Instituto Universitário de Lisboa

Departamento de Ciências e Tecnologias da Informação

Sistema para Deteção de Zonas de Estacionamento para Smart Cities

João Filipe Machado de Matos Fernandes

Dissertação submetida como requisito parcial para obtenção do grau de

Mestre em Engenharia Informática

Orientador:
Carlos José Corredoura Serrão, Professor Auxiliar,
ISCTE-IUL

Coorientador:
Nuno Miguel de Figueiredo Garrido, Professor Auxiliar,
ISCTE-IUL

Setembro, 2018

"Não importa trabalhar muito. Devemos utilizar o tempo disponível para trabalhar bem."

José de Matos Fernandes

Agradecimentos

Ao meu orientador, Professor Carlos Serrão, que se mostrou prontamente disponível para me orientar, agradeço imenso todo o apoio ao longo deste tempo. Ao meu coorientador, Professor Nuno Garrido, que sempre esteve disponível para me ajudar e que me acompanhou em primeira pessoa ao longo de todo este percurso, agradeço imenso e um especial "muito obrigado" pela sua diligência em se ter deslocado até Madrid para me substituir na apresentação do artigo científico. Ao ISCTE-IUL, que possibilitou a concretização de todo este trabalho e forneceu os materiais e o seu laboratório, agradeço todo o apoio.

À família, à minha mãe, ao meu pai e ao meu irmão, que me apoiaram nas decisões, deram-me a liberdade de poder seguir o meu caminho, agradeço por tudo. Ao meu avô, a quem deixo uma especial homenagem, que sempre puxou por mim, sempre me orientou para a vida, nunca criticou as minhas opções, deu-me conselhos que guardo para sempre e tenho a certeza que iria adorar este trabalho, agradeço profundamente por tudo porque hoje em dia dou o devido valor!

Aos meus amigos, que me apoiaram ao longo do desenvolvimento desta dissertação e me ajudaram a encontrar problemas no protótipo durante a fase de testes, tornando o trabalho mais completo e mais bem concebido, agradeço imenso por tudo.

À minha namorada, Ângela, um enorme muito obrigado pela força que me deu na recta final desta dissertação. Ajudou-me imenso a superar os meus medos e a dar-me confiança para mostrar aquilo que realmente sou e fiz ao longo deste tempo. Um agradecimento especial a uma pessoa que entrou há pouco tempo na minha vida, mas já fez tanto por mim!

À EMEL e Câmara Municipal de Lisboa, particularmente, ao Nuno Sardinha e ao Pedro Machado, que se mostraram recetivos para nos receberem e conhecerem este trabalho, ajudando a compreender mais sobre este tema, agradeço por tudo.

A todos os que enumerei o meu sincero "Muito Obrigado".

Resumo

Os sistemas de *Smart Parking* otimizam o uso do espaço de estacionamento, melhoram a eficiência das suas operações de estacionamento e auxiliam o tráfego na cidade a fluir de forma mais livre. Ao longo dos anos, algumas cidades levaram a cabo estudos sobre este conceito e que impactos um sistema de *Smart Parking* pode trazer, em termos sociais, económicos, políticos e ambientais.

Contudo, existem alguns problemas inerentes à implementação deste tipo de sistemas de estacionamento inteligente que preocupam os responsáveis políticos das cidades. O elevado investimento é algo que perturba as entidades reguladoras do estacionamento nas cidades, o que dificulta a evolução das *Smart Cities* a nível global.

O caso de estudo desta dissertação apresenta uma possível solução para um sistema de *Smart Parking* com o menor custo possível, tendo em conta a sua viabilidade e escalabilidade, e, posteriormente, a sua implementação numa grande cidade como Lisboa. O protótipo desenvolvido durante o trabalho representa uma prova de conceito para a futura implementação do sistema proposto, o qual pode ser usado em Lisboa, ou ser adaptado para uma qualquer outra cidade no mundo.

O estudo e as sugestões realizados não dispensam, ainda assim, outros dados relevantes para o crescimento deste tema, havendo, por conseguinte, um longo trilho a percorrer por forma a tornar o *Smart Parking* uma realidade no quotidiano das cidades.

Palavras-Chave: *Smart Parking*; *Smart Cities*; protótipo; estacionamento; Lisboa.

Abstract

Smart Parking systems optimize the parking space usage, improve the efficiency of the parking operations and help city traffic to flow more freely. Over the years, some cities have carried out studies about this concept and what impacts can a *Smart Parking* system cause in terms of social, economic, political and environmental aspects.

However, there are some problems related to *Smart Parking* systems implementation which concerns politicians in major cities. High investment is something that disturbs parking regulators in cities, which hampers the evolution of *Smart Cities* globally.

The case study of this thesis presents a low-cost solution for a *Smart Parking* system, considering its viability and scalability, and later its implementation in a city like Lisbon. The prototype developed during this work is considered as proof of concept for the future implementation of the proposed system in Lisbon or any other city in the world.

The study and the suggestions made do not discard any other data relevant to the growth of this topic. There is still a long way to go for *Smart Parking* to become a reality in everyday cities.

Keywords: *Smart Parking*; *Smart Cities*; prototype; parking; Lisbon.

Índice

Índice de Quadros

Índice de Figuras

Lista de Abreviaturas e Siglas

API – *Application Programming Interface*

AWS – *Amazon Web Services*

BLE – *Bluetooth Low Energy*

CLI – *Command Line Interface*

CLS – *Common Language Specification*

FAN – *Field Area Networks*

GCP – *Google Cloud Platform*

GPIO – *General Purpose Input/Output*

HAN – *Home Area Networks*

I2C – *Inter-Integrated Circuit*

IoT – *Internet of Things*

LoRaWAN – *Long Range Wide Area Network*

M2M – *Machine to Machine*

MBaaS – *Mobile Backend-as-a-Service*

NFC – *Near Field Communication*

OTA – *Over-The-Air*

PANID – *Personal Area Network Identification*

PoE – *Power over Ethernet*

RFID – *Radio-frequency Identification*

SDK – *Software Development Kit*

SI – Sistemas de Informação

SPI – *Serial Peripheral Interface*

TI – Tecnologias de Informação

TIC – Tecnologias de Informação e Comunicação

UOS – *Urban Operating System*

WAN – *Wide Area Networks*

WSN – *Wireless Sensor Network*

1. Introdução

Assistimos, nas zonas mais urbanas, a um fluxo cada vez mais estrangulado de tráfego devido ao aumento do número de veículos [1] e à circulação *parasita* associada à procura de lugares de estacionamento. É indispensável dentro do contexto das *Smart Cities*, que se encontrem soluções que tornem o quotidiano dos seus habitantes mais agradável o que terá, consequentemente, efeitos na economia e no desenvolvimento dos grandes centros urbanos.

Este projeto consiste na investigação e desenvolvimento de um sistema de gestão inteligente de estacionamento, que tem por base sensores distribuídos por zonas de parqueamento em espaços públicos e/ou privados, nas cidades. Este trabalho irá permitir observar se este tipo de sistemas, aliado à utilização de aplicações móveis, poderá influenciar a otimização do estacionamento em grandes cidades e a mobilidade dos cidadãos.

O sistema a desenvolver pretende aumentar a rentabilização dos espaços de estacionamento em zonas de grande tráfego e redução do tempo de espera, por via do recurso a uma aplicação móvel que informa, previamente, os utilizadores/condutores da existência de um lugar de estacionamento, numa certa zona da cidade, para a qual este pretenda deslocar-se de automóvel e parquear.

Estes sistemas facilitam a procura de lugares de estacionamento, permitindo, também, que os condutores não tenham de percorrer extensivamente a cidade, economizando tempo e reduzindo o gasto de combustível, cujos gastos desnecessários trazem enormes consequências para o meio ambiente e aumento da dependência energética do Estado.

Este trabalho baseia-se no desenvolvimento de um sistema de *Smart Parking* que apresente uma solução de menor custo e mais aberta, em termos de *hardware* e

software, quando comparada com os sistemas existentes e, eventualmente, a sua aceitação por parte dos utilizadores.

1.1. Identificação e descrição do problema

Dois dos principais problemas que as cidades, hoje em dia, enfrentam são os seguintes: a mobilidade e o estacionamento [2] [3]. Todos os dias, milhares de veículos e os respetivos ocupantes circulam nas cidades e a busca de lugares de estacionamento em cidades congestionadas é bastante problemática. As denominadas *Smart Cities* procuram tirar partido das novas Tecnologias de Informação e Comunicação (TIC) para serem mais eficientes e otimizarem os seus próprios recursos. A mobilidade e o estacionamento são problemas que afetam não apenas os condutores, mas também as cidades, em termos de distribuição e ocupação dos locais de estacionamento, a melhoria da fluidez de tráfego e a redução de gases nocivos.

Como acontece em todas as grandes cidades, Lisboa é um exemplo que sempre apresentou dificuldades, no que respeita ao estacionamento na via pública e ao congestionamento do trânsito durante os períodos da manhã e final da tarde. Isto acontece devido ao número elevado de veículos que entram na cidade e que, posteriormente, procuram aceder a lugares de estacionamento. O número de lugares de estacionamento na via pública é de 152500, dos quais 112350 são não tarifados e os restantes 40150 são tarifados [4]. Os parques de estacionamento de acesso público contêm, no total, 51500 lugares. Estes números apresentam um total de 203900 lugares de estacionamento público. A população de Lisboa é de, aproximadamente, 564650 habitantes, sendo o número de veículos na cidade de, estimadamente, 159000, entrando e saindo, diariamente, em média, 790000 viaturas. Da figura 1.1 resulta, de forma clara, a evolução da procura diária de estacionamento ao longo do dia, em Lisboa:

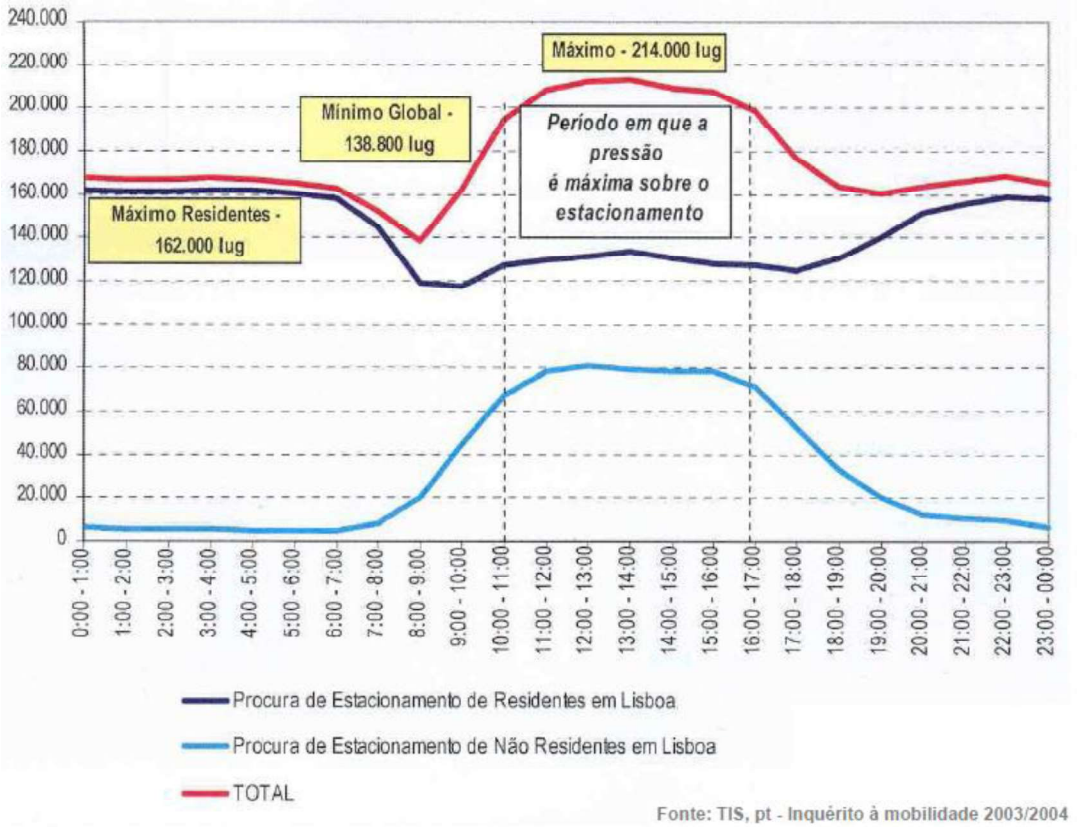

Figura 1.1: Procura diária de estacionamento em Lisboa [4].

Assim se verifica que, na cidade de Lisboa, o número de lugares de estacionamento, por habitante, é de 0.27, um valor excessivamente reduzido quando comparado com a procura diária de lugares de acesso público.

Outra conclusão resultante da observação da figura 1.1 está ligada com a pressão da procura de estacionamento ao longo do dia. A figura 1.2 apresenta um mapa da cidade de Lisboa que mostra as freguesias onde a procura diurna ou noturna de lugares de estacionamento acontece com maior frequência. Assim, é possível verificar quais as zonas da cidade que carecem de uma particular atenção e que possam usufruir de um sistema de estacionamento mais avançado, auxiliando na procura de lugares de estacionamento.

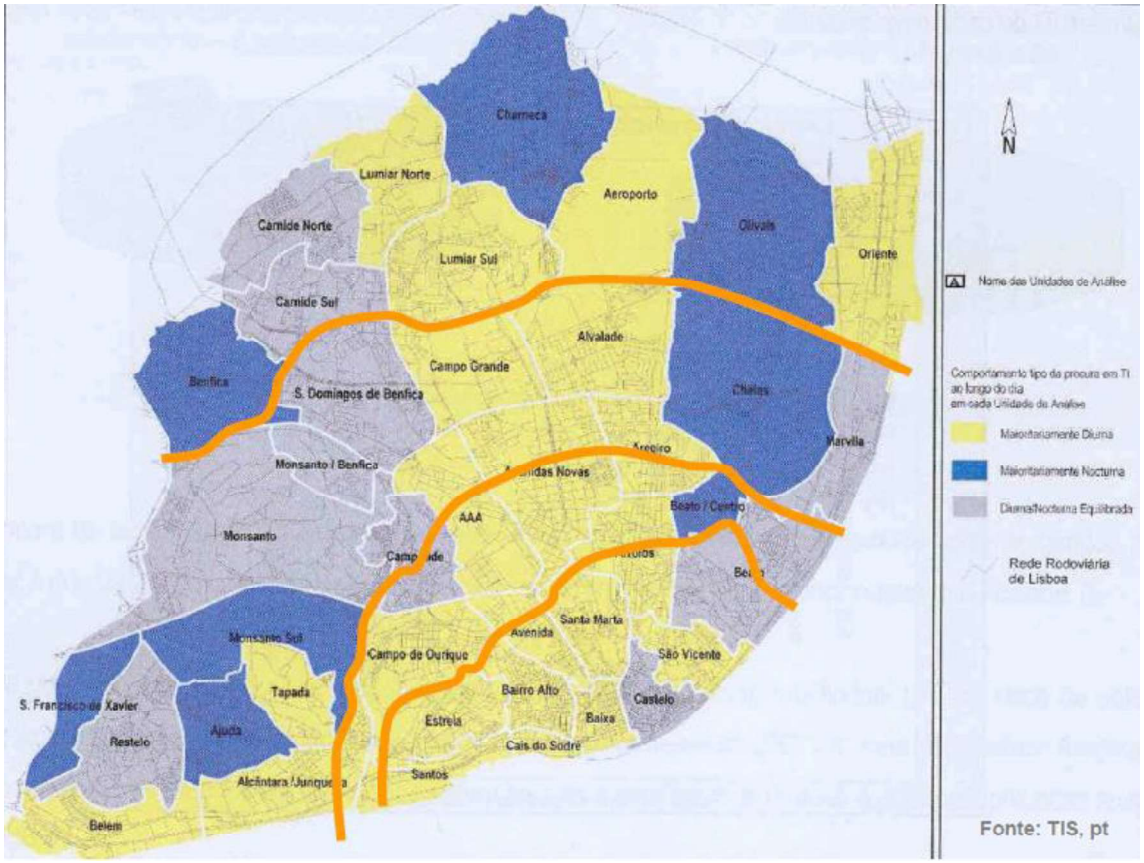

Figura 1.2: Pressão da procura de estacionamento ao longo do dia [4].

Partindo, então, do exemplo da cidade de Lisboa, a existência de mais de 204 mil lugares de estacionamento aí existentes, aliada a uma procura que ronda os 220 mil veículos em determinadas alturas do dia, além da maior pressão em determinadas zonas, sobretudo em zonas históricas e menos habilitadas ao tráfego, exige uma maior capacidade de gerir o estacionamento dentro da cidade. Este problema é mitigado, parcialmente, com a ajuda de ferramentas tecnológicas usadas pela empresa de gestão de estacionamento em Lisboa – a EMEL – em estreita cooperação com a Câmara Municipal de Lisboa (designada por CML). A aplicação '*ePark*' possibilita o pagamento das tarifas de estacionamento por intermédio dos dispositivos móveis dos utilizadores. Todavia, não resolve o problema do tráfego e do tempo de espera em busca de uma vaga de estacionamento.

Este projeto procura endereçar e mitigar problemas similares ao caso de Lisboa e, assim, criar uma solução viável, de baixo custo e aberta de modo a integrar outros elementos tecnológicos consoante as necessidades de cada cidade.

1.2. Objetivos da Dissertação

O objetivo desta dissertação engloba dois pontos essenciais: investigar o problema do estacionamento nas cidades e desenvolver um protótipo de um sistema de *Smart Parking* com o intuito de diminuir, significativamente, o tempo despendido na busca de uma vaga em local de estacionamento, otimizando, dessa forma, o consumo de combustível e o percurso a realizar pelo interessado no estacionamento do veículo. Este problema real e que afeta, diretamente, a população pode, também, ser visto como um problema de mobilidade mais abrangente e abstrato ao nível do meio ambiente e da dependência energética de combustíveis fósseis. O protótipo a ser desenvolvido irá estudar a melhor combinação de *hardware* e *software* para oferecer uma solução de baixo custo para uma gestão mais eficiente do estacionamento em *Smart Cities*.

Assim, os principais objetivos deste trabalho são os seguintes:

1. Rever o estado da arte dos principais mecanismos existentes para a monitorização e gestão de lugares de estacionamento;
2. Investigar quais os principais sistemas de parqueamento inteligente que são utilizados e disponibilizados para os cidadãos que possuem as características em estudo;
3. Investigar as capacidades das aplicações existentes e as suas principais limitações;
4. Desenhar e implementar um sistema conectado a uma aplicação móvel que ofereça uma solução de baixo custo e mais aberta em relação ao *hardware* e *software* utilizados em sistemas existentes;
5. Realizar testes para validar a eficácia da aplicação móvel a nível conceptual e de desenho.

Para consolidar os objetivos, anteriormente, enumerados, é importante ter em conta algumas questões de investigação a que o trabalho procura dar resposta:

1. Será possível desenvolver uma plataforma tecnológica, baseada em sensores e em tecnologias móveis de baixo custo que seja capaz de gerir, de uma forma inteligente, o estacionamento em cidades?
2. Esta plataforma será aberta o suficiente para permitir a instalação de diferentes componentes consoante as necessidades de cada cidade e dos seus cidadãos?

1.3. Metodologias de investigação

O Design Science Research é um conjunto de técnicas analíticas muito utilizadas em investigação para Tecnologias de Informação (TI). Este método consiste no desenvolvimento de artefactos inovadores e analisar se a utilização e performance desses mesmos artefactos trazem benefícios aos Sistemas de Informação (SI) [5].

Como tal, o método de investigação a ser utilizado neste projeto é o Design Science Research. O artefacto a ser utilizado é uma instanciação, pois pretende-se concretizar e orientar os artefactos para a realidade atual.

O projeto inicia-se na fase de identificação do problema. O estado da arte permite identificar quais os problemas inerentes ao estacionamento, nomeadamente, em sistemas de estacionamento atuais e sistemas de *Smart Parking* existentes. A segunda fase passa por definir os objetivos para a solução e, dentro do que foi descrito na secção anterior sobre os propósitos da dissertação, a redução dos custos e a abertura do sistema em termos de *hardware* e *software* para inclusão de outros componentes alternativos enquadram-se nas principais metas para a solução em estudo. O capítulo 3 está relacionado com a terceira fase da metodologia do Design Science Research, descrevendo-se a fase de desenho e desenvolvimento do artefacto. As fases de demonstração e avaliação do artefacto são referidas nos capítulos 4 e 5 da dissertação, onde são detalhados os testes e as conclusões, relativamente ao protótipo funcional desenvolvido, tendo em conta, também, o trabalho futuro sobre esta matéria. A última fase da metodologia do Design Science Research refere-se à comunicação do problema e isso será conseguido através da elaboração de um artigo científico que expõe este problema e apresenta uma solução com base no protótipo desenvolvido ao longo do trabalho.

1.4. Estrutura e organização da dissertação

Esta dissertação está organizada, essencialmente, em cinco capítulos principais que, por sua vez, estão estruturados de forma a transmitir, fielmente, as fases realizadas durante o percurso de trabalho.

O capítulo introdutório desta dissertação servirá, essencialmente, para se fazer uma referência à identificação do problema e objetivos da mesma, bem como a descrição das metodologias de investigação utilizadas durante todo o trabalho.

No capítulo seguinte, referente à análise do estado da arte, começa-se por efetuar uma descrição geral do conceito de *Smart Cities*, que engloba o ponto seguinte inerente ao tema principal do trabalho: *Smart Parking*. Dentro deste ponto, far-se-á uma breve explicação do que é um sistema de *Smart Parking*, incluindo algumas referências a estudos já efetuados neste âmbito. Seguidamente, elaborar-se-á uma lista de sistemas e aplicações existentes e será feita, no final deste capítulo, uma comparação entre cada sistema, apresentando, por conseguinte, as vantagens e desvantagens de cada um.

O terceiro capítulo refere-se ao trabalho prático realizado, indicando, em cada secção, a solução apresentada e, bem assim, a descrição dos elementos do protótipo e os resultados obtidos. Optou-se por separar o capítulo em quatro partes essenciais inerentes às fases do trabalho prático realizado: levantamento de requisitos, arquitetura do sistema global, sensores de estacionamento, *gateway, backend* e aplicação móvel.

De seguida, é apresentado o capítulo onde é aplicado o trabalho desenvolvido, efetuando uma análise coerente do problema que se procura resolver e como o protótipo funcional construído pode colmatar esse problema, enumerando as vantagens, desvantagens e custos da solução proposta. Posteriormente, será feita a validação da solução obtida, com descrição do resultado dos testes e apresentar-se-á uma lista com os aspetos positivos e negativos das tecnologias aplicadas.

Por fim, serão apresentadas as principais conclusões após o desenvolvimento desta dissertação e sugestões para trabalho futuro a ser realizado.

2. Estado da Arte

Neste capítulo dá-se a conhecer, primeiramente, a área global em estudo, denominada por *Smart Cities*, bem como as suas principais caraterísticas e tecnologias envolventes. No ponto seguinte, insere-se a componente principal da dissertação – *Smart Parking* – onde é apresentado, com mais detalhe o conteúdo relacionado com o desenvolvimento e estudos recentemente efetuados nesta área. Para além da componente teórica desta matéria, será listado um conjunto de sistemas e aplicações existentes onde, no final, será feita uma análise comparativa entre eles, permitindo identificar e descrever os seus principais aspetos positivos e negativos. O objetivo desta comparação consistirá na determinação dos principais requisitos da solução a desenvolver, resolvendo os aspetos menos positivos das soluções existentes.

2.1. Smart Cities

2.1.1. Introdução

O conceito de *Smart City* ainda está a surgir e o trabalho envolvente nesta matéria está a progredir [2]. É, por vezes, construída uma definição baseada no significado de cidade como um conjunto de componentes inteligentes que se interrelacionam através das tecnologias [6]. Outra definição passa por juntar as TIC e as tecnologias *Web* que ajudem a agilizar os processos burocráticos e identificar novas e inovadoras soluções que melhorem a sustentabilidade e o quotidiano dos cidadãos [7]. Os conceitos, anteriormente, descritos referem-se à *Smart City* como nível de inteligência da cidade através do uso racional dos seus recursos e a qualidade dos serviços prestados aos cidadãos, beneficiando o aspeto económico, social e político.

É, atualmente, possível determinar que uma *Smart City* se apresenta como um conjunto de subsistemas e componentes interligados. A coesão entre as infraestruturas e os serviços, juntamente com a utilização de dispositivos para monitorizar a atividade dentro da cidade, faz com que esta cresça relativamente à sua sustentabilidade e

eficiência [8]. O desenvolvimento progressivo das áreas urbanas e da população obrigou os governos e autarquias a apostar em soluções mais inteligentes para o aumento da sua produtividade e redução dos problemas relacionados com o quotidiano dos cidadãos [9]. "A dificuldade na gestão de resíduos, a escassez de recursos, a poluição, a preocupação da saúde humana, o congestionamento do trânsito e as infraestruturas inadequadas, deterioradas e envelhecidas estão dentro dos problemas considerados básicos" [2] [10]. Por isso, numa tentativa de atenuar estes obstáculos, procuram-se, constantemente, soluções tecnológicas e inteligentes que aumentem a qualidade de vida nas cidades.

2.1.2. Principais características das *Smart Cities*

Considerando o ponto anterior, uma *Smart City* é encarada como "uma cidade que, ao monitorizar e integrar as condições de todas as suas infraestruturas críticas, incluindo estradas, pontes, túneis, linhas de comboio, metropolitanos, aeroportos, portos, comunicações, água, energia e até grandes edifícios, pode otimizar melhor os seus recursos, planear as suas atividades preventivas de manutenção e controlar os aspetos de segurança, enquanto o serviço prestado aos cidadãos é maximizado" [2]. A figura 2.1 ilustra os principais componentes de uma *Smart City*.

Figura 2.1: Principais componentes de uma *Smart City* [8].

As características de uma *Smart City* resumem-se em seis pontos essenciais, relacionados com aspetos em comum entre as cidades:

1. Infraestruturas conectadas que tornam a política e a economia mais eficientes e permitem evoluir as componentes social, cultural e urbana [9];
2. Atrair novos negócios aumenta o desempenho socioeconómico e o desenvolvimento urbano;
3. Aumento da inclusão social em serviços públicos;
4. Fomentar o sucesso com a aposta nas indústrias tecnológicas e criativas;
5. Garantir que a comunidade consegue aprender e adaptar-se às novas tecnologias;
6. Elaborar estratégias de sustentabilidade social e ambiental, bem como, a utilização estável e moderada dos recursos naturais.

Em suma e aliando a figura 2.1 com os pontos listados, anteriormente, considera-se que uma *Smart City* se divide em seis dimensões fundamentais: economia inteligente; mobilidade inteligente; ambiente inteligente; comunidade inteligente; vida inteligente; governo inteligente. Ou seja, uma cidade que apresente estas características, poderá, certamente, ser identificada como uma *Smart City* que evoluiu em prol do crescimento e aumento da qualidade de vida dos seus cidadãos.

Na perspetiva do autor em [6], uma *Smart City* é entendida como sendo a conexão entre todas as infraestruturas da cidade (físicas, tecnológicas, sociais e de negócio) de modo a tornar a cidade mais inteligente na sua totalidade. A ideia é obter dados provenientes de aparelhos físicos e de *software* avançado para auxiliar nas decisões a tomar, aumentando a qualidade de vida nas cidades. Para que esta evolução aconteça, sugerem-se três propriedades de TI para o desenvolvimento das *Smart Cities* [6]:

1. Instrumentação: conectar o mundo físico e virtual com o uso de dispositivos que obtêm e recebem dados, disponibilizando essa informação aos cidadãos através de *smartphones*;
2. Interconexão: a recolha de dados a partir dos dispositivos distribuídos pela cidade e das redes sociais são organizados de forma a que a fonte dos mesmos não seja apenas de um sistema em particular, mas sim do conjunto de todos os sistemas interligados;
3. Inteligência: gerar novos conceitos, decisões e ações a partir da análise da informação obtida, valorizando ainda mais os dados adquiridos durante o processo.

Para que todas as propriedades referidas sejam aplicadas de forma coerente e eficaz, é necessário compreender os desafios e estratégias (tabela 2.1) que um projeto de *Smart City* apresenta, não só, na fase de planeamento como também *a posteriori*, no início do seu desenvolvimento. Estas duas etapas são essenciais para perceber se um dado projeto é exequível em tempo útil, se apresenta um orçamento acessível ao Estado, câmaras municipais ou freguesias, se, porventura, a equipa de desenvolvimento apresenta os conhecimentos suficientes para a concretização do projeto, bem como se os cidadãos e comunidades se adaptarão, facilmente, aos mecanismos tecnológicos dos produtos e serviços. Para que isso aconteça, é necessário educar os cidadãos para as novas tecnologias antes de avançar para projetos de complexidade elevada que necessitam de conhecimentos mais profundos da matéria.

Desafios	Estratégias
• Tamanho do projeto • Atitude e comportamento do gestor de projeto • Diversidade dos utilizadores ou organizacional • Falta de alinhamento dos objetivos organizacionais e de projeto • Objetivos múltiplos ou em conflito • Resistência na mudança • Disputas e conflitos	• Perícia e experiência da equipa de projeto • Líder de TI respeitado e competente (técnica e socialmente) • Objetivos claros e realistas • Identificação de decisores relevantes • Envolvimento do utilizador final • Planear • Etapas claras e entregas mensuráveis • Boa comunicação • Melhoria do processo de negócio anterior • Treino adequado • Financiamento adequado • Revisão das práticas atuais

Tabela 2.1: Desafios e estratégias na gestão e organização numa *Smart City* [2].

2.1.3. Tecnologia envolvida

O uso das TIC e de tecnologias de inteligência computacional que permitem obter dados em tempo-real, fazendo uso das redes sociais [6], é uma prática comum dentro dos projetos de *Smart Cities*, o que "torna todos os componentes e serviços da cidade – administração, educação, saúde, segurança pública, transportes e utilidades − mais inteligentes, interligados e eficientes" [11]. Uma infraestrutura de TIC consiste, essencialmente, num conjunto de tecnologias usadas que permitem a comunicação entre

os componentes das cidades e os cidadãos, designadamente o uso de redes Wi-Fi, canais de fibra ótica, pontos de Internet sem-fios. Tudo isto faz com que sejam criados serviços mais céleres e eficientes para os cidadãos, aumentando a capacidade de resposta da cidade aos problemas causados, diariamente, e que afetam a população no seu quotidiano, tanto no emprego como na sua vida pessoal, saúde, educação e transportes [2]. Dentro do contexto de *Internet of Things* (IoT), a comunicação entre os sensores depende da localização e cobertura necessária, contando com três tipos de redes: *Home Area Networks* (HAN), *Wide Area Networks* (WAN) e *Field Area Networks* (FAN) [8]. Os protocolos de comunicação mais utilizados são descritos seguidamente:

a. 3G e LTE

Estas tecnologias foram concebidas para a conexão por via de banda larga, evitando a sua utilização por parte de sistemas de curto alcance e baixa potência. Estes serviços são fornecidos pelos operadores de telecomunicações e o custo dos dados ainda é um pouco elevado. Em aplicações de IoT, estas tecnologias são descartadas, devido à enorme quantidade de dados transferidos diariamente para a *cloud* e ao consumo elevado de energia em redes de sensores. Particularmente, o LTE tem sido muito favorável em aplicações M2M de alto desempenho, "devido à sua cobertura, alta taxa de transferência e baixa latência" [8].

b. ZigBee

O ZigBee é um protocolo de rede usado para transmissão de pequenas quantidades de dados, através do padrão IEEE 802.15.4. A tecnologia utilizada pode ser comparada às redes Wi-Fi e Bluetooth, distinguindo-se, porém, pelo seu baixo consumo e com um alcance de cerca de 100 metros [12].

Os XBee representam um conjunto de dispositivos da empresa Digi [13] que utilizam o protocolo ZigBee, entre outros, para comunicação de dados dentro de uma rede sensorial [14].

c. Dash7

O Dash7 é um protocolo que tem sido alvo de estudo para aplicações militares. É um modelo orientado para *Wireless Sensor Network* (WSN) e pretende-se incluí-lo em aplicações de deteção de baixa potência e de longo alcance. Com um alcance de aproximadamente 1km e operando a 433MHz (uma frequência que permite uma melhor penetração do que 2,4GHz), torna-se apelativo para redes de área doméstica [8].

d. RFID e NFC

O *Radio-frequency Identification* (RFID) consiste no uso de uma etiqueta que contém dados que são lidos através da indução de um campo eletromagnético por parte de um leitor designado. Esta tecnologia pode ser utilizada no contexto das *Smart Cities* para localização de objetos, aplicações de cuidados de saúde, gestão de ativos e *Smart Parking*. As etiquetas de RFID apresentam, não só, um baixo custo para o utilizador, como também um baixo consumo de energia, ou seja, podem trazer muitos benefícios na área de IoT.

O *Near Field Communication* (NFC) é utilizado em dispositivos móveis para comunicação bidirecional de muito curto alcance. Os utilizadores de *smartphones* podem usar esta tecnologia para transferência de dados entre terceiros. A carteira digital é uma aplicação que utiliza esta tecnologia, permitindo que os seus clientes façam pagamentos, diretamente, com o seu *smartphone*. Em ambiente IoT, é possível colocar etiquetas NFC em lugares estratégicos nas habitações para detetar a presença humana e gerir os recursos consoante a necessidade (água, luz, Wi-Fi, etc.).

A tabela 2.2 faz uma comparação entre os protocolos de comunicação listados, anteriormente, para que se tenha uma maior percepção das características principais de cada um.

	Dash7	**ZigBee**	**LTE**	**3G**	**NFC**
Padrão	ISO/IEC 18007-7	IEEE 802.15.4	3GPP-LTE	Vários	ISO/IEC 18092
Frequência (MHz)	433	868/915/2400	700-2600	700-2600	13.56
Penetração	Alta	Baixa	Baixa/alta	Baixa/alta	Alta
Alcance	1 km	500 km	Vários quilómetros	Vários quilómetros	10 cm
Taxa de dados	200 Kbps	250 Kbps	100 Mbps +	3.6-21 Mbps	106-424 Mbps

Tabela 2.2: Comparação entre os protocolos de comunicação [8].

Todas estas tecnologias podem ser introduzidas em algumas aplicações no âmbito das *Smart Cities*: sistemas de distribuição de água; sistema de distribuição de

eletricidade; edifícios e habitações inteligentes; monitorização de pontes e atividade sísmica; monitorização ambiental; transportes inteligentes; *Smart Parking*; vigilância; cuidados de saúde. A tabela 2.3 permite analisar, de forma mais clara, os benefícios que estas poderão receber, caso tenham os mecanismos incorporados, e as desvantagens em querer manter os processos atuais.

	Sem as tecnologias	**Com as tecnologias**
Monitorização da Saúde Estrutural	- Custos elevados devido ao número de pessoal necessário para inspeções programadas; - Inspeção visual nem sempre é eficaz.	- Sistema de monitorização autónomo reduz custos de inspeções programadas e fornece monitorização contínua; - Permite uma análise mais precisa do estado da estrutura do que a inspeção visual.
Distribuição de água	- Custos elevados em desastres causados por deteções de fendas falhadas ou tardias.	- Permite a mitigação dos custos causados por possíveis acidentes devido a deteções de fendas tardias; - Monitorizar a qualidade da água garante que esta é segura para consumo humano.
Distribuição de eletricidade	- Medição e previsão imprecisas.	- Sensor avançado de energia permite uma medição e previsão mais precisas.
Edifícios inteligentes	- Consumo elevado de água e eletricidade.	- Redução do consumo de água e eletricidade.
Transportes inteligentes	- Controlo de tráfego ineficiente.	- Esquemas de controlo de tráfego melhorados e adaptáveis às condições de tráfego.
Vigilância	- Necessidade de um operador humano propenso a distrações.	- Deteção inteligente de situações anormais sem a necessidade de um operador.
Monitorização ambiental	- Condições perigosas, como a presença de gases perigosos, que possam ser detetadas tarde demais.	- A deteção contínua de gases garante que condições perigosas possam ser detetadas a tempo.

Tabela 2.3: Benefícios das tecnologias nas *Smart Cities* [8].

2.1.4. Exemplos de *Smart Cities*

A ideia de *Smart City* está a espalhar-se pelo Mundo inteiro e alguns países já adotaram esta visão e prosseguiram com projetos piloto [8]. A figura 2.2 demonstra alguns exemplos de novas *Smart Cities* e as suas principais características [11].

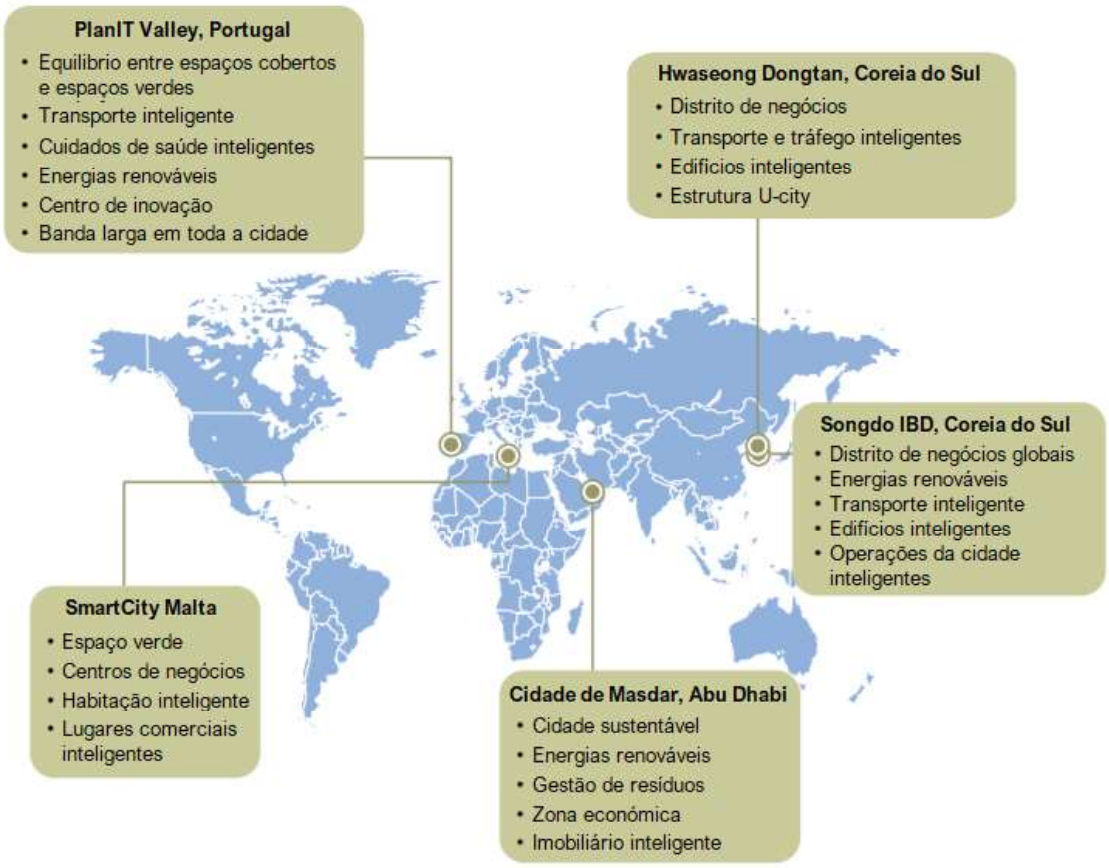

Figura 2.2: Exemplos de *Smart Cities* [11].

Estas cidades utilizam vários componentes de TIC, nomeadamente "identificação por radiofrequência (RFID), Wi-Fi, WiMax, LTE, ZigBee, comunicação máquina a máquina (M2M), gestão de ativos, gestão de energia, comunicações unificadas e colaboração" [11]. Para clarificar os aspetos mais importantes de cada cidade representada na figura 2.2, os pontos a seguir descrevem, sucintamente, esses tópicos.

a. Songdo IBD, Coreia do Sul

Esta cidade foi criada com o intuito de remover a circulação de veículos, utilizando apenas meios de transporte amigos do ambiente. Além de promover a conservação do ambiente, a cidade possui um mecanismo de recolha de lixo dentro das habitações que

facilita a reciclagem e a produção de energia. Songdo é considerada um centro de negócios globais e fomenta o crescimento económico e sustentável [8] [15].

b. Hwaseong Dongtan, Coreia do Sul

Sendo a primeira cidade a adotar o conceito de U-City [16], Hwaseong Dongtan ainda está em processo de crescimento. Ao implementar a estrutura de U-City, esta cidade introduziu uma rede de comunicação global em que os cidadãos interagem para obter informações em tempo útil. Os objetivos passam pela prevenção do crime, controlo do tráfego e estacionamento de forma inteligente [16].

c. Cidade de Masdar, Abu Dhabi

Começou a ser desenvolvida em 2008 com o propósito de vir a ser "a cidade mais sustentável do Mundo" [17]. O principal objetivo seria reduzir as emissões de carbono, com a remoção de todos os veículos. Porém, os atrasos na construção dificultaram, em certa medida, o processo e "a data de conclusão foi adiada para 2030" [17]. As grandes apostas estão viradas para o uso de energias renováveis, gestão de resíduos e investimento em empresas de tecnologia e imobiliárias [18].

d. SmartCity Malta

Esta é uma *Smart City* que aposta, fortemente, na eficiência energética, segurança, centros de negócios, turismo, comércio e entretenimento [19]. Lamentavelmente, não existe informação fidedigna disponível relativamente a esta cidade devido ao impedimento no acesso ao *website* da SmartCity Malta (www.smartcity.ae/Malta).

e. PlanIT Valley, Portugal

O PlanIT Valley é um projeto de *Smart City* desenvolvido pela empresa 'Living PlanIT' (www.living-planit.com) que pretendia implementá-lo a Este da cidade do Porto. O objetivo seria apresentar a plataforma *Urban Operating System* (UOS), constituída por sensores espalhados pela cidade que recolheriam informações, posteriormente, disponibilizadas nas aplicações destinadas para o efeito [20]. Com o apoio do município de Paredes e do Governo português, procurou-se criar um centro de inovação e de apoio a *startups* de tecnologia, com edifícios pré-fabricados e equipados com a tecnologia avançada e inteligente adequada ao âmbito [20]. Infelizmente, devido a dificuldades no financiamento, planeamento e a existência de conflitos de ideias entre os intervenientes, o projeto nunca chegou a progredir [21].

2.2. Smart Parking

2.2.1. Introdução

Dentro da gama de produtos e serviços relacionados com *Smart Cities*, o *Smart Parking* ou estacionamento inteligente tem sido alvo de algum estudo e já se observam alguns produtos desenvolvidos no mercado. A preocupação em desenvolver projetos de *Smart Parking*, além de ter como objetivo o crescimento das cidades, passa também pela necessidade de resolver problemas existentes no sistema de parqueamento atual, controlar melhor o trafego diário dentro das cidades [1], reduzir o tempo de espera na procura de um lugar de estacionamento, ajudar os indivíduos não residentes na cidade em busca de lugares de estacionamento e, consequentemente, melhorar o meio ambiente, reduzindo os gases de escape dos veículos [3].

Os estudos [1] [3] [24] sobre sistemas de *Smart Parking* começam, precisamente, por referir a necessidade de melhorar os sistemas de parqueamento atuais e o que estes sistemas trazem como benefício para o desenvolvimento das *Smart Cities*. O crescimento das cidades e o aumento do número de veículos permitiram o desenvolvimento de tecnologias relacionadas com o estacionamento, aumentando cada vez mais a necessidade de implementar sistemas de parqueamento mais avançados e inteligentes nas zonas metropolitanas das cidades [22]. O aumento do tráfego em zonas residenciais das cidades criou alguns problemas de gestão de estacionamento, devido à necessidade de estacionamento ilegal por parte dos condutores aliado à falta de lugares disponíveis, o que dificulta a circulação normal de veículos [3].

Em [24] é feita uma proposta para criação de um sistema como uma rede de IoT que auxilia os condutores a encontrar um lugar de estacionamento livre com o menor custo possível baseado em novas métricas de performance para o cálculo do custo de estacionamento, considerando a localização GPS do veículo, a distância entre as áreas de estacionamento e o número total de lugares livres num parque de estacionamento. Caso o parque esteja lotado, o utilizador é conduzido para um outro parque até este conseguir estacionar o veículo, através do algoritmo apresentado na figura 2.3.

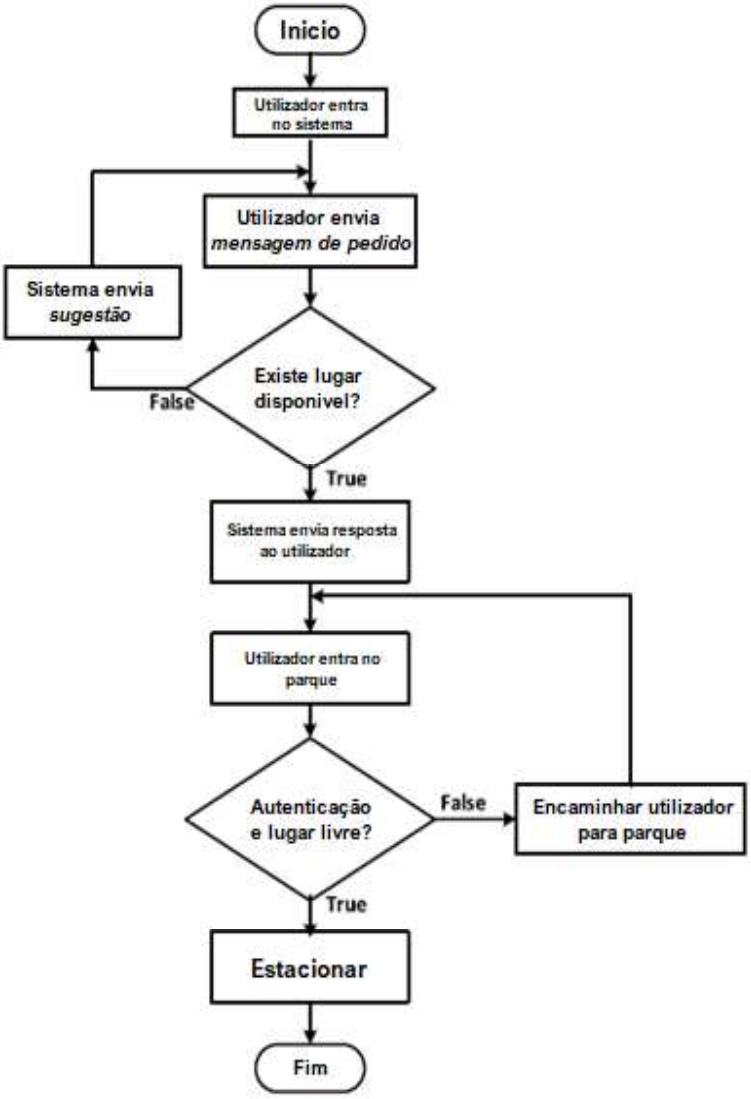

Figura 2.3: Algoritmo das operações do sistema [24].

Cada parque de estacionamento utiliza tecnologia WSN que consiste na monotorização dos parques de estacionamento através de RFID [25]. O sistema funciona em tempo-real e proporciona ao utilizador a escolha do lugar de estacionamento mais adequado, enviando, posteriormente, as direções até ao destino [24]. Sempre que um veículo entra ou sai, os dados são atualizados através da comunicação com o WSN do parque de estacionamento, tal como representado no algoritmo da figura 2.4.

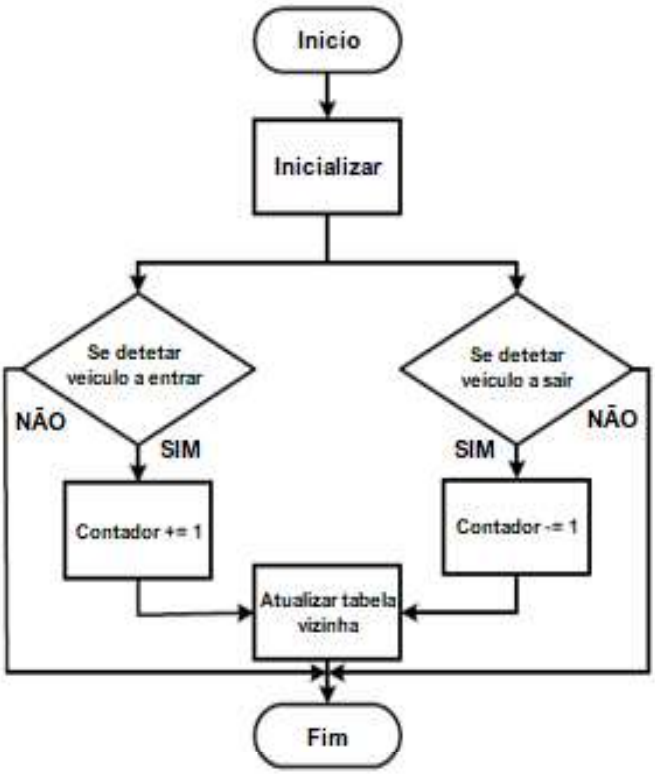

Figura 2.4: Algoritmo de atualização do estado do parque de estacionamento [24].

Qualquer proposta para a implementação de um sistema de *Smart Parking*, numa dada cidade, requer uma avaliação minuciosa em relação a alguns aspetos importantes [22]:

- O sistema em causa tem de conseguir, obrigatoriamente, detetar veículos nos lugares de estacionamento em tempo-real;
- Fornecer as direções corretas até ao destino escolhido pelo condutor;
- Criar decisões inteligentes com base nos dados recolhidos para melhorar a experiência de utilização e facultar outras alternativas ao utilizador, caso seja necessário;
- Assegurar o funcionamento total e segurança do sistema nos momentos críticos do dia, em que existe maior utilização e acessos por parte dos condutores.

Ao garantir que o sistema de *Smart Parking* a desenvolver abrange estes requisitos enumerados, poder-se-á acreditar que este será eficiente e irá satisfazer as expectativas dos utilizadores, ao longo do tempo [22].

2.2.2. Tecnologia de *Smart Parking*

2.2.2.1. Sensores

Analisando os sistemas de *Smart Parking* existentes, é possível constatar algo em comum entre eles: uma rede sensorial que comunica com um sistema central para recolha e análise de dados relativamente ao estado de cada lugar de estacionamento e, por conseguinte, o envio dessa informação aos utilizadores. Dentro das tecnologias para deteção de veículos, existem dois grupos essenciais: intrusivos e não intrusivos. Tal como o nome indica, os sensores intrusivos são aqueles que obrigam a instalação nos pavimentos e implica a alteração na estrutura dos mesmos. Os aparelhos não intrusivos são instalados facilmente e não requerem modificações estruturais nas ruas. Dos principais mecanismos existentes, temos alguns exemplos dentro dos intrusivos, tais como, "os sensores infravermelhos ativos, circuitos indutivos, sensores magnéticos, tubo de estrada pneumático, cabos piezoelétricos e sensores de peso em movimento. Dentro da gama de sensores não intrusivos, englobam-se o radar de micro-ondas, sensores infravermelhos passivos, RFID, sensores ultrassónicos e processamento de vídeo e imagem" [1].

Para perceber qual o mecanismo mais adequado para um sistema de *Smart Parking* viável é preciso, em primeiro lugar, averiguar quais os possíveis mecanismos à disposição e, neste caso, que se adequam melhor ao âmbito da dissertação. A lista seguinte de sensores descreve alguns dos exemplos enumerados, previamente, de forma a compreender melhor as características de cada um [1].

a. <u>Sensor infravermelho ativo</u>

O sensor infravermelho ativo consegue detetar um objeto emitindo uma energia infravermelha e, em seguida, medindo a quantidade de energia que foi refletida. Porém, é também sensível ao ambiente exterior e condições atmosféricas.

b. <u>Detetores de tráfego com circuito indutivo</u>

Estes dispositivos permitem detetar veículos a partir de um circuito isolado instalado no pavimento. Ao aproximar-se, por exemplo, de um semáforo, a viatura causa um distúrbio no circuito indutivo e, consequentemente, deteta a sua presença. Esta poderá ser uma alternativa em sistemas de parqueamento para controlar o acesso e ocupação de veículos num parque de estacionamento.

c. Sensor magnético

Os sensores magnéticos destacam-se pela indução de um campo magnético que deteta alterações no fluxo através da passagem de materiais ferromagnéticos existentes, por exemplo, em veículos. Devido ao facto deste sensor ser insensível às condições atmosféricas e impedir a deteção de elementos não metálicos, faz com que esta seja uma solução bastante interessante para a implementação de uma rede sensorial para um sistema de *Smart Parking*.

d. Sensor infravermelho passivo

Estes sensores controlam o estado de ocupação num parque de estacionamento através da deteção de mudanças na energia emitida pelo veículo e pelo chão. Embora estes sensores funcionem de forma eficaz em condições meteorológicas normais, o mesmo não acontece em condições agravadas, tais como chuvas fortes, neve e nevoeiro. Portanto, este tipo de sensor é utilizado para deteção de veículos em alguns parques de estacionamento de acesso público.

e. Processamento de vídeo e imagem

Esta é uma tecnologia que utiliza processamento de imagem e vídeo para a leitura dos dados de fotografias obtidas pela câmara, retirar os números da matrícula e enviar para a base de dados. Pode ser usada em parques de estacionamento para controlar as entradas, evitando visitantes não desejados ou que pudessem entrar no espaço com possível comportamento impróprio ou ação criminosa.

2.2.2.2. Gateways

As *gateways* são os elementos de um sistema de *Smart Parking* que fazem a ligação entre os sensores de estacionamento associados à rede e os servidores através da Internet. Estas recolhem todas as informações essenciais dos sensores, possibilitando o funcionamento em tempo-real do sistema. Estes dados definem tanto o estado do lugar de estacionamento como o próprio estado do sensor em relação ao nível de energia, permitindo que haja um controlo total sobre os elementos físicos do sistema e prosseguir com a manutenção dos mesmos, caso seja necessário.

No mercado, existem algumas opções relativamente às *gateways*. Também é possível desenvolver, integralmente, o dispositivo. Todavia, são necessários certos cuidados, tanto em termos de *software*, como também de *hardware*. Para o sistema operativo,

pede-se um *software* simples, fluído e seguro. Isto permite que a *gateway* seja o mais autónoma possível e, dessa forma, se possam utilizar soluções energéticas mais adequadas e sem consumos exagerados de energia. O nível de segurança do sistema determina a probabilidade de acesso aos dados por via de terceiros e garante a integridade dos mesmos durante o funcionamento global do sistema. O crime informático é uma realidade nos dias de hoje e é imperativo ter controlo sobre os dados que são, posteriormente, disponibilizados aos utilizadores. O fornecimento de informações alteradas pode equivocar os condutores no momento em que estes acedem à aplicação em busca de lugares disponíveis numa certa zona. Em termos de estrutura física, é preciso ter atenção à qualidade do material usado no fabrico do dispositivo. As condições meteorológicas podem danificar facilmente os aparelhos eletrónicos e, por isso, é obrigatório proteger o dispositivo contra a humidade, temperaturas elevadas e reduzidas, pressão e ventos fortes. Estes são os pontos principais a analisar antes de tomar uma decisão sobre o produto a adquirir ou para a construção do dispositivo. Os pontos a seguir representam uma lista de produtos existentes no mercado e que garantem o funcionamento adequado desta parte dos sistemas de *Smart Parking*, possuindo as características referidas.

a. Fastprk *Gateway*

O sistema Fastprk possui todos os elementos necessários para a implementação de um sistema de *Smart Parking*. A frequência de operação é de 868MHz na União Europeia, 902MHz nos Estados Unidos da América e 868.8MHz na Rússia. Possui uma área de comunicação de 1km^2. O fornecimento de energia pode ser feito por uma de duas vias:

1º − Através da tecnologia PoE (*Power over Ethernet*) descrita pelo padrão IEEE 802.3af, que permite receber energia elétrica juntamente com os dados provenientes do cabo ligado à rede Ethernet. A tensão elétrica recebida é de 48V e pode atingir um máximo de 15W de potência;

2º − Utilização de uma fonte de energia de corrente contínua. Por exemplo, bateria recarregável, painel solar, etc.

Contém uma entrada para ligação à rede através de Ethernet e também possibilita a conexão à Internet por meio de um *modem* 3G e antena (HSDPA, EDGE, GPRS) *quad band*.

O sistema operativo utilizado é o Linux 3.10.33. Possui um processador ARM v5 ou ARM 9 e uma memória RAM de 128MB.

O dispositivo é avaliado em IK08 no que toca à resistência de impacto, o que significa que o aparelho poderá permanecer intacto caso este seja largado a mais ou menos 1 metro de distância do chão. A temperatura de funcionamento e armazenamento está entre os −30ºC e os 70ºC, contando também com uma resistência total à humidade. Este possuiu ainda 21cm de largura, 31cm de altura, 17cm de espessura e um peso de 2kg [26].

Figura 2.5: *Gateway* do produto Fastprk [26].

b. SmartSpot Gateway

A SmartSpot Gateway é um produto de IoT altamente flexível que permite a implantação de sistemas de *Smart Parking* e atua como a unidade de comunicação central. O dispositivo recolhe os dados dos sensores de estacionamento através de comunicação sem-fios e transmite-os ao sistema de serviços de estacionamento SmartCloud. A possibilidade de utilização de protocolos de comunicação, tais como Ethernet, Wi-Fi, Zigbee 802.15.4, *Long Range Wide-Area Network* (LoRaWAN), 3G, 4G e, futuramente, 5G, faz com que o SmartSpots também possa atuar como uma plataforma de *gateway* IoT convencional. A configuração e atualizações de *firmware* são efetuadas através de uma ferramenta *Over-The-Air* (OTA) [27].

Este dispositivo pode ser alimentado de três formas distintas:

1º − Energia elétrica proveniente dos postes de eletricidade: 90V a 264V de corrente alternada;

2º − Bateria recarregável;

3º − Painéis solares externos.

Existe, ainda, a possibilidade do uso da tecnologia PoE. Contudo, poderão existir problemas durante o funcionamento do aparelho. À semelhança do dispositivo anterior, este também apresenta uma alta resistência ao impacto e às condições atmosféricas extremas (IP67+).

Figura 2.6: Características da *gateway* do produto SmartSpot [27].

c. Meshlium

A Libelium possui uma larga gama de produtos inerentes à IoT e o Meshlium é a *gateway* que permite a comunicação entre os próprios componentes, assim como de outros fora do leque de produtos da Libelium. Sem dúvida, uma grande solução tecnológica para integração em sistemas de *Smart Parking*. Porém, a sua versatilidade poderá resultar num preço demasiado elevado em relação ao espectável.

Com um processador de 4 núcleos e 1GHz de velocidade, 2GB de memória RAM e 16GB de memória interna, o Meshlium tem a total capacidade de processar grandes quantidades de dados provenientes dos sensores de estacionamento e enviá-los, rapidamente, para o servidor. O sistema operativo é baseado em Linux (mais concretamente, Debian) e o *software* de gestão é livre. Do ponto de vista da segurança,

o sistema contém autenticação WEP, WPA, WPA2 e HTTPS. O aparelho pode ser alimentado por PoE, corrente alternada com 220V de tensão ou corrente contínua com 12V. Relativamente ao aspeto físico, o Meshlium é envolvido numa estrutura de alumínio, um peso de 2,2kg, 30cm de largura, 22cm de altura e 8,7cm de espessura. Tem um grau de proteção IP65, ou seja, é bastante resistente à poeira e a jatos de água pouco potentes, e também suporta temperaturas entre −20ºC e 50ºC. Para a ligação à Internet, é possível fazê-lo através de Ethernet, Wi-Fi ou dados móveis com protocolos 4G, LTE, 3G, WCDMA, HSPA, UMTS, GPRS e GSM. Existem três opções de escolha para o módulo de radiofrequência usado para comunicar com os sensores de estacionamento:

1º − Modelo XBee-PRO 802.15.4, frequência de 2,4GHz e um alcance de 1,6km (750 metros nos modelos da União Europeia);

2º − Modelo XBee 868LP, frequência de 868MHz e um alcance de 8,4km;

3º − Modelo XBee-PRO 900HP, frequência de 900MHz e um alcance de 15,5km [28].

Figura 2.7: Meshlium – *Gateway* da gama de produtos da Libelium [28].

2.2.2.3. Backend

Dentro do conjunto de serviços de *backend* existentes, podemos contar com um grupo que possui a segurança e armazenamento necessários para o bom funcionamento de sistemas de IoT. É claro que, em regra, este tipo de servidores requer um investimento que aumenta consoante as características internas dos mesmos.

Felizmente, temos uma enorme variedade de serviços de *backend* que se ajustam às necessidades de cada produto que surja no mercado, nomeadamente, os *Mobile Backend-as-a-Service* (MBaaS), que oferecem serviços de *backend* com armazenamento em *cloud* e *Application Programming Interface* (API). Além dos serviços principais, os MBaaS ainda possibilitam o envio de notificações, análise em tempo-real da performance das aplicações, monetização e outros [29]. Este ponto permite apenas enumerar alguns sistemas existentes e dos mais pretendidos ao longo dos anos.

a. Kinvey

A empresa Kinvey foi fundada em 2010 e, recentemente, lançou uma ferramenta para empresas que ajuda a gestão externa e interna das equipas de desenvolvimento. Caso se pretenda experimentar o serviço, a Kinvey oferece uma versão gratuita e limitada para esses utilizadores. Os preços variam consoante o número de utilizadores ativos e poderão rondar os 200US$ e os 1500US$ por mês.

b. Kii

A Kii é uma empresa japonesa focada em IoT e suporta o desenvolvimento, testes e aquisição de utilizadores dentro de um único ambiente de trabalho. O envio de notificações, análise das aplicações, localização geográfica e monetização estão, igualmente, incluídos no pacote de serviços da Kii.

c. Built.io

A Built.io é uma solução de desenvolvimento de aplicações MBaaS que permite aos clientes colocar aplicações móveis em ambientes de *cloud* privados, públicos ou híbridos. São disponibilizados três planos de pagamento: Padrão, Dedicado (para armazenamento em *cloud* privada) e Estudantes. Os preços começam, para 5000 utilizadores, em 99US$ por mês, atingindo os 399US$ por mês para a mesma quantidade de utilizadores, mas com maior número de recursos.

d. Parse

A Parse foi adquirida pelo Facebook em 2013 e oferece três serviços principais:

1. Parse Core: serviço básico de MBaaS, permitindo armazenar dados de forma segura, ligação através das redes sociais e outros;
2. Parse Push: serviço de notificações;

3. Parse Analytics: análise das métricas dos utilizadores e das próprias aplicações.

O Parse Core é oferecido, gratuitamente, para período de experimentação com 30 acessos por segundo. Quando se chega a 80 acessos por segundo, o preço aumenta para 500US$ e 210 pedidos por segundo chega aos 1800US$ por mês.

e. Firebase

O Firebase é um serviço BaaS adquirido pela Google cujo objetivo é oferecer um serviço de *backend* que ajude os programadores a desenvolver as suas aplicações mais rapidamente. Esta plataforma possui um serviço de análise de aplicações, notificações, base de dados em tempo-real, armazenamento, etc. A API utilizada para a chamada às funções de inicialização e execução da plataforma é bastante intuitiva e facilita imenso a integração nas aplicações móveis. O Firebase apresenta três planos de preços [30]:

1. Spark Plan: Grátis, mas limitado a 1GB de conteúdo na base de dados em tempo-real e 5GB de armazenamento, bem como o número de acessos simultâneos;
2. Flame Plan: 25US$ por mês, com um aumento de armazenamento nos serviços referidos em relação ao Spark Plan;
3. Blaze Plan: O preço é calculado consoante a utilização dos serviços.

2.2.2.4. Desenvolvimento de aplicações móveis

Hoje em dia, temos múltiplas hipóteses de desenvolver aplicações e, felizmente, maior parte delas são gratuitas. É possível utilizar o *software* recomendado pelas plataformas e, no caso da Apple, é mesmo obrigatório o uso do programa XCode para compilar e publicar as aplicações. O Android Studio é um *software* bastante intuitivo que facilita imenso o trabalho dos programadores. O estilo *drag and drop* para a organização do *layout* não é algo que funcione a 100% no Android Studio. Porém grande parte dos programadores Android prefere manipular os elementos da aplicação através da escrita de código. O XCode facilita um pouco mais esta abordagem de *drag and drop*. Atualmente, vemos inúmeras alternativas para o desenvolvimento de aplicações multiplataformas, isto é, desenvolver para Android e iOS através do mesmo código e linguagem. Contudo, ainda não existe uma *framework* que junte ambas as plataformas de forma perfeita. Será, sempre, necessário verificar alguns aspetos individuais, durante o desenvolvimento, que estejam apenas ligados àquela plataforma. Ao longo do tempo, observamos que as *frameworks* trabalham fortemente para anular

estas lacunas e os programadores cada vez mais optam por escolher uma *framework* de desenvolvimento híbrido. Existem inúmeras opções, porém a lista abaixo apresenta quatro das *frameworks* mais utilizadas e em crescimento nos últimos anos [31].

a. Xamarin

A Microsoft é proprietária do *software* Xamarin, que contém implementações multiplataforma da *Command Line Interface* (CLI) e *Common Language Specification* (CLS) para implementações em Microsoft .NET.

A utilização da linguagem C# para desenvolvimento de aplicações nativas de Android, iOS e Windows, através de código compartilhado, é uma grande vantagem no que toca ao tempo economizado e ferramentas utilizadas durante a fase de construção.

b. PhoneGap

Esta é uma *framework* multiplataforma bastante popular para desenvolvimento de aplicações híbridas móveis. O Phonegap permite reutilizar as funcionalidades de desenvolvimento *Web* para a criação de aplicações híbridas a partir do uso de HTML, CSS e JavaScript para as plataformas com um único código-fonte.

c. Ionic Framework

A Ionic Framework é um *Software Development Kit* (SDK) de código aberto para o desenvolvimento de aplicações móveis híbridas. Este utiliza Angular.js e Apache Cordova, fornecendo ferramentas e serviços para desenvolver aplicações móveis híbridas através de tecnologias *Web* como CSS, HTML5 e Sass (definição de variáveis apenas uma vez, podendo ser utilizadas em múltiplos ficheiros do projeto).

d. React Native

A React Native é uma *framework* gerida pelo Facebook para criação de aplicações móveis multiplataforma e utiliza JavaScript como linguagem de programação. Esta *framework* ainda está em crescimento. Porém o uso do mesmo código para o desenvolvimento de aplicações em ambiente Android e iOS é uma vantagem clara para grande parte dos programadores e empresas que pretendem contratá-los, sem necessidade de encontrar alguém específico para uma dada plataforma [32].

e. Sencha

A Sencha é uma empresa que apresenta soluções para desenvolvimento de aplicações móveis através do uso de tecnologias *Web*, tais como HTML5 e JavaScript. O grande objetivo desta empresa é oferecer aos seus clientes os serviços necessários para o desenvolvimento rápido dos seus produtos e proporcionar uma solução MBaaS para testes e gestão de aplicações desenvolvidas em HTML e JavaScript. A Sencha oferece um período experimental de 30 dias [33].

f. Appcelerator

A Appcelerator oferece uma solução de desenvolvimento de aplicações multiplataforma. O Titanium SDK é uma *framework* aberta e utiliza JavaScript como linguagem de programação, comum no desenvolvimento de aplicações em todas as plataformas. Esta empresa também oferece um serviço MBaaS com base de dados, armazenamento de ficheiros, notificações e uma API disponibilizada para chamada a funções de acesso. A Appcelerator dispõe de três planos para obtenção de funcionalidades de desenvolvimento de aplicações:

1. Indie: Grátis, oferece as ferramentas base para programadores particulares que pretendem começar a desenvolver as próprias aplicações;
2. Pro: 99US$ por mês, direcionado a empresas de desenvolvimento de aplicações móveis em crescimento. Para além das ferramentas oferecidas no plano Indie, também contém *software* de apoio ao desenho e testes *beta* das aplicações, assim como suporte às equipas de desenvolvimento por *e-mail* ou *chat*;
3. Enterprise: Preço customizado consoante as necessidades, possui as mesmas ferramentas que o plano Pro e ainda organização de dados, aplicações e utilizadores, deteção de falhas e performance, e automatização de testes.

Para os serviços de *backend*, a Appcelerator também contém algumas opções, destacando-se pela sua elasticidade conforme as necessidades das equipas de desenvolvimento e das próprias aplicações [34].

2.2.3. Sistemas e aplicações de *Smart Parking* existentes

Este ponto descreve alguns sistemas de *Smart Parking* existentes, bem como aplicações móveis disponíveis na Europa. Pretende-se obter algumas informações sobre os mecanismos utilizados nestes sistemas para compreender as principais tecnologias envolventes.

2.2.3.1. Sistemas de *Smart Parking*

a. ParkAnyWhere

Esta *startup* portuguesa trouxe um conceito e um modelo de negócio bastante interessantes no que toca ao estacionamento numa cidade. Os fundadores desta empresa também foram alertados para o problema de estacionamento em Lisboa, tendo construído um projeto que, apesar de ser cativante, dentro do seu âmbito, não apresenta as características essenciais que um projeto de *Smart Parking* exige. O sistema baseia-se na reserva de uma garagem privada num qualquer ponto da cidade, anunciada através do *website* da ParkAnyWhere, e a rede de garagens aderentes é apresentada na aplicação que permite a reserva por parte dos utilizadores. Isto é, este projeto depende, em muito, da adesão de clientes e, bem assim, que estes disponibilizem os seus parques privados, possibilitando, dessa forma, o surgimento de vantagens no estacionamento em Lisboa. O acesso a lugares de estacionamento na via pública ou de acesso público não é permitido nesta aplicação, o que não se apresenta como algo útil para o caso de estudo em progresso.

b. ParkWhiz

O ParkWhiz foi fundado por Aashish Dalal em 2006 e disponibiliza informação sobre o estacionamento em algumas cidades nos EUA, tais como Chicago, Nova Iorque, Boston, São Franscisco e Washington [35]. Os utilizadores podem instalar a aplicação disponível para Android e iOS com o objetivo de encontrarem lugares de estacionamento distribuídos pela cidade. Ao efetuarem o pagamento, os condutores recebem um bilhete eletrónico para que possam entrar no parque de estacionamento. Sendo este um conceito bastante apelativo, é um sistema semelhante à ideia transmitida pela ParkAnyWhere, porém a uma grande escala, causando maior impacto no que toca ao controlo do estacionamento nas grandes cidades.

c. Parkopedia

A empresa fundada por Eugene Tsyrklevich em 2009 anunciou em julho de 2016 que, em parceria com a Apple, vai fornecer os seus dados relativos aos parques de estacionamento através de AppleMaps. Isto significa que os utilizadores de iOS poderão encontrar mais de 40 milhões de lugares de estacionamento em 75 países situados na América do Norte, Europa, Ásia e América Latina. O AppleMaps permite aos

utilizadores procurar parques e garagens de estacionamento, informando-os da sua localização, método de pagamento, número de lugares e mais. Este sistema oferece, ainda, a possibilidade de reservar um lugar de estacionamento através do fornecimento de um *link*, que direciona os utilizadores para a página da Parkopedia, ou para efetuar a instalação da aplicação que possui outras informações complementares, designadamente preços, comentários de utilizadores, ofertas especiais, lugares disponíveis em tempo-real, permitindo, ainda, o pagamento da taxa de estacionamento. A reserva de um lugar dentro do parque de estacionamento é conseguida através do envio de um passe de estacionamento por *e-mail*, após o pagamento da taxa. Ou seja, este sistema não permite a reserva de um lugar específico dentro da cidade, mas sim dentro de um parque de estacionamento privado ou de acesso publico. É uma abordagem que traz vantagem no acesso dos utilizadores a parques de estacionamento de acesso público tarifados. Todavia, a reserva de lugares na via pública pode trazer alguns problemas de gestão de veículos que se encontrem fora da rede de estacionamento.

d. SmartParking Systems

Esta empresa surgiu em 2004 para oferecer uma solução que torne a experiência de estacionamento mais agradável através da recolha e armazenamento de dados em tempo-real [36]. A solução consiste num sistema de navegação que revela a disponibilidade de zonas de estacionamento e encaminha os condutores em direção à vaga mais próxima do destino final estabelecido. Dentro deste sistema, existe uma rede de sensores sem-fios, subterrâneos e magnéticos de 3 eixos que enviam as alterações de estado para um sistema central. São alimentados por uma bateria com autonomia até 10 anos e comunicam através de um sinal de rádio com um protocolo encriptado. Os sensores de estacionamento não são afetados pelas condições atmosféricas (entre −40ºC e 80ºC), sendo completamente invisíveis em superfícies pavimentadas, asfalto, pedras de pavimentação e outros tipos de superfície, evitando problemas para a limpeza das ruas e vandalismo. O sistema SmartParking utiliza a tecnologia LoRaWAN, capaz de conectar sensores a longas distâncias, exigindo uma estrutura mínima, entregando a melhor vida útil da bateria. Isso oferece vantagens como mobilidade, segurança e localização/posicionamento otimizados, bem como redução de custos. Os sistemas LoRaWAN são capazes de comunicar a distâncias de mais de 100km em ambientes favoráveis, 15km no meio rural e para mais de 2km em ambientes urbanos densamente povoada a uma velocidade entre 27 e 50 Kbps [37].

A aplicação para *smartphones* e *tablet* permite que sejam consultadas, em tempo-real, as vagas de estacionamento e auxilia os motoristas a escolher o melhor local, sem terem de se deslocar de um lado para o outro para verificar as vagas. O tempo desperdiçado na busca de lugares disponíveis é eliminado; o utilizador ganha tempo e o tráfego *parasita* na cidade é aliviado. O utilizador pode pagar o estacionamento de forma fácil, usando a aplicação que está associado a um cartão de crédito.

Este sistema contém as características essenciais para o estudo do sistema de *Smart Parking* dentro do âmbito desta dissertação. Todos os componentes principais estão bem descritos, no entanto não existe uma pormenorização em relação à estrutura do sensor de estacionamento e preço por unidade.

e. Fastprk

A empresa WorldSensing foi fundada em 2008 na cidade de Barcelona e engloba três setores no mercado: *Smart Cities* com o produto Fastprk, Smart Infrastructures e Smart Resiliency.

O Fastprk é um sistema de *Smart City* que ajuda os condutores a encontrar um lugar de estacionamento de forma rápida, possibilitando uma maior eficiência na gestão das zonas de estacionamento [26]. O sistema informa os condutores, via *smartphones* ou painéis eletrónicos nas ruas, da existência de lugares vagos numa certa zona de estacionamento. Também é permitido efetuar o pagamento através de uma plataforma móvel que, por sua vez, está ligada ao sistema, fazendo com que a cidade obtenha mais receitas.

O sistema é composto por uma rede de sensores de estacionamento que detetam a presença de um veículo através de um campo magnético e enviam um sinal ao *gateway* mais próximo, que, por sua vez, envia a informação à base de dados em tempo útil através da Internet. A ocupação do lugar de estacionamento é reportada imediatamente aos utilizadores através de aplicações e dos painéis eletrónicos nas ruas. Caso exista um estacionamento, o sistema controla todo o processo, informando através de uma aplicação para *tablet*s fornecidos às autoridades responsáveis por controlar as zonas de estacionamento e o pagamento por parte dos condutores.

O sensor de estacionamento suporta uma temperatura mínima e máxima de −30ºC e 70ºC, uma humidade entre 0% a 100%, possui um alcance de 500 metros até ao

gateway mais próximo e apresenta uma altura de 13cm, um diâmetro de 8cm e um peso de 315g.

A possibilidade de instalação dos sensores de estacionamento debaixo do asfalto garante que estes sejam impermeáveis, que suportem 60 toneladas de peso e impede o vandalismo e, assim, não existe a necessidade de reparação constante. A não utilização de cabos permite que o sistema seja escalável. Os sensores de estacionamento são de baixo consumo de energia, alimentados por uma bateria de autonomia até 4 anos e a sua eventual substituição passa pelo uso de uma chave especial de acesso.

O Fastprk é um sistema bastante interessante em termos conceptuais. Apresenta bons argumentos na descrição do seu produto. Porém pode conter alguns problemas relativamente ao sensor de estacionamento, pois este pode não estar totalmente protegido do vandalismo e a sua autonomia e manutenção não são muito convincentes.

f. CivicSmart

A CivicSmart é uma empresa que adquiriu, em 2015, a Duncan Parking Technologies, uma empresa de gestão de parqueamento nos EUA. O seu sistema de gestão de parqueamento é baseado em *cloud* que obtém todas as informações necessárias para o funcionamento correto do sistema. Além do sistema de parquímetros já implementado, a CivicSmart desenvolveu um sistema de sensores sem-fios, de baixo custo, alimentados por baterias e resistentes às condições atmosféricas adversas, para deteção de veículos que comunicam com *gateways* alimentadas por energia solar. É possível montar estes sensores de estacionamento debaixo ou na superfície do pavimento, em postes ou nos próprios parquímetros, o que permite maior precisão na deteção de veículos e tempo de espera reduzido. O sistema inclui, ainda, um sistema de gestão que auxilia os administradores de estacionamento a monitorizar as zonas de estacionamento e gerar relatórios [38].

A descrição do produto apresentada pela CivicSmart no seu *website* não contém pormenores sobre a especificação dos sensores de estacionamento, no entanto demonstra garantias no funcionamento correto do sistema e na integração do mesmo em sistemas de parqueamento existentes, conseguindo, assim, criar um produto versátil sem prejudicar os utilizadores e os responsáveis de parqueamento acostumados aos sistemas anteriores.

2.2.3.2. Aplicações de *Smart Parking*

a. Aplicação 'Smart Parking (ZTE G&E)' para a Roménia e Hungria

Esta aplicação apresenta um desenho muito simples, com apenas três ecrãs principais:

- Mapa (API da 'Google Maps') com a zona de *Smart Parking* indicando o número de lugares disponíveis (figura A.1.1 em anexo);
- Opções de pagamento do lugar após o estacionamento (figura A.1.2 em anexo);
- Perfil do utilizador, com matrícula do veículo e nacionalidade (figura A.1.3 em anexo).

Ao escolher uma zona de estacionamento, a aplicação permite mostrar ao utilizador quantos lugares de estacionamento existem e indicar as direções até ao local, através da aplicação 'Google Maps'. Também é indicado, previamente, o preço da tarifa de estacionamento para informar os custos de parqueamento ao utilizador.

Não é possível avaliar a aplicação, do ponto de vista do correto funcionamento dos sensores de estacionamento. Ainda assim, a aplicação apresenta os elementos principais para funcionar corretamente.

b. Aplicação 'SmartParking Warsaw' para a cidade de Varsóvia, Polónia

A aplicação abre, inicialmente, o mapa da cidade de Varsóvia e mostra os contornos do que parece ser a zona da cidade que possui o sistema de *Smart Parking* em produção. Mais uma vez, esta aplicação usa a API do 'Google Maps' como base. O ecrã inicial contém uma barra superior onde é possível fazer pesquisa de zonas de estacionamento através da introdução de uma morada (figura A.2.1 em anexo). Também é possível fazer pesquisa por voz e a barra ainda inclui um botão de menu lateral que possui algumas opções tais como pesquisa avançada, favoritos, vista do mapa em modo terreno ou satélite, definições e formulário de contacto (figura A.2.2 em anexo). Ainda no ecrã principal, existem dois botões em baixo: informação no lado esquerdo e pesquisa avançada no lado direito. Testando a pesquisa avançada (figura A.2.3 em anexo), ocorre um erro, pois parece que a aplicação só permite fazer pesquisa caso o utilizador esteja perto das zonas de estacionamento.

c. Aplicação '*Smart Parking* Dubrovnik' para a cidade de Dubrovnik, Croácia

Ao abrir a aplicação, o utilizador é confrontado com duas opções de escolha (figura A.3.1 em anexo):

- *'Drive to nearest spot'*;
- *'Browse nearby spot'*.

Ao clicar na primeira opção, a aplicação 'Google Maps' é iniciada, apresentando o trajeto mais curto desde a localização do utilizador até ao lugar de estacionamento disponível mais próximo, dentro da rede de *Smart Parking* inserido na aplicação.

Escolhendo a segunda opção, a aplicação apresenta, também, um mapa, através do uso da API do 'Google Maps', onde indica as zonas de estacionamento a partir de *markers* que contêm o número de lugares disponíveis na zona (figura A.3.2 em anexo). Em baixo, mostra um botão a vermelho *'Filter Parking'* que dá a opção ao utilizador de filtrar o tipo de parqueamento que pretende: garagem, parque de estacionamento ou na rua (figura A.3.4 em anexo). O utilizador pode escolher as opções que pretende. Ao clicar num *marker*, é apresentado um ecrã com as informações de parqueamento: rua, preço, horas de funcionamento (figura A.3.3 em anexo). No lado direito deste ecrã, existem dois botões: *'Go'* e *'Pay'*. O botão *'Go'* redireciona o utilizador para a aplicação 'Google Maps' que indica o trajeto desde a sua localização até ao lugar de estacionamento selecionado. Ao clicar *'Pay'*, é pedido ao utilizador para indicar o número de matrícula para efetuar o pagamento.

d. Aplicação 'SmartPark Pozuelo' para a cidade de Pozuelo, Espanha

Ao abrir a aplicação, repara-se, imediatamente, no escasso conteúdo que esta apresenta. Inicialmente, mostra, apenas, um mapa através da API do 'Google Maps', que preenche todo o ecrã (figura A.4.1 em anexo). No canto superior esquerdo existem dois botões: pesquisa (ícone geral de pesquisa) e informação (ícone geral de informação). Ao clicar no botão de informação, é apresentada uma caixa ilustrativa do que representam os diferentes *markers* incluídos no mapa da aplicação (figura A.4.5 em anexo). Na opção de pesquisa, o utilizador pode procurar os lugares de estacionamento por zona, rua e lugares específicos: Cargas e Descargas ou Lugares para Incapacitados (figura A.4.3 em anexo). Ao efetuar a procura, o mapa apresenta a zona escolhida com vários

markers que representam cada lugar em específico (figura A.4.2 em anexo). O utilizador faz a escolha do lugar pretendido e a aplicação desenha o trajeto desde a sua localização até ao lugar de estacionamento (figura A.4.4 em anexo). No canto inferior direito, aparece um botão que abre a aplicação 'Google Maps' e mostra o percurso até ao lugar de estacionamento. Ao contrário das outras aplicações, esta não permite efetuar o pagamento da taxa de estacionamento.

2.2.4. Comparação entre os sistemas e aplicações de *Smart Parking*

Numa primeira análise aos sistemas de *Smart Parking* referidos na dissertação, verificam-se algumas conformidades em relação às especificações apresentadas. Importa aqui realçar as características principais dos sensores de estacionamento: as tecnologias utilizadas para a deteção de veículos e comunicação entre os componentes do sistema, a resistência às condições atmosféricas, o fornecimento de energia para alimentação dos aparelhos (autonomia) e, por conseguinte, o custo por lugar de estacionamento inerente ao investimento nos próprios sensores e gastos na instalação e manutenção. Portanto, devido à clara importância que esta componente detém num sistema de *Smart Parking*, os critérios de comparação aplicados, no seguimento deste ponto, são os enumerados, anteriormente, na frase respeitante às propriedades fundamentais dos sensores de estacionamento.

Sistemas de *Smart Parking*	**Tecnologias**	**Resistência**	**Alimentação**	**Instalação**
SmartParking Systems	- Sensor magnético de 3 eixos; - LoRaWAN	Temperaturas: −40ºC a 80ºC	Bateria: autonomia até 10 anos[b]	Subterrâneo: invisíveis e não sujeitos a vandalismo
Fastprk	- Sensor magnético; - Alcance: 500 metros até ao *gateway*	- Temperaturas: −30ºC a 70ºC; - Humidade: 0% a 100%	Bateria: autonomia até 4 anos[b]	Subterrâneo: invisíveis e não sujeitos a vandalismo
CivicSmart	*Wireless*[a]	Todas as condições atmosféricas[a]	Bateria: autonomia não especificada[b]	Subterrâneo ou superficial: pavimento, postes ou parquímetros

a. Não especificado.
b. Não existem garantias quanto à veracidade destes dados.

Tabela 2.4: Comparação entre os sistemas de *Smart Parking* existentes.

Examinando a tabela 2.4, conclui-se que, no que respeita às características relacionadas com a resistência, alimentação e instalação dos sensores de estacionamento dos sistemas de *Smart Parking* considerados, estes apresentam propriedades bastante semelhantes. A autonomia das baterias é uma incógnita, pois não foi possível garantir que as informações reveladas são verdadeiras. Os três sistemas asseguram que os sensores de estacionamento podem ser subterrâneos, evitando obstruções nas ruas, vandalismo e suportam condições atmosféricas agressivas. Relativamente às tecnologias, todos os sistemas dedicaram-se ao uso de sem-fios para a comunicação e a deteção dos veículos é feita por meio de sensores magnéticos, ao contrário do sistema CivicSmart que não entrou em detalhes em relação a este tópico. Os custos para aquisição dos sensores de estacionamento não foram revelados pelas empresas. A instalação e manutenção são despesas obrigatórias e não é possível fazer um cálculo minucioso dos valores.

As empresas SmartParking Systems e Fastprk apontam as suas soluções de *Smart Parking* como algo construído para os cidadãos e não para as empresas de controlo de estacionamento. Ou seja, os sistemas apresentados não impedem a integração de outros já existentes. Todavia, não referem, com clareza, a versatilidade do seu produto. A empresa CivicSmart garante facilidade na incorporação em sistemas de parqueamento atuais e isso pode ser um ponto positivo a considerar para um trabalho futuro.

As aplicações móveis diferenciam-se, principalmente, pelo desenho e pela forma como estas revelam o conteúdo aos utilizadores. A base de uma aplicação de *Smart Parking* é o mapa que sinaliza, através de *markers*, as zonas de estacionamento e o número de lugares disponíveis. A aplicação 'SmartPark Pozuelo' coloca um *marker* para cada lugar no mapa, indicando, individualmente, a sua disponibilidade, ao invés das restantes, que apresentam um *marker* respeitante a cada zona de estacionamento. A ideia de utilizar o método de separar cada lugar no mapa é bastante interessante. Contudo, poderá dificultar o processamento e a fluidez dos dados dentro da aplicação, tornando-a mais lenta em dispositivos limitados. As restantes funcionalidades são altamente similares, desde a pesquisa de zonas de estacionamento até à condução dos utilizadores ao local. As grandes diferenças residem no desenho global das aplicações e em uma ou outra funcionalidade, não sendo significativo para o que pretende: procurar zonas e lugares de estacionamento livres, e orientar os condutores até ao destino.

2.3. Sistemas de estacionamento existentes em Lisboa

Os atuais sistemas de estacionamento em Lisboa caracterizam-se por zonas na via pública onde é possível estacionar um veículo e que pode ser, ou não, tarifada. Caso seja tarifada, o pagamento é obrigatório para que se possa usufruir do lugar de estacionamento e é controlado pela empresa EMEL que, em parceria com a CML, é responsável pela gestão das zonas de estacionamento designadas. Inicialmente, o pagamento era apenas efetuado através de parquímetros, mas, recentemente, foi criada uma aplicação móvel que permite o pagamento da tarifa mais facilmente. Os restantes parques de estacionamento de acesso público podem usar um dos seguintes sistemas de parqueamento:

- Zeag: Sistema de parque com barreira em banda magnética ou código de barras muito robusto ideal para parques de média e grande dimensão. Sistema modular com diversas funcionalidades, nomeadamente via verde, pré-reserva, pagamentos com cartão de débito/credito, cartões sem contacto, leitura de matrículas, entre outras [39];
- FAAC: Sistema de parque com barreira em código de barras, ideal para parques pequenos. Embora seja simples, é, também, modular e permite diversas funcionalidades como os cartões sem contacto, entre outros [39];
- JMS – *Software* de gestão JANUS: Inovador e baseado em *Web* sendo possível o seu acesso de qualquer equipamento com ligação à Internet e com diferentes níveis de acesso (incluindo *smartphone* e/ou *tablet*) [39].

Alguns destes parques já utilizam um sistema de sensores para deteção de lugares de estacionamento. Porém, este está, apenas, ao nível interno do parque. Cada lugar possui um sinalizador luminoso em cima que comunica com um painel a LED para informar os condutores do número de lugares disponíveis numa dada zona do parque de estacionamento [40].

2.4. Conclusão

O conceito de *Smart City* está a emergir e alguns países estão a apostar na criação de cidades que possuem componentes inteligentes [9]. A integração de TIC numa cidade auxilia o controlo da poluição, aumenta a sustentabilidade e melhora o aproveitamento de recursos naturais. A análise dos dados recolhidos através dos aparelhos existentes em

toda a cidade, permite agir de forma rápida perante os problemas encontrados, estudar soluções que poderão melhorar consideravelmente o funcionamento da cidade [10]. São estas tecnologias que permitem tomar decisões capazes de elaborar alternativas que possam otimizar os processos de negócio e que alavancam o uso das TIC nas iniciativas de *Smart Cities* [2]. A gestão de esgotos e de espaços verdes ajudam a melhorar o desempenho da cidade, tornando-a mais atrativa, aumenta a habitabilidade e fomenta o turismo. Ter uma cidade economicamente inteligente possibilita o aumento do número de postos de trabalho através da criação de novos negócios, e do aumento do empreendedorismo e da produtividade [2]. Todos estes fatores elevam a necessidade de avançar para projetos de *Smart Cities* e, com a chegada de novos planos, juntando aos já em curso, a determinação dos governos em investir pode ser ainda maior [11].

O *Smart Parking*, sendo uma componente dentro do âmbito de transportes inteligentes das *Smart Cities* [8], também tem tido muita atenção por parte de algumas cidades e, neste capítulo, foram descritos alguns sistemas existentes. Estes projetos ofereceram uma visão mais realista do que pode ser um sistema de *Smart Parking* e permitiram observar algumas das suas tecnologias, vantagens e desvantagens [1] [22] para o estudo dentro desta dissertação. Com a implementação de sistemas de *Smart Parking* em zonas mais centrais das cidades e, consequentemente, mais problemáticas no que toca ao estacionamento, é possível que haja melhorias no controlo do tráfego e eliminação da circulação nas zonas residenciais em busca de lugares de estacionamento, dificultando o trânsito e o acesso aos habitantes [24]. A diminuição das emissões de carbono devido à redução de consumo de combustível e associada ao corte no tempo de espera na procura de lugares de estacionamento vagos [1], é um fator importante na decisão em investir neste tipo de projetos. Todavia, os custos elevados podem afastar os governos e outras entidades. Apesar de algumas contrariedades que possam surgir, considera-se que a implementação de um sistema de *Smart Parking* poderá resolver parte dos problemas das cidades, mais especificamente o congestionamento do trânsito em zonas centrais, históricas e residenciais inerente à busca de lugares de estacionamento [3].

3. Desenvolvimento do sistema de *Smart Parking* em estudo

Este capítulo apresenta o desenvolvimento do protótipo do sistema de *Smart Parking* como prova de conceito para a implementação de um sistema com o menor custo e o mais aberto possível. Optou-se por dividi-lo em três pontos referentes aos requisitos, desenho da arquitetura e componentes do sistema proposto e do protótipo elaborado durante o percurso do trabalho, para que se compreendam as características principais do sistema e os motivos que levaram a optar pelas soluções sugeridas. Após a descrição da solução proposta e dos itens do protótipo, são indicados os resultados obtidos para cada elemento do sistema.

3.1. Requisitos do sistema

Antes de partir para o desenho da solução em estudo, é necessário extrair alguns requisitos de alto nível que são importantes para o desenvolvimento da solução. Para a análise de requisitos, será feita a divisão entre *hardware* e *software* para compreender melhor as necessidades dessas duas componentes gerais do trabalho a desenvolver na dissertação.

3.1.1. Hardware

O *hardware* do sistema é constituído pela rede sensorial – sensores de estacionamento e *gateways*. Pretende-se que os sensores de estacionamento sejam de baixo consumo de energia, fiáveis no decorrer da deteção de veículos, de alcance suficiente para comunicar com a *gateway* e com custo o mais baixo possível. Durante a fase de instalação, pede-se que os elementos não estorvem a via pública, permitindo que não haja dificuldades na deslocação dos cidadãos e na limpeza das ruas. Isto é conseguido com a colocação dos sensores de estacionamento no interior do solo, fazendo com que os mesmos sejam invisíveis e inacessíveis por parte de terceiros e as *gateways* poderão ser introduzidas dentro dos parquímetros, semáforos ou outros

elementos distribuídos pelas cidades que possam proteger os dispositivos. As configurações dos sensores de estacionamento e das *gateways* serão efetuadas durante a instalação dos mesmos através de um *software* específico para o efeito e não irá perturbar a via publica durante o processo. O ponto de colocação ótimo dos sensores de estacionamento será aquele em que a probabilidade de deteção é máxima, o que significa minimizar as probabilidades de deteção falsa (causada por outros veículos ou objetos próximos) e falsa rejeição (devida a uma variação não suficientemente elevada perto do sensor com um veículo estacionado no local). Esse local ideal de colocação dependerá do tipo de estacionamento. No caso de parques de estacionamento paralelos, o sensor de estacionamento deve ser colocado abaixo de um dos lados do carro. Para locais de estacionamento perpendiculares, o local mais adequado será o mais próximo do centro do motor ou do lado de trás do veículo [41].

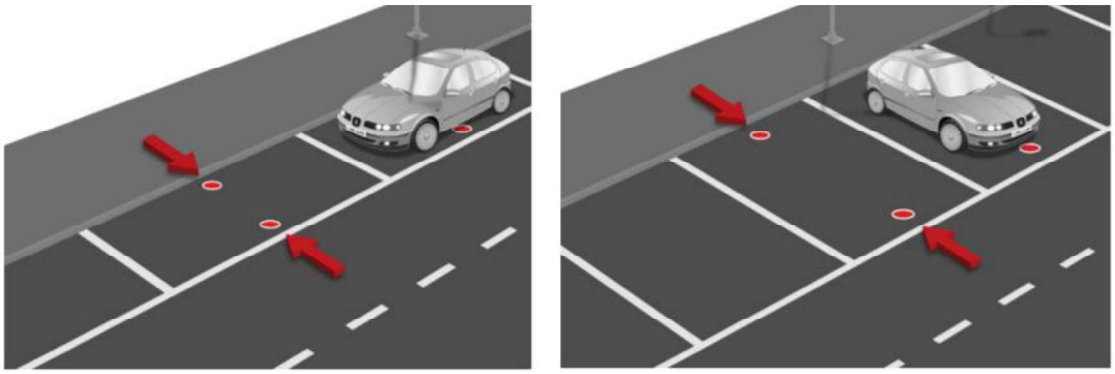

Figura 3.1: Opções de pontos de colocação ótimos do sensor de estacionamento no pavimento [41].

3.1.2. Software

O *software* do sistema é constituído pelo *backend* e aplicação móvel. Na base de dados será colocada a informação necessária para o funcionamento global do sistema. Parte desta informação é confidencial e, por isso, pede-se um serviço de *backend* rápido e seguro. A aplicação móvel constitui uma parte fundamental do sistema para os utilizadores, pois é através dos seus dispositivos que estes irão entrar em contacto com o sistema. Pede-se uma aplicação atrativa, intuitiva e que satisfaça as necessidades dos utilizadores, informando-os sobre os lugares de estacionamento vagos numa dada localização e apresentar o trajeto até ao destino. A aplicação deverá ser segura e respeitar as normas de utilização, impedindo o acesso por parte de terceiros a informações relativas aos utilizadores. Deverá existir uma experiência de utilizador

agradável, evitando *bugs* desnecessários e tornar a aplicação fluida durante a utilização. Toda a informação essencial deverá ser guardada na base de dados específica de forma estável e sem acessos alheios.

3.2. Desenho global da arquitetura do sistema

A figura 3.2 mostra a arquitetura global do sistema e que serve de base para a construção do protótipo [42]. Cada componente da imagem será objeto de análise, individualmente, ao longo deste capítulo.

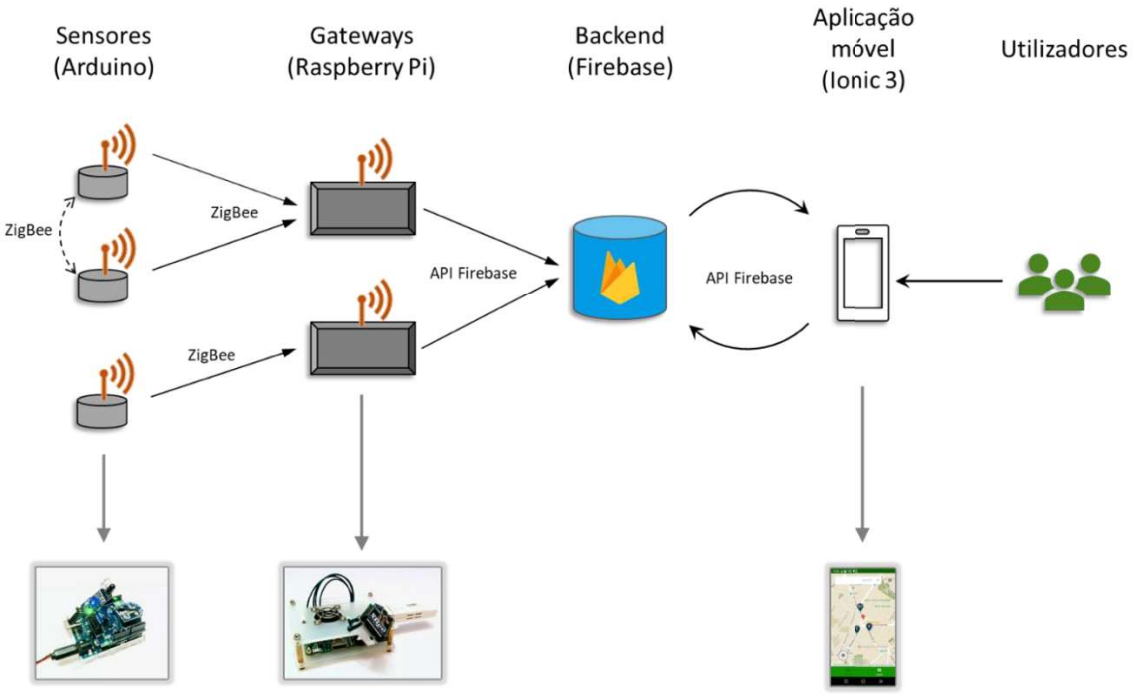

Figura 3.2: Arquitetura global do sistema de *Smart Parking*.

3.3. Componentes do protótipo de *Smart Parking*

Tal como ilustrado na imagem anterior, o protótipo do sistema de *Smart Parking* em desenvolvimento inclui quatro componentes essenciais: sensores de estacionamento, *gateways*, *backend* e aplicação móvel. Pretende-se apresentar algumas soluções para cada um dos elementos e, consequentemente, fazer a descrição do protótipo, incluindo os resultados obtidos.

3.3.1. Sensores de estacionamento

3.3.1.1. Soluções propostas

Os sensores de estacionamento são, sem dúvida, um elemento muito importante na construção de um sistema de *Smart Parking*, pois estão diretamente ligados à parte

física do mesmo. A complexidade técnica de um sensor de estacionamento leva a crer que a produção integral destes não seja viável em termos monetários, de funcionamento adequado e de tempo despendido dentro do contexto da dissertação. Verdadeiramente, o tempo consumido e o investimento aplicado num sensor de estacionamento concebido de raiz poderão não compensar, caso se pretenda implementar um sistema o mais rápido possível numa cidade. Mesmo tendo os conhecimentos e os recursos para tal, a produção em massa dificulta um pouco a redução clara dos custos. Portanto, o que esta dissertação pretende demonstrar é que, apesar da impossibilidade de se diminuir, consideravelmente, os custos na obtenção do sensor de estacionamento, a implementação de um sistema de *Smart Parking* de baixo custo é exequível, por via da redução de despesa na produção e desenvolvimento dos restantes elementos do sistema.

Um sensor de *Smart Parking* tem de oferecer, obrigatoriamente, 3 características principais: viabilidade, autonomia e proteção. A garantia de que os dados enviados são verdadeiros e a deteção de veículos acontece sem erros permite que este cumpra os requisitos pretendidos para o funcionamento global do sistema [42]. Um simples sensor de proximidade por radiação infravermelha não assegura que o sensor de estacionamento seja viável devido ao facto de este detetar qualquer obstáculo e até mesmo a própria luz, sendo que este fenómeno foi observado durante a fase de testes do próprio sensor. A necessidade de colocá-lo à superfície torna-o vulnerável ao vandalismo e, assim, exclui-se das opções para um produto final. Todos os outros sensores descritos no capítulo do estado da arte não são válidos para esta solução. Existe, porém, um sensor que garante a viabilidade da deteção de, somente, veículos e passível de ser introduzido debaixo do asfalto: o sensor geomagnético RM3100 da PNI Sensor Corporation [43].

a. Sensor Geomagnético RM3100

Os sensores geomagnéticos são utilizados para medir o campo magnético da Terra, mas existem elementos exteriores que podem alterar os campos magnéticos e distorcer a informação, temporariamente, designadamente, peças metálicas, veículos de passagem ou dispositivos eletrónicos próximos.

O RM3100 é um conjunto de sensores magnéticos de 3 eixos com tecnologia magneto-indutiva que oferece alta resolução, baixo consumo de energia e grande imunidade ao ruído de sinal. Este sensor incorpora as interfaces *Inter-Integrated Circuit*

(I2C) e *Serial Peripheral Interface* (SPI) para a flexibilidade do desenho do sistema [43].

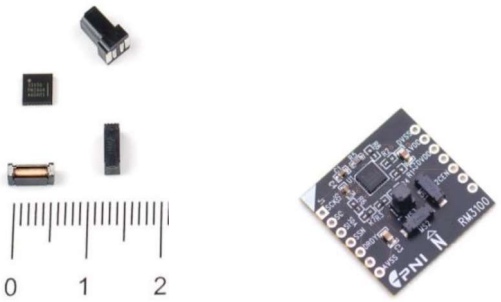

Figura 3.3: RM3100, conjunto de sensores magnéticos [43].

Este sensor apresenta um custo de 15,50US$, podendo ser reduzido consoante a quantidade de aparelhos encomendados. Infelizmente, caso se opte pela construção do sensor de estacionamento através da ligação de elementos individuais, incluído o RM3100, será necessário um adaptador, representado na figura 3.3, com um preço de 20US$, o que aumenta significativamente os custos para implementação do sistema de *Smart Parking* final. Porém, este sensor revela características interessantes para outros projetos de *Smart Cities* que possam surgir ao longo do tempo e, no contexto desta dissertação, é, inquestionavelmente, uma solução bastante razoável.

b. Dispositivo sensor com emissor/receptor ZigBee

Como descrito, anteriormente, o ZigBee designa o protocolo de comunicação de curto alcance entre os sensores de estacionamento e, posteriormente, a *gateway*, através do dispositivo XBee, que apresenta uma lista de modelos ao longo do tempo. Os custos de cada XBee podem rondar entre os 17€ e os 28€ para modelos mais simplistas [13]. A configuração dos aparelhos é efetuada, facilmente, a partir de um computador pessoal com sistema Windows, MacOS ou Linux, utilizando o programa XCTU fornecido gratuitamente pela Digi, uma empresa de suporte a desenvolvimento de *software* específico. Esta configuração passa por, em primeira instância, definir qual o papel que o XBee vai desempenhar na rede: Coordinator (Coordenador), Router ou End Device (Dispositivo Final). Uma rede de dispositivos XBee deve conter apenas um Coordinator, que define o valor do *Personal Area Network Identification* (PANID) e o modo de segurança da rede. O coordenador é o elemento central de confiança da rede, através da utilização do protocolo de segurança Zigbee 2007, e é responsável pela autenticação de novos nós e distribuição de chaves de rede à medida que estes se

conectam à rede [44]. Seguidamente, temos os Routers que permitem a comunicação de dados entre os nós restantes. Não existindo a possibilidade de entrarem em modo de suspensão, não se aconselha o uso deste tipo de configuração para aparelhos alimentados por bateria ou outra fonte pouco autónoma. Os Routers também possibilitam a entrada de novos nós na rede. Por fim, os End Devices, ao contrário dos Routers, não possuem a capacidade de receber e enviar dados provenientes de outros nós, porém estes podem entrar em modo de suspensão, o que possibilita a configuração em dispositivos alimentados por baterias recarregáveis. Caso um End Device entre noutra rede, o novo nó-pai é informado pelo dispositivo que este fará parte da rede.

Estes são os primeiros passos a seguir para a configuração inicial dos XBee, tanto para os sensores de estacionamento como para as *gateways*. Conclui-se, então, que os XBee dos sensores de estacionamento vão estar configurados, maioritariamente, como End Devices, garantindo, dessa forma, uma poupança de energia e, por conseguinte, um aumento da autonomia. Dependendo das zonas de instalação dos sensores de estacionamento e *gateways*, alguns terão de ser configurados como Router para garantir que a zona é escalável, permitindo a introdução de outros nós e aumentar o raio de comunicação da rede, sem a necessidade de instalação de outras *gateways*.

c. Sensor PlacePod RF 868MHz

O sensor de *Smart Parking* PlacePod contém o sensor geomagnético RM3100 da PNI. Ao contrário dos sensores magnéticos padrão, este sensor da PNI tem a capacidade de eliminar interferências eletromagnéticas ou “ruídos”. Os sensores e algoritmos são otimizados para a deteção de veículos de forma precisa, reduzindo as falsas deteções [45]. A tabela 3.1 apresenta as especificações do sensor de estacionamento e, posteriormente, é apresentada uma lista das características principais.

	Especificações
Comunicação	Long-Range 915MHz LoRa Module Compatível com LoRaWAN 1.0 Utiliza bandas Sub-GHz ISM na América do Norte e Europa
Outputs	Ocupado/Livre
Autonomia da bateria	10 anos dependendo da configuração e distância ao *gateway*
Dimensões (subterrâneo)	10.92cm de diâmetro (buraco com 11.43cm de diâmetro no mínimo) 3cm de altura (buraco com 6.35cm de altura no mínimo)
Dimensões (superficial)	22.86cm de diâmetro e 3cm de altura
Posição de instalação	Centrado na área de estacionamento
Temperatura de funcionamento	-30°C a +70° C
Temperatura de armazenamento	-40°C a +85° C

Tabela 3.1: Tabela de especificações do sensor de estacionamento PlacePod [45].

O preço da PlacePod para quantidades de implementação está disponível para orçamento. Os preços podem variar entre 100US$ e 200US$, dependendo da quantidade total da encomenda. Para implementações piloto ou final, um representante de engenharia PNI poderá ser disponibilizado como consultor no auxílio à configuração inicial. No entanto, a PNI não é responsável pelo seguinte:

- Instalação física do PlacePod;
- Criação e/ou manutenção da rede LoRaWAN, incluindo *gateways*;
- Integração de *software* de terceiros.

O PlacePod é acompanhado pelo *software* PNI's Parking Cloud, sem custos adicionais. A PNI's Parking Cloud inclui uma aplicação de gestão de estacionamento e uma API de estacionamento para comunicação com qualquer sistema externo e interno.

Atualmente, não há nenhum custo associado ao PNI Parking Cloud. O PlacePod é compatível com LoRaWAN, sendo a PNI um membro oficial da LoRa Alliance.

Esta é, com certeza, uma excelente solução no que diz respeito aos sensores do sistema de *Smart Parking*, pois revela completude e todas as características essenciais para o bom funcionamento do sistema. Apesar de apresentar um preço elevado, o PlacePod oferece todos os elementos necessários e traz garantias relativamente à autonomia da bateria, *software* incluído e permite a utilização de elementos externos, tais como *gateways* e outro *software* desenvolvido. A compatibilidade é um ponto positivo deste aparelho e pode ser determinante numa fase inicial de desenvolvimento de um projeto de *Smart Parking*. Outros produtos existentes no mercado não revelam esta liberdade de opção, forçando a aquisição dos dispositivos restantes para o funcionamento correto do sistema. Assim sendo, o PlacePod é visto, nesta dissertação, como uma alternativa para o que se pretende do sistema de *Smart Parking* final, não excluindo, obviamente, a alternativa de construção de um sensor de estacionamento de raiz, com os principais elementos apresentados, anteriormente, nesta secção.

3.3.1.2. Descrição do protótipo

O protótipo do sensor de estacionamento é constituído por 3 elementos principais: sensor de proximidade por radiação infravermelha, Arduino Uno R3 e XBee Série 2 [42]. A descrição de cada elemento será feita nos pontos seguintes, para que se compreenda a função que cada um desempenha dentro do sensor de estacionamento.

a. Sensor de proximidade por radiação infravermelha

O sensor de proximidade é um simples sensor que deteta a presença de obstáculos com base na reflexão de radiação infravermelha emitida por um LED transparente e recolhida pelo LED mais escuro ao seu lado. É um sensor básico e de baixo custo que apenas é utilizado para efeitos de prototipagem e não é aconselhável o seu uso num sistema de *Smart Parking* final.

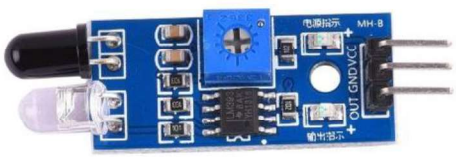

Figura 3.4: Sensor de proximidade por radiação infravermelha.

b. Arduino Uno R3

O Arduino é uma placa eletrónica de *hardware* livre, que suporta a entrada/saída de elementos periféricos e linguagem de programação C/C++. Esta placa permite a criação de protótipos funcionais de baixo custo e que facilitem o uso por parte de principiantes e profissionais [46]. É através do Arduino que todos os componentes do sensor vão comunicar para obtenção e envio de dados, e alimentação. O programa compilado e carregado no Arduino permite, em primeiro lugar, a inicialização das variáveis inerentes ao sensor de proximidade e XBee. As restantes variáveis globais estão em conformidade com o funcionamento do programa, incluindo a recolha e tratamento de dados.

```
#include <SoftwareSerial.h>

SoftwareSerial xbee(2, 3); // RX, TX
const int ProxSensor = 7;
const String data = "1;38.74771449916544;-9.153176236770606;";
int dgRead = -1;
int counter = 0;
int numReads = 0;
```

Figura 3.5: Variáveis globais do programa compilado para Arduino.

Na figura 3.5, retiram-se, intuitivamente, 4 variáveis importante:

- `SoftwareSerial xbee(2,3)`: declaração da variável respeitante ao XBee que está localizada nas portas 2 e 3, indicando a ligação RX e TX do Arduino, respetivamente;
- `const int ProxSensor = 7`: inicialização da variável constante do sensor de proximidade em que o inteiro 7 designa a porta do Arduino onde este está ligado para a recolha digital do estado do mesmo;
- `const String data`: esta variável constante indica a cadeia de caracteres que irá ser enviada para a *gateway* através do XBee. Cada dado é separado por ';', onde o primeiro inteiro representa o identificador do sensor, os dois seguintes representam as coordenadas da posição do sensor de estacionamento e o último irá representar, mais à frente, o estado do estacionamento (0 → ocupado, 1 → livre, −1 → sensor indisponível);
- `int dgRead`: aqui irá ser guardado o estado do estacionamento obtido pelo sensor de proximidade, inicializado a −1.

As restantes variáveis fazem parte do programa lógico para o tratamento dos dados e serão avaliados mais à frente.

Após a declaração e inicialização de algumas variáveis, um programa desenvolvido para Arduino é estruturado por duas funções principais: '`setup()`' e '`loop()`'. Na função '`setup()`' são feitas as últimas inicializações. É dentro do '`loop()`' que, como a própria palavra indica, o programa irá correr, infinitamente, em ciclo e, por isso, é necessário tomar as devidas precauções, relativamente ao modo como os dados são recolhidos e enviados, evitando-se, assim, erros de transmissão e garantido a integridade dos dados.

```
void setup() {
  pinMode(ProxSensor, INPUT);
  Serial.begin(115200);
  xbee.begin( 115200 );
}
```

Figura 3.6: Função '`setup()`' do programa compilado para Arduino.

Como se pode observar na figura 3.6, a função '`setup()`' consiste na inicialização dos elementos constituintes do sensor de estacionamento. É feita, primeiramente, a inicialização do sensor de proximidade através da função '`pinMode`', que recebe como argumentos o inteiro que indica onde o sensor está conectado e o segundo argumento refere-se ao modo como o Arduino encara o elemento conectado (neste caso, '`INPUT`', pois a intenção aqui é obter o estado do sensor como entrada de dados para o Arduino). As linhas seguintes indicam a taxa *baud* (*bit* por segundo) da transmissão de dados.

```
void loop() {
  char* bufSend;
  String dataToSend = data;
  int status = digitalRead(ProxSensor);
  if(dgRead != status){
    numReads++;
    if(numReads >= 2){
      dgRead = status;
    }
  }
  else{
    numReads = 0;
  }
  if(counter == 10 || numReads >= 2){
    dataToSend += dgRead;
    dataToSend.toCharArray(bufSend, dataToSend.length());
    xbee.write( bufSend, dataToSend.length() );
    Serial.println( dataToSend );
    counter = 0;
    numReads = 0;
  }
  counter++;
  delay(2000);
}
```

Figura 3.7: Função '`loop()`' do programa compilado para Arduino.

É na função '`loop()`' que se irá efetuar a recolha, tratamento e envio de dados. É importante destacar no código acima as funções '`digitalRead()`' e '`XBee.write()`', pois ambas estão relacionadas com a leitura do estado do sensor de proximidade e envio da informação através do XBee. A última instrução condicional do programa contém a função de envio de dados por parte do XBee. Para que isso seja possível, é imperativo que a variável '`counter`' seja igual a 10 ou '`numReads`' seja maior ou igual a 2, que especificam, respetivamente, um contador que é incrementado a cada iteração do ciclo e o número de leituras feitas, nas quais o estado anterior do sensor representado por '`dgRead`' é diferente do estado atual do sensor indicado pela variável '`status`'. O objetivo é garantir que não existem erros de deteção de obstáculos, verificando se a variável '`status`', respeitante ao estado atual do sensor, permaneceu alterada em relação ao estado anterior, ou seja, se na primeira iteração do ciclo houve uma alteração de estado que foi confirmada na segunda iteração do ciclo, garantindo, assim, uma deteção verdadeira. O contador tem como função enviar dados a cada 10

iterações, caso não haja alterações de estado, informando à *gateway* que o sensor de estacionamento está ativo na rede. Isto permite que o sensor não esteja constantemente a enviar informação inalterada e, assim, poupar energia. A última função do programa '`delay(2000)`', refere-se a um atraso de 2 segundos antes de voltar à primeira instrução do ciclo.

c. Emissor/receptor XBee Série 2

O XBee, como referido anteriormente, é o dispositivo de comunicação de dados para a *gateway* através do protocolo ZigBee. Foi configurado como Router, devido à existência de somente 3 sensores de estacionamento na rede sensorial para o protótipo e aumentando, assim, o raio de comunicação entre os mesmos e a *gateway*. Embora esta configuração não seja totalmente eficiente em termos energéticos, justifica-se pelo facto de o consumo dos dispositivos ser baixo e aumentar o alcance do dispositivo mais distante, relativamente, à *gateway*.

Figura 3.8: Dispositivo de comunicação XBee.

Após a ligação de todos os elementos, o resultado do sensor de estacionamento encontra-se na figura 3.10. Através da consola do *software* disponibilizado para a programação e configuração do Arduino, é possível verificar que os dados impressos são coerentes com aquilo que se pretende. Nesta fase, ainda não é possível determinar se essa informação está a ser enviada pelo XBee, pelo que esta conclusão só será averiguada no ponto seguinte, referente à *gateway*.

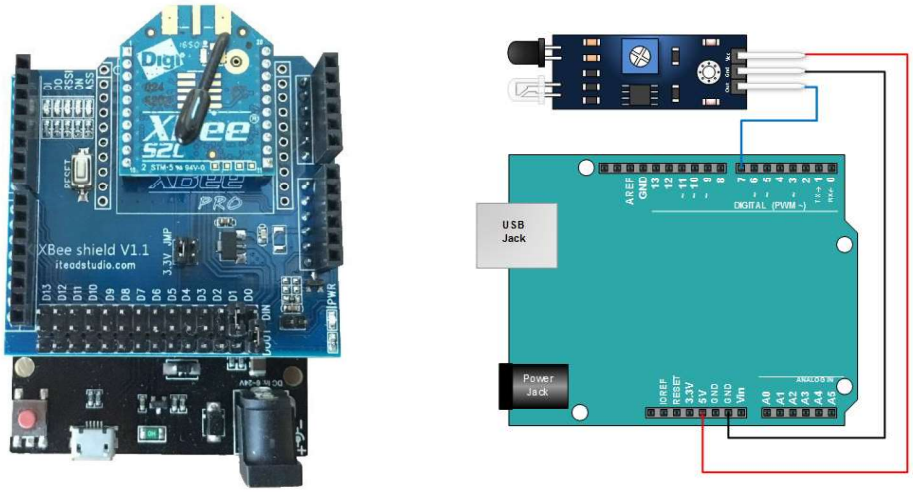

Figura 3.9: Ligação do sensor e do *shield* ao Arduino [47] [48].

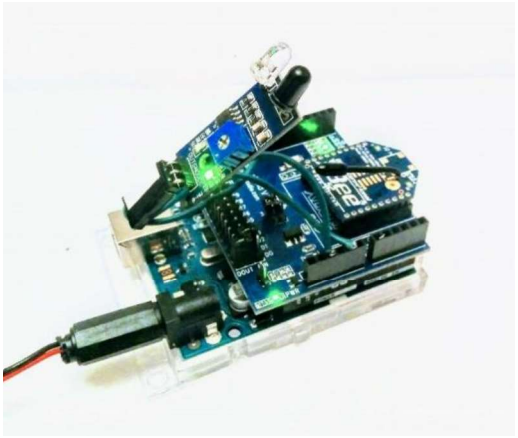

Figura 3.10: Protótipo do sensor de estacionamento incluindo XBee, sensor IV, *shield* e Arduino.

3.3.2. Gateway

3.3.2.1. Soluções propostas

À semelhança dos sensores de estacionamento, a *gateway* também pode ser construída de raiz. Existe a opção de aquisição de um produto final. Contudo, poderá ser muito dispendiosa. No capítulo 2, faz-se referência a alguns exemplos de produtos existentes, contudo o seu preço não é revelado. Em conformidade com a lista de possibilidades para o sensor de estacionamento e para além dos aparelhos apresentados no capítulo 2, esta dissertação considera 2 opções, para as *gateways* do sistema de *Smart Parking* final, descritas seguidamente.

a. *Gateway* Multitech da PNI

Apesar deste produto não estar especificado no *website* da PNI [45], o Diretor de Desenvolvimento de Negócios da PNI, em resposta a um *e-mail* enviado, referiu a existência deste aparelho pré-configurado por 450US$, um valor demasiado elevado para o que se pretende nesta dissertação. Todavia, esta seria a escolha a tomar, caso se optasse pela aquisição dos sensores de estacionamento da PNI, devido à maior compatibilidade entre os componentes e à possibilidade de obtenção de um acordo interessante por parte da empresa.

b. Solução com base em Raspberry Pi – opção para o protótipo funcional

O Raspberry Pi é um pequeno computador de baixo-custo que abriu portas a quem se quer dedicar à programação, mas não tem possibilidades para adquirir um computador convencional para o efeito.

Para a construção de uma *gateway* de raiz, esta é uma solução viável, pois não é necessário um computador poderoso para o tratamento de dados provenientes dos sensores de estacionamento. O preço de um Raspberry Pi varia consoante os modelos existentes. O modelo 3 não ultrapassa os 35€ e o recente modelo Zero custa menos de 5€ [49], contudo é limitado ao uso obrigatório de um recetor de Wi-Fi para ligação à rede. Para o sistema operativo, existe um leque de opções de sistemas baseados em Linux. O mais comum e fácil de instalar é o Raspbian, um sistema operativo baseado em Debian e bastante intuitivo. Este sistema operativo já traz várias ferramentas de programação e *software* instalado, o que facilita a configuração inicial. O dispositivo contém 40 portas de *General Purpose Input/Output* (GPIO) que servem de interface física entre o Raspberry Pi e outros elementos electrónicos [50].

3.3.2.2. Descrição do protótipo

Para o protótipo, foi escolhido o Raspberry Pi 3, onde foi instalado o sistema operativo Raspbian [42]. Este sistema já traz um compilador Python, o que facilita a implementação do programa que irá recolher os dados provenientes dos sensores de estacionamento e enviá-los para a base de dados. A figura 3.11 representa o processo de ligação do XBee às portas GPIO do Raspberry Pi [51]. Após as configurações necessárias do sistema para que o acesso ao XBee seja bem-sucedido, o próximo passo é a implementação do programa em Python para recolher os dados provenientes do XBee e tratá-los como devido.

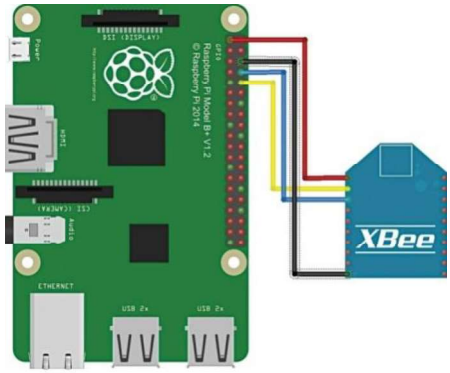

Figura 3.11: Ligação do XBee ao Raspberry Pi [51].

Para a explicação do programa, em Python, desenvolvido para a *gateway*, optou-se por dividi-lo em 5 partes essenciais. Os pontos seguintes descrevem as secções de código representadas pelos excertos acima de cada um, respetivamente.

```
import serial
from firebase import firebase
import json
import time
from threading import Timer
import socket

## Setup
ser = serial.Serial ("/dev/ttyAMA0", 115200)
idZ = 'z1'
idG = 'g1'
i = 0
sensors = []

data = {idG : {"id": "1",
               "rua":"Av. das Forças Armadas 376",
               "tipo_zona":"Verde",
               "tarifa":"0.25€/15 min.",
               "horario":"09:00-19:00",
               "lat":"38.74771449916544",
               "lng":"-9.153176236770609",
               "sensores":"" } }

auth = firebase.FirebaseAuthentication('***************',
                                       '************',
                                    extra={'uid': '************'})
firebase = firebase.FirebaseApplication('*****************', auth)
```

Figura 3.12: *Setup* do programa desenvolvido para a *gateway*, em Python.

A secção do código representada na figura 3.12 está, diretamente, ligada à inicialização das variáveis e elementos importantes que irão ser relevantes ao longo da execução do programa. A variável '`ser`' está associada à porta GPIO do Raspberry Pi, onde o XBee se encontra conectado e com uma taxa *baud* de 115200, estando coerente com as configurações do programa dos sensores de estacionamento e do dispositivo XBee. As variáveis constantes '`idZ`' e '`idG`' são respeitantes ao identificador da zona de estacionamento e ao identificador da *gateway*. O vetor '`sensors`', inicialmente vazio, representa a lista de sensores de estacionamento dentro da rede sensorial da *gateway*. A variável '`data`' é um objeto JSON com informações sobre a *gateway* e que será enviado, em primeira instância, à base de dados para a declaração da *gateway* como novo elemento, caso este não exista. Finalmente, é feita a inicialização da API da plataforma Firebase que será utilizada ao longo do programa para envio dos dados recolhidos. Esse excerto de código será analisado com mais detalhe no ponto referente ao protótipo da base de dados.

```
class Sensor:

    def __init__(self, id, lat, lng):
      self.id = id
      self.lat = lat
      self.lng = lng
      self.timer = Timer(100.0, self.removeSensor)
      self.timer.start()

    def getId(self):
        return self.id

    def startTimer(self):
        self.timer = Timer(100.0, self.removeSensor)
        self.timer.start()

    def cancelTimer(self):
        self.timer.cancel()

    def removeSensor(self):
        self.cancelTimer()
        idSens = "s" + self.id
        result = firebase.patch("/zonas/"+idZ+"/gateways/" + idG +
"/sensores", {idSens: {"id": self.id,
                       "lat":self.lat,
                       "lng":self.lng,
                       "status":"-1"}})
```

Figura 3.13: Classe '`Sensor`' do programa para a *gateway*, em Python.

A secção de código ilustrada pela figura 3.13 contém uma classe chamada '`Sensor`' e, como o próprio nome indica, está relacionada com os sensores de estacionamento. É importante destacar os métodos '`startTimer`', '`cancelTimer`' e '`removeSensor`'. Os dois primeiros dizem respeito a um temporizador de 100 segundos que representa o tempo limite que a *gateway* deve esperar para receber dados dos sensores de estacionamento. Caso o tempo de um determinado sensor esgote, este será considerado inativo dentro da rede sensorial da *gateway*, ou seja, será enviado o estado '`-1`' para a base de dados, através da chamada do método '`removeSensor`', informando, posteriormente, o utilizador que não é possível recolher informações sobre o estado daquele lugar de estacionamento.

```
    def addSensor(list, sensorData):
        hasSensor = False
        if len(list) > 0:
            for member in list:
                if member.getId() == sensorData[0]:
                    hasSensor = True
                    break
        if not hasSensor:
            list +=
[Sensor(sensorData[0],sensorData[1],sensorData[2])]

    def getSensorPosition(list, sensorId):
        sensorPos = -1
        hasSensor = False
        if len(list) > 0:
            a = 0
            for member in list:
                if member.getId() == sensorId:
                    hasSensor = True
                    break
                a += 1
        if hasSensor:
            sensorPos = a
        return sensorPos
```

Figura 3.14: Métodos 'addSensor' e 'getSensorPosition' do programa desenvolvido para a *gateway*, em Python.

Na figura 3.14, destacam-se dois métodos: 'addSensor' e 'getSensorPosition'. O primeiro método recebe como argumentos uma lista de sensores de estacionamento ('list') e um vetor que representa os dados recebidos pelo XBee ('sensorData'). Tal como o nome indica, este método adiciona um elemento à lista de sensores de estacionamento da rede sensorial da *gateway*, verificando se o mesmo existe na lista e, caso contrário, adiciona um objeto do tipo 'Sensor' ao final daquela. O segundo método procura um sensor com identificador 'sensorId' na lista de sensores de estacionamento e devolve a posição desse mesmo sensor de estacionamento dentro da lista. Estes métodos irão ser utilizados ao longo da execução do programa para que seja possível colocar novos sensores de estacionamento na rede, sem a necessidade de modificações diretas no programa da *gateway*. Assim, é possível que o programa esteja em execução de forma contínua.

```
result = firebase.get("/zonas/"+idZ+"/gateways/" + idG, None)
if result == None:
    result = firebase.patch("/zonas/"+idZ+"/gateways/", data)
```

Figura 3.15: Inicialização da *gateway* na base de dados.

Com a base de dados inicializada na primeira secção de código, a figura 3.15 representa o primeiro envio de dados, neste caso, o objeto '`data`'. Isto acontece, apenas, se a *gateway* não existir dentro da base de dados.

```
    ## Loop
    while True:
        i = i + 1
        dataReceived = str(ser.readline().strip()).replace("b","")
        dataReceived = dataReceived.replace("'","").split(";")

        if dataReceived and len(dataReceived) == 4 and dataReceived[0]
!= "":
            idS = "s" + dataReceived[0]
            addSensor(sensors, dataReceived)

sensors[getSensorPosition(sensors,dataReceived[0])].cancelTimer()
            result = firebase.patch("/zonas/"+idZ+"/gateways/" + idG +
"/sensores",
                                    {idS: {"id": dataReceived[0],
                                           "lat":dataReceived[1],
                                           "lng":dataReceived[2],
                                           "status":dataReceived[3]}})
            print ("%d Firebase: data uploaded to DB: %s" % (i,
dataReceived))

sensors[getSensorPosition(sensors,dataReceived[0])].startTimer()
```

Figura 3.16: *Loop* do programa desenvolvido para a *gateway*, em Python.

A figura 3.16 mostra o ciclo infinito que permite a execução contínua do programa. É dentro deste ciclo que se fará a recolha de informação proveniente dos sensores de estacionamento, a partir do dispositivo XBee. Esses dados serão, então, transferidos para um vetor através do método '`split()`' que separa uma cadeia de caracteres a partir de um caracter definido – neste caso, ';'. A instrução condicional '`if`' verifica a receção correta dos dados, adiciona o sensor de estacionamento à lista e cancela o temporizador associado ao mesmo. Seguidamente, serão enviados, para a base de dados, informações relativas ao sensor de estacionamento em causa e procede-se ao reinício do

temporizador. A função '`print`' é apenas um indicador visual na consola do editor de Python para assegurar que os dados são recebidos e enviados para a base de dados.

O resultado do protótipo da *gateway* encontra-se na figura 3.17. O programa em Python desenvolvido funciona, corretamente, como esperado e é possível verificar o seu funcionamento através da consola do Firebase, que será descrito no ponto seguinte. A *gateway* é alimentada por uma *power bank* com células fotovoltaicas que obtêm energia para a bateria polímero de lítio recarregável. É possível, controlar a *gateway*, remotamente, através da aplicação 'VNC Viewer', que acede ao ambiente de trabalho sem a necessidade de ecrã, teclado e rato. Porém, para que este acesso seja conseguido com sucesso, a *gateway* terá de estar ligada à Internet.

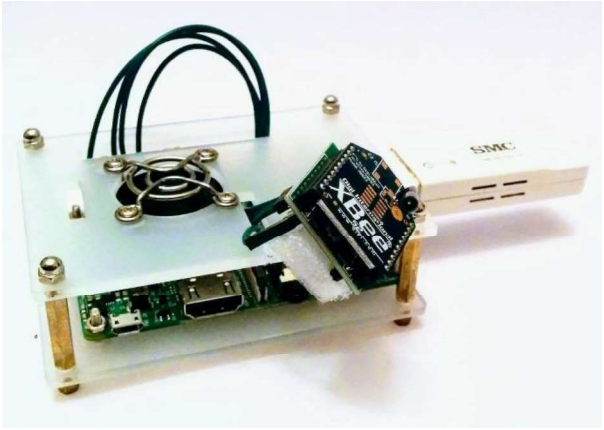

Figura 3.17: Protótipo da *gateway* incluindo XBee, ligação Wi-Fi e Raspberry.

3.3.3. Backend

3.3.3.1. Soluções propostas

No capítulo 2, foram listados alguns sistemas MBaaS disponíveis que oferecem serviços de armazenamento em *cloud* e outras características. Os MBaaS fomentam o crescimento das aplicações móveis, pois além de disponibilizarem acessos à *cloud* para armazenamento de conteúdo, também contêm serviços para notificações e análise de aplicações móveis que ajudam a aumentar, diariamente, o sucesso das mesmas. Uma aplicação com um serviço de *backend* apropriado garante a interligação com os utilizadores, o que permite o desenvolvimento progressivo da aplicação ao longo do tempo. Geralmente, o custo para aquisição destes serviços é bastante elevado. Porém, é obrigatória a sua integração em projetos IoT.

Para a proposta de solução, relativamente ao serviço de *backend* para um sistema de *Smart Parking*, esta dissertação considera quatro possibilidades de acordo com o contexto do projeto em estudo:

1. Utilização do serviço de *backend* adotado num sistema de estacionamento existente;
2. Optar por um serviço MBaas existente, nomeadamente, os serviços listados no capítulo 2;
3. Analisar os *Web Services* disponíveis que também oferecem produtos diversos para o pretendido;
4. Desenvolver o próprio *backend*.

No que toca à segunda opção, a lista incluída no capítulo 2 apresenta alguns sistemas dos mais utilizados em aplicações móveis e qualquer um poderá ser uma solução viável. Contudo, em relação à terceira opção, temos disponíveis alguns sistemas muito interessantes que oferecem produtos diversos para cada área de desenvolvimento de *software*: Amazon Web Services (AWS), Google Cloud Platform (GCP) e Azure. É importante realçar que existem outras ofertas, porém consideram-se estas três as mais adequadas para um sistema de *Smart Parking* totalmente funcional. Todos os serviços enumerados contêm produtos para desenvolvimento de projetos de IoT, o que facilita a integração no sistema de *Smart Parking* em estudo. Os custos adjacentes dependem das escolhas efetuadas e podem variar entre os 600€ e os 1700€ anuais. Independentemente da escolha tomada, existe uma plataforma denominada DreamFactory [52]. Esta poderá ser uma solução vantajosa, no que toca à organização de bases de dados utilizadas em sistemas de parqueamento existentes, não obrigando a alterações estruturais das tabelas e auxilia o desenvolvimento de uma API global para a aplicação a desenvolver. A última opção refere-se ao desenvolvimento do próprio *backend* e, no caso de se querer reduzir, substancialmente, os custos, esta seria a escolha a seguir. Porém, para desenvolver o próprio *backend* são necessários recursos e conhecimentos que não estão ao alcance de todos. A segurança e a viabilidade do serviço são extremamente importantes durante a execução do sistema e qualquer falha no *backend* poderá ser fatal. Por isso, a análise aos serviços de *backend* existentes permite optar pelo que se considera mais adequado ao sistema desenvolvido e também compreender as características essenciais no desenvolvimento do próprio *backend*.

A próxima secção descreve a plataforma utilizada para o protótipo do sistema de *Smart Parking* desenvolvido – Firebase. Este serviço, fornecido pela Google, ajuda o crescimento das aplicações móveis e adequa-se às necessidades pretendidas no desenvolvimento do protótipo funcional.

3.3.3.2. Descrição do protótipo

Como em qualquer serviço de *backend*, a segurança é uma das caraterísticas mais importantes, pois garante a autenticidade e integridade dos dados. O Firebase não foge à regra e este permite bloquear acessos indesejados aos conteúdos existentes através da definição de regras de leitura e escrita, restringindo esse acesso a utilizadores registados.

A inicialização da API também exige a autenticação por parte de um utilizador, neste caso, o gestor da base de dados. Na componente da *gateway*, é feita, inicialmente, a autenticação no Firebase e, em caso de sucesso, segue-se para a inicialização da API, o que permite, posteriormente, a chamada de funções para recolha e envio de dados para a base de dados [42].

```
auth = firebase.FirebaseAuthentication('***************',
                                       '************',
                                  extra={'uid': '***********'})
firebase = firebase.FirebaseApplication('****************', auth)
```

Figura 3.18: Autenticação e inicialização da API do Firebase na *gateway*.

A figura 3.18 mostra um excerto do código, em Python, da *gateway* que efetua a autenticação a partir da chamada à função '`FirebaseAuthentication`' e a inicialização com a chamada à função '`FirebaseApplication`'. Os argumentos da função '`FirebaseAuthentication`' são específicos para cada projeto. O primeiro remete para a chave secreta da base de dados do Firebase, seguido do endereço de *e-mail* e identificador do utilizador que pretende aceder, remotamente, ao Firebase. A função '`FirebaseApplication`' recebe o endereço referente ao projeto Firebase e o resultado da autenticação realizada anteriormente.

```
result = firebase.get("/zonas/"+idZ+"/gateways/" + idG, None)
if result == None:
    result = firebase.patch("/zonas/"+idZ+"/gateways/", data)
```

Figura 3.19: Recolha e envio de informação para o Firebase.

No extrato de código da figura 3.19, destacam-se as funções '`get`' e '`patch`' da API do Firebase, que apontam para a recolha e envio de informação para a base de dados, respetivamente.

```
firebase.initializeApp({
      apiKey: "**********************",
      authDomain: "**************",
      databaseURL: "***************",
      projectId: "smartpark-lx",
      storageBucket: "***********",
      messagingSenderId: "***********"
    });
```

Figura 3.20: Código de inicialização da API do Firebase em Ionic 3.

Nas configurações de um projeto Firebase, são fornecidos pedaços de código para introdução nos projetos de aplicações Android, iOS e *Web* que permitem a inicialização da API do Firebase. A figura 3.20 mostra esse excerto de código que contém os elementos necessários para inicialização da API. No caso de Ionic 3, é retirado o código para aplicações *Web*, pois este utiliza tecnologias *Web* para o desenvolvimento de aplicações móveis híbridas.

A figura 3.21 representa a estrutura da base de dados construída no programa 'Enterprise Architect' referente ao protótipo do sistema de *Smart Parking*. Durante o funcionamento da *gateway*, é possível verificar as alterações do estado dos sensores de estacionamento em tempo-real. Todas as informações contidas na base de dados são recolhidas e tratadas pela aplicação e mostradas ao utilizador de forma organizada. O ponto seguinte descreve as características da aplicação e revelar-se-ão alguns aspetos importantes do código desenvolvido para o protótipo.

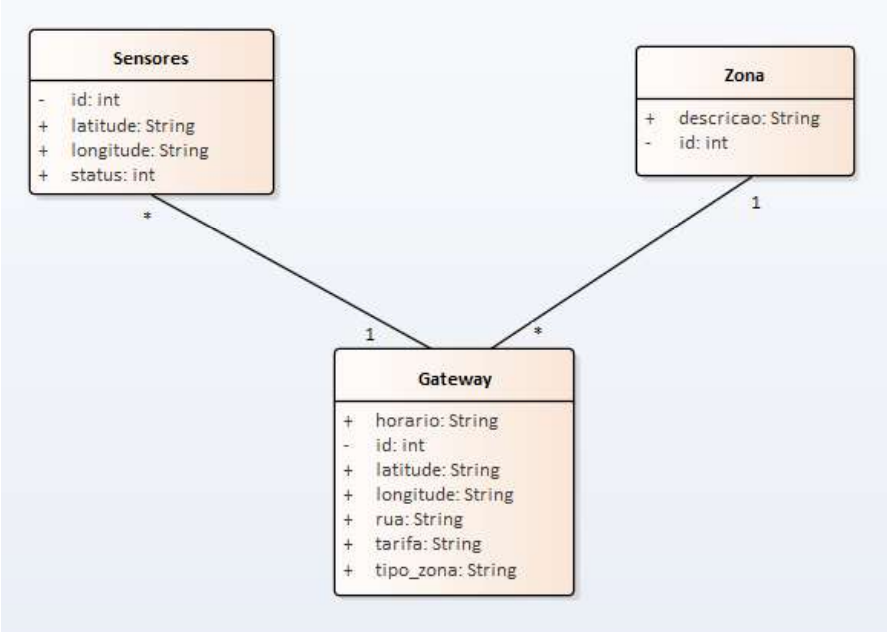

Figura 3.21: Estrutura da base de dados implementada em Firebase.

3.3.4. Aplicação móvel

3.3.4.1. Soluções propostas

Para o desenvolvimento da aplicação móvel, existem várias possibilidades. A decisão de escolha para o SDK a utilizar depende de dois fatores essenciais e cada um apresenta duas hipóteses:

1. Integração de um sistema de *Smart Parking* num sistema de parqueamento existente:
   a) Utilização do(s) SDK em uso para o desenvolvimento e manutenção da aplicação do sistema de parqueamento, sem alterações na estrutura base;
   b) Utilização de um SDK distinto, que obriga a reestruturação total da aplicação existente e, por conseguinte, o desenvolvimento adicional dos elementos relativos ao sistema de *Smart Parking*.
2. Criação de um sistema de *Smart Parking* de origem:
   a) Escolha do(s) SDK a utilizar por parte da equipa de programadores, consoante as suas preferências de linguagens e/ou *frameworks*;
   b) Escolha do(s) SDK que se adequa(m) às exigências do cliente, ou seja, a partir do levantamento de requisitos que determina a opção que se enquadra melhor na solução final.

Independentemente da decisão tomada, a solução final deve cumprir algumas regras de funcionamento geral que garantem a aceitação por parte dos utilizadores e a fiabilidade durante a execução da aplicação. O *layout* e a estrutura devem ser atrativos para os utilizadores, deve apresentar fluidez nas transições entre ecrãs e outros elementos durante o funcionamento global e, indispensavelmente, segurança e proteção dos dados dos utilizadores.

3.3.4.2. Descrição do protótipo

Para o desenvolvimento do protótipo da aplicação móvel, foi utilizado o Ionic Framework [42]. Como referido, anteriormente, no capítulo 2, o Ionic Framework é um SDK para o desenvolvimento de aplicações móveis híbridas que utiliza Angular 2 e Apache Cordova. Como qualquer SDK para desenvolvimento de aplicações híbridas, o Ionic permite que se construa aplicações em ambiente Android e iOS com, praticamente, o mesmo código-fonte. Para a construção de uma página respeitante a um ecrã na aplicação, é necessário incluir uma pasta com três ficheiros fundamentais: HTML, TypeScript e CSS. O ficheiro HTML estrutura os elementos contidos no ecrã, tal como numa página *Web* convencional. O ficheiro TypeScript é utilizado para implementação das ações durante a execução da aplicação, que inclui modificações visuais, lógicas e funcionais da aplicação. Por fim, o ficheiro CSS, assim como numa página *Web*, representa a estilização dos elementos contidos na aplicação.

a. Desenho da aplicação móvel

Como em qualquer projeto de desenvolvimento de aplicações móveis, o desenho é uma fase essencial para o planeamento do que se pretende desenvolver. Desenhar os ecrãs da aplicação (*mockups*) antes de esta ser implementada, auxilia os programadores na visualização inicial da ideia que se pretende desenvolver. O processo de análise e desenho da aplicação poderá ser moroso, porém fomenta um desenvolvimento mais acelerado.

O desenho da aplicação foi efetuado na plataforma Ionic Creator [53], que permite desenhar aplicações em Ionic através de *drag and drop*, gerando o código-fonte com as configurações iniciais para um projeto em Ionic.

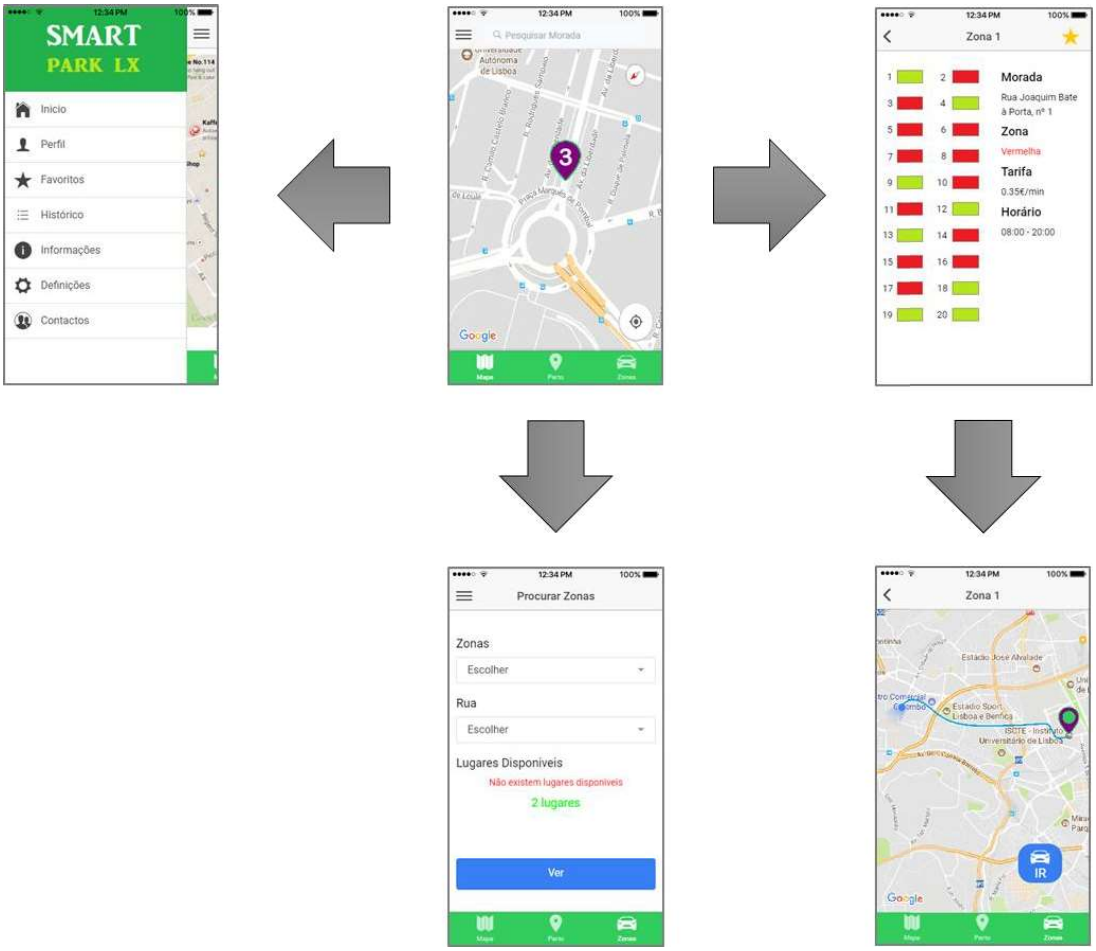

Figura 3.22: Desenho e navegação da aplicação.

A figura 3.22 contém o desenho e navegação de cinco ecrãs principais da aplicação. Pretende-se, então, uma aplicação com um menu lateral que, por sua vez, contém uma lista de opções de navegação. No fundo do ecrã, colocou-se uma barra com três abas principais. 'Mapa', 'Perto' e 'Zonas'. A aba 'Mapa' irá apresentar um mapa, através da conexão à API do Google Maps, que posiciona os marcadores nas zonas de estacionamento. Estes marcadores contêm um número representativo dos lugares vagos de estacionamento numa determinada zona. A aba 'Perto' apenas desencadeia a ação de levar o utilizador à sua localização e mostrar as zonas de estacionamento num determinado raio. Por fim, a aba 'Zona' permite que o utilizador procure lugares de estacionamento, listando as zonas e ruas de forma organizada e sem recurso ao mapa existente no ecrã principal. O objetivo é oferecer ao utilizador duas formas de pesquisa de lugares de estacionamento, sem a obrigação de explorar apenas o mapa. Cada lugar listado apresenta uma cor (vermelho ou verde) e um número de identificação. A cor verde indica que o lugar de estacionamento se encontra livre, possibilitando a navegação do utilizador até à vaga. Assim, o vermelho indica que o lugar está ocupado,

impedindo a navegação do utilizador até esse ponto. Finalmente, o ecrã restante descreve a linha de navegação desde a localização do condutor até ao lugar de estacionamento selecionado. O botão introduzido no canto inferior direito do mapa tem a ação de início da navegação, através da aplicação Google Maps, caso esta esteja instalada no dispositivo.

b. Implementação da aplicação móvel

Após a finalização do desenho no *website* da Ionic Creator [53] e exportação do projeto criado, procede-se para o desenvolvimento da aplicação móvel. O primeiro passo a tomar é compreender as APIs mais importantes a utilizar. Analisando as aplicações de *Smart Parking* existentes, todas elas contêm um mapa fornecido pela API da Google Maps. Ora, em primeira instância, prossegue-se com a inclusão da biblioteca do Google Maps para que o utilizador possa navegar no mapa em busca de zonas de estacionamento representadas por *markers* customizados (figura 3.23).

Figura 3.23: Locais de estacionamento disponíveis no mapa.

É possível verificar o número de lugares disponíveis representado por um número inserido nos *markers* de cada zona. Esse número será atualizado consoante a mudança de estado dos sensores de estacionamento referentes a uma determinada zona. Sempre que o utilizador pressiona um *marker*, este será conduzido a outro ecrã que mostra, com mais detalhe, as informações respeitantes à zona selecionada, ou seja, a lista de lugares e o seu estado atual, e as informações relativas à zona de estacionamento (figura 3.24).

Figura 3.24: Informações relativas à zona selecionada.

Caso o utilizador de facto selecione um lugar, aparecerá, novamente, o ecrã com o mapa anterior. Contudo, este mostra o trajeto desde a localização do condutor até ao destino, podendo esta operação ser confirmada ou cancelada (figura 3.25).

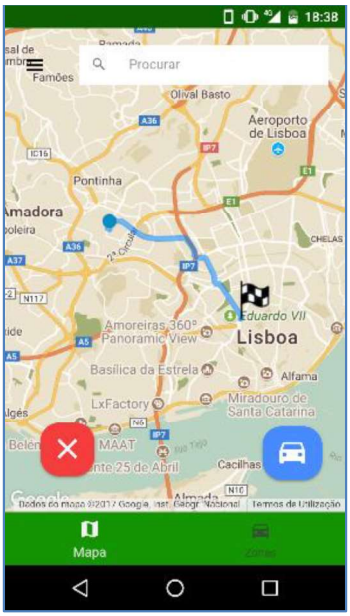

Figura 3.25: Localização do percurso através da API do 'Google Maps'.

Existe outra alternativa para a procura de zonas e essa encontra-se na segunda aba localizada no fundo do ecrã (figura 3.26). Após a seleção da zona e rua, é recolhida a informação sobre o número de lugares disponíveis e o botão 'Ver' que encaminha, mais uma vez, para o ecrã com os detalhes respeitantes à zona selecionada.

Figura 3.26: Informação sobre o número de lugares disponíveis.

O menu lateral (figura 3.27) oferece completude à aplicação, mostrando as secções mais relevantes para uma aplicação deste contexto e cada uma foi implementada separadamente.

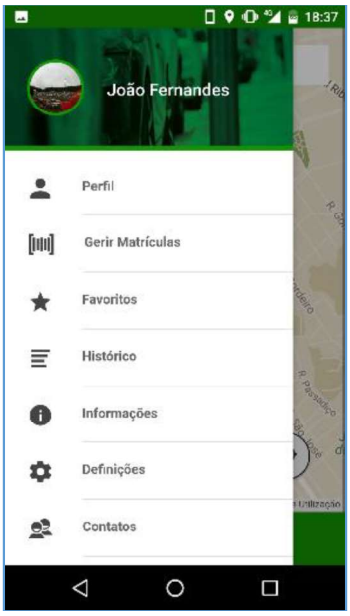

Figura 3.27: Dados do utilizador.

Todos os dados gerados e alterados pelo utilizador são, posteriormente, atualizados na base de dados, onde cada utilizador registado possui uma secção reservada. Dando maior relevância à comunicação com a base de dados, a figura B.1 em anexo mostra a classe '`DatabaseService`'. Esta classe lista um conjunto de métodos para extração e envio de dados provenientes do Firebase. Isto é possível através da chamada de

métodos implementados na biblioteca da API do Firebase. A aplicação inicia com a obtenção dos dados relativos ao utilizador e às zonas de estacionamento. Outros métodos auxiliam o processamento destes dados para apresentação correta ao utilizador. Durante a execução da aplicação, as informações sobre o estado do estacionamento em cada zona poderão ser atualizadas em tempo-real, pois foi implementado um *listener* que deteta se a base de dados foi alterada. Assim, o utilizador possui sempre as informações em tempo útil, sem esquecer, obviamente, os atrasos da receção de pacotes da rede.

Para exemplificar a implementação das funcionalidades listadas no menu lateral, observa-se, assim, a figura B.2 em anexo que representa um excerto da classe '`FavoritosService`'. Esta classe apresenta dois métodos importantes: '`storeFavoritos`' e '`fetchFavoritos`'. Como o nome indica, o primeiro método envia, para a base de dados, a informação das zonas favoritas do utilizador e o segundo método recolhe a lista de favoritos guardados na sessão anterior.

Finalmente, após o desenvolvimento das componentes principais da aplicação, colocando de lado alguns *bugs* que foram surgindo, optou-se por integrar um mecanismo de autenticação fornecido pelo Firebase. Um utilizador que abra a aplicação pela primeira vez será obrigado a registar-se na base de dados para que tenha autorização de acesso a conteúdos do sistema. Esta funcionalidade torna a aplicação mais robusta e completa, apesar de não ser necessário em modo de protótipo. A figura B.3 em anexo mostra a classe '`AuthService`' utilizada para o registo, autenticação e fim de sessão por parte do utilizador. O método '`getActiveUser`' permite que a aplicação obtenha a sessão anterior do utilizador guardada em cache, permitindo que este não necessite de efetuar o início de sessão sempre que entrar na aplicação.

No final do desenvolvimento da aplicação móvel, foram realizados alguns testes ao funcionamento interno da mesma. Após a correção de alguns erros que impediam a execução fluida da aplicação, analisou-se o comportamento do protótipo, incluindo, assim, os outros componentes para testes globais. Os resultados foram positivos, no que toca à receção dos dados provenientes do Firebase para a aplicação. Contudo, ainda existiam alguns pormenores visuais e funcionais a modificar na própria aplicação que a tornou mais apelativa e fluida na transição entre ecrãs e no acesso à base de dados. Porém, ainda contém um erro relativamente à procura de localização do dispositivo. Existem algumas explicações para este acontecimento que podem justificar a demora na

procura da localização, nomeadamente, a ligação a uma rede de fraca intensidade, o GPS do dispositivo não funcionar corretamente e a própria API de acesso através do código pode não estar completamente operacional. Também existe a hipótese de o acesso à localização do aparelho ser deteriorada em modo teste da aplicação.

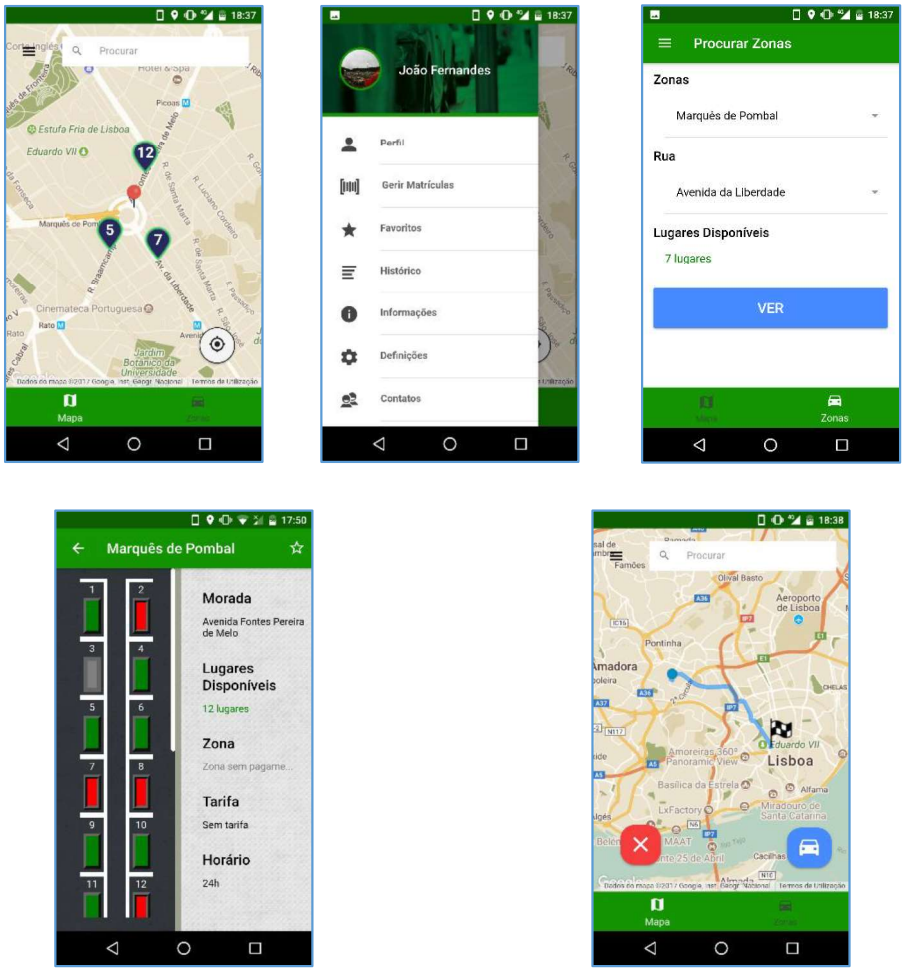

Figura 3.28: Resultados do protótipo da aplicação.

4. Caso de estudo para sistema de *Smart Parking*

No seguimento do capítulo 3, o protótipo funcional desenvolvido será descrito na sua totalidade, onde é feita a junção de todos os elementos, clarificando-se, assim, o funcionamento do sistema. O mesmo será feito para as soluções propostas de cada elemento do sistema, obtendo uma solução apropriada para um sistema de *Smart Parking* real. De seguida, serão apresentadas listas de vantagens e desvantagens do *Smart Parking,* consolidando o estudo e trabalho realizados no decorrer da dissertação e, por fim, a validação da solução obtida com os testes efetuados, os pontos positivos e negativos da tecnologia usada para o protótipo e informações obtidas em reunião com a empresa EMEL.

4.1. Protótipo funcional

No capítulo 3, foram descritos, singularmente, os componentes do protótipo funcional. O resultado é apresentado na figura seguinte, que inclui todos os elementos a funcionar em conjunto.

Figura 4.1: Protótipo funcional completo.

O protótipo funcional final é, então, constituído por duas *gateways*, três sensores de estacionamento e a aplicação móvel instalada num dispositivo Android. A *gateway* em cima e à direita da imagem recebe informações dos dois sensores logo abaixo à direita, e a *gateway* em cima e à esquerda recebe informações, unicamente, do sensor situado em baixo à esquerda. Pretende-se simular a existência de duas zonas de estacionamento ativas, ambas situadas nas coordenadas em redor do ISCTE-IUL. Uma das *gateways* é alimentada por um *power bank* com células fotovoltaicas com o objetivo de apresentar uma solução energética renovável e amiga do ambiente. Tendo em conta a existência deste método para fornecimento de energia para semáforos, por exemplo, estes aparelhos requerem uma quantidade de energia bastante mais baixo, o que viabiliza o uso de energias renováveis para a alimentação das *gateways*. Os sensores de estacionamento são alimentados por uma pilha de 9V, sendo suficiente para o funcionamento dos aparelhos.

Após a realização de testes e correção de alguns erros de performance, o protótipo final descreve, fielmente, o que se pretende para um sistema de *Smart Parking* final. Obviamente, os elementos do protótipo não são totalmente úteis para um sistema real, porém o intuito desta dissertação foi o de demonstrar que é possível construir um sistema de *Smart Parking* rudimentar de baixo custo. Retirando alguns aspetos essenciais durante a elaboração do protótipo final, é então possível apresentar uma solução para um sistema de *Smart Parking* final, que será detalhado na secção seguinte, bem como as suas vantagens, desvantagens e custos no ponto posterior.

4.2. Solução para o sistema de *Smart Parking* em estudo

Tal como no ponto anterior, a proposta de solução para o sistema de *Smart Parking* junta todos os componentes inseridos no capítulo 3.

Um sistema de *Smart Parking* é constituído por sensores de estacionamento, *gateways* para receção e envio de dados provenientes dos sensores, serviço de *backend* e aplicação móvel. Todos os elementos devem funcionar em harmonia para garantir a estabilidade, segurança e escalabilidade do sistema. Para isso, é necessário optar pelos melhores produtos, tanto pela qualidade física e tecnológica, como pelo preço o mais baixo possível sem comprometer o funcionamento global do sistema.

Recolhendo as soluções de cada componente descritas no capítulo 3, esta dissertação sugere o uso de sensores de estacionamento PlacePod RF 868MHz fornecido pela

empresa PNI que, apesar do custo um pouco elevado, oferece garantias no que toca ao seu desempenho e autonomia. Também se considera uma boa solução a utilização do sensor geomagnético RM3100 como elemento do sensor de estacionamento final, porém poderá elevar os custos de produção. Para a *gateway*, sugerem-se duas alternativas: a aquisição de *gateways* disponíveis na empresa PNI ou a construção das *gateways* a partir de um pequeno computador como o Raspberry Pi ou equivalente. A primeira hipótese é sem dúvida mais dispendiosa, contudo garante o funcionamento correto desta parte do sistema e não necessita de trabalho de produção adjacente. A última hipótese é mais rentável, porém obriga a que haja um trabalho complementar de configuração e criação de uma estrutura para proteção da *gateway*. Para a alimentação das *gateways*, propõe-se o uso de painéis fotovoltaicos semelhantes aos utilizados em parquímetros, por exemplo, ou outra forma de energia alternativa. O uso de uma bateria com autonomia elevada, tal como utilizada nos sensores de estacionamento, também apresenta uma solução viável de fornecimento de energia. No caso do serviço de *backend*, existem várias opções no mercado. Assumindo que o sistema de *Smart Parking* é construído de raiz, uma escolha possível seria o uso do AWS, que oferece um vasto leque de produtos que satisfazem as necessidades do sistema. No caso de se pretender integrar o sistema num outro de estacionamento já existente, é possível utilizar a base de dados em atividade e acrescentar outro serviço de *backend* mais adequado, sem excluir a atual. O DreamFactory é um serviço que ajuda a coligar bases de dados diferentes, criando uma API única para uso posterior na aplicação móvel e *gateway*. Finalmente, a aplicação móvel poderá ser implementada à preferência da equipa de desenvolvimento em relação às *frameworks* existentes. O protótipo da aplicação desenvolvida durante a dissertação apresenta as características essenciais para a aplicação final e poderá servir de modelo para construção da aplicação final, todavia esta necessita de melhorias nos aspetos visuais e funcionais relativamente às suas características secundárias.

4.2.1. Custos

Analisando os componentes propostos para o sistema de *Smart Parking*, o custo para cada um deles é um fator muito importante para quem pretende investir num sistema deste tipo. No capítulo 3, já foram revelados alguns dados sobre esta parte, contudo esta secção apresenta esses mesmos dados de forma mais organizada. A tabela seguinte revela os custos de cada componente, com as alternativas propostas, anteriormente,

nesta secção. Os elementos selecionados a verde representam os considerados mais aptos para o sistema de *Smart Parking* em estudo.

Componente		Custo
Sensor de estacionamento	RM3100 + adaptador	12.60€ + 16.25€
	XBee	17€ a 28€
	PlacePod RF 868MHh da PNI	81€ a 175€
Gateway	*Gateway* Multitech da PNI	365.50€
	Raspberry Pi ou equivalente	5€ a 35€
	XBee	17€ a 28€
Servidor/Base de Dados	AWS	519.40€ a 747.08€ por ano
	GCP	1175.30€ a 1424.33€ por ano
	Azure	493.07€ a 742.10€ por ano
Aplicação móvel	Ionic 3 desenvolvida para o protótipo	0€*

*Sujeito a cálculo posterior dependendo do custo de desenvolvimento, manutenção e operação.

Tabela 4.1: Custos dos componentes para o sistema de *Smart Parking*.

Os custos foram calculados com base nos dados disponibilizados nos *websites* dos produtos. Relativamente ao componente do servidor, os custos foram calculados a partir da imagem C.1 em anexo, onde foram retirados os valores anuais de subscrição dos produtos. No que toca à aplicação móvel, o custo está, diretamente, ligado ao resultado obtido durante a construção do protótipo final. Este valor pode variar consoante a empresa ou programador contratado para o desenvolvimento da aplicação.

4.3. Vantagens e desvantagens do *Smart Parking*

4.3.1. Vantagens

Analisando o capítulo 2, nomeadamente a secção respeitante à descrição de sistemas de *Smart Parking* existentes, é possível retirar algumas vantagens comuns que fortificam a necessidade de implementação de um sistema de *Smart Parking* nas cidades. De seguida, listam-se as vantagens que esse sistema genérico poderá trazer para as cidades [54]:

- Otimização de estacionamento – Os utilizadores economizam tempo, recursos e esforço, pois alcançam o melhor local disponível. Ao saber a localização exata da vaga, o utilizador recebe o benefício imediato de reduzir para 5 minutos o tempo médio em busca de um lugar em vez dos 10 minutos nos casos atuais, ou seja, o tempo de busca por lugares é reduzido em 43%. Adicionalmente, a indicação direta ao local de estacionamento permite uma direção mais tranquila e um trajeto, em média, 21% mais curto [36]. As entidades comerciais e corporativas usufruem dos seus espaços designados sem que haja problemas inerentes ao estacionamento indevido, pois o estacionamento é ocupado de forma eficiente;
- Redução do tráfego − O tráfego em redor flui de forma harmoniosa com a diminuição de veículos em busca de lugares de estacionamento livre;
- Poluição reduzida – "A procura de estacionamento consome cerca de um milhão de barris de petróleo por dia. Uma ótima solução de estacionamento diminuirá, significativamente, o tempo de condução, reduzindo em mais de 30% da quantidade de emissões diárias de veículos e, em última instância, reduzindo a pegada de carbono" [54];
- Experiência de utilizadores aprimorada – O condutor tem à sua disposição uma integração total da experiência de utilizador numa única ação. O pagamento do estacionamento, a identificação do local, a procura de local e as notificações de horário estão contemplados no mesmo mecanismo fornecido ao utilizador;
- Pagamentos integrados − Os utilizadores regulares podem optar por pagamentos através de uma aplicação, o que agiliza este processo, permite a inclusão em programas de fidelização de clientes e retirar *feedback* de

utilizadores valiosos. Para além disso, por possuir a oportunidade de pagar, efectivamente, pelo tempo utilizado, o cidadão economiza, aproximadamente, 22% do custo de estacionamento [36];

- Aumento da segurança – Os funcionários e os guardas de segurança obtêm dados em tempo real sobre o parqueamento, o que possibilita a prevenção de violações no estacionamento e atividades suspeitas. Com a redução do tráfego aliado à busca de lugares livres nas ruas pode diminuir o número de acidentes causados pela distração durante a procura de estacionamento;
- Dados em tempo real e observação de tendências – A recolha de dados provenientes de uma solução de estacionamento inteligente permite descobrir tendências de utilizadores e parques que podem vir a ser inestimáveis para os proprietários de estacionamento, pois consegue-se efetuar ajustes e melhorias de forma constante;
- Redução dos custos de gestão – Automatizar alguns processos, previamente, designados para trabalhadores economiza custos e reduz a exaustão de recursos;
- Aumento de serviços e imagem de marca – Ao oferecer uma boa experiência de utilização é possível criar uma imagem de marca corporativa ou comercial bastante atrativa para a aquisição novos utilizadores.

Para além destas vantagens, também é importante abordar os pontos positivos relativamente ao sistema propriamente dito. Os métodos de instalação, manutenção e expansão do sistema são, igualmente, relevantes para compreender os incentivos reais na implementação de um sistema de *Smart Parking* numa cidade.

- Instalação facilitada – O sistema poderá entrar em funcionamento alguns dias após a colocação dos componentes, posteriormente, configurados;
- Recolocação facilitada – Flexibilidade em mover, facilmente, o sistema para outras zonas da cidade;
- Escalável – Tendo em conta a sua fácil instalação e configuração, o sistema é altamente escalável, permitindo adicionar novas unidades sem interferir com os elementos existentes [55];

- Manutenção periódica – Tendo em conta a autonomia da bateria de um sensor de estacionamento, a manutenção desta componente só será necessária num período entre 3 a 10 anos, dependendo da capacidade das baterias. As *gateways*, caso sejam alimentadas por uma fonte de energia ilimitada e estejam bem protegidas contra as condições atmosféricas, não necessitam de manutenção física. Contudo, terão de manter um *software* seguro e estável e, para isso, a sua atualização é imperativa e poderá ser feita remotamente.

Tendo em conta a lista de vantagens apresentada nesta secção, a implementação da solução de *Smart Parking* proposta seria, certamente, um grande investimento para qualquer governo ou empresa da cidade. Porém, existem algumas contrariedades que se devem estudar antes de partir para um projeto desta dimensão.

4.3.2. Desvantagens

Um sistema tão invasivo como o *Smart Parking* não apresenta apenas vantagens na sua integração. Existem alguns aspetos que é preciso ter em conta antes de implementar um sistema deste tipo nas cidades:

- Pode ser um pouco confuso para utilizadores desconhecidos – Como qualquer novidade no mundo aplicacional, a aceitação por parte dos utilizadores poderá ser lenta, no entanto, em conjunto com uma aplicação de pagamento já existente, este facto será, com certeza, suprimido;

- A informação poderá ser menos precisa nas horas de ponta – Nas horas em que a procura de lugares de estacionamento é elevada, os acessos à aplicação e as alterações constantes de estado de lugares pode dificultar o processamento de dados provenientes do servidor. Por vezes, o utilizador poderá ser induzido em erro com informações que não são totalmente corretas;

- Reservas não serão possíveis ao princípio – Esta é, provavelmente, a questão mais pertinente que um possível utilizador colocará. Sem dúvida que a reserva de lugares potenciaria ainda mais o conceito de *Smart Parking*, porém é algo que poderá criar alguns conflitos e problemas entre os condutores. Esta será uma questão abordada com mais detalhe com o intuito de facilitar um trabalho futuro sobre esta matéria;

- Impedimento prematuro de estacionamento durante a instalação – A manutenção da cidade é reconhecida, pela população em geral, como um problema comum nas cidades e, ao longo do tempo, tem sido atenuado com o desenvolvimento na área de construção civil. Por conseguinte, considera-se este ponto uma desvantagem forçada, pois, neste momento, não é possível encontrar uma solução que não cause importuno aos condutores diários numa certa zona;
- Zonas férreas podem interferir com a comunicação – As zonas férreas podem interferir com a comunicação entre os sensores de estacionamento e as *gateways*, pois os campos eletromagnéticos gerados nessas zonas distorcem as ondas emitidas pelos dispositivos, impedindo a receção e emissão corretos dos dados na *gateway*. Assim sendo, a tecnologia ZigBee não será eficaz nestas zonas, o que invalida o uso dos sensores de estacionamento propostos, anteriormente, neste capítulo;
- Controlo das zonas residenciais – A identificação de condutores com dísticos para zonas residenciais poderá ser um problema de controlo do estacionamento nas zonas tarifadas. Apesar de este caso ter sido apresentado pela empresa EMEL, esta dissertação considera que o problema poderá ser mitigado, facilmente, através da identificação da matrícula na própria aplicação e, efetuando o cruzamento de dados, possibilita a diferenciação entre condutores com direito a estacionamento em zonas residenciais e outros que não possuem esta regalia. No futuro, o uso de um aparelho no veículo que comunica com o sistema de *Smart Parking* (através do sensor de estacionamento, *gateway* ou diretamente pela Internet) poderá resolver este problema, porém será, seguramente, mais dispendioso.

Como em qualquer projeto de *software*, as falhas e os erros são inevitáveis e, com o tempo, vão sendo retificados para oferecer a melhor experiência ao utilizador. Algumas destas desvantagens são comuns a qualquer projeto de *Smart Cities* e a outros fora deste âmbito, podendo vir a ser colmatadas no futuro com a evolução das tecnologias e dos métodos de trabalho.

4.4. Validação da solução apresentada

Após a realização de alguns testes, verificou-se que o protótipo funciona como esperado. Todos os componentes interligam-se de forma coesa e os dados transmitidos e recebidos na aplicação nunca são afetados. Foram efetuados testes ao funcionamento global do sistema, com todos os componentes ligados e em comunicação entre si. Testaram-se, primeiramente, os sensores de estacionamento para verificar se os dados eram gerados, de forma coerente, através da visualização dos mesmos na consola de *output.* De seguida, verificou-se a transmissão de dados para a *gateway*, onde se detetaram alguns erros de envio e recepção. Com a regulação da taxa *baud* e, imediatamente, a validação dos dados na *gateway,* partiu-se para o teste do alcance dos sensores de estacionamento e o funcionamento da configuração Router que permite o envio de dados entre os próprios nós da rede. Após estes testes, concluiu-se que a rapidez de resposta a alterações do estado de sensores está dentro dos padrões mínimos e a tecnologia ZigBee garante a comunicação entre os componentes físicos a tempo inteiro. O alcance da rede de sensores de estacionamento construída para o protótipo é aceitável. Porém, os obstáculos podem influenciar, negativamente, a comunicação. Não foi possível efetuar testes em ambiente real. Contudo, os resultados obtidos superaram as espectativas.

Para validar a afirmação anterior, foi apresentada uma proposta de reunião com a empresa EMEL, em pareceria com a CML, que mostrou interesse em compreender o que foi elaborado. O responsável da empresa que prontamente se disponibilizou a estar presente nareunião revelou algumas informações úteis e colocou questões importantes sobre as vantagens que este sistema teria em relação a outros existentes. O primeiro argumento utilizado foi o facto de este sistema conseguir ser aberto o suficiente para incluir outros componentes, designadamente sensores e dispositivos diversos para outras medições ou deteção diferente da implementada no protótipo. Apresentar uma solução de baixo custo foi, também, um dado impressionante. Todavia não é viável, em larga escala, na ótica da EMEL, pois são milhares de lugares de estacionamento espalhados pela cidade, ou seja, muito investimento a ser introduzido. Uma questão colocada pelo responsável da EMEL, em relação à instalação das *gateways*, aponta para o facto de a CML não autorizar a colocação destes aparelhos nos postes luminosos da cidade. Em resposta a esta adversidade, afirmou-se que a solução passaria pela colocação destes dispositivos nos parquímetros situados nas zonas de estacionamento

tarifadas. Assim, não haverá nenhuma objeção, por parte dos responsáveis da CML, e anula-se, diretamente, a contrariedade colocada em relação à alimentação das *gateways*, pois os parquímetros utilizam painéis solares para obtenção de energia. Foi revelado, ainda, que a Avenida da República é uma das zonas de Lisboa com maior taxa de ocupação de lugares ao longo do dia e considerada uma forte possibilidade para integração primordial de um sistema de *Smart Parking*. Porém, a primeira fase passará pela realização de testes com várias soluções na zona destinada para o efeito. Assim, a recolha destes dados determinará se a decisão parte pela introdução do sistema na cidade (a começar na Avenida da República) ou se será considerada impraticável por razões económicas ou outras encontradas no decorrer da análise de resultados.

Resumidamente, a reunião revelou-se de extrema importância para compreender alguns aspetos consideráveis em conjunto com uma entidade interessada em investir num projeto de *Smart Parking*. Os argumentos apresentados em defesa do sistema proposto e das tecnologias incluídas foram considerados esclarecedores relativamente a algumas dúvidas ainda persistentes. Contudo, não foi apontado como solução viável para um sistema real por ser demasiado rudimentar. Não obstante, o trabalho efetuado foi altamente elogiado, devido ao resultado conseguido, à abertura em termos de *hardware* e *software* – que aumenta as possibilidades dentro do conjunto de sensores e dispositivos eletrónicos que se podem incluir – e ao baixo custo de todo o protótipo.

5. Conclusões

Esta dissertação teve como principal objetivo a apresentação de uma solução de *Smart Parking* com o custo mais baixo possível e aberta em relação ao *hardware* e *software* utilizados. O estudo de sistemas existentes e o protótipo realizado permitiram concluir que o conceito de *Smart Parking* será uma realidade no futuro e que beneficiará, seguramente, os cidadãos e as *Smart Cities*. Procurou-se desenvolver uma solução aberta, que permita a integração com outros componentes, para além dos que foram escolhidos para o protótipo da solução, tornando-a completamente escalável. As outras componentes poderão ser alvo de alterações. Todavia, neste momento, a dissertação considera as escolhas mais viáveis e fiáveis para um sistema de *Smart Parking* final. A integração do sistema em Lisboa ou noutra cidade poderá redefinir o paradigma que afirma que o tráfego e o estacionamento são os maiores problemas das cidades, valendo, inquestionavelmente, a continuação do seu estudo para melhorar as soluções existentes e propostas ao longo do tempo.

5.1. Principais conclusões

Para mitigar o problema do estacionamento nas cidades e a circulação parasita em busca de lugares, esta dissertação propôs uma solução para um sistema de *Smart Parking* a partir do estudo de sistemas existentes e análise do mercado em redor deste conceito.

Em resposta à primeira questão de investigação colocada, inicialmente, que se refere à possibilidade de desenvolver uma plataforma tecnológica, baseada em sensores e em tecnologias móveis que seja capaz de gerir, eficazmente, o estacionamento inteligente nas cidades, demonstrou-se, através da construção do protótipo funcional, ser possível elaborar um sistema de *Smart Parking* totalmente funcional e implementá-lo nas cidades. A proposta apresentada traz garantias de funcionamento do sistema, assim como a gestão do estacionamento. No caso lisboeta, a empresa EMEL, que gere o

estacionamento na cidade, contém o seu próprio sistema de gestão e com a entrada de um sistema de *Smart Parking* poderão obter dados que completam os obtidos diariamente.

A segunda questão de investigação está relacionada com a solução apresentada ser aberta o suficiente que permita a inclusão de outros componentes tendo em conta as necessidades de cada cidade e dos cidadãos. A utilização de Arduino e Raspberry Pi para os sensores de estacionamento e *gateways,* respetivamente, possibilita a inclusão de outros tipos de sensores e extrair dados a partir de métodos similares aos aplicados durante a implementação do protótipo funcional. Ou seja, em relação ao *hardware* do protótipo desenvolvido, o Arduino permite a ligação de outros tipos de sensores consoante as características que se pretendem implementar. O uso de sensores infravermelhos serviu como elemento para o protótipo, porém existe a possibilidade de substitui-lo por um sensor geomagnético (como o RM3100, referido no capítulo 3), que apenas deteta materiais ferromagnéticos, trazendo maior dimensão e completude ao protótipo desenvolvido durante esta dissertação. Relativamente ao *software*, é possível desenvolver os programas de forma diversa, utilizando linguagens de programação ou *frameworks* distintas e outros métodos e algoritmos, mesmo com a utilização das próprias linguagens de programação do protótipo.

Houve uma tentativa de apresentar uma proposta com o menor custo possível. Porém, a solução exposta transmite viabilidade e versatilidade, no que toca ao seu desenvolvimento e implementação. O sensor de estacionamento continua a ser problemático devido ao seu custo elevado e é necessário, ainda, criar uma solução inovadora para esta componente do sistema de *Smart Parking*.

Na fase final desta dissertação, foi elaborado um artigo científico sobre o conteúdo exposto ao longo do trabalho, tendo este sido submetido, aceite e apresentado na conferência MCCSIS 2018 em Madrid (http://smartcities-conf.org/).

5.2. Considerações

O protótipo construído ao longo da dissertação demonstrou o uso de tecnologia ZigBee para comunicação entre os elementos físicos do sistema. Este protocolo de comunicação é, comumente, utilizado em outros sistemas de *Smart Parking* existentes e em diferentes aplicações para *Smart Cities*. Considera-se o método mais adequado de

comunicação, em termos de poupança de energia e de custos, apesar de algumas desvantagens que possam existir.

Demonstrou-se que a implementação de um sistema de *Smart Parking* de baixo custo e aberto é exequível, através da prova de conceito divulgada pelo sucesso resultante do protótipo funcional. A solução proposta, apesar de não ser ótima, é, teoricamente, viável, o que obriga a testes práticos do seu funcionamento para que sejam retiradas as conclusões apropriadas. O desenvolvimento de uma aplicação que revela, previamente, a disponibilidade de lugares de estacionamento nas zonas com maior taxa de ocupação e procura diária é algo que poderá agradar aos cidadãos e que, sem dúvida, é uma solução vantajosa para reduzir a circulação parasita associada à busca de lugares de estacionamento.

Finalmente, esta dissertação defende e acredita que implementar este sistema em Lisboa poderá trazer muitos benefícios para a cidade, independentemente das dificuldades iniciais e do investimento elevado.

5.3. Limitações do Trabalho

Ao longo da dissertação, foi sempre defendido que a abertura dos sensores de estacionamento para o *hardware* e *software* é uma vantagem clara desta proposta. Porém, existem algumas limitações que são importantes enumerar e poderão servir de estudo para um outro trabalho no futuro.

O sensor de estacionamento contém elementos bastante simples, no entanto o fornecimento de energia para o mesmo não apresenta a autonomia desejada. Para efeitos de prototipagem, é suficiente, mas, caso fosse implementado em ambiente real, não apresentaria as qualidades necessárias. A comunicação ZigBee pode apresentar alguns problemas, como foi referido anteriormente. Os obstáculos e as zonas férreas podem interferir com a comunicação, contudo esta dissertação defende o uso desta tecnologia. No caso da *gateway*, ao receber dados provenientes dos sensores de estacionamento, poderão ocorrer alguns erros na receção de dados, nomeadamente, se existir um elevado número de sensores de estacionamento dentro da rede sensorial. Com os ajustes feitos à taxa *baud* durante os testes, foi possível atenuar um pouco este problema. Contudo, acredita-se que não está totalmente resolvido para uma situação de implementação em ambiente real. No que toca à aplicação móvel, o desenho pode ser melhorado e é possível que surjam alguns *bugs* durante a sua execução.

5.4. Trabalho futuro

Durante o percurso da dissertação, foi possível analisar alguns dos maiores problemas que as empresas de estacionamento e as próprias cidades enfrentam. Este trabalho demonstrou que existem vantagens na implementação de um sistema de *Smart Parking* em qualquer cidade e, consequentemente, possibilitar o crescimento das *Smart Cities*. Infelizmente, esta dissertação não conseguiu resolver algumas questões importantes, nomeadamente, a reserva de lugares. Por enquanto, isto poderá ser algo considerado impossível de implementar. Todavia, poderá ser um fator determinante para a aceitação total dos cidadãos. O desenvolvimento do *software* de gestão do estacionamento também não foi alvo de estudo, pois considerou-se um elemento secundário para o sistema de *Smart Parking* em estudo, devido à sua relação direta com os sistemas de gestão existentes. Porém, o desenvolvimento de um protótipo para um *software* de gestão de estacionamento poderá ser interessante para as empresas de estacionamento atuais, pois poderão emergir algumas características ou algoritmos que façam parte da evolução do *software* em atividade. Outra questão não abordada durante o trabalho foi a criação de uma tecnologia inovadora para o sensor de estacionamento. O desenvolvimento de uma tecnologia viável e de baixo custo que não existe no mercado pode despoletar o interesse para o investimento neste tipo de sistemas, o que levará, com certeza, as entidades de estacionamento a seguir este conceito. Caso este seja um tema a considerar, propõe-se o estudo para um sensor de estacionamento com uma tecnologia de baixo custo por lugar de estacionamento, o mais autónomo possível e viável, ou outra tecnologia de controlo de parqueamento e tráfego em redor, tendo em conta algumas questões legais que possam surgir, como por exemplo, o direito à privacidade.

Para fomentar a investigação contínua sobre este tema, o código desenvolvido para o protótipo foi colocado no repositório GitHub, permitindo o acesso público a quem poderá ter interesse em efetuar um estudo mais aprofundado sobre *Smart Parking* ou outras tecnologias inerentes a *Smart Cities* e IoT (https://github.com/joaomf24/smartpark_lx).

Encontrar uma solução que satisfaça estes requisitos poderá ser a chave para o sucesso do *Smart Parking* a nível Mundial e esta dissertação defende, proactivamente, este conceito e está aberta ao estudo de soluções vantajosas para o futuro das *Smart Cities* e dos seus cidadãos.

Bibliografia

[1] Idris, Mohd & Y.Y, Leng & E.M, Tamil & N.M, Noor & Razak, Zaidi. (2009). Car Park System: A Review of Smart Parking System and its Technology. Information Technology Journal. 8. 10.3923/itj.2009.101.113

[2] Chourabi, H., Nam, T., Walker, S., Gil-Garcia, J. R., Mellouli, S., Nahon, K., ... & Scholl, H. J. (2012, January). Understanding *Smart Cities*: An integrative framework. In System Science (HICSS), 2012 45th Hawaii International Conference on (pp. 2289-2297). IEEE.

[3] Giuffrè, T., Siniscalchi, S. M., & Tesoriere, G. (2012). A novel architecture of parking management for *Smart Cities*. Procedia-Social and Behavioral Sciences, 53, 16-28.

[4] Câmara Municipal de Lisboa, Estacionamento em Lisboa: Análise, 2010

[5] Isabel Machado Alexandre, "Design Science Research", 2015

[6] Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., & Williams, P. (2010). Foundations for Smarter Cities. IBM Journal of Research and Development, 54(4).

[7] Toppeta, D. (2010). The *Smart City* Vision: How Innovation and ICT Can Build Smart, "Livable", Sustainable Cities. The Innovation Knowledge Foundation. Available from http://www.thinkinnovation.org/file/research/23/en/Top peta_Report_005_2010.pdf.

[8] Hancke, G. P., & Hancke Jr, G. P. (2012). The role of advanced sensing in *Smart Cities*. Sensors, 13(1), 393-425.

[9] Caragliu, A., Del Bo, C., & Nijkamp, P. (2011). *Smart Cities* in Europe. Journal of urban technology,18 (2), 65-82.

[10] Zanella, A., Bui, N., Castellani, A., Vangelista, L., & Zorzi, M. (2014). Internet of things for *Smart Cities*. IEEE Internet of Things Journal, 1(1), 22-32.

[11] Washburn, D., Sindhu, U., Balaouras, S., Dines, R. A., Hayes, N. M., & Nelson, L. E. (2010). Helping CIOs Understand "*Smart City*" Initiatives: Defining the *Smart City*, Its Drivers, and the Role of the CIO. Cambridge, MA: Forrester Research, Inc. Available from http://public.dhe.ibm.com/partnerworld/pub/smb/smart erplanet/forr_help_cios_und_smart_city_initiatives.pdf.

[12] Muthu Ramya, C & Madasamy, Shanmugaraj & Prabakaran, R. (2011). Study on ZigBee technology. 297-301. 10.1109/ICECTECH.2011.5942102

[13] Digi International, https://www.digi.com/, visitado a 29.06.2017

[14] What are Xbee Modules?, https://core-electronics.com.au/tutorials/what-are-XBee-modules.html, visitado a 29.06.2017

[15] Discover Songdo, the $35 Billion South Korean City Built to Banish Cars, https://interestingengineering.com/discover-songdo-the-35-billion-south-korean-city-built-to-banish-cars, visitado a 10.03.2018

[16] Jang, Myungjun & Suh, Soon-Tak. (2010). U-City: New Trends of Urban Planning in Korea Based on Pervasive and Ubiquitous Geotechnology and Geoinformation. 6016. 262-270. 10.1007/978-3-642-12156-2_20.

[17] A half-built futuristic 'eco-city' is sitting empty in the Arabian Desert, http://www.businessinsider.com/masdar-city-photos-of-abu-dhabis-green-city-in-the-arabian-desert-2016-7/, visitado a 10.03.2018

[18] Masdar – A Mubadala Company, http://www.masdar.ae/, visitado a 10.03.2018

[19] Frank Salt – Smart City Malta, https://franksalt.com.mt/high-profile-developments/smart-city-malta/, visitado a 10.03.2018

[20] Smart City Strategy: PlanlT Valley (Portugal), http://www.urenio.org/2015/01/26/smart-city-strategy-planlt-valley-portugal/, visitado a 10.03.2018
[21] PlanIT Valley: The smartest city never been built, http://smartcityhub.com/governance-economy/planit-valley-the-smartest-city-never-been-built/, visitado a 10.03.2018
[22] Fraifer, Muftah & Fernström, Mikael. (2016). Investigation of Smart Parking Systems and their technologies
[24] Thanh Nam Pham, Ming-Fong Tsai, Duc Binh Nguyen, Chyi-Ren Dow, and Der-Jiunn Deng. "A *Cloud*-based Smart-parking System based on Internet-of-Things technologies." digital object identifier 10.1109/access.2015.2477299 (2015): 1581-1591.
[25] I. F. Akyildiz, I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges", Ad Hoc Netw., vol. 2, no. 4, pp. 351-367, Oct. 2004.
[26] World Sensing – Fastprk, http://www.worldsensing.com/product/fastprk/, visitado a 19.12.2016
[27] SmartSpot Gateway – Helping Cities to Operate Efficiently, https://www.smartparking.com/technologies/smartspot-*gateway*, visitado a 16.06.2017
[28] Libelium: Meshlium Xtreme Technical Guide, http://www.libelium.com/downloads/documentation/meshlium_technical_guide.pdf, visitado a 19.12.2016
[29] The Top Mobile Backend-as-a-Service (MBaaS) Providers, http://www.businessofapps.com/top-mobile-*backend*-service-mbaas-providers/, visitado a 07.04.2017
[30] Firebase, https://firebase.google.com, visitado a 08.04.2017
[31] 10 Frameworks for Mobile Hybrid Apps, https://blog.jscrambler.com/10-frameworks-for-mobile-hybrid-apps/, visitado a 07.04.2017
[32] What is React Native?, https://www.techworld.com/apps-wearables/what-is-react-native-3625529/, visitado a 12.02.20018
[33] Sencha, https://www.sencha.com, visitado a 07.04.2017
[34] Appcelerator, https://www.appcelerator.com/, visitado a 07.04.2017
[35] ParkWhiz, https://www.parkwhiz.com, visitado a 20.07.2018
[36] Smart Parking Systems, http://www.smartparkingsystems.com/, visitado a 19.12.2016
[37] Adelantado, Ferran, Xavier Vilajosana, Pere Tuset, Borja Martínez and Joan Melià. "Understanding the Limits of LoRaWAN." *IEEE Communications Magazine* 55 (2017): 34-40.
[38] CivicSmart – Wireless Vehicle Detection Sensors, http://www.civicsmart.com/wireless-vehicle-detection-sensors/, visitado a 19.12.2016
[39] Multifrota Parking: Sistemas de gestão de parques de estacionamento, http://multifrota.pt/multifrota-parking/sistemas-de-gestao-de-parques-de-estacionamento-barreiras/, visitado a 29.11.2016
[40] Infocontrol: Sistema de ajuda ao estacionamento, http://www.infocontrol.pt/1176/sistema-de-ajuda-ao-estacionamento.htm, visitado a 29.11.2016
[41] Libelium: Smart Parking Technical Guide – Waspmote, http://www.libelium.com/v11-files/documentation/waspmote/smart-parking-sensor-board_eng.pdf, visitado a 19.12.2016

[42] J. M. Fernandes, C. Serrão, N. Garrido, "Development of a Low-Cost Smart Parking solution for Smart Cities", International Conference on Connected Smart Cities, 2018
[43] RM3100 – Geomagnetic Sensor, https://www.pnicorp.com/rm3100/, visitado a 13.06.2017
[44] What is the difference between an end device, a router, and a coordinator?, https://www.silabs.com/community/wireless/zigbee-and-thread/knowledge-base.entry.html/2012/07/02/what_is_the_differen-IYze, visitado a 25.07.2017
[45] PlacePod – Smart Parking Sensor, https://www.pnicorp.com/placepod/, visitado a 13.06.2017
[46] M. Banzi, Geting started with Arduino, 2nd Edition, Make:Books, 2011.
[47] Arduino IR Obstacle Sensor: Tutorial and Manual, http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-ir-obstacle-sensor-tutorial-and-manual/, visitado a 29.06.2017
[48] Which jumper to set on the ITEAD XBee shield v1.1 for use with a 3.3V Arduino, https://vxlabs.com/2018/03/23/which-jumper-to-set-on-the-itead-XBee-shield-v1-1-for-use-with-a-3-3v-arduino/, visitado a 25.07.2017
[49] Raspberry Pi, https://www.raspberrypi.org/, visitado a 29.06.2017
[50] Raspberry Pi – GPIO, https://www.raspberrypi.org/documentation/usage/gpio/, visitado a 22.01.2018
[51] XBee interfacing Raspberry Pi Model 2, https://electronicsforu.com/electronics-projects/XBee-interfacing-raspberry-pi-model-2/2, visitado a 30.06.2017
[52] DreamFactory – Features, https://www.dreamfactory.com/features, visitado a 07.04.2017
[53] Ionic Creator, https://creator.ionic.io, visitado a 11.07.2017
[54] 10 Benefits of a Smart Parking Solution, http://www.plasmacomp.com/blogs/benefits-of-smart-parking-solution, visitado a 05.12.2017
[55] Smart Parking Solution Inc. – The advantages of Smart Parking, http://www.smartparkingsolution.com/advantages/, visitado a 05.12.2017
[56] AWS vs Azure vs Google Cloud Pricing: Compute Instances, https://www.rightscale.com/blog/cloud-cost-analysis/aws-vs-azure-vs-google-cloud-pricing-compute-instances, visitado a 07.04.2017

Anexo A – Capturas de ecrãs das aplicações de *Smart Parking* existentes

Figura A.1.1: 'Smart Parking (ZTE G&E)' – Mapa da zona de *Smart Parking*.

Figura A.1.2: 'Smart Parking (ZTE G&E)' – Opções de pagamento do lugar.

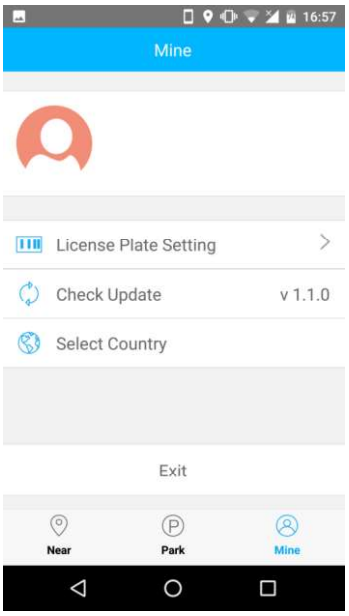

Figura A.1.3: 'Smart Parking (ZTE G&E)' – Perfil do utilizador.

Figura A.2.1: 'SmartParking Warsaw' – Mapa da zona de *Smart Parking*.

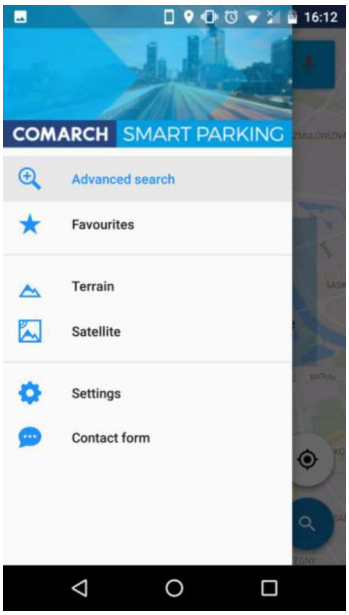

Figura A.2.2: 'SmartParking Warsaw' – Menu lateral.

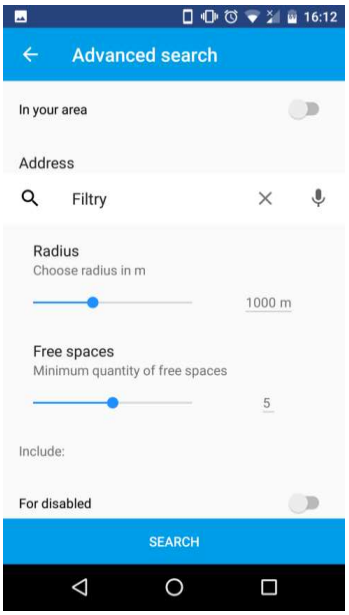

Figura A.2.3: 'SmartParking Warsaw' – Pesquisa avançada.

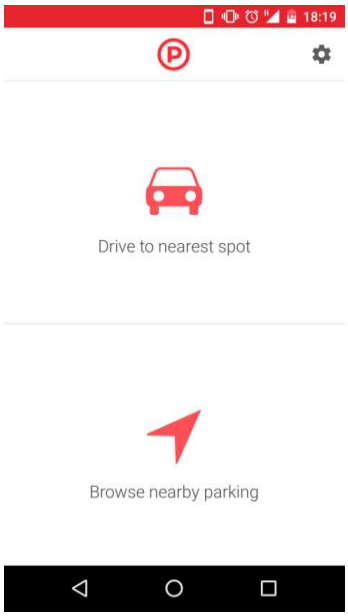

Figura A.3.1: 'Smart Parking Dubrovnik' – Ecrã inicial.

Figura A.3.2: 'Smart Parking Dubrovnik' – Mapa da zona de *Smart Parking*.

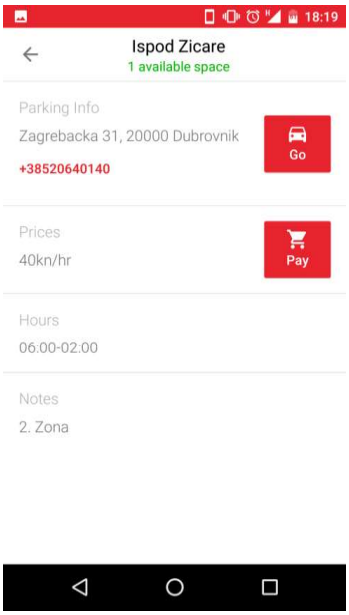

Figura A.3.3. 'Smart Parking Dubrovnik' – Informações de estacionamento.

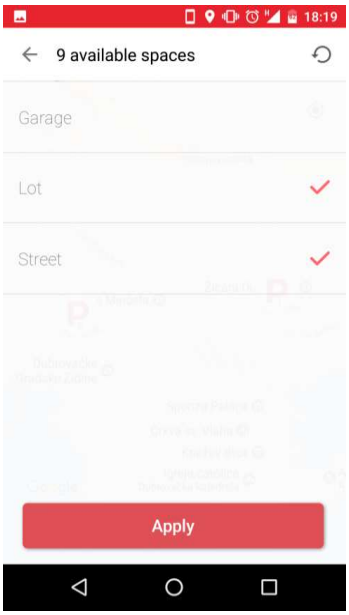

Figura A.3.4: 'Smart Parking Dubrovnik' – Filtro para o tipo de parqueamento.

Figura A.4.1: 'SmartPark Pozuelo' – Localização do utilizador.

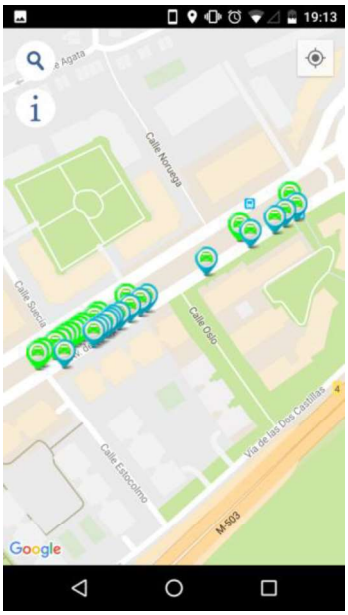

Figura A.4.2: 'SmartPark Pozuelo' – Mapa da zona de *Smart Parking*.

Figura A.4.3: 'SmartPark Pozuelo' – Pesquisa de zonas de estacionamento.

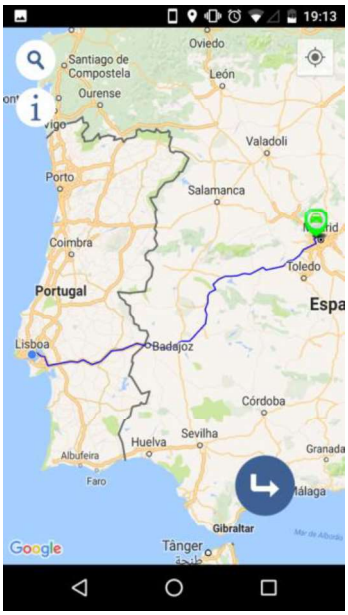

Figura A.4.4: 'SmartPark Pozuelo' – Trajeto até ao lugar de estacionamento.

Figura A.4.5: 'SmartPark Pozuelo' – Informações sobre a aplicação.

Anexo B – Excertos de código-fonte do protótipo

```
.....
@Injectable()
export class DataBaseService {
    private _gateways$: any;
    private _db: any;
    private _gatewaysRef: any;
    public data    : FirebaseListObservable<any[]>;
    public zonasList : ZonaModel[] = [];
    public isDataLoaded:boolean = false;
    public loading = this.loadingCtrl.create({
       content:  this.translationServ.currentLang == 'pt'?  'A
contatar servidor...': 'Connecting to server...'
    });
    constructor(private angFire: AngularFireDatabase,
        private loadingCtrl: LoadingController,
        private translationServ: TranslateService,
        private perfilServ:PerfilService,
        private utilsServ:UtilsService) {
        this.loading.present();
        this.data = this.angFire.list('/zonas');
        this._db = firebase.database().ref('/');
        this._gatewaysRef = firebase.database().ref('zonas');
        this._gatewaysRef.on('child_changed', this.handleData, this);
        this._gateways$ = new ReplaySubject();
    }
.....
    showData(){
        let dataLoaded = this.isDataLoaded;

        return new Promise(resolve => {
            this.data.forEach(data => {
                if(!this.isDataLoaded){
                    for(let d of data){
                        this.zonasList.push(new ZonaModel(d.id,
d.descricao, this.getGateways(d.gateways)));
                    }
                    this.isDataLoaded = true;
                }
                return true;
            })
            resolve(this.zonasList);
        });
    }

    updateData(data){
        return new Promise(resolve => {
            for(let zona of this.zonasList){
                if(zona.id == data.id){
                    zona.markersList =
this.getGateways(data.gateways);
                    zona.description = data.descricao;
                    for(let marker of zona.markersList)
                        marker.setNumFreeSpots();
                }
            }
            resolve(this.zonasList);
        });
    }
.....
}
```

Figura B.1: Ficheiro 'database.service.ts'

```
import { Injectable } from "@angular/core";
import { FavoritoModel } from "../models/favorito";
import { Http, Response } from "@angular/http";
import { AuthService } from "./auth.service";
import 'rxjs/Rx';

@Injectable()
export class FavoritosService {
    private favoritos : FavoritoModel[] = [];

    constructor(private http: Http,
        private authService: AuthService){}

    storeFavoritos(token: string){
        const userId = this.authService.getActiveUser().uid;
        return this.http
            .put('https://smartpark-lx.firebaseio.com/users/' + userId
+ '/favoritos.json?auth=' + token, this.favoritos)
            .map((response: Response) => {
                return response.json();
            });
    }

    fetchFavoritos(token: string){
         const userId = this.authService.getActiveUser().uid;
        return this.http.get('https://smartpark-
lx.firebaseio.com/users/' + userId + '/favoritos.json?auth=' + token)
        .map((response: Response) => {
            const favoritos: FavoritoModel[] = response.json() ?
response.json() : [];
            return favoritos;
        })
        .do((favoritos: FavoritoModel[]) => {
            if(favoritos){
                this.favoritos = favoritos;
            }
            else{
                this.favoritos = [];
            }
        });
    }
 }
```

Figura B.2: Ficheiro 'favoritos.service.ts'

```
import firebase from 'firebase';

export class AuthService{
    signup(email: string, password: string){
        return firebase.auth().createUserWithEmailAndPassword(email,
password);
    }

    signin(email: string, password: string){
        return firebase.auth().signInWithEmailAndPassword(email,
password);
    }

    logout(){
        firebase.auth().signOut();
    }

    getActiveUser(){
        return firebase.auth().currentUser;
    }
}
```

Figura B.3: Ficheiro 'auth.service.ts'

Anexo C – Serviços de *backend*

AWS vs. Azure vs. Google Discounted/Year

Resource Type (us-east, Linux)	AWS Instance	Azure Instance	Google Instance	AWS 1Y RI No Upfront Annual	Azure EA 30% Annual	Google 100% SUD Annual	AWS RI /GB RAM	Azure EA /GB RAM	Google SUD /GB RAM
Standard 2 vCPU w SSD	m3.large	D2 v2	n1-standard-2	$832.20	$699.05	$1,594.20	$104.03	$99.86	$212.56
Highmem 2 vCPU w SSD	r3.large	D11 v2	n1-highmem-2	$919.80	$913.67	$1,753.63	$61.32	$65.26	$134.89
Highcpu 2 vCPU w SSD	c3.large	F2	n1-highcpu-2	$639.48	$607.07	$1,447.03	$170.53	$151.77	$803.91
Standard 2 vCPU no SSD	m4.large	D2 v2	n1-standard-2	$648.24	$699.05	$613.20	$81.03	$99.86	$81.76
Highmem 2 vCPU no SSD	r4.large	D11 v2	n1-highmem-2	$814.68	$913.67	$772.63	$53.42	$65.26	$59.43
Highcpu 2 vCPU no SSD	c4.large	F2	n1-highcpu-2	$613.20	$607.07	$466.03	$163.52	$151.77	$258.91

As of Dec 2, 2016 *Source: RightScale*

Figura C.1: Comparação entre os serviços de *backend* AWS, Azure e GCP [56].