo *linha* (Figura 5.70) é possível a visualização do número de campanhas que um cliente recebeu por hora durante o período de tempo escolhido.

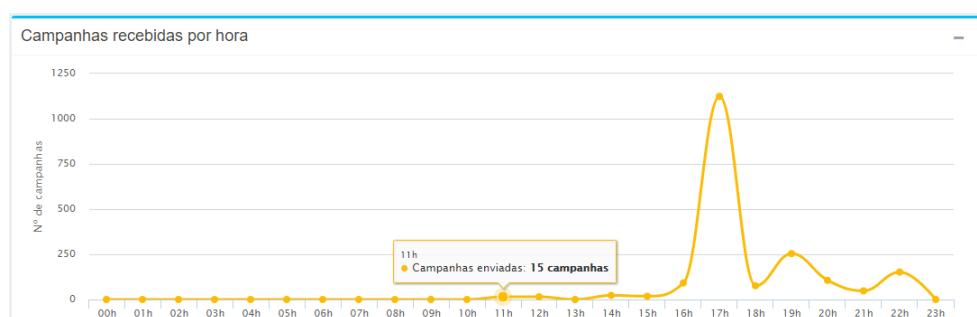


Figura 5.70 – Número de campanhas recebidas por hora

Por exemplo, de forma a ter-se noção das horas em que os clientes frequentam a loja, pode se criada uma campanha do tipo *bem-vindo*, enviando a cada cliente uma mensagem de boas vindas após entrada na loja. Sempre que o cliente recebe essa campanha é registado a sua interação com o sistema, guardando o dia e hora exata da comunicação.

O CMS e a aplicação móvel permitem controlar a granularidade do envio de campanhas, de modo a não enviar demasiadas campanhas num curto intervalo de tempo para o mesmo

cliente. Assim, por exemplo, se um cliente se dirigir à loja várias vezes no mesmo dia apenas irá receber uma campanha de boas-vindas na primeira vez que entrar na loja.

Com base nestas interações, este *dashboard*, permite obter expressões do tipo:

O cliente frequentou a sua loja entre as 11 horas e as 22 horas.

Recorrendo a algoritmos de *data mining*, por exemplo K-Means, é possível a criação de padrões de conhecimento agrupando clientes, por exemplo, por dias da semana que efetuam compras, possibilitando o envio de campanhas personalizadas a cada um. A integração do CMS com algoritmos de *data mining* consiste num trabalho futuro, atualmente já em planeamento.

Taxa de campanhas abertas por cliente

Através da relação entre o número de campanhas recebidas e o número de campanhas abertas é possível perceber, por exemplo, se o cliente se encontra “satisfeito” com as campanhas recebidas. No caso de um cliente possuir uma taxa de abertura bastante baixa (Figura 5.71) pode, indicar que as campanhas que lhe estão a ser direcionadas não vão ao encontro das suas necessidades. Neste caso, o *Gestor* deve analisar o perfil de compra do cliente e alterar o seu grupo. Esta alteração leva a que o cliente receba outro tipo de campanhas que, possivelmente, irão ao encontro das suas necessidades aumentando a satisfação do cliente e a estratégia de negócio assente nesta tecnologia.



Figura 5.71 – Taxa de abertura de campanhas por cliente

TOP 3 dos tipos de campanhas preferidas por cliente

Este *dashbord* apresenta ainda um *podium* com o tipo de campanhas “preferidas” pelo cliente. Esta análise é conseguida através da contabilização o número de aberturas de cada tipo de campanha recebida, sendo os três tipos de campanhas preferidas as que possuírem maior número de aberturas.

O *podium* é construído de forma dinâmica através de um gráfico de barras, sendo contabilizado no seu topo o número de aberturas por tipo. Por exemplo, pela Figura 5.72, conclui-se que o cliente possui preferência por campanhas cujo *layout* é do tipo 2. Pelo contrário não possui qualquer interesse por campanhas recebidas com *layout* do tipo 1.

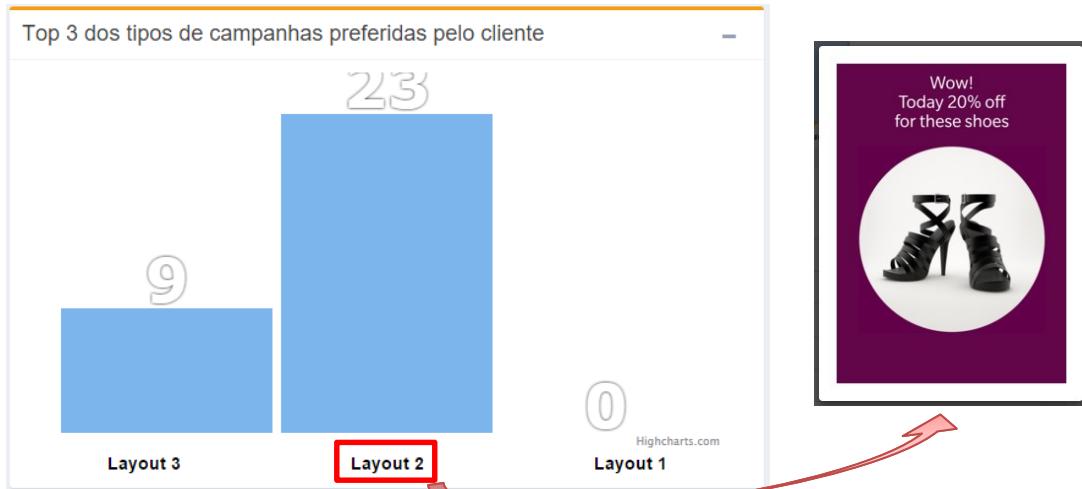


Figura 5.72 – Tipos de campanhas preferidas pelo cliente

O Gestor, neste caso, deve colocar o cliente num grupo onde sejam enviadas campanhas maioritariamente do tipo 2, de forma a receber campanhas “que mais gosta” e, com isso, elevar a sua satisfação de compra bem como o uso e interesse pela solução.

Visualização global do Dashboard por Cliente

Apresenta-se, na Figura 5.73, a página completa no CMS relativamente ao *dashboard* descrito.

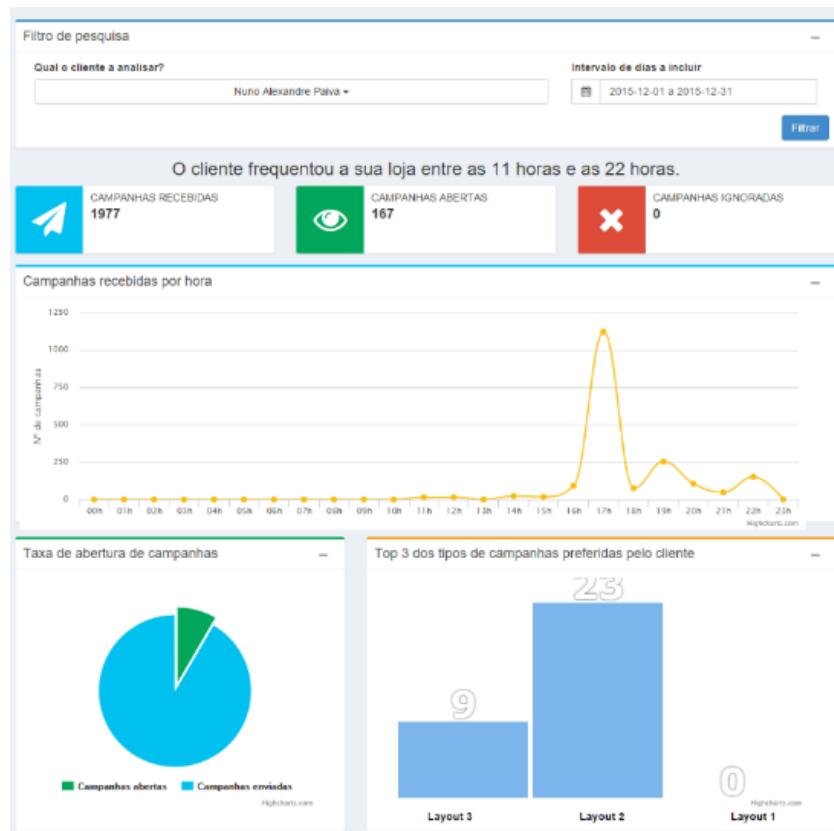


Figura 5.73 – Dashboard por Cliente

5.4.3.10.4. Dashboard por Região

Recorrendo a regiões de *beacons*, este *dashboard* permite a extração das seguintes informações:

- Total de clientes que estiveram muito próximo, perto ou longe da região de *beacons* escolhida;
- Número de clientes por *beacon* e proximidade;
- Taxa de clientes por proximidade;
- Taxa de clientes por género;
- Afluência de clientes por hora.

O seu funcionamento assenta no *dashboard* anteriormente apresentado, distinguindo-se na escolha da região de *beacons* e respetivo intervalo de tempo.

Total de clientes: muito próximo, perto e longe

Neste *dashboard* é apresentado (através dos quadros da Figura 5.74) o total de clientes que se encontraram muito próximo, perto ou longe tendo em conta todos os *beacons* pertencentes à região previamente escolhida.



Figura 5.74 – Total de clientes por região de proximidade

Taxa de clientes por proximidade

Sob a forma de gráfico do tipo *pie* (Figura 5.75) é apresentada a taxa de proximidade de todos os clientes que frequentaram a região de *beacons* selecionada.

Por exemplo, através do envio de uma campanha de boas vindas associada à região “entrada”, e estando esta definida com uma zona de proximidade até 3 metros, é possível obter a percentagem de clientes que efetivamente entrou na loja.

Por outro lado, no caso de se criar uma região específica para o envio de um vale de desconto a clientes que se aproximem de um produto (menos de 1 metro), é possível verificar a taxa de clientes que efetivamente se deslocou junto ao mesmo de forma a receber o vale.

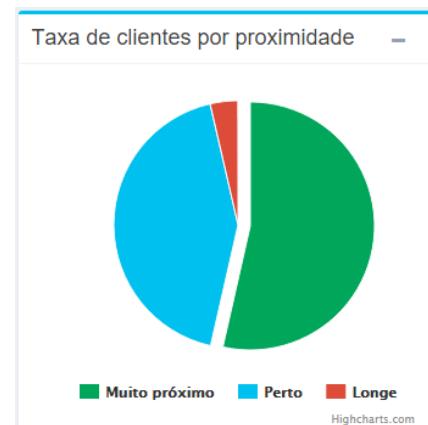


Figura 5.75 – Taxa de clientes por proximidade

Número de clientes por beacon e proximidade

De forma a aumentar a granularidade da informação é apresentada uma tabela, representada na Figura 5.76, onde se encontram listados todos os *beacons* pertencentes à respetiva região selecionada, sendo para cada um contabilizado o número de clientes para as três zonas de proximidade.

Usando o exemplo anterior, e assumindo que cada entrada se encontra associada a um *beacon*, é possível perceber, no caso de um dia de baixa afluência de clientes, qual a entrada que recebeu menos clientes e tomar medidas e forma a melhorar essa tendência.

Por outro lado, no caso de uma região associada a vales de desconto e assumindo que cada *beacon* se encontra associado a um produto, é possível perceber, por exemplo, qual o produto que teve maior interesse por parte dos clientes, ou seja, que se deslocaram junto ao mesmo de forma a receber o vale.

Número de clientes por beacon e proximidade

#	Beacon	Imediato	Perto	Longe
	Verde	12	7	1
	Ciano	1	3	0
	Onyx Beacon 2	0	0	0
	Onyx Beacon 1	2	2	0

Figura 5.76 – Número de clientes por beacon e proximidade

Taxa de clientes por género

A percentagem de clientes que frequentaram uma determinada região por género (masculino ou feminino) é dada por um gráfico do tipo *pie* (Figura 5.77).

A identificação do cliente por parte da aplicação móvel e a inserção do seu género (quando do registo no CMS) permitem responder a perguntas do tipo:

- Qual o género de clientes por tipo de produto?
 - Os homens compram mais produtos eletrónicos?
 - As mulheres compram mais produtos alimentares?
- Quais as seções da loja mais frequentadas por homens e mulheres?



Figura 5.77 – Taxa de clientes por género

A resposta às questões anteriores obtém-se com recurso a regiões de *beacons* associadas a cada seção da loja, e a envio de vales de desconto por tipo de produto, efetuando posteriormente a análise da taxa de clientes por género resultante da interação dos clientes com a solução.

Afluência de clientes por hora

Mostra-se ainda um gráfico linear (Figura 5.78), onde se apresenta o número de clientes que frequentou uma determinada região por hora, permitindo responder a questões do tipo:

- Qual a hora de maior afluência de clientes na loja (ou numa seção)?
- Qual a hora que os clientes se encontram mais interessados num determinado produto?

Esta análise consiste num somatório de:

- Beacons pertencentes à região selecionada;
- Intervalo de dias selecionados;
- Clientes que entraram em contato com cada beacon.



Figura 5.78 – Afluência de clientes por hora

Visualização global do Dashboard por Região

A Figura 5.79 apresenta uma visão global do *dashboard* descrito.

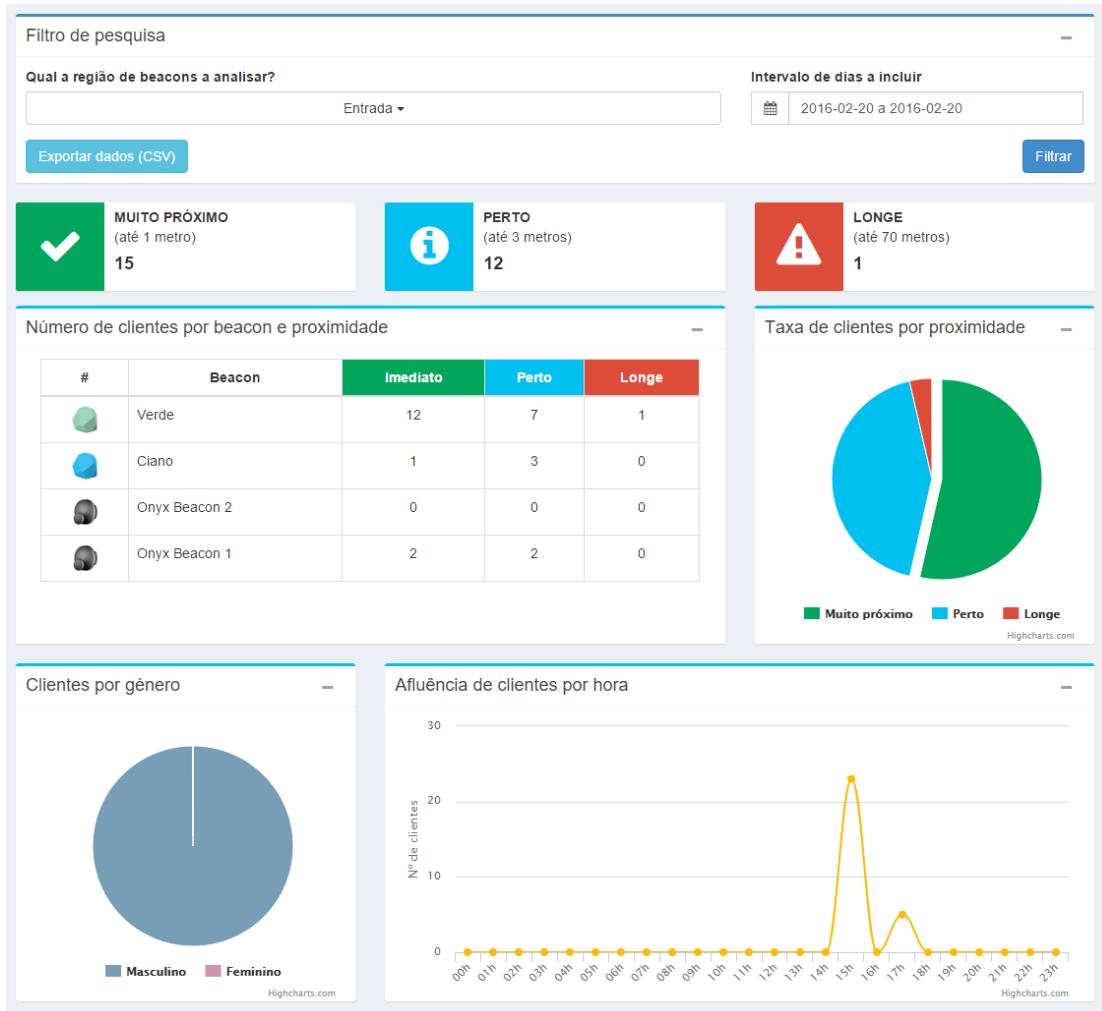


Figura 5.79 – Dashboard por Região

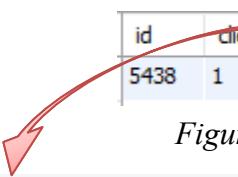
5.4.3.10.5. Percurso dos Clientes

Este módulo permite rastrear o percurso efetuado por um determinado cliente (por piso e num determinado intervalo de tempo), sendo composto por duas áreas:

- Filtragem de dados:
 - Área onde é possível a escolha do cliente sobre o qual se deseja analisar o percurso;
 - Planta onde deseja efetuar essa análise;
 - Data de início e fim do rastreamento;
- Resultado:
 - Área onde é apresentado o percurso efetuado pelo cliente consoante os dados escolhidos no filtro anterior.

Explicação técnica

Sempre que o dispositivo do cliente se aproxima de um *beacon*, é registada na tabela *percursos_clientes* essa ocorrência, identificando o cliente, o *beacon*, a data e hora dessa interação e a distância aproximada (em metros) a que o cliente se encontrava do *beacon*.



id	cliente_id	beacon_id	data_hora_contacto	distancia_aproximada
5438	1	27	2016-02-20 17:16:37	0.38

Figura 5.80 – Entrada na tabela “percursos_clientes”



id	loja_id	plant_id	designacao	uuid	major	minor	pos_x	pos_y
27	6	17	Onyx Beacon 1	xxxxxxxxxxxxxxxx	xxxxx	xxxx	410	86

Figura 5.81 – Entrada na tabela “beacons”



id	loja_id	designacao	local_imagem	piso
17	6	Piso 6	545396a7-cc...	6

Figura 5.82 – Entrada na tabela “plantas”

Através dos regtos apresentados na tabela *percursos_clientes* (Figura 5.80), sabe-se que o cliente com ID 1 (Nuno Paiva) se aproximou do *beacon* com ID 27 (*Onyx Beacon 1*) no dia 20 de fevereiro 2016 pelas 17 horas e 16 minutos. Verificando os dados do *beacon* (tabela *beacons*, Figura 5.81) com ID 27, conclui-se que este se encontra localizado na planta com ID 17 numa determinada posição XX e YY. Através da tabela *plantas* (Figura 5.82) verifica-se que o *beacon* se encontra situado no piso 6.

Os dados anteriores mostram que o cliente Nuno Paiva esteve no dia 20 de fevereiro 2016 pelas 17 horas e 16 minutos no piso 6, na área de venda de roupa (dado pelas coordenadas XX e YY do *beacon* mapeado na planta).

Uma vez que esta análise se torna complexa aquando da existência de milhares de regtos, para além de ser possível a exportação para o formato *Comma-separated values* (CSV)

(para servir de *input (dataset)* a algoritmos de análise de dados), o CMS permite a sua representação de forma visual.

Funcionamento do módulo

Após a seleção dos dados pretendidos no filtro e clicar no botão *Filtrar* é despoletado um evento via AJAX enviando todos os dados escolhidos via POST para a função *percurso_utilizadores_ajax* pertencente ao controlador *BeaconsController*. Esta é responsável por efetuar a pesquisa na DB, processar e devolver os dados à View respetiva.

Fazendo uso da biblioteca *excanvas.js* é desenhada a planta escolhida, seguida dos *beacons* nas suas posições, e por fim as setas correspondentes ao percurso efetuado pelo cliente, ordenado pela data indicada.

De forma a ser perceptível o respetivo percurso, ou seja, perceber-se qual o lugar onde o cliente esteve primeiramente e para onde se deslocou de seguida, foi usado um gradiente de cor nas setas representativas. Assim, o início do percurso é representado pela cor azul, sendo feito um gradiente até à cor vermelha à medida que o percurso se aproxima do final, tal como mostra a Figura 5.83.



Figura 5.83 – Legenda “Percursos dos clientes”

Todo este processo ocorre sem que haja um recarregamento da página. A Figura 5.84 mostra, de forma resumida, o código que permite o funcionamento do módulo.

Filtragem de dados

Cliente
Nuno Alexandre Paiva

Escolha a planta
Piso 6

Data de inicio
2016-01-23

Data de fim
2016-01-23

Filtrar

View: percursos_utilizadores_ajax

```
$("button").click(function () {
    var utilizador = $("select option:selected").val();
    var data_inicio = document.getElementById("data_inicio").value;
    var plantas = document.getElementById("plantas").value;
    var data_fim = document.getElementById("data_fim").value;
    $.ajax({
        type: "POST",
        url: "/beacons/percurso_utilizadores_ajax/" * 
        data: {
            data_inicio: data_inicio,
            data_fim: data_fim,
            plantas: plantas,
            utilizador: utilizador},
        success: function (result) {
            $("#percurso_cliente").html(result);
        }
    });
});
```

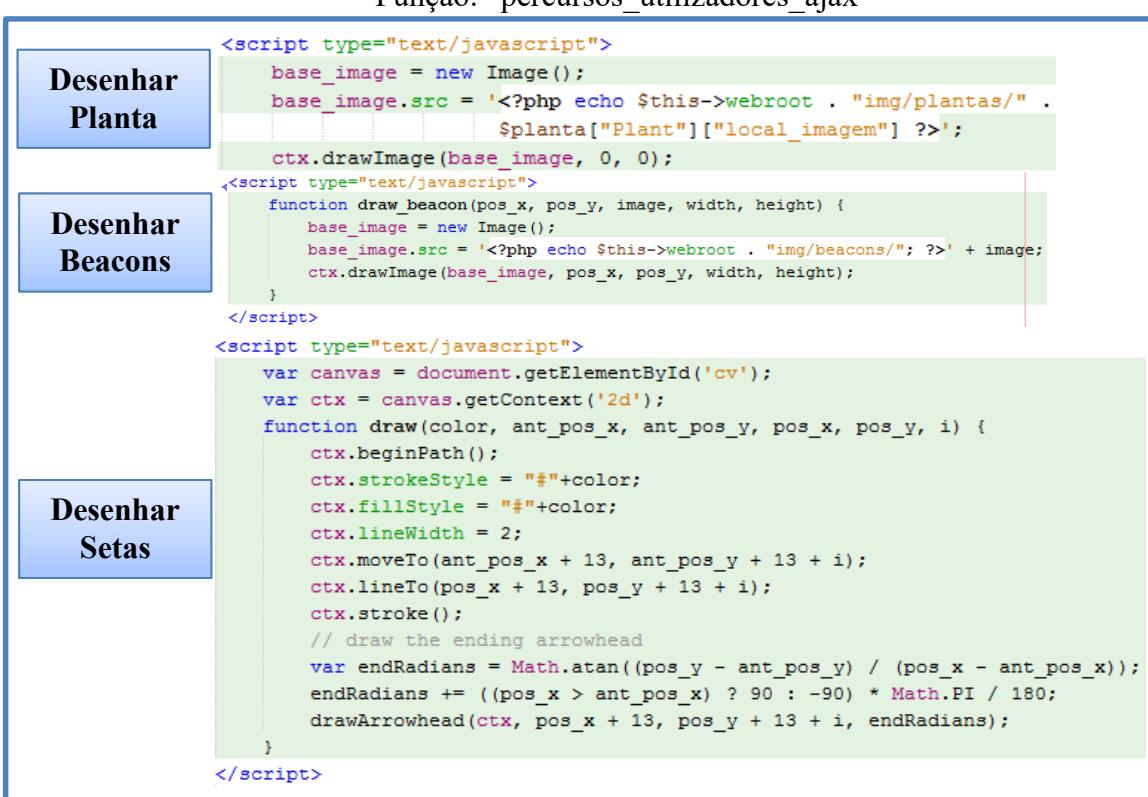


Figura 5.84 – Esquema do funcionamento da seção de percurso de utilizadores

Exemplo prático

Por exemplo, se se desejar saber o percurso efetuado pelo cliente Nuno Paiva no Piso 6 no dia 23 de janeiro 2016, após escolha desses valores, na área de filtragem de dados, são apresentadas as setas correspondentes ao percurso efetuado pelo cliente com base na hora de proximidade com cada beacon.

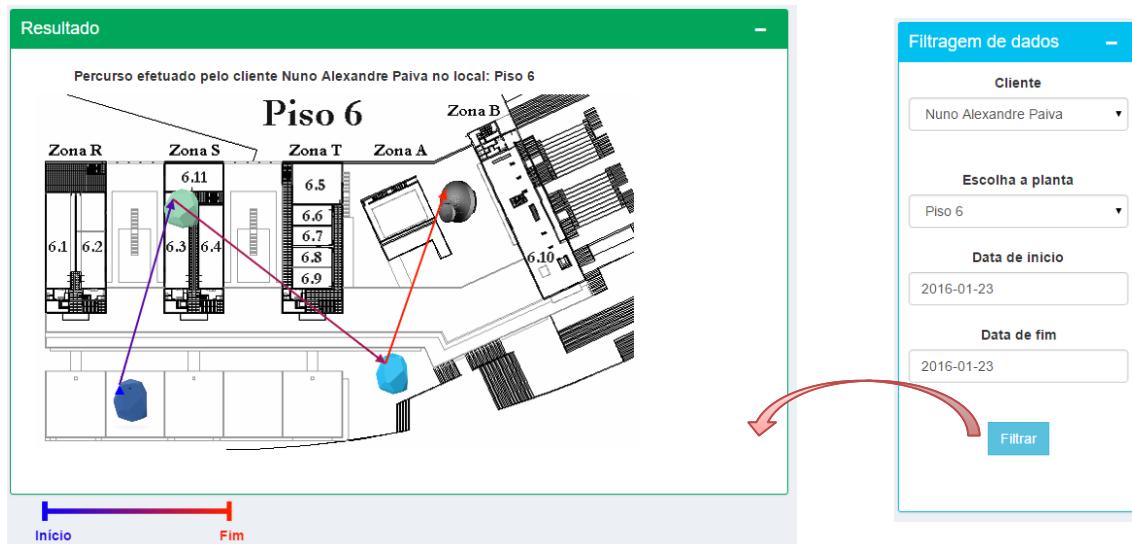


Figura 5.85 – Exemplo de um percurso efetuado por um cliente

Através do resultado obtido (Figura 5.85), verifica-se que o cliente entrou pela porta Este (localização do *beacon* Roxo), dirigiu-se seguidamente à seção de roupa (*beacon* verde), passando pela área de alimentos (*beacon* azul), acabando por sair pela porta Oeste (*beacon* preto).

A análise do percurso dos clientes possibilita a tomada de medidas que possam melhorar a experiência de compra do cliente. Por exemplo, se um cliente entra sempre pela porta Este e se dirige a maior parte das vezes à seção de roupa, então após a entrada do cliente, poderá receber uma campanha de desconto referente a essa secção.

Analizando o comportamento de outros clientes, pode-se chegar à conclusão que a porta Oeste possui uma baixa taxa de utilização, devendo nesse caso o *Gestor* tomar medidas como forma de elevar o seu negócio, por exemplo, a alteração de disposição dos produtos da porta Oeste para a porta Este.

A vista apresentada é útil para análise de um cliente em específico. Para uma análise completa do comportamento de diversos clientes, é possível a exportação de dados sob a forma de CSV de modo a ser usado como *dataset* em algoritmos de *data mining*, conseguindo obter padrões e extração de conhecimento mais pormenorizados.

5.4.3.10.6. Heatmap

O CMS permite ainda a criação de *heatmaps*. Um *heatmap* (mapa de calor, em português) consiste numa representação gráfica recorrendo ao uso de cores associadas a um conjunto de dados.

O esquema de cores mais usado é o arco-íris dado a sua fácil percepção para o ser humano. Fazendo uso deste esquema, quanto maior a quantidade de dados a representar pela cor, mais quente será, ou seja, cores frias representam poucos dados (ou valores baixos) sendo que cores quentes representam enormes quantidades de dados (ou valores elevados). Entre o valor mínimo e valor máximo da amostra é efetuado um gradiente entre a cor fria (por exemplo, branco) e a cor quente (por exemplo, vermelho).

Para a criação de *heatmaps* no CMS foi usada a biblioteca *heatmap.js*.

Funcionamento do módulo

O funcionamento deste módulo assemelha-se ao *dashboard* por cliente e campanha. Neste, o *Gestor* possui uma área de filtro de dados onde é possível a escolha da planta (que deseja ser alvo da criação do *heatmap*) e o respetivo intervalo de tempo que se deve ter em conta. Após clicar no botão “Criar Heatmap” os dados são enviados via POST para o controlador *PagesController* (e respetiva função *heatmap*) onde serão pesquisados e devolvidos à mesma *View*.

O resultado é conseguido através da apresentação da planta escolhida, seguido dos *beacons* (nas suas posições, associados a esta), finalizando o processo com invocação de

funções *Javascript* que permitem a mostragem de cores consoante a quantidade de dados por *beacon*.

Cada *heatmap* é criado com base no número de clientes que se aproximaram de cada um dos *beacons*. Assim, locais que foram frequentados por bastantes clientes terão uma cor mais quente (mais avermelhada) face a locais onde a frequência de clientes foi menor (cor mais fria, podendo ser nula).

Exemplo de utilização

Por exemplo através do filtro na Figura 5.86 criou-se um *heatmap* onde é possível visualizar as áreas mais frequentadas pelos clientes na planta “Piso 6” durante o mês de janeiro.



Figura 5.86 - Heatmap

Como forma de precisar o número de clientes por local, existe uma tabela (Figura 5.87) onde é apresentado o número de clientes que comunicou com cada *beacon*.

Número de clientes por beacon		
#	Beacon	Nº Clientes
	Roxo	10
	Verde	0
	Ciano	9
	Onyx Beacon 1	14

Figura 5.87 – Número de clientes por beacon

É ainda apresentado o número total de clientes que frequentaram a planta selecionada no período de tempo respetivo (dada pela expressão abaixo):

Frequentaram 33 clientes na planta "Piso 6" entre 2016-01-01 e 2016-01-31.

Tendo em conta o *heatmap* anterior, é notória a baixa afluência de clientes na área de venda de alimentos (*beacon* verde) durante o mês de janeiro. Importa entender se foi uma situação anormal ou a tendência tem vindo a repetir-se ao longo do ano. O *Gestor* deve verificar outros intervalos de tempo, por exemplo, selecionar todo o ano anterior e verificar se essa baixa afluência de clientes nessa área se mantém. Caso se verifique deve ponderar possibilidades como: fecho da área; alteração do local; alteração do tipo de produtos vendidos na mesma, entre outros.

Esta análise possibilita ao *Gestor* visualizar áreas que estão a prejudicar o seu negócio e tomar decisões de forma a minimizar as suas perdas.

Visualização global do módulo de Heatmap

A Figura 5.88, mostra a página completa correspondente ao módulo de *heatmap*.



Figura 5.88 – Heatmap: Seção completa

5.4.3.11. Exportação de dados

Para além de fornecer os *dashboards* anteriormente apresentados o CMS permite, para todos eles, e respeitando sempre todos os filtros efetuados, a exportação de dados no formato CSV.

Esta funcionalidade permite dar continuidade à análise de dados de forma mais aprofundada em *softwares* específicos para o efeito, por exemplo *rapidminer*.

Assim, em todos os *dashboards*, após efetuada a filtragem da informação que se deseja, é apresentado um botão “Exportar dados (CSV)” responsável por descarregar para a máquina local um ficheiro no formato CSV.

Os dados existentes no ficheiro não consistem no conteúdo tratado e apresentado no respetivo *dashboard*, mas em toda a informação armazenada em DB que permite a sua construção.

A Figura 5.89 mostra o procedimento de exportação para o *dashboard* por região.

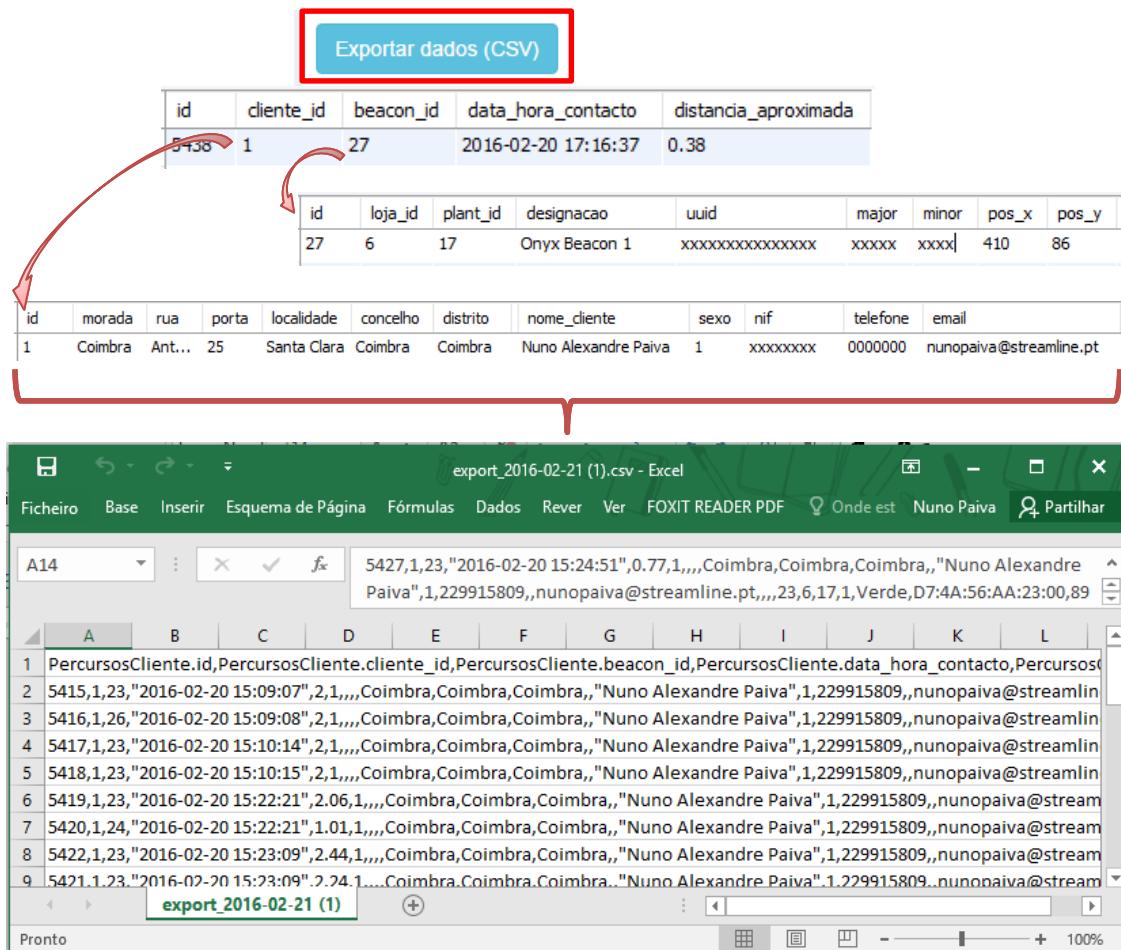


Figura 5.89 – Processo de exportação de ficheiro CSV

5.4.3.12. Integração com sistemas externos

O CMS, para além de possibilitar a criação de conteúdo para utilização própria da solução, possibilita ainda que o seu conteúdo seja disponibilizado de forma a ser acedido por sistemas externos.

Até à data o CMS foi integrado com um sistema de *CorporateTV* desenvolvido pela **Streamline**, cujo *workflow* é apresentado na Figura 5.90. Concretamente, através da receção de argumentos via POST (tais como: data e hora) e no formato JSON, o CMS permite devolver campanhas e respetivos conteúdos associados (igualmente no formato JSON) de acordo com as necessidades.

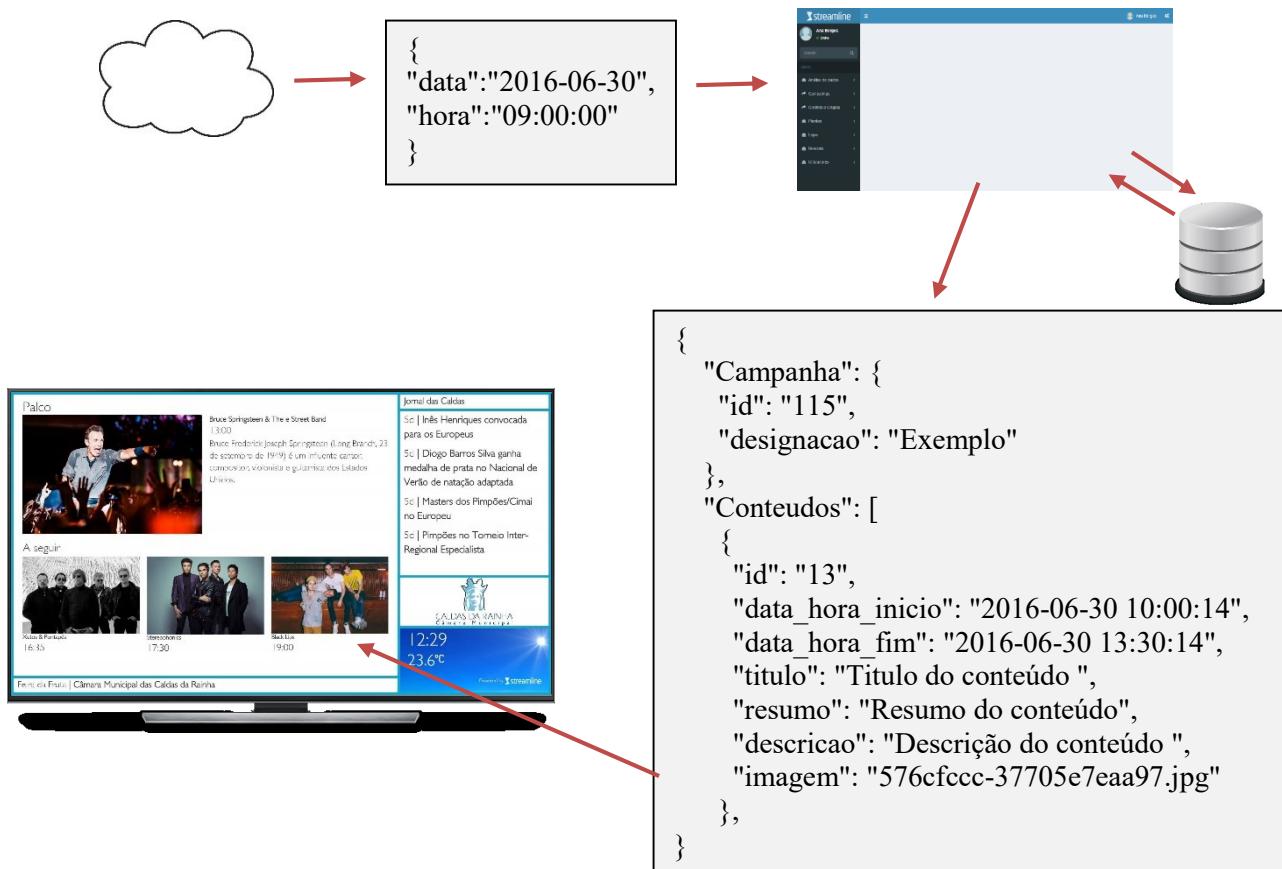


Figura 5.90 – Integração do CMS com sistemas externos

Esta flexibilidade permite aumentar o alcance da distribuição da informação. Por exemplo, campanhas informativas destinadas ao público em geral, para além de serem distribuídas através dos respetivos *beacons* (e obtidas nos *smartphones*), podem ser apresentadas em tempo real e de forma dinâmica em *CorporateTV's* existentes na loja. Assim, por exemplo, enquanto o cliente aguarda pela sua vez (numa fila), pode visualizar informação útil, melhorando a sua experiência de compra. Para o gestor consiste numa mais-valia no sentido em que permite alcançar clientes que, por exemplo, não fazem uso regular do seu *smartphone*.

6. APLICAÇÃO ANDROID

O capítulo que agora inicia destina-se a apresentar a aplicação móvel desenvolvida para o sistema *Android*, organizando-se da seguinte forma: Descrição da componente; Requisitos da aplicação; Conceção da aplicação e Comunicação com o CMS e Receção de campanhas.

6.1. Descrição da componente

Como parte integrante da solução apresentada, foi definido que deveria ser desenvolvida uma aplicação móvel, para o sistema *Android* responsável pela comunicação com os *beacons* e o CMS.

Sendo uma solução genérica, adaptável a qualquer cliente da área do retalho, estabeleceu-se que a aplicação deveria possuir um *design* genérico onde o seu principal objetivo visa a comunicação com todo o tipo de *beacons*, comunicação com o CMS e respetiva receção e amostragem de campanhas.

Dado que cada cliente possui os seus requisitos (por exemplo, nome, logotipo e *design*), a aplicação deve ser customizada, acrescentando funcionalidades específicas, à medida de cada cliente.

6.2. Requisitos da aplicação

No que diz respeito aos requisitos funcionais da aplicação, a identificação do cliente através da obtenção do seu *Email* (registado no *smartphone*), e do seu número de telefone, de forma transparente para o utilizador, foi definido como prioridade máxima. Com prioridade elevada foi também definida a compatibilidade com todos os *beacons* (independentemente do seu fabricante), desde que respeitem a norma *iBeacon*, obtendo e enviando para o CMS os parâmetros: *UUID*, *Major*, *Minor*, *TXPower* e *Mac Address*. Foi definido ainda que a aplicação deveria receber campanhas de três tipos, nomeadamente *Standard*, *Passbook* e *Web* operando e enviando dados de interação do cliente com as mesmas para o CMS.

O Apêndice III apresenta detalhadamente os requisitos funcionais e não funcionais especificados e implementados na aplicação *Android*.

6.3. Conceção da Aplicação

Esta seção documenta as principais opções tomadas no desenvolvimento da aplicação, bem como as principais áreas de desenvolvimento.

6.3.1. Verificações de serviços ativos

Uma vez que tecnologia *iBeacon* faz uso da tecnologia BLE, a primeira verificação feita pela aplicação consiste em detetar se o *Bluetooth* do dispositivo se encontra ligado. Caso isso

não se verifique, então é apresentada uma mensagem ao utilizador (tal como apresentado na Figura 6.1) alertando-o que esse serviço deve encontrar-se ligado de modo a poder usar a aplicação.

Tecnicamente, essa verificação é executada na função *onResume()* da *Main Activity* sendo invocada sempre que a aplicação é executada. Para efetuar essa verificação foi usada a *class BluetoothAdapter*, efetuando essa verificação através do código abaixo.

```
protected void onResume() {
    super.onResume();
    if ((mBTAdapter != null) && (!mBTAdapter.isEnabled())) {
        Toast.makeText(this, R.string.bt_not_enabled, Toast.LENGTH_SHORT).show();
    }
}
```

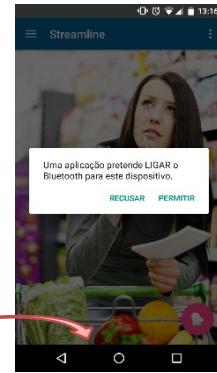


Figura 6.1 – Verificar se o Bluetooth se encontra ativo

Para que as aplicações possuam acesso ao serviço de *Bluetooth* do dispositivo é necessário que sejam inseridas as seguintes linhas no ficheiro *AndroidManifest.xml*.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Com o lançamento da versão 6 (*marshmallow*) do sistema Android tornou-se necessário acrescentar as linhas destacadas no código acima, permitindo informar o utilizador, aquando da instalação, que a aplicação poderá detetar a sua localização, pedindo permissões de acesso ao utilizador.

6.3.2. Identificação do cliente

Na fase de levantamento de requisitos foi definido que a aplicação deveria obter uma identificação do dispositivo de forma autónoma e transparente para o cliente. Assim, após a aplicação iniciar, obtém os seguintes dados do dispositivo:

- Número de telemóvel do dispositivo;
- *Email* registado no dispositivo;
- *Mac Address* do dispositivo.

Com estas informações o cliente encontra-se identificado sem que tenha de introduzir quaisquer dados, por exemplo efetuar um login na aplicação.

Assim, quando a aplicação comunicar com o CMS, (e enviar os dados obtidos), através de uma pesquisa na DB será possível identificar o cliente (Figura 6.2), mesmo que apenas tenha sido registado no CMS um dos três dados, por exemplo o número de telemóvel.

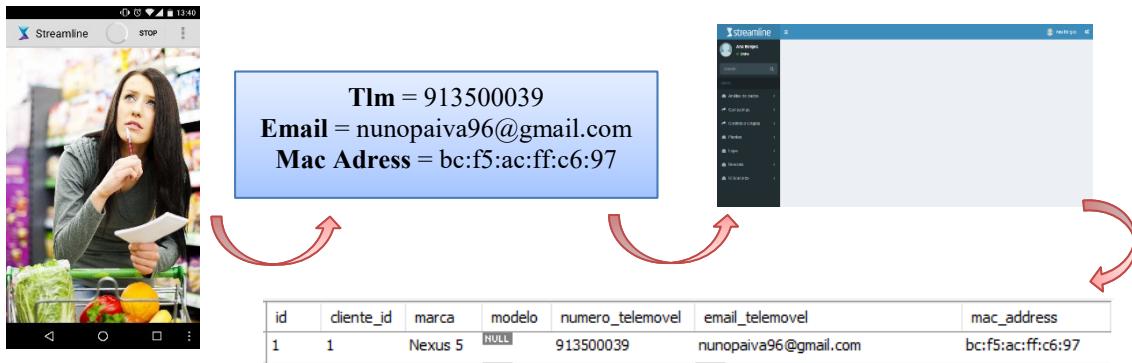


Figura 6.2 – Identificação do cliente

6.3.3. Detecção de beacons

Existem bastantes API's que permitem efetuar a comunicação com *beacons*. Os principais fabricantes, por exemplo *Estimote*, aquando da compra dos seus *beacons* fornecem uma API proprietária que permite a comunicação com os respetivos *beacons*. No entanto, essa API (como todas as API's de fabricantes de *beacons*) apenas permite a comunicação com os seus próprios *beacons*, não detetando *beacons* de outros fabricantes mesmo que respeitem a norma *iBeacon*.

Para que a solução desenvolvida fosse compatível com quaisquer *beacons*, independentemente do seu fabricante, foi usada a biblioteca *Android Beacon Library* [58].

6.3.3.1. Como é feita a deteção?

De modo a que um dispositivo móvel detete a presença de um *beacon* é necessário que efetue um *scan* procurando por *beacons* nas proximidades. Tal como os *beacons* emitem um sinal periódico informando da sua presença, também o dispositivo que deseja encontrar o *beacon* deve efetuar um *scan* por dispositivos BLE com uma determinada periodicidade.

Assim, após a aplicação iniciar é instanciada a classe que permite gerir a forma como é feito o scan de *beacons*, representado pelo código seguinte:

```
beaconManager = BeaconManager.getInstanceForApplication(this)
beaconManager.bind(this);
```

6.3.3.2. Poupança de bateria

Um dos pontos principais que contribui para o bom funcionamento da aplicação consiste na boa gestão do consumo de energia do dispositivo.

O processo de *scan* de dispositivos BLE consiste num processo “pesado” que provoca um elevado consumo de bateria quando executado intensivamente.

Assim, na aplicação desenvolvida foi usada a classe *BackgroundPowerSaver* existente na biblioteca utilizada, sendo criada uma instância no início do arranque da aplicação.

```
backgroundPowerSaver = new BackgroundPowerSaver(this);
```

Através desta classe é possível alterar o tempo do *scan*, bem como a periodicidade com que é executado. A alteração destes parâmetros deve ter em conta os seguintes pressupostos: quanto maior for o tempo entre cada *scan* maior será o tempo para detetar um *beacon* (se for efetuado um *scan* de 5 em 5 minutos a deteção de um *beacon* só ocorrerá 5 minutos após a entrada na zona de proximidade); quanto mais curto for o tempo de execução de um *scan* maior a probabilidade de ser perdido um anúncio enviado pelo *beacon*. Segundo a norma *iBeacon* não é aconselhável que o tempo de execução de um *scan* seja inferior a 1,1 segundos uma vez que um *beacon* envia em média 1 anúncio por segundo. Como o *scan* é feito com uma determinada periodicidade, permite obter anúncios de *beacons* no *scan* seguinte caso não tenha sido detetado no *scan* anterior. O código seguinte permite a configuração de ambos os parâmetros.

```
beaconManager.setBackgroundScanPeriod(11001);  
beaconManager.setBackgroundBetweenScanPeriod(60001);
```

Perante o código acima, a aplicação irá efetuar o *scan* de *beacons* de 60 em 60 segundos com um tempo por *scan* de 1,1 segundos. Caso a aplicação esteja em *background* estes tempos podem ser alterados, normalmente aumentados, de forma a otimizar o consumo de bateria.

6.3.3.3. Cálculo da distância

O cálculo da distância a que o utilizador se encontra de um *beacon* possibilita priorizar as campanhas a mostrar ao utilizador.

Por exemplo, se o utilizador se encontra na proximidade de dois *beacons* qual o *beacon* a ter em conta? Foi definido na fase de levantamento de requisitos que a aplicação, nestes casos, deve priorizar os *beacons* com base na distância a que se encontra. Assim, um utilizador irá receber a campanha sempre do *beacon* que se encontre mais perto.

A aplicação faz uso de uma função fornecida pela API que permite estimar aproximadamente a distância entre o dispositivo e o *beacon* com base na potência do sinal recebida.

Não deve ser esperada uma alta precisão no cálculo da distância devido a ruido, reflexões ou obstruções do sinal poderem influenciar a sua precisão, tal como foi apresentado no capítulo 3 (seção 3.3.5). Com o aumento da distância entre o *beacon* e o dispositivo receptor, o cálculo da precisão diminui de forma exponencial. Por exemplo, quando é estimada uma distância de 1 metro entre o dispositivo e o *beacon*, na realidade pode-se encontrar, entre 50 centímetros e 2 metros. Quando a distância aumenta, a precisão diminui. Por exemplo, se for estimada uma distância de 20 metros, na realidade o dispositivo pode-se encontrar entre 10 a 30 metros [59].

Para além de fatores externos influenciarem o cálculo da distância, também os dispositivos podem influenciar nesse cálculo. A mesma aplicação em dispositivos diferentes e à mesma distância de um *beacon* pode apresentar distâncias diferentes. A origem dessa variação encontra-se no *chipset Bluetooth* e na sua antena (ambos diferentes nos vários dispositivos) o que pode levar a receberem potências de sinais distintas. Como forma de uniformizar o cálculo, a API usada possui um ficheiro de configuração onde irá usar valores consoante o dispositivo onde se encontre a ser executado.

Fórmula do cálculo de distância

A fórmula que permite aproximar o cálculo da distância em metros é dada através de uma regressão de potência e um conjunto de dados constantes para cada dispositivo, nomeadamente:

$$d = A \times \frac{r}{t}^B + C$$

d → Distância em metros;

r → TX Power, potência do sinal recebida pelo dispositivo;

t → RSSI, potência do sinal de referência quando o dispositivo se encontra a 1 metro do *beacon*;

A, B e C → Constantes definidas por dispositivo. Assinaladas no ficheiro *model-distance-calculations.json*.

model-distance-calculations.json

```
{
  "models": [
    {
      "A": "coefficient1": 0.42093,
      "B": "coefficient2": 6.9476,
      "C": "coefficient3": 0.54992,
      "version": "4.4.2",
      "build_number": "LPV79",
      "model": "Nexus 5",
      "manufacturer": "LGE",
      "default": true
    },
    {
      "coefficient1": 0.9401940951,
      "coefficient2": 6.170094565,
      "coefficient3": 0.0,
      "version": "5.0.2",
      "build_number": "LXG22.67-7.1",
      "model": "Moto X Pro",
      "manufacturer": "XT1115",
      "default": false
    }
  ]
}
```

A função “*beacons.iterator().next().getDistance()*” usada na aplicação faz uso da fórmula apresentada facilitando o processo ao programador.

6.4. Comunicação com o CMS e receção de campanhas

O processo de comunicação com o CMS inicia-se após a deteção de um *beacon* nas proximidades do dispositivo móvel. Consoante a distância estimada a que se encontre o dispositivo do *beacon* são tomadas decisões distintas. Para efeitos de explicação será abordado o processo de deteção de sinal de um *beacon*.

O código seguinte permite desencadear uma determinada ação caso seja detetado um *beacon* nas proximidades.

```

beaconManager.setRangeNotifier(new RangeNotifier() {
    @Override
    public void didRangeBeaconsInRegion(Collection<Beacon> beacons, Region region) {
        if (beacons.size() > 0) {
            Log.i(TAG, "Encontrei um beacon a "+beacons.iterator().
                next().getDistance()+" metros.");
        }
    }
});

```

Cada *scan* executado pela aplicação é efetuado em *background* por uma *thread* gerida pela API usada.

Após a deteção do *beacon*, é executada a função *send_data_to_server()* responsável por enviar as seguintes informações para o CMS:

- Identificação do dispositivo:
 - Número de telemóvel do dispositivo;
 - *Email* registado no dispositivo;
 - *Mac Address* do dispositivo;
- Identificação do Beacon:
 - *UUID*;
 - *Major*;
 - *Minor*;
- Dados da interação:
 - Data e hora da comunicação;
 - Distância aproximada entre o dispositivo e o *beacon*.

A comunicação com o CMS é feita via HTTP, sendo enviados todos os valores (inseridos num *ArrayList*) através do método POST e recorrendo ao serviço *AsyncTask* de forma a não bloquear a interface do utilizador enquanto é executada a comunicação.

Os dados são recebidos pela função *receive_data_client()* existente no controlador *campanhas* do CMS. Após identificação do cliente, registo de atividade e processamento dos dados, é enviado sob o formato JSON uma resposta para a aplicação. Essa resposta contém o ID da campanha que irá receber, o título e detalhes a mostrar na notificação *push* (Figura 6.3), bem como o tipo de campanha (*Standard* ou *Passbook*) irão ser desencadeados processos distintos.

```

jsonObjRecv = new JSONObject(data_from_server);
String campanha_id = jsonObjRecv.getString("id");
String tipos_campanha_id = jsonObjRecv.getString("tipo");
String titulo = jsonObjRecv.getString("notificacao_titulo");
String detalhe = jsonObjRecv.getString("notificacao_detalhes");

showNotification(titulo,detalhe);

if(tipos_campanha_id.equals("1")){
    // Standard
} else if(tipos_campanha_id.equals("2")){
    // Passbook
}

```

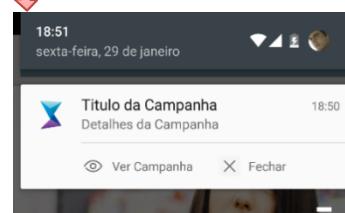


Figura 6.3 – Receção de notificação da campanha

6.4.1. Tipo de campanha: Standard

Tal como indicado, a solução disponibiliza três *layouts* para este tipo de campanhas, criados de uma forma programática (Figura 6.4) não sendo possível a receção de campanhas para além destes.

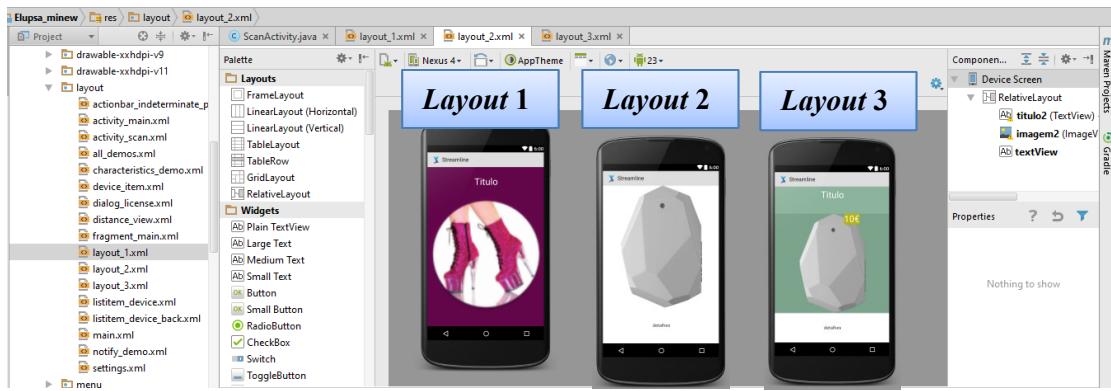


Figura 6.4 – Campanha Standard: Layouts

Assim, caso o tipo de campanha enviada pelo CMS seja do tipo *Standard*, o passo seguinte consiste em verificar o *layout* da campanha recebida carregando-o de acordo com a mesma. Posteriormente serão carregadas as respetivas informações para cada elemento (imagem e texto) do *layout*, efetuando uma leitura das informações enviadas no formato JSON pelo CMS.

No que diz respeito à imagem da campanha, esta é descarregada para a memória interna do dispositivo (caso ainda não exista), para uma pasta automaticamente criada pela aplicação, tal como apresentado no código seguinte.

```

if (layout_id.equals("1")) {
    setContentView(R.layout.layout_1);
    ImageView img = (ImageView) findViewById(R.id.imagem1);
    File imgFile = new File("/sdcard/streamline/" + imagem);
    if (imgFile.exists()) {
        Bitmap myBitmap = BitmapFactory.decodeFile(imgFile.getAbsolutePath());
        myBitmap=getRoundedShape(myBitmap);
        img.setImageBitmap(myBitmap);
    }
    TextView titulo_txt = (TextView) findViewById(R.id.titulo1);
    titulo_txt.setText(titulo);
    TextView valor_txt = (TextView) findViewById(R.id.valor);
    valor_txt.setText(valor);
    TextView detalhe_txt = (TextView) findViewById(R.id.detalhes1);
    detalhe_txt.setText(detalhe);
}

...

```

6.4.1.1. Abertura de campanha

Para que seja possível, no CMS, verificar se um cliente abriu a campanha, é inserido um link na imagem da campanha recebida, através do seguinte código.

```
img.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setAction(Intent.ACTION_VIEW);
        intent.addCategory(Intent.CATEGORY_BROWSABLE);
        intent.setData(Uri.parse(url_site));
        startActivity(intent);
    }
})
```

O URL inserido na imagem não corresponde ao URL inserido pelo lojista na campanha, mas sim num URL intermédio representado abaixo.

```
final String url_site = "http://beacons.streamline.pt/clientes_campañas/click_campaña/?id="+_tlm_base64;
```

Este irá enviar, através do método GET o ID da campanha e o respetivo número de telefone do dispositivo codificado em *base 64* (de forma a não ser legível caso o pacote seja capturado). Através destas informações, o CMS identifica o cliente, a campanha, regista a atividade e devolve ao dispositivo o URL correspondente ao inserido pelo lojista. Todo este processo, ilustrado na Figura 6.5, acontece de forma transparente para o utilizador.

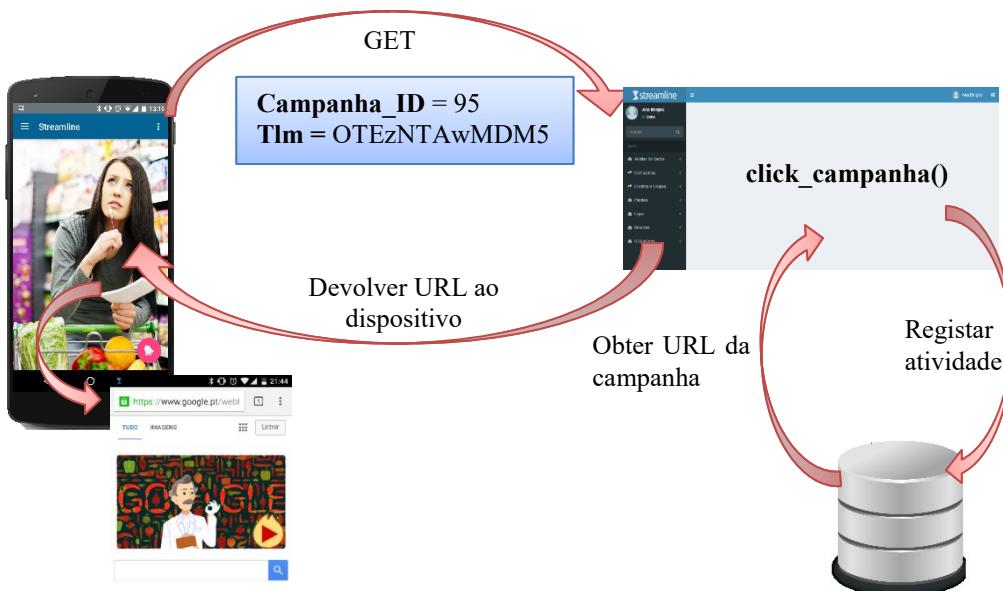


Figura 6.5 – Esquema de abertura de uma campanha

6.4.2. Tipo de campanha: Passbook

Caso seja devolvido pelo CMS que a campanha a receber na aplicação será do tipo *Passbook*, o próximo passo consiste em fazer o *download* do ficheiro *pkpass* para a memória interna do dispositivo.

6.4.2.1. Download do Passe

O processo de *download* do passe, é executado recorrendo a uma função no CMS de nome *download_passe* pertencente ao controlador *campanhas*, no qual permite devolver o ficheiro do passe (*.pkpass*) através da receção do ID da campanha. De modo a não interferir com a interface da aplicação, o *download* é executado em *background* sendo criada uma *thread* exclusiva para o efeito. Usando a função *DataInputStream* o passe é descarregado para uma pasta criada pela aplicação existente na memória interna do dispositivo. O código seguinte é responsável pela invocação da função no CMS e respetivo descarregamento do ficheiro.

```

final String aux_download_file_path =
    "http://beacons.streamline.pt/campanhas/download_passe/" + campanha_id;
final String aux_dest_file_path =
    "/sdcard/streamline/" + "passe " + campanha_id + ".pkpass";
```

URL do CMS

Localização e nome do passe

```

passbook(aux_download_file_path, aux_dest_file_path);

private void passbook(final String download_file_path,
                      final String dest_file_path) {

    dialog = ProgressDialog.show(ScanActivity.this, "",
        "Obtendo o seu passe...", true);
    new Thread(new Runnable() {
        public void run() {
            downloadFile(download_file_path, dest_file_path);
            dialog.dismiss();
        }
    }).start();
```

public void downloadFile(String url, String dest_file_path) {
 try {
 File dest_file = new File(dest_file_path);
 URL u = new URL(url);
 URLConnection conn = u.openConnection();
 int contentLength = conn.getContentLength();
 DataInputStream stream = new DataInputStream(u.openStream());
 byte[] buffer = new byte[contentLength];
 stream.readFully(buffer);
 stream.close();
 DataOutputStream fos = new DataOutputStream(new FileOutputStream(
 dest_file));
 fos.write(buffer);
 fos.flush();
 fos.close();
 hideProgressIndicator();
 ...
}

De forma ao nome do passe ser único e garantir que não existem dois passes respeitantes á mesma campanha, o nome deste é sempre do tipo *passe_id_campagna].pkpass*.

A Figura 6.6 mostra a interação entre o CMS e a aplicação móvel durante este processo.



Figura 6.6 – Esquema de download do ficheiro .pkpass

6.4.2.2. Abertura do Passe

Antes de iniciar o processo de abertura é executada uma rotina de forma a verificar que o processo de *download* foi finalizado com êxito. Caso se verifique, é executado o serviço *Runnable* de forma a abrir o passe numa nova *thread*, possibilitando à aplicação executar outras funções em paralelo durante a abertura.

```
new Runnable() {
    @Override
    public void run() {
        open_file(aux_dest_file_path);
    }
};
```

De modo a que seja possível abrir o passe é necessário que o utilizador possua a aplicação responsável por abrir este tipo de ficheiros, ou seja, a aplicação *PassWallet*. Durante a abertura da campanha é verificado se a aplicação se encontra instalada no dispositivo. Em caso negativo, o utilizador é redirecionado para a página de *download* da aplicação no *Google Play*. Caso a aplicação *PassWallet* se encontre instalada então é automaticamente aberta mostrando o passe previamente descarregado.

A Figura 6.7 apresenta, de forma esquemática, o comportamento da aplicação face à existência ou não da aplicação *PassWallet* no dispositivo.

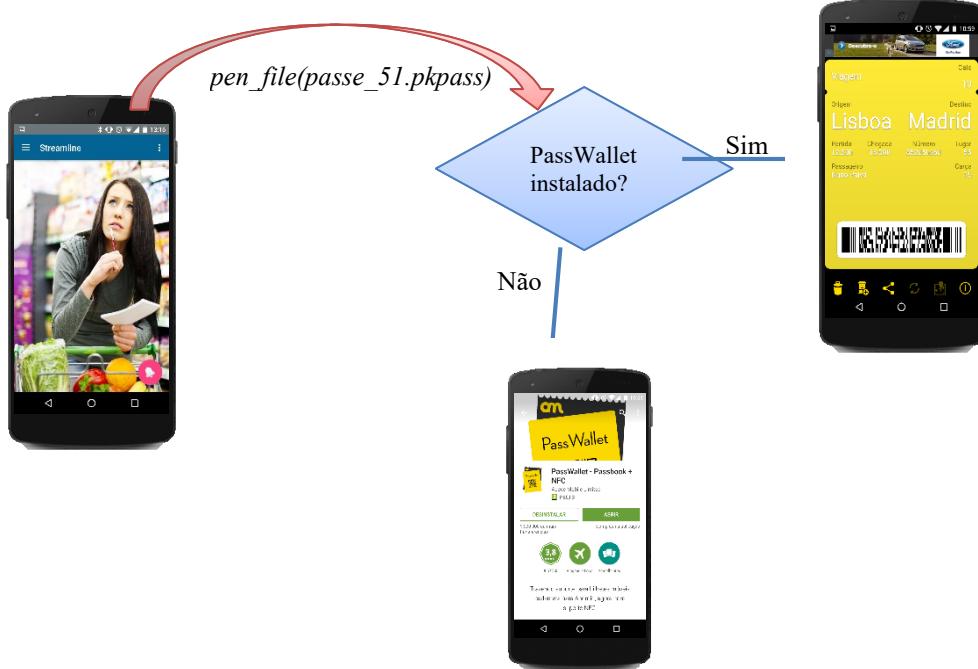


Figura 6.7 – Opções tendo em conta a instalação da aplicação *Passwallet*

6.4.3. Tipo de campanha: Web

Embora este tipo de campanhas não necessite de uma aplicação do lado do cliente para as receber, existe, no entanto, a obrigatoriedade de uma aplicação responsável por sincronizar os URL's difundidos por cada *beacon*.

Atualmente apenas o SDK do fabricante *Estimote* permite a alteração de parâmetros de configuração *Eddystone*, nomeadamente o campo URL.

Assim, recorrendo ao SDK deste fabricante, foi implementada a compatibilidade da tecnologia *Eddystone* na aplicação desenvolvida, conseguindo efetuar a deteção dos URL's emitidos por cada *beacon* bem como efetuar a sua alteração.

6.4.3.1. Abertura de campanhas Web

O processo de abertura de uma campanha Web consiste em abrir o *browser*, existente por defeito no *smartphone*, passando por argumento o URL detetado na aplicação que foi enviado pelo *beacon*. O código abaixo representa o descrito.

```

Eddystone nearestBeacon = eddystones.get(0);
String nearestBeacon_url = String.valueOf(nearestBeacon.url);
Intent browserIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(nearestBeacon_url));
startActivity(browserIntent);
  
```

6.4.3.2. Sincronização de Eddystone URL

Tal como mencionado no capítulo 5 (secção 5.4.3.9.2), após criação de uma campanha do tipo Web é necessário inserir o respetivo *short URL* no *beacon*, de modo a que este o divulgue.

Assim, na aplicação móvel desenvolvida foi criado um módulo responsável por efetuar o sincronismo dos URL's detetados de cada *beacon*. Este módulo inicia-se com a execução de uma *AsyncTask* responsável por enviar para o CMS, via POST, os dados de identificação do *beacon* com o qual contactou (neste caso: *mac address* e URL transmitido pelo *beacon*). O CMS irá verificar se o URL que possui na DB associado ao respetivo *beacon* é igual ao recebido pela aplicação, devolvendo a FLAG *_synchronized_cms* = *FALSE* em caso negativo e *_synchronized_cms* = *TRUE* em caso afirmativo.

Após receção dos dados do CMS e caso a FLAG *_synchronized_cms* possua o valor *FALSE*, então a aplicação irá proceder ao envio do URL, devolvido pelo CMS, para o *beacon*.

Tal como indicado no capítulo 3 (secção 3.3.8) os dados transmitidos pelo *beacon* são configuráveis exclusivamente através do seu *firmware*. Todas as alterações a configurações do *beacon* carecem de uma prévia autenticação na API do fabricante. Assim o primeiro passo consiste em criar uma ligação ao *beacon* recorrendo às credenciais previamente definidas na API (visível no código abaixo). Esta autenticação garante que nenhuma entidade consegue alterar os dados dos *beacons* sem autorização.

```
EstimoteSDK.initialize(getApplicationContext(), "USER", "PASSWORD");
connection = new BeaconConnection(MainActivity.this, beacon_mac_address,
    createConnectionCallback());
...
private BeaconConnection.ConnectionCallback createConnectionCallback() {
    return new BeaconConnection.ConnectionCallback() {
        @Override public void onConnected(final BeaconInfo beaconInfo) {
            runOnUiThread(new Runnable() {
                @Override public void run() {
                    Log.e("Status", "Connected to eddystone");
                    updateEddystone(eddystone_sync_url);
                }
            });
        }
    };
}
...
```

Caso a autenticação na API seja bem-sucedida e consiga ser feita uma ligação ao respetivo *beacon*, então irá ser executada a função *updateEddystone* passando por argumento o novo URL a transmitir.

Através do método *edit()* pertencente à class *BeaconConnection* e respetivo *commit* é possível enviar o novo URL para o *beacon*, tal como descrito no código abaixo.

```
private void updateEddystone(String new_url) {
    connection.edit()
        .set(true ? connection.eddystoneUrl() : connection.eddystoneNamespace(), new_url)
        .commit(new BeaconConnection.WriteCallback() {
            @Override public void onSuccess() {
                runOnUiThread(new Runnable() {
                    @Override public void run() {
                        Log.e("Success", "URL updated!");
                    }
                });
            }
        });
    ...
}
```

Uma vez que o processo de sincronismo dos URL's de cada *beacon* necessita de ser efetuado no raio de alcance do seu sinal, através da implementação deste módulo é possível que a sincronização seja feita de forma automática e transparente pelos clientes ao se aproximarem de cada *beacon*. Evita-se assim, por exemplo, que seja o gestor de loja a percorrer toda a superfície (entrando em contacto com cada *beacon*). Alargando a solução a uma cadeia de lojas existente em diversos pontos do país, através desta solução o sincronismo de todos os *beacons* será feito automaticamente, pelos clientes, após a criação de uma campanha no CMS.

7. CONCLUSÕES E TRABALHO FUTURO

O aumento exponencial da IoT, o uso cada vez maior de equipamentos *smart*, e atualmente a criação da tecnologia *iBeacon* permitiu abrir um novo capítulo no que diz respeito à interação entre equipamentos e pessoas.

Desenvolvida tendo em vista a área do retalho, a solução apresentada distingue-se de outras semelhantes devido ao facto de fornecer um módulo de análise de dados responsável por apresentar ao *Gestor* um conjunto de informações que lhe permitem otimizar o seu negócio, possibilitando não só prevenir perdas, como melhorar a experiência de compra do cliente. Diferencia-se ainda por ser compatível com qualquer tipo de *beacons*, independentemente do fabricante, integrar o sistema *Passbook*, assim como a recente tecnologia *Eddystone*. Através desta solução é possível efetuar a ponte entre o meio físico e o digital, diminuindo a latência de comunicação entre a loja e o cliente.

Esta solução destina-se a ser utilizada de forma simultânea em diversas cadeias de hipermercados independentes entre si. Como trabalho futuro, surge o desenvolvimento de requisitos adicionais na aplicação móvel, tendo em vista a utilização num contexto real na área de retalho.

Dependendo dos requisitos estabelecidos pela loja onde a solução for implementada, o desenvolvimento da uma aplicação para o sistema iOS prevê-se necessário estando já o seu plano de desenvolvimento em evolução.

Com o aumento de número de dados na plataforma, torna-se importante a integração com um sistema de *data mining* de modo a efetuar o processamento detalhado dos dados e respetiva extração de conhecimento. Algoritmos como *K-means*, *C4.5*, *kNN* ou *Naive Bayes* são fundamentais na criação e segmentação de perfis de clientes.

De forma a automatizar o processo de registo de clientes no CMS torna-se importante, para cada situação, a integração com *softwares* (por exemplo, de faturação) onde já existam dados de todos os clientes.

O facto da tecnologia *iBeacon* ter surgido recentemente, a falta de maturidade dos meios de comunicação com os *beacons*, a sua constante evolução, e ultimamente o surgimento da tecnologia *Eddystone*, levou a uma árdua tarefa de desenvolvimento, obrigando a uma constante adaptação, de forma a manter a compatibilidade com quaisquer tecnologias equivalentes que surjam na área.

Devido à estratégia de desenvolvimento da solução, esta permite ser facilmente adaptável a outras áreas de atuação. Desta forma, encontram-se a ser estabelecidos procedimentos de modo a adaptar a solução à área dos transportes públicos tendo em vista a sua implementação nos transportes da cidade de Coimbra. Pretende-se também implementar uma versão adaptada da solução, recorrendo à tecnologia *Eddystone*, numa feira que irá decorrer nas Caldas da Rainha.

REFERÊNCIAS BIBLIOGRÁFICAS

- [1] “Instituto Superior de Engenharia de Coimbra,” 2016. [Online]. Disponível: <http://www.isec.pt/>. [Acedido em 16 Abril 2016].
- [2] “Instituto Politécnico de Coimbra,” 2016. [Online]. Disponível: <http://portal.ipc.pt/portal>. [Acedido em 16 Abril 2016].
- [3] “Streamline,” 2015. [Online]. Disponível: <http://www.streamline.pt>. [Acedido em 16 Abril 2016].
- [4] E. Gomes, “Retalho: uma nova realidade,” 2014. [Online]. Disponível: <http://www.plotcontent.com/retalho-nova-realidade/>. [Acedido em 16 Abril 2016].
- [5] S. Mark e N. Paul, “Accenture: Understanding the Changing Consumer,” 2015. [Online]. Disponível: https://www.accenture.com/t20151028T034558_w_fi-en_acnmedia/Accenture/Conversion-Assets/DotCom/Documents/Global/PDF/Dualpub_21/Accenture-Understanding-the-Changing-Consumer.pdf. [Acedido em 16 Abril 2016].
- [6] B. Walker, “Forbes: Retail In Crisis,” 2014. [Online]. Disponível: <http://www.forbes.com/sites/jeremybogaisky/2014/02/12/retail-in-crisis-these-are-the-changes-brick-and-mortar-stores-must-make>. [Acedido em 16 Abril 2016].
- [7] B. Galipeau, “Business Insider: Nordstrom,” 2013. [Online]. Disponível: <http://www.businessinsider.com/nordstroms-pinterest-in-stores-plan-2013-11>. [Acedido em 16 Abril 2016].
- [8] S. Ramaswamy, “Shopping Then and Now,” Junho 2013. [Online]. Disponível: <https://www.thinkwithgoogle.com/articles/five-ways-retail-has-changed-and-how-businesses-can-adapt.html>. [Acedido em 16 Abril 2016].
- [9] “Licenciatura em STI - Universidade Atlântica,” Dezembro 2011. [Online]. Disponível: <http://ssti1-1112.wikidot.com/a-internet-das-coisas>. [Acedido em 22 Dezembro 2015].
- [10] J. Pontin, “MIT Technology Review,” 2005. [Online]. Disponível: <https://www.technologyreview.com/s/404694/etc-bill-joys-six-webs/>. [Acedido em 22 Novembro 2015].
- [11] K. Ashton, “That 'Internet of Things' Thing,” *RFID Journal*, 1999.
- [12] L. Atzori, A. Iera e G. Morabito, “The Internet of Things: A survey,” *Computer Networks*, pp. 1-3, 2010.
- [13] C. Valente, “Capital Intelectual,” 2013, pp. 171-174.

- [14] D. Evans, “Como a próxima evolução da Internet está mudando tudo,” *Cisco IBSG*, pp. 2-4, 2011.
- [15] J. Gantz e D. Reinsel, “THE DIGITAL UNIVERSE IN 2020,” *IDC View*, nº Big Data, Bigger Digital Shadows, and Biggest Growth in the Far East, pp. 1-4, 2012.
- [16] N. Newman, “Apple iBeacon technology briefing,” *Journal of Direct*, pp. 1-4, 2014.
- [17] A. Fujihara e T. Yanagizawa, “Proposing an extended iBeacon system for indoor route guidance,” em *International Conference on Intelligent Networking and Collaborative Systems*, 2015.
- [18] C. F. Hughes, “Bluetooth Low Energy for Use with MEM Sensors,” Novembro 2015. [Online]. Disponível:
https://repository.asu.edu/attachments/163965/content/Hughes_asu_0010N_15636.pdf. [Acedido em 3 Fevereiro 2016].
- [19] R. Faragher e R. Harle, “Location Fingerprinting with Bluetooth Low Energy,” *IEEE Journal on Selected Areas in Communications*, pp. 1-3, 2015.
- [20] M. Siekkinen, M. Hiienkari, J. K. Nurminen e J. Nieminen, “How Low Energy is Bluetooth Low Energy?,” em *Workshop on Internet of Things Enabling Technologies*, 2012.
- [21] F. Martelli, “Wireless Sensor Networks: BLUETOOTH LOW ENERGY,” [Online]. Disponível: http://www.deyisupport.com/cfs-file.ashx/_key/CommunityServer-Components-PostAttachments/00-00-03-93-04/handouts_5F00_WSNs_5F00_BT_2D00_LE.pdf. [Acedido em 5 Fevereiro 2016].
- [22] Wireshark, “Wireshark,” [Online]. Disponível: <https://wiki.wireshark.org/Bluetooth>. [Acedido em 6 Fevereiro 2016].
- [23] A. Inc., “Xcode,” [Online]. Disponível: <https://developer.apple.com/xcode/>. [Acedido em 5 Fevereiro 2016].
- [24] M. S. Gast, Building Applications with iBeacon, Estados Unidos da América: O'Reilly Media, Inc., 2015, pp. 13-18.
- [25] A. Inc., “Getting Started with iBeacon,” Junho 2014. [Online]. Disponível:
<https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>. [Acedido em 7 Fevereiro 2016].
- [26] M. Köhne e J. Sieck, “Location-based Services with iBeacon Technology,” em *Second International Conference on Artificial Intelligence, Modelling and Simulation*, Alemanha, 2014.

- [27] aislelabs, “The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs,” Maio 2015. [Online]. Disponível: <http://www.aislelabs.com/reports/beacon-guide/>. [Acedido em 22 Fevereiro 2016].
- [28] aislelabs, “iBeacon and Battery Drain on Phones: A Technical Report,” Julho 2014. [Online]. Disponível: <http://www.aislelabs.com/reports/ibeacon-battery-phones/>. [Acedido em 25 Fevereiro 2016].
- [29] aislelabs, “iBeacon Battery Drain on Apple vs Android: A Technical Report,” Agosto 2014. [Online]. Disponível: <http://www.aislelabs.com/reports/ibeacon-battery-drain-iphones/>. [Acedido em 26 Fevereiro 2016].
- [30] C. Martin, “Harvard Business Review: How Beacons Are Changing the Shopping Experience,” Setembro 2014. [Online]. Disponível: <https://hbr.org/2014/09/how-beacons-are-changing-the-shopping-experience/>. [Acedido em 1 Março 2016].
- [31] P. Chaney, “Beacons: Local Retail Game-changing Technology?,” Setembro 2015. [Online]. Disponível: <http://webmarketingtoday.com/articles/118077-Beacons-Local-Retail-Game-changing-Technology/>. [Acedido em 7 Março 2016].
- [32] R. Johnson, “Harvard Business Review: Retail Isn’t Broken. Stores Are,” Dezembro 2011. [Online]. Disponível: <https://hbr.org/2011/12/retail-isnt-broken-stores-are>. [Acedido em 8 Março 2016].
- [33] D. Girish, “How Beacons can make the Mall Experience more Delightful,” Agosto 2014. [Online]. Disponível: <http://blog.beaconstac.com/2014/08/how-beacons-can-make-the-mall-experience-more-delightful/>. [Acedido em 10 Março 2016].
- [34] “McDonald’s Customers Treated to a Better Dining Experience with Piper Beacon Technology,” Dezembro 2014. [Online]. Disponível: <http://www.businesswire.com/news/home/20141218005195/en/McDonald%20%80%99s-Customers-Treated-Dining-Experience-Piper-Beacon#.VJMpXcAAA>. [Acedido em 14 Março 2016].
- [35] K. Taylor, “McDonald’s Boosts McNuggets Sales With iBeacon Test,” Dezembro 2014. [Online]. Disponível: <https://www.entrepreneur.com/article/241145>. [Acedido em 18 Março 2016].
- [36] “LONDON’S REGENT STREET ADOPTS IBEACON,” Junho 2014. [Online]. Disponível: <http://www.ibeacon.com/londons-regent-street-adopts-ibeacon/>. [Acedido em 16 Março 2016].
- [37] J. Purcher, “Apple’s iBeacon Wins Big in ANKAmall in Turkey Thanks to Boni,” Abril 2014. [Online]. Disponível: <http://www.patentlyapple.com/patently-apple/2014/04/apples-ibeacon-wins-big-in-ankamall-in-turkey-thanks-to-boni.html>. [Acedido em 17 Março 2016].

- [38] M. Panzarino, “inMarket Rolls Out iBeacons To 200 Safeway,” Janeiro 2014. [Online]. Disponível: <http://techcrunch.com/2014/01/06/inmarket-rolls-out-ibeacons-to-200-safeway-giant-eagle-grocery-stores-to-reach-shoppers-when-it-matters/>. [Acedido em 20 Março 2016].
- [39] L. Johnson, “Hillshire Brands Sees 20% Jump in Purchase Intent With Beacons,” Julho 2014. [Online]. Disponível: <http://www.adweek.com/news/technology/hillshire-brands-sees-20-jump-purchase-intent-beacons-159042>. [Acedido em 20 Março 2016].
- [40] H. Milnes, “Shopkick's iBeacons Brought in \$1B in Sales for Its Retail Partners,” Outubro 2014. [Online]. Disponível: <http://bostinno.streetwise.co/2014/10/15/report-shopkicks-ibeacons-drov-1b-in-revenue/>. [Acedido em 22 Março 2016].
- [41] A. Souppouris, “Apple's iBeacon location-aware shopping goes live today,” Dezembro 2013. [Online]. Disponível: <http://www.theverge.com/2013/12/6/5181302/apple-store-ibeacon-rollout>. [Acedido em 28 Março 2016].
- [42] P. Burzacca, M. Mircoli, S. Mitolo e A. Polzonetti, “iBeacon technology that will make possible Internet of Things,” em *Internation Conference on Software Intelligence Technologies and Applications*, Itália, 2014.
- [43] “What Is The Difference Between iBeacon and NFC ?,” Janeiro 2014. [Online]. Disponível: https://rapidnfc.com/blog/100/what_is_difference_between_ibeacon_and_nfc. [Acedido em 3 Abril 2016].
- [44] D. Thompson, “iBeacon: Is Bluetooth On?,” Março 2014. [Online]. Disponível: <http://beekn.net/2014/03/ibeacon-bluetooth-insights-empatika/>. [Acedido em 22 Março 2016].
- [45] C. Gallen, “ABI Research Predicts Wireless Connectivity Gap to Widen as Bluetooth Enabled Device Shipments Reach 19 Billion over the Next Five Years,” Dezembro 2015. [Online]. Disponível: <http://www.prnewswire.com/news-releases/abi-research-predicts-wireless-connectivity-gap-to-widen-as-bluetooth-enabled-device-shipments-reach-19-billion-over-the-next-five-years-300193828.html>. [Acedido em 28 Março 2016].
- [46] Google, “Eddystone,” Junho 2015. [Online]. Disponível: <https://github.com/google/eddystone>. [Acedido em 29 Março 2016].
- [47] mubaloo, “Beacons Technical Overview,” *Beacons Technical White Paper*, pp. 2-3, 2015.
- [48] “W3Schools,” 2016. [Online]. Disponível: <http://www.w3schools.com/html/>. [Acedido em 30 Março 2016].
- [49] Y. Fan, “Cascading Style Sheets,” 2010.

- [50] “Bootstrap,” 2016. [Online]. Disponível: <http://bootstrapdocs.com/v2.0.2/docs/>. [Acedido em 30 Março 2016].
- [51] P. Neves, N. Paiva e J. Durães, “A comparison between JAVA and PHP,” em *C3S2E*, 2013.
- [52] J. Niederauer, Desenvolvendo Websites com PHP, Brasil: Novatec Editora Lda, 2009, pp. 23-26.
- [53] P. Neves e R. Ruas, O guia prático do MySQL, 1º ed., Famalicão: Centro Atlântico, Lda, 2005, pp. 22-25.
- [54] J. Garrett, “Ajax: A New Approach to Web Applications,” pp. 1-3, Fevereiro 2005.
- [55] “Scrum,” 2016. [Online]. Disponível: <https://www.scrumalliance.org/why-scrum>. [Acedido em 1 Abril 2016].
- [56] “GIT,” 2005. [Online]. Disponível: <http://gitref.org/>. [Acedido em 2 Abril 2016].
- [57] Apple, “Passbook,” [Online]. Disponível:
<https://developer.apple.com/passbook/Getting Started with Passbook.pdf>. [Acedido em 22 Abril 2016].
- [58] “AltBeacon,” [Online]. Disponível: <http://altbeacon.org/>. [Acedido em 22 Abril 2016].
- [59] T. Ming, From “Where I am” to “Here I am”: Accuracy study on location-based services with iBeacon, vol. 22, Hong Kong: HKIE Transactions, 2015, pp. 23-31.
- [60] “iBeacon in Hotels?,” Maio 2014. [Online]. Disponível:
<http://www.ibeacon.com/ibeacon-in-hotels/>. [Acedido em 4 Abril 2016].
- [61] “Bar de Toronto cria programa de recompensa via app iBeacon,” Setembro 2014. [Online]. Disponível: <https://ibeaconmania.wordpress.com/2014/09/30/bar-de-toronto-cria-programa-de-recompensa-via-app-ibeacon/>. [Acedido em 5 Abril 2016].
- [62] N. Mallik, “Museums using Beacons to Enhance Interactivity,” Fevereiro 2015. [Online]. Disponível: <http://blog.beaconstac.com/2015/02/3-museums-using-beacons-to-enhance-interactivity/>. [Acedido em 6 Abril 2016].
- [63] P. Perez, “Schools creating perfect interactive experiences using beacons,” [Online]. Disponível: <http://www.securedgenetworks.com/blog/3-schools-creating-perfect-interactive-experiences-using-beacons>. [Acedido em 9 Abril 2016].
- [64] J. Kaye, “BLE and Beacon Technology: Great Potential in the Medical Market,” Fevereiro 2014. [Online]. Disponível:
<http://www.techbriefs.com/component/content/article/mdb/features/19167>. [Acedido em 10 Abril 2016].

- [65] C. Swedberg, “Dutch Hospital Uses Beacons to Track Treatment for Cardiac Patients,” Setembro 2015. [Online]. Disponível: <http://www.rfidjournal.com/articles/view?13503>. [Acedido em 12 Abril 2016].
- [66] R. Ghee, “American Airlines undertakes industry’s biggest deployment of iBeacons at DFW Airport,” Junho 2014. [Online]. Disponível: <http://www.futuretravelexperience.com/2014/06/american-airlines-undertakes-industrys-biggest-deployment-ibeacons-dfw-airport/>. [Acedido em 11 Abril 2016].
- [67] R. Ghee, “easyJet’s multi-airport iBeacon trial lays the foundations for a major rollout,” Julho 2014. [Online]. Disponível: <http://www.futuretravelexperience.com/2014/07/easyjets-multi-airport-ibeacon-trial-lays-foundations-major-rollout/>. [Acedido em 14 Abril 2016].
- [68] “SITA Lab,” Março 2014. [Online]. Disponível: <http://www.futuretravelexperience.com/2014/03/sita-lab-beacon-technology-can-personalise-airport-experience-common-use-approach-needed/>. [Acedido em 11 Abril 2016].
- [69] “Fujitsu World Tour,” Julho 2014. [Online]. Disponível: <http://blog.mylocalsocial.com/post/91449477489/ibeacons-conferences-and-fujitsu-world-tour>. [Acedido em 14 Abril 2016].
- [70] Devina, “Silvercar Enhances Its Services With iBeacon,” Outubro 2014. [Online]. Disponível: <http://blog.cubeacon.com/silvercar-enhances-its-services-with-ibeacon.html>. [Acedido em 10 Abril 2016].
- [71] Simonato, “Beacons: a procura de estatísticas que justifiquem seu uso,” Agosto 2014. [Online]. Disponível: <https://ibeaconmania.wordpress.com/2014/08/29/beacons-a-procura-de-estatisticas-que-justifiquem-seu-uso/>. [Acedido em 15 Abril 2016].
- [72] C. Velazco, “MLB’s iBeacon Experiment May Signal A Whole New Ball Game For Location Tracking,” Setembro 2013. [Online]. Disponível: <http://techcrunch.com/2013/09/29/mlbs-ibeacon-experiment-may-signal-a-whole-new-ball-game-for-location-tracking/>. [Acedido em 11 Abril 2016].
- [73] N. Lomas, “Can iBeacons Be Used To Help The Visually Impaired Navigate Public Transport,” Agosto 2014. [Online]. Disponível: <http://techcrunch.com/2014/08/06/ibeacon-navigation/>. [Acedido em 16 Abril 2016].
- [74] R. Fera, “Nivea's Protection ad,” Junho 2014. [Online]. Disponível: <http://www.fastcocreate.com/3032008/cannes/niveas-protection-ad-wins-mobile-grand-prix-at-cannes>. [Acedido em 11 Abril 2016].

- [75] B. SIG, “GATT,” 2016. [Online]. Disponível:
<https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx>. [Acedido em 19 Fevereiro 2016].

APÊNDICES

Apêndice I – Outras áreas de atuação e casos de uso da tecnologia *iBeacon*

Apêndice II – Requisitos do CMS

Apêndice III – Requisitos da aplicação *Android*

Apêndice IV – Diagrama de atividades da aplicação *Android*

Apêndice V – Diagrama da base de dados

Apêndice I: Outras áreas de atuação e casos de uso da tecnologia iBeacon

Esta secção apresenta outras áreas de atuação, para além do retalho, que a tecnologia *iBeacon* permite abranger. Para cada área de atuação será exposto de que forma a poderá melhorar, apresentando-se casos de estudo práticos e implementações de soluções realizadas em diversos países.

Área de atuação: Hotelaria

Mostram-se de seguida algumas ações que podem ser desencadeadas nesta área:

- Possibilidade de enviar incentivos aos clientes, por exemplo, “Pratique minigolfe às 15h e receberá 10% de desconto na próxima noite”;
- No caso de um hotel, aquando do *check-in* do cliente poderá ser feito um guia de apresentação do hotel (apenas a primeira vez que o cliente passar nos corredores) de forma ao cliente se dirigir a todos os espaços, possibilitando ao gestor direcionar os clientes e acrescentar valor a áreas com melhor afluência de clientes;
- Efetuar uma surpresa ao cliente. Através da deteção do cliente nas proximidades do hotel pode ser preparada uma receção especial, ou mesmo dirigir-se ao local onde o cliente se encontra, surpreendendo-o, por exemplo, com as chaves do quarto;
- Abrir portas automaticamente quando o cliente se encontra por perto;
- Enquanto o cliente espera pelo seu pedido enviar um jornal, para que o cliente fique ocupado enquanto espera;
- Baseado nas definições de língua configuradas no dispositivo, verificar qual a língua nativa do cliente possibilitando preparar atempadamente a comunicação com este.

Casos de estudo

Nesta secção serão apresentados diversos casos de estudo para a área de hotelaria.

James Hotels

James Hotels localizado nos EUA, nomeadamente em Chicago, Miami e New York encontra-se a fazer uso da tecnologia *iBeacon* possibilitando aos seus clientes receber sugestões de jantar, atividades e compras baseadas na sua localização, *check-in* e *check-out* através da aplicação, prolongar a estadia, abrir a porta do quarto quando se encontra perto (até 1 metro) [60], ou mesmo proporcionar ao cliente um guia pelas áreas do hotel.

Bar Hop

O *Bar Hop* situado na *King Street West* em Toronto, implementou em outubro de 2014 a tecnologia *iBeacon* na sua aplicação possibilitando “verificar o perfil dos clientes, quais os mais frequentes e efetuar-lhes recompensas pela sua lealdade” [61].

Área de atuação: Turismo e atrações turísticas

Ações que a tecnologia *iBeacon* pode permitir nesta área:

- Proporcionar um guia turístico ao cliente;
- Encontrar pontos de interesse ou receber mapas à medida que um visitante caminha;
- Permitir aos visitantes ler e escrever comentários, acerca de uma exposição, que serão recebidos por outros visitantes que se encontrem igualmente a ver, ou venham num futuro a faze-lo;
- Mostrar ao visitante quanto tempo falta para finalizar a visita;
- Nas praias, mostrar informação do tempo, locais de interesse tais como: polícia, nadador salvador, hospital mais próximo, entre outros.

Casos de estudo

Nesta seção serão apresentados casos de estudo no que diz respeito área de turismo.

National Slate Museum, Wales

O museu *National Slate* existente na Snowdonia (País de Gales) foi o primeiro museu a integrar a tecnologia *iBeacon*, em 2015. A aplicação do museu inclui patrocínio digital e é apresentado aos visitantes à medida que estes caminham pelo museu, sendo cada conteúdo mostrado como suporte ao que se encontra a visualizar em cada local [62].

Brooklyn Museum

O museu *Brooklyn* existente em New York adotou a tecnologia *iBeacon* tendo em vista satisfazer as dúvidas dos seus visitantes. Através da aplicação, um visitante recebe informação consoante o seu local, podendo efetuar perguntas relativas ao local onde se encontra a serem respondidas por visitantes ou empregados [62].

Área de atuação: Educação

Alguns exemplos possíveis na área de educação:

- Útil para os pais. Por exemplo, verificar se os filhos chegaram atrasados à sala de aula;
- Melhorar a experiência entre o professor e o aluno tornando as aulas mais interativas podendo fornecer aos alunos, que se encontram na sala, conteúdos digitais complementares.

Casos de estudo

Nesta seção serão apresentados diversos casos de estudo para a área de educação.

Clevedon School

A *Clevedon School* existente no Reino Unido integrou em 2015 a tecnologia *iBeacon* na sua aplicação, de modo a melhorar o uso dos 1200 *iPads* distribuídos por todos os alunos. Foi criado um *Beacon Management Interface* onde os professores colocam conteúdo das aulas, sendo distribuído por determinado período de tempo e num determinado local. Desta forma os professores conseguem fornecer, por exemplo, material teórico quando um aluno entra na sala de aula e fornecer matéria de estudo quando se desloca à sala de estudo [63].

Dulwich College

A *Dulwich College* em Singapura foi a primeira escola a adotar a tecnologia *iBeacon* na Ásia possibilitando a matrícula na escola, guia turístico de todo o campus (com informações de cada local à medida que é visitado) e distribuir conteúdo de estudo em tempo real aos seus alunos consoante o seu local [63].

Área de atuação: Cuidados de saúde

Jonathan Kaye, no artigo *BLE and Beacon Technology: Great Potential in the Medical Market* [64] explica de que forma a tecnologia BLE pode melhorar não só a experiência no atendimento do paciente nos hospitais, como também todo o seu funcionamento.

Possíveis melhorias que a tecnologia *iBeacon* pode trazer:

- Quando um cliente entra no hospital poderá receber um mapa de forma a saber onde se dirigir;
- Na sala de espera, o paciente pode receber informações como: tempo médio de espera; jornais; conteúdos úteis relacionados com o local, ou área da consulta;
- Quando um cliente sai do hospital poderá receber informações como: local mais próximo de compra de medicamentos; cupões de desconto resultante de parcerias entre o hospital e determinada indústria farmacêutica; notas da consulta enviadas pelo médico, bem como a receita médica.

Caso de estudo

O *Dutch Hospital Shopping Precinct* situado no Sri Lanka (India), nomeadamente o departamento de cardiologia, iniciou em setembro de 2015 o uso da tecnologia *iBeacon* de forma a monitorizar o local frequentado pelos seus pacientes [65]. Aliado à tecnologia *Radio Frequency Identification (RFID)*⁴ e através da solução *Zebra Technologies* a tecnologia *iBeacon* permite fornecer aos médicos em tempo real informação do seu tratamento e despoletar alertas caso haja alguma inconformidade.

Área de atuação: Viagem

Apresentam-se alguns exemplos de aplicação prática para esta área:

- Num aeroporto, informar o passageiro sobre: em que local deve efetuar o *check-in*; qual o local de embarque; dados relevantes do voo, tais como, atrasos e cancelamentos; estado da bagagem, entre outros;
- Uma agência de viagens, com base no histórico do cliente poderá enviar campanhas / promoções direcionadas quando este frequentar um aeroporto;
- Na estação / paragem de comboios, metro ou autocarros, os passageiros podem receber notificações de atrasos, cancelamentos, acidentes, ou outras informações relevantes.

Casos de estudo

Seguidamente serão apresentados casos de estudo onde a tecnologia *iBeacon* foi aplicada nesta área.

American Airlines

A *American Airlines* foi a primeira companhia aérea a implementar a tecnologia *iBeacon* [66]. O projeto implementado no aeroporto internacional de Dallas, com duração de 6 meses destinou-se inicialmente a ajudar os passageiros a encontrar o seu caminho ao longo do terminal, bem como a acompanharem em tempo real o estado do voo, e a sua bagagem.

Segundo Phil Easter, diretor de aplicações moveis da *American Airlines*, 65% dos passageiros chegam demasiado cedo ao aeroporto por falta de informação do estado do voo, afluência de passageiros e serem desconhecedores do percurso a efetuar [66]. Easter indica ainda que quando os passageiros estão sentados, através da tecnologia *iBeacon*, podem ser informados se se encontram na porta certa, se o seu voo está atrasado, ou receber informações

⁴ RFID – É uma tecnologia de identificação automática que funciona através de sinais de rádio, permitindo recuperar e armazenar dados remotamente através de dispositivos denominados etiquetas RFID.

relevantes para o seu voo de acordo com o seu local. Todas estas informações irão melhorar a experiência de voo bem como reduzir o *stress* no passageiro.

EasyJet

A *EasyJet* adotou a tecnologia *iBeacon* em julho 2014, colocando um projeto em testes nos aeroportos *Luton London*, *London Gatwick* e *Paris Charles*. James Millett, indica que o projeto se destina a abranger todos os aeroportos da companhia aérea dado o sucesso ocorrente da experiência. Millett afirma que a aplicação *easyJet* foi descarregada 9 milhões de vezes tendo um aumento significativo com a introdução da nova tecnologia [67].

A experiência introduzida pela *easyJet* teve em conta o relatório [68] desenvolvido pela SITA Lab, no qual salienta as potencialidades da tecnologia *iBeacon* nos aeroportos, cuja solução que as incluem se encontra em desenvolvimento, nomeadamente:

- Localização dos passageiros: com base na localização do dispositivo e da comunicação com cada *beacon* é possível detetar a localização do passageiro no aeroporto e com isso informa-lo, por exemplo, que se encontra atrasado;
- Envio de cartões de embarque: através do uso da aplicação *Passbook* é possível visualizar cartões de embarque digitais recebidos quando entra em contacto com um *beacon* e assim ser apresentado no momento de embarque;
- Recuperação da bagagem: quando um passageiro chega a um local, ao entrar em contato com um *beacon* pode receber uma notificação que a sua bagagem está a chegar e quanto tempo terá de esperar.

Área de atuação: Conferências

Através da tecnologia *iBeacon* é possível fornecer ao participante de uma conferência dados como: diplomas de participação no final da conferência; informações contextuais distintas em cada apresentação, por exemplo, enviar notas de apresentação para os participantes no final da exposição; informações sobre a audiência presente na sala, entre outros.

Caso de estudo

A conferência realizada em Julho 2014 pela Fujitsu na Irlanda, em parceria com a LocalSocial, teve como principal premissa fornecer aos participantes uma forma de ligação entre a camada física e a camada digital.

A tecnologia responsável por efetuar essa ligação e que foi usada em toda a conferência foi a tecnologia *iBeacon*, tendo sido colocados dezenas de *beacons* em todo espaço. Através do uso de uma aplicação e de um CMS os participantes obtiveram informações como: dados complementares do orador em cada apresentação; detalhes adicionais de produtos que se encontravam “à sua frente”; cartão de boas vindas; dados auxiliares de cada apresentação [69].

Área de atuação: Automotivo

A *Silvercar* consiste numa empresa de aluguer de automóveis cuja implementação da solução *iBeacon* foi efetuada em Austin, Dallas, Denver, Los Angeles, Miami e San Francisco funcionando apenas com um único modelo de carro, o Audi A4. Embora seja um projeto de testes, este permite, após descarregar a aplicação, destrancar o carro e enviar dados do condutor para a *Silvercar*, tais como, consumo médio, percurso efetuado (através de GPS), estação de radio preferida e usar posteriormente esses dados para análise do perfil do cliente [70].

Área de atuação: Eventos

Apresentam-se de seguida eventos onde a tecnologia *iBeacon* já foi utilizada

Bonnaroo Festival

No festival de música dos EUA, ocorrido em Junho 2014 foram instalados 100 *beacons* resultando no envio de mais de 97.000 notificações para os utilizadores. Cada utilizador interagiu em média 102 minutos com os conteúdos enviados via *iBeacon*, aceitando 20% das notificações *push* das 12.6 notificações que recebeu em média.

Verificou-se ainda o espaço de ocupação em determinadas áreas do evento, como pistas, áreas VIP e setores de alimentação, obtendo-se, por exemplo, que a pista “What” foi a mais popular com cerca de 1980 utilizadores e uma média de tempo de permanência de 102 minutos por utilizador [71].

Major League Baseball

Em 2015 a liga de beisebol nos EUA implementou em 20 dos 30 estádios centenas de *beacons*. Cada equipa efetua a gestão dos *beacons* instalados no seu estádio possibilitando aos adeptos receber mapas do estádio, vídeos, informações de capacidade do estádio, desconto em bilhetes, entre outros [72].

Área de atuação: Melhorar a experiência do utilizador

A *Royal London Society* desenvolveu uma solução que permite a pessoas com incapacidade visual serem guiadas pelo metro de Londres com recurso à tecnologia *iBeacon*. Para tal, desenvolveu a aplicação *Wayfindr* que através da comunicação com dezenas de *beacons* distribuídos pelo metro, permite guiar o utilizador desde a entrada até ao local desejado, recorrendo a mensagens de voz [73].

Área de atuação: Uso pessoal

Nesta área podem ser desencadeadas notificações que poderão ajudar as tarefas diárias de cada utilizador, nomeadamente:

- Receber uma notificação quando se encontra próximo do lixo indicando que o deve levar num determinado dia;
- Através da implantação de um *beacon* no automóvel é possível “descobrir” onde o estacionou (numa distância máxima de 70 metros);
- Desligar / ligar luzes quando sai / entra em casa;
- Detetar possíveis roubos. Por exemplo, ao colocar um *beacon* num objeto, é possível a receção de um alerta quando este se afastar mais de X metros de casa.

Caso de estudo

The Protection Ad consiste num projeto desenvolvido pela *Nivea* cujo objetivo se destina a impedir que as crianças se afastem mais de X metros dos seus pais. Para isso, é colocado uma pulseira na criança (que possui um *beacon*) e, através da aplicação, os pais recebem um alerta quando a criança se afasta mais do que os metros configurados na aplicação [74].

Apêndice II: Requisitos do CMS

A secção atual apresenta os requisitos de interface, não funcionais e funcionais estipulados para o CMS.

Requisitos de interface

Os requisitos de interface destinam-se a estabelecer um conjunto de regras visuais que a plataforma deve seguir. Nas Tabelas II.1-8 serão apresentados os requisitos inicialmente definidos.

Tabela II.1 - CMS: Requisito de interface: Página de login

Página de login	
Tipo	Interface
Valor	Baixo
Descrição	A página de login do CMS consiste na primeira página que o retalhista visualiza. Deve estar presente a área do CMS (retalho), possuindo uma imagem de fundo relacionada, bem como um formulário de login no centro da página.

Tabela II.2 – CMS: Requisitos de interface: Imagem do CMS

Imagen do CMS	
Tipo	Interface
Valor	Médio
Descrição	A imagem do CMS deve respeitar as cores predominantes da Streamline , ou seja, o azul deve ser a cor predominante no CMS. O logotipo da Streamline deve ser facilmente identificável.

Tabela II.3 – CMS: Requisitos de interface: Layout do CMS

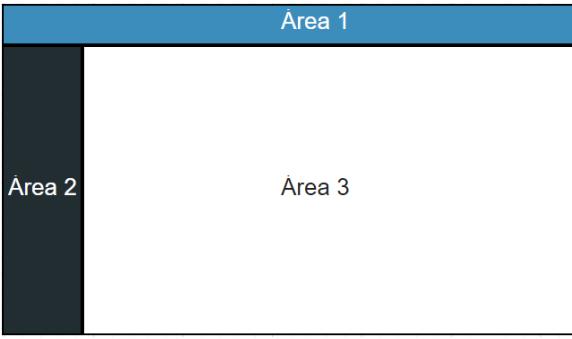
Layout do CMS	
Tipo	Interface
Valor	Alto
<p>O <i>layout</i> do CMS deve ser composto por três áreas.</p> <p><u>Área 1</u>: Topo da página;</p> <p><u>Área 2</u>: Menu de navegação com os principais módulos;</p> <p><u>Área 3</u>: Conteúdo respeitante ao menu selecionado.</p> <p>A área 3 deve ser a área predominante no ecrã, ocupando aproximadamente 80% do espaço disponível.</p>	
Descrição	 <p>O diagrama mostra um layout dividido horizontalmente. A parte superior é azul e rotulada como "Área 1". Abaixo disso, uma faixa preta à esquerda é rotulada como "Área 2". A maior parte da área é branca e rotulada como "Área 3".</p>

Tabela II.4 – CMS: Requisitos de Interface: Página inicial

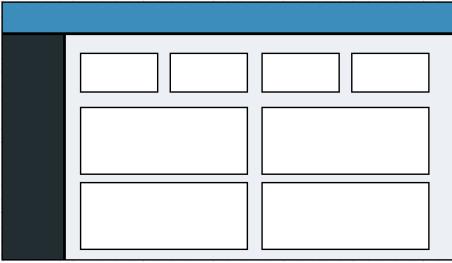
Página inicial	
Tipo	Interface
Valor	Médio
<p>A página inicial do CMS (após feito login) deverá conter um <i>dashboard</i> com informação útil para o utilizador. Deverá ser segmentada em diversas secções organizadas pelo seu conteúdo.</p>	
Descrição	 <p>O diagrama mostra uma estrutura de dashboard com uma barra azul no topo. Abaixo, uma faixa preta à esquerda e uma estrutura de 3 linhas por 4 colunas formando 12 caixas brancas.</p>

Tabela II.5 – CMS: Requisitos de interface: Design responsivo

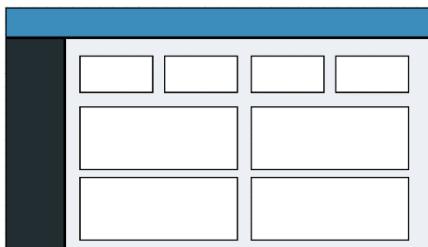
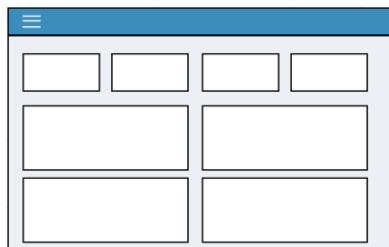
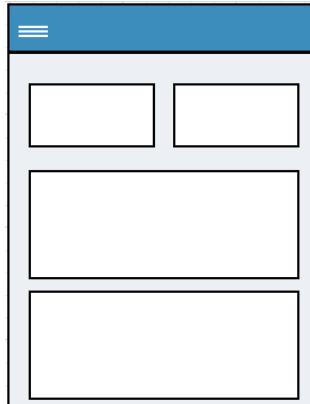
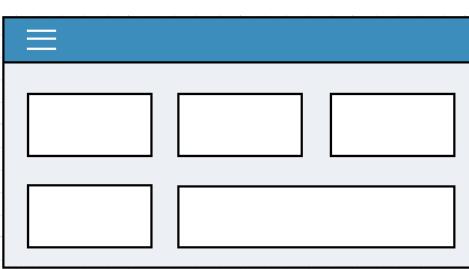
Design responsivo	
Tipo	Interface
Valor	Alto
<p>O CMS deve possuir um <i>design</i> responsivo adaptando-se a todo o tipo de ecrãs e plataformas, inclusive dispositivos móveis. Consoante o dispositivo e as suas medidas de ecrã, o <i>design</i> e <i>layout</i> devem ajustar-se automaticamente mantendo todas as funcionalidades, apresentando-as de forma adequada a cada situação.</p>	
Ecrã de um computador	Ecrã de um tablet
	
Descrição	
Ecrã de um smartphone (vertical)	Ecrã de um smartphone (horizontal)
	

Tabela II.6 – CMS: Requisitos de Interface: Módulos

Módulos	
Descrição	Interface
Os módulos existentes no CMS devem ser parte integrante da área 2 sendo de fácil visualização para o utilizador. Cada item do menu deve consistir num módulo devendo ser mostrado em submenu ações relacionadas com o módulo escolhido.	Médio

Tabela II.7 – CMS: Requisitos de interface: Funcionalidades acessíveis

Funcionalidades acessíveis	
Descrição	Interface
Em qualquer página em que o utilizador se encontre, devem ser percetíveis os módulos e funcionalidades existentes no CMS.	Médio

Tabela II.8 – CMS: Requisitos de interface: Localização do utilizador

Localização do utilizador	
Descrição	Interface
Em qualquer secção do CMS, deverá ser identificado de forma inequívoca a secção em que o utilizador se encontra. Deve ser usado o conceito de <i>breadcrumbs</i> de múltiplos níveis.	Baixo



Requisitos não funcionais

Apresenta-se nas Tabelas II.9-13 os requisitos não funcionais definidos para o CMS.

Tabela II.9 – CMS: Requisitos não funcionais: Usabilidade

Usabilidade	
Tipo	Não funcional
Valor	Médio
Descrição	O CMS deve ser de fácil utilização mesmo para utilizadores menos experientes em tecnologias de informação. Todas as funcionalidades devem ser intuitivas e de fácil acesso.

Tabela II.10 – CMS: Requisitos não funcionais: Desempenho

Desempenho	
Tipo	Não funcional
Valor	Alto
Descrição	A criação de <i>dashboards</i> dinâmicos, bem como resultados de filtros aplicados, deve ser rápida. Salvo informação expressa no CMS, todas as operações devem ser executadas num tempo máximo de 2 segundos. O uso de ferramentas para melhorar o aspetto visual não deve prejudicar o desempenho do CMS.

Tabela II.11 – CMS: Requisitos não funcionais: Portabilidade

Portabilidade	
Tipo	Não funcional
Valor	Alto
Descrição	O CMS deve funcionar em qualquer sistema operativo desde que possua os serviços MySQL, PHP e Apache. Deve fazer uso apenas de tecnologias OSS. Deve ainda poder comunicar com qualquer tipo de <i>beacons</i> independentemente do seu fabricante.

Tabela II.12 - CMS: Requisitos não funcionais: Fiabilidade

Fiabilidade	
Descrição	
<p>Valor Alto</p> <p>O CMS deve possuir uma elevada taxa de disponibilidade uma vez que toda a solução depende do funcionamento deste. O sistema deve ser suficientemente estável de forma, à sua disponibilidade se situar acima dos 99%.</p>	<p>Tip Não funcional</p>

Tabela II.13 – CMS: Requisitos não funcionais: Segurança

Segurança	
Descrição	
<p>Valor Muito Alto</p> <p>Devem ser implementadas técnicas de segurança de forma a impedir ataques que, por exemplo, comprometam a integridade / confidencialidade dos dados existentes na DB, bem como possam prejudicar o bom funcionamento da plataforma.</p>	<p>Tip Não funcional</p>

Requisitos funcionais

As Tabelas II.14-21 apresentam os requisitos funcionais inicialmente definidos.

Tabela II.14 – CMS: Requisitos funcionais: Utilizadores

Utilizadores		
ID	Funcionalidades	Valor
RF1.1	<p>Perfis de utilizador</p> <p>O CMS deve conter três perfis de utilizadores. Cada perfil corresponde a um nível de acesso com privilégios distintos.</p> <p>Lojista</p> <ul style="list-style-type: none"> • Visualiza apenas o conteúdo da loja a que se encontra associado; • Deve poder aceder á área de gestão de clientes, podendo: <ul style="list-style-type: none"> ○ Listar, adicionar, editar e apagar* clientes; • Deve poder aceder á área de gestão de grupos, podendo: <ul style="list-style-type: none"> ○ Listar, adicionar, editar e apagar* grupos; ○ Listar, associar, editar e remover clientes de um determinado grupo; 	Alto

	<ul style="list-style-type: none"> • Deve poder aceder á área de gestão de campanhas, podendo: <ul style="list-style-type: none"> ○ Listar, adicionar, editar e apagar todo o tipo de campanhas; <p>Gestor</p> <ul style="list-style-type: none"> • Todas as funcionalidades do perfil “Lojista” ao que acresce: <ul style="list-style-type: none"> ○ Deve poder aceder à área de gestão de utilizadores, podendo: <ul style="list-style-type: none"> ■ Listar, editar e apagar* utilizadores associados à sua loja; ■ Adicionar utilizadores apenas à sua loja; ○ Deve poder aceder á área de gestão de <i>beacons</i>, podendo: <ul style="list-style-type: none"> ■ Listar, adicionar, editar e apagar* <i>beacons</i>; ■ Mapear ou desmapear <i>beacons</i> em plantas; ○ Deve poder aceder á área de gestão de plantas, podendo: <ul style="list-style-type: none"> ■ Listar, adicionar, editar e apagar* plantas; ○ Deve poder aceder á área de análise de dados, podendo: <ul style="list-style-type: none"> ■ Visualizar o <i>dashboard</i> geral; ■ <i>Dashboard</i> por campanha: <ul style="list-style-type: none"> ● Selecionar a campanha, data e visualizar o <i>dashboard</i> respetivo; ■ <i>Dashboard</i> por região: <ul style="list-style-type: none"> ● Selecionar a região, data e visualizar o <i>dashboard</i> respetivo; ■ <i>Dashboard</i> por cliente: <ul style="list-style-type: none"> ● Selecionar o cliente, data e visualizar o <i>dashboard</i> respetivo; ■ Percurso dos clientes: <ul style="list-style-type: none"> ● Selecionar o cliente, planta, data e visualizar o percurso efetuado pelo respetivo cliente; ■ <i>Heatmap</i>: <ul style="list-style-type: none"> ● Selecionar a planta, data e criar o respetivo <i>heatmap</i>; ○ Visualiza e altera unicamente dados referentes à loja a que se encontra associado; <p>Administrador</p> <ul style="list-style-type: none"> • Todas as funcionalidades do perfil “Gestor” ao que acresce: <ul style="list-style-type: none"> ○ Visualiza e altera dados de qualquer uma das lojas registadas no CMS; ○ Deve poder aceder á área de gestão de lojas, podendo: <ul style="list-style-type: none"> ■ Listar, adicionar, editar e apagar* lojas; 	
--	---	--

	*O processo de remoção de lojas, utilizadores, grupos, clientes e plantas deve consistir apenas na alteração do estado de uma FLAG na DB que permita eliminar o conteúdo da página Web mas mantenha o mesmo na DB preservando todo o histórico. Devem ser feitas verificações aquando da remoção de dados de forma a garantir a estabilidade da plataforma, por exemplo: não deve ser possível apagar um grupo se possuir clientes.	
RF1.2	Visualizar todos os utilizadores numa lista onde seja possível efetuar um ordenamento por Nome, <i>Email</i> e Telemóvel.	Baixo
RF1.3	Possibilidade do utilizador Administrador poder assumir o papel de qualquer perfil / utilizador. Ver como: Perfil lojista.	Médio
RF1.4	Cada utilizador deve estar associado obrigatoriamente a uma e única loja.	Alto
RF1.5	Apenas o utilizador “Administrador” pode criar utilizadores do tipo “Gestor”.	Alto
RF1.6	Não existe a possibilidade de um utilizador se registar na plataforma. Todos os utilizadores devem ser criados por outros já existentes.	Alto
RF1.7	Cada utilizador deve estar associado obrigatoriamente a um e único perfil.	Alto
RF1.8	Um utilizador deve usar como dados para se autenticar no CMS o seu <i>Email</i> e <i>Password</i> .	Alto
RF1.9	Cada utilizador deve possuir um Nome, <i>Email</i> e <i>Password</i> obrigatoriamente.	Alto
RF1.10	Um utilizador do tipo “Administrador” deve poder alterar a <i>password</i> de qualquer utilizador.	Médio
RF1.11	Um utilizador do tipo “Gestor” deve poder alterar a <i>password</i> de qualquer utilizador associado à sua loja.	Baixo

Tabela II.15 - CMS: Requisitos funcionais: Lojas

Lojas		
ID	Funcionalidades	Valor
RF2.1	O CMS deve poder operar com várias lojas em simultâneo e de forma independente.	Alto
RF2.2	Apenas o utilizador “Administrador” deve gerir todas as lojas existentes no CMS, podendo: listar, adicionar, editar e apagar lojas.	Alto

Tabela II.16 – CMS: Requisitos funcionais: Plantas

Plantas		
ID	Funcionalidades	Valor
RF3.1	Cada planta deve estar associada a uma e só loja.	Alto
RF3.2	Cada loja pode possuir mais de uma planta.	Alto
RF3.3	Cada planta é identificada por nome, piso e uma única imagem.	Alto
RF3.4	O CMS deve permitir listar, adicionar, editar e apagar plantas com respetiva imagem associada.	Médio

Tabela II.17 – CMS: Requisitos funcionais: Beacons

Beacons		
ID	Funcionalidades	Valor
RF4.1	Cada <i>beacon</i> deve estar associado a uma e só loja.	Alto
RF4.2	Cada loja pode possuir mais de um <i>beacon</i> .	Alto
RF4.3	O CMS deve permitir listar, adicionar, editar e apagar <i>beacons</i> .	Alto
RF4.4	Um <i>beacon</i> deve ser identificado no CMS por: Nome, <i>UUID</i> , <i>Major</i> , <i>Minor</i> e uma imagem.	Alto
RF4.5	Deve ser possível mapear / desmapear um <i>beacon</i> numa planta.	Alto

Tabela II.18 – CMS: Requisitos funcionais: Clientes

Clientes		
ID	Funcionalidades	Valor
RF5.1	O CMS deve permitir listar, adicionar, editar e apagar clientes.	Alto
RF5.2	A pesquisa de um cliente deve ser feita pelo número de telemóvel.	Médio
RF5.3	Deve ser possível associar um ou mais dispositivos a cada cliente.	Alto
RF5.4	O dispositivo do cliente deve ser identificado, pelo menos, por um dos seguintes dados: Número de telemóvel; <i>Email</i> registado no telemóvel; <i>Mac Address</i> do telemóvel.	Alto
RF5.5	Um cliente pode pertencer a um ou mais grupos.	Alto

Tabela II.19 – CMS: Requisitos funcionais: Grupos

Grupos		
ID	Funcionalidades	Valor
RF6.1	O CMS deve permitir listar, adicionar, editar e apagar grupos.	Alto
RF6.2	Um grupo deve conter um ou mais clientes.	Alto

Tabela II.20 – CMS: Requisitos funcionais: Campanhas

Campanhas		
ID	Funcionalidades	Valor
RF7.1	O CMS deve permitir listar, adicionar, editar e apagar campanhas.	Alto
RF7.2	Cada campanha deve possuir uma data de validade sendo a sua listagem separada em campanhas ativas e inativas.	Médio
RF7.3	<p>O CMS deve permitir enviar três tipos de campanhas:</p> <p><u>Standard</u></p> <ul style="list-style-type: none"> • Deve existir pelo menos três layouts diferentes; • Cada <i>layout</i> possui dados e <i>design</i> diferentes; <p><u>Passbook</u></p> <ul style="list-style-type: none"> • O CMS deve permitir criar ficheiros do tipo pkpass; • O CMS deve permitir criar os cinco modelos de <i>Passbook</i>, nomeadamente: Genérico; Cartão de loja; Bilhete de evento; Cupão de desconto; Bilhete de viagem; • O aspetto do <i>Passbook</i> deve ser configurado no CMS, escolhendo, por exemplo, cor de fundo bem como imagem de capa; • O CMS deve possibilitar escolher o tipo de código que o bilhete possuirá: QR, PDF 417 ou AZTEC; <p><u>Web</u></p> <ul style="list-style-type: none"> • Deve existir pelo menos dois layouts diferentes; • Cada <i>layout</i> possui dados e <i>design</i> diferentes. 	Muito Alto
RF7.4	Uma campanha deve poder ser divulgada em um ou mais <i>beacons</i> .	Alto
RF7.5	Uma campanha deve poder ser divulgada para um ou mais grupos de clientes.	Alto
RF7.6	O CMS deve permitir pré-visualizar a campanha antes da sua publicação.	Alto

Tabela II.21 – CMS: Requisitos funcionais: Análise de dados

Análise de dados		
ID	Funcionalidades	Valor
RF8.1	<p>Deve existir um <i>dashboard</i> geral onde seja possível visualizar informações tais como:</p> <ul style="list-style-type: none"> • Afluência de clientes por loja por um determinado período de tempo; • Taxa de abertura de campanhas; • Top de campanhas enviadas / abertas; • Tipos de campanhas mais bem sucedidas. 	Alto
RF8.2	<p>Deve existir um <i>dashboard</i> por campanha onde seja possível visualizar informações tais como:</p> <ul style="list-style-type: none"> • Número de vezes que uma campanha foi enviada; • Número de vezes que uma campanha foi aberta; • Número de vezes que uma campanha foi ignorada; • Taxa de abertura de cada campanha; • Alcance de determinada campanha em número e taxa de clientes; • Número de vezes que uma campanha foi enviada por hora. 	Alto
RF8.3	<p>Deve existir um <i>dashboard</i> por região onde seja possível visualizar informações tais como:</p> <ul style="list-style-type: none"> • Total de clientes que estiveram muito próximo, perto ou longe de uma região de <i>beacons</i>; • Número de clientes por <i>beacon</i> e proximidade; • Taxa de clientes por proximidade; • Taxa de clientes por género; • Afluência de clientes por hora. 	Alto
RF8.4	<p>Deve existir um <i>dashboard</i> por cliente onde seja possível visualizar informações tais como:</p> <ul style="list-style-type: none"> • Número de campanhas que um cliente recebeu; • Número de campanhas que um cliente abriu; • Número de campanhas que um cliente ignorou; • Taxa de abertura de campanhas por cliente; • Número de campanhas que um cliente recebeu por hora; • Tipos de campanhas preferidas por cliente. 	Alto
RF8.5	O CMS deve possibilitar desenhar aproximadamente o percurso efetuado por um determinado cliente numa loja e num determinado intervalo de tempo.	Alto
RF8.6	O CMS deve permitir a criação de <i>heatmaps</i> por planta e num determinado intervalo de tempo.	Alto

Apêndice III: Requisitos da Aplicação Android

Esta secção apresenta os requisitos não funcionais e funcionais estipulados para a aplicação *Android*.

Requisitos não funcionais

Mostra-se nas Tabelas III.1-3 os requisitos não funcionais da aplicação *Android*.

Tabela III.22 – Aplicação Android: Requisitos não funcionais: Usabilidade

Usabilidade	
Tipo	Não funcional
Valor	Alto
Descrição	A aplicação deve ser de fácil utilização mesmo para utilizadores menos experientes em tecnologias de informação.

Tabela III.23 – Aplicação Android: Requisitos não funcionais: Desempenho

Desempenho	
Tipo	Não funcional
Valor	Alto
Descrição	Salvo informação expressa na aplicação, a operação de receção das campanhas deve ser executada num tempo máximo e 5 segundos.

Tabela III.24 - Aplicação Android: Requisitos não funcionais: Portabilidade

Portabilidade	
Tipo	Não funcional
Valor	Alto
Descrição	A aplicação deve ser compatível com a última versão atual do sistema <i>Android</i> , nomeadamente 6.0. Deve poder ser executada sem problemas em versões superiores a 4.0 desde que os dispositivos suportem tecnologia BLE.

Requisitos funcionais

As Tabelas III.4-7 apresentam os requisitos funcionais por área de desenvolvimento.

Tabela III.25 – Aplicação Android: Requisitos funcionais: Identificação do cliente

Identificação do utilizador		
ID	Funcionalidades	Valor
RF9.1	A aplicação não deve pedir nenhuma informação de identificação ao utilizador.	Alto
RF9.2	Deve obter automaticamente o número de telefone do dispositivo no qual se encontra a ser executada.	Alto
RF9.3	Deve obter automaticamente o <i>Email</i> registado no dispositivo.	Médio
RF9.4	Deve obter automaticamente o <i>Mac Address</i> do dispositivo.	Baixo

Tabela III.26 - Aplicação Android: Requisitos funcionais: Beacons

Beacons		
ID	Funcionalidades	Valor
RF10.1	A aplicação deve poder detetar qualquer tipo de <i>beacon</i> independentemente do fabricante, desde que respeite a norma <i>iBeacon</i> .	Alto
RF10.2	Deve ser capaz de recolher todos os dados de identificação do <i>beacon</i> com o qual comunicou perante a norma <i>iBeacon</i> .	Alto
RF10.3	Deve ser capaz de calcular a distância aproximada, em metros, a que se encontra do <i>beacon</i> com o qual comunicou, com base na potência do sinal recebida (<i>TX Power</i>).	Alto
RF10.4	O protótipo deve possuir um botão onde deve ser possível parar e iniciar a procura de <i>beacons</i> nas proximidades.	Médio

Tabela III.27 – Aplicação Android: Comunicação com o CMS

Comunicação com o CMS		
ID	Funcionalidades	Valor
RF11.1	A aplicação deve comunicar com o CMS de forma transparente para o utilizador.	Alto
RF11.2	Deve enviar os dados de identificação do cliente e do <i>beacon</i> detetado nas proximidades.	Alto
RF11.3	Durante a comunicação com o CMS, a aplicação não deve impedir o utilizador de efetuar outra ação.	Médio

Tabela III.28 – Aplicação Android: Campanhas

Campanhas		
ID	Funcionalidades	Valor
RF12.1	A aplicação deve receber campanhas enviadas pelo CMS.	Alto
RF12.2	Deve conseguir abrir campanhas do tipo <i>Standard</i> , com três tipos de <i>design</i> e <i>layout</i> diferentes.	Alto
RF12.3	Deve receber campanhas do tipo <i>Passbook</i> , independentemente do tipo de passe.	Alto
RF12.4	Deve enviar informação ao CMS quando uma campanha é aberta pelo utilizador.	Alto
RF12.5	Deve enviar informação ao CMS quando uma campanha é ignorada pelo utilizador.	Alto

Apêndice IV: Diagrama de atividades da aplicação *Android*

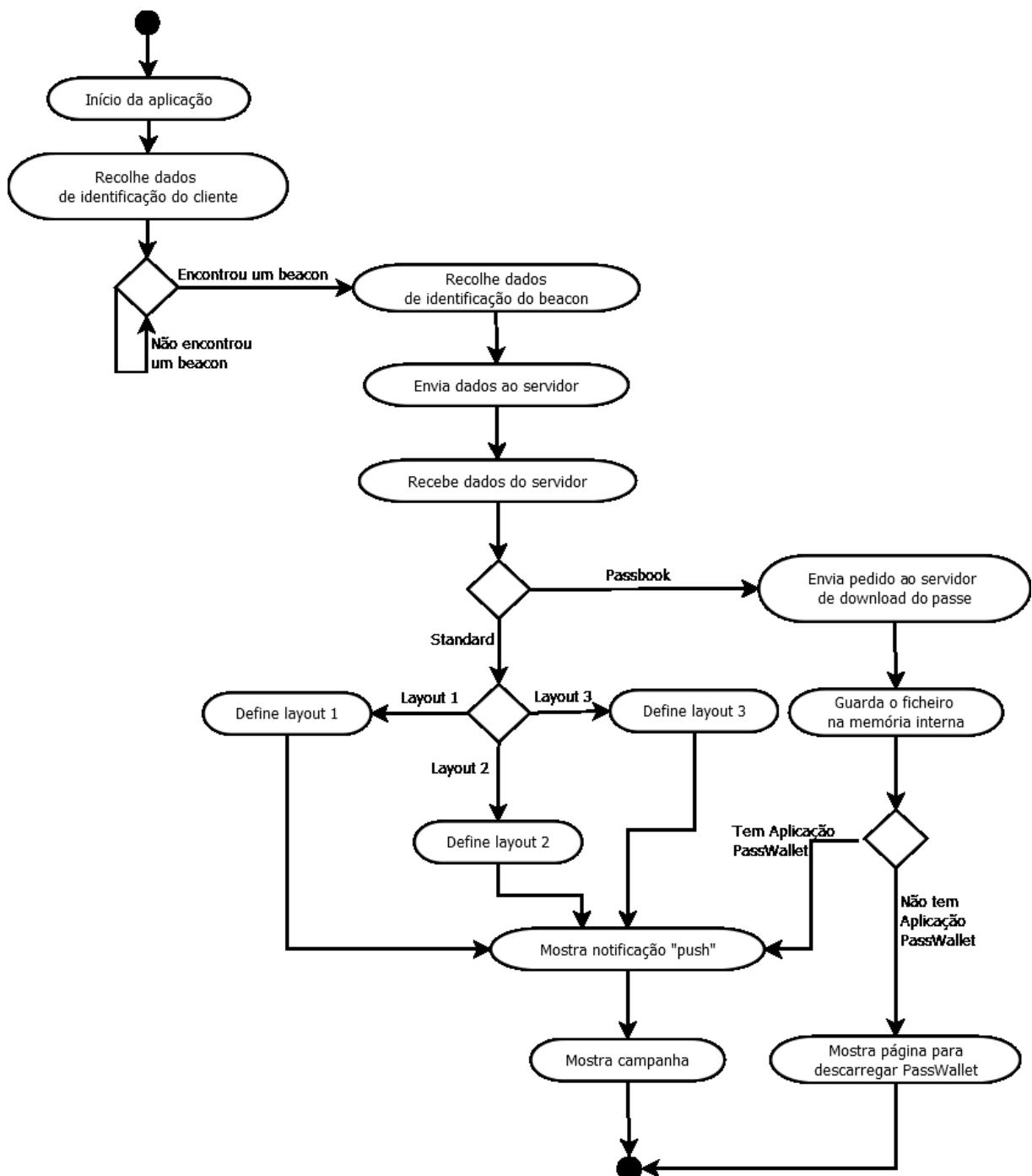


Figura IV.1 - Diagrama de atividades da aplicação Android

Apêndice V: Diagrama da base de dados do CMS

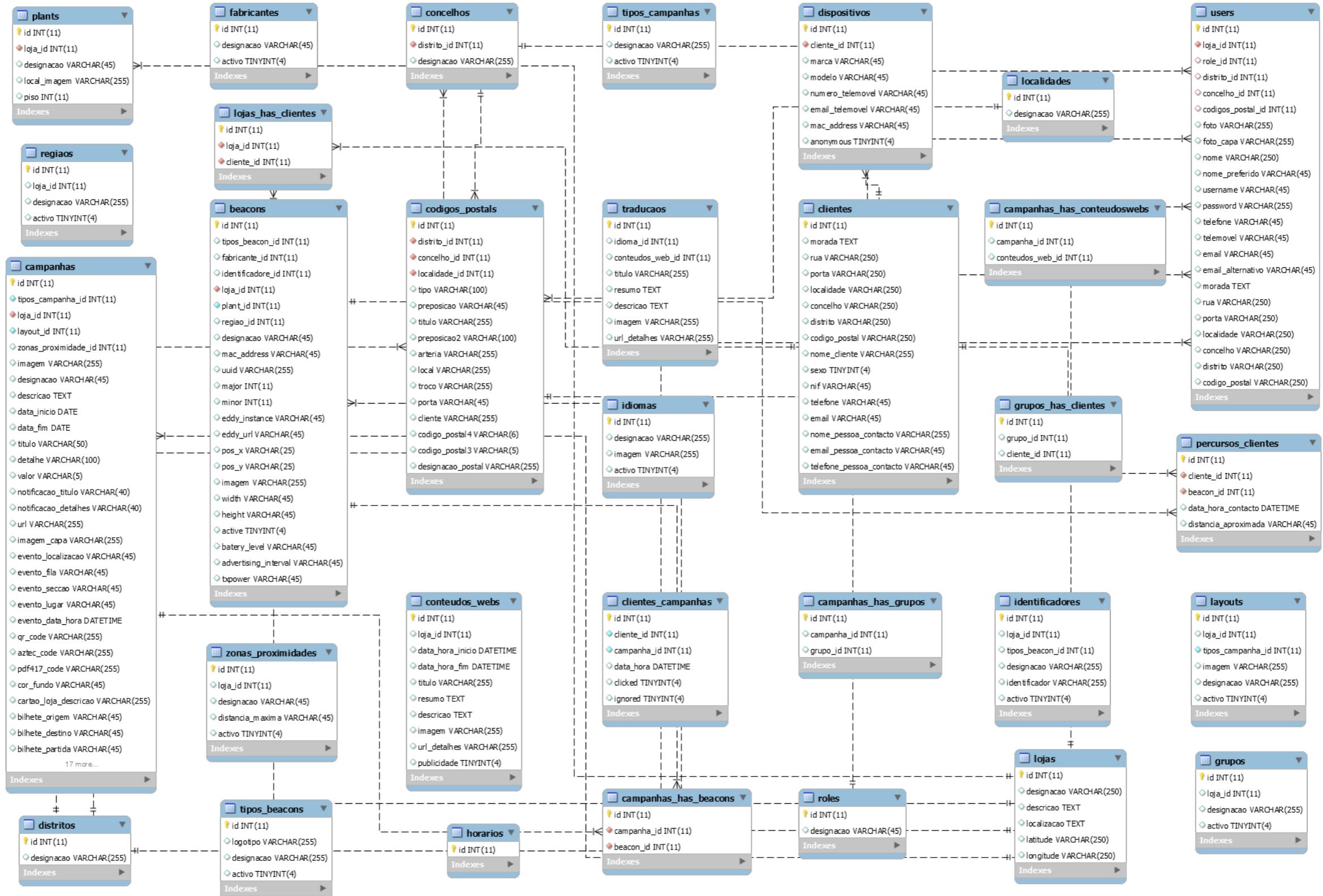


Figura V.1 - Diagrama da base de dados do CMS

Nuno Alexandre Lemos de Paiva