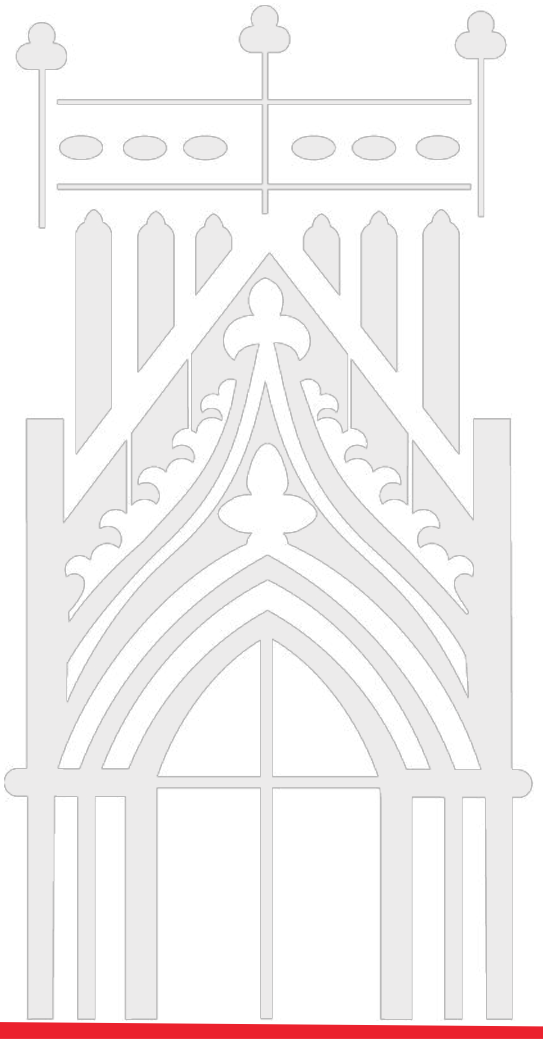
IPG Politécnico da Guarda
Polytechnic of Guarda

Mestrado em Computação Móvel

Meu Táxi



João Paulo Campos Miranda

junho | 2013

Escola Superior
de Tecnologia e Gestão

Abstract

This document describes the work done within the Master's degree in Mobile Computing at the Polytechnic Institute of Guarda and consisted of developing an application for mobile devices, especially smartphones with the Android operating system, allowing to a citizen to call quickly and easily a taxi through their own mobile device, without using the service provider's or service company's telephone numbers, or to go to a specific location to access this type of service, commonly called a taxi rank.

The application takes advantage of the geolocation system, usually known as GPS, to determine the citizen´s position; thereby facilitating the usability of this type of application (it is not necessary to input manually the citizen's location).

It was also developed a platform that stores customer orders and with which the mobile application communicates, either to save customers' orders, or to get the taxis data. With this platform is also possible to select the taxi that will perform the service, based on the distance and estimated time it takes to reach the customer.

With the aim of approaching the mobile application as real as possible, was made a survey of prices charged by taxis, to be able to perform the calculation of the amount expected to be paid by the customer, depending on the distance it will go, being shown a brief summary with this information as well as a set of options that the client has in order to satisfy your request.

Keywords Android, Taxi, Mobile Computing, Urban Mobility, GPS

Resumo

O presente documento descreve o trabalho realizado no âmbito do Mestrado em Computação Móvel do Instituto Politécnico da Guarda e consistiu no desenvolvimento de uma aplicação para dispositivos móveis, nomeadamente para *smartphones* com o sistema operativo *Android*, que permite, a um determinado cidadão, chamar um táxi de forma simples e rápida com recurso ao uso do seu próprio dispositivo móvel, sem que o cidadão tenha necessidade de ter um contacto telefónico do prestador do serviço ou da empresa responsável pela prestação desse serviço ou ainda de se deslocar a um local específico para ter acesso a este tipo de serviço, vulgarmente chamado de praça de táxis.

A aplicação tira partido do sistema de geolocalização, usualmente conhecido por GPS, para determinar a posição do cidadão, facilitando desta forma a usabilidade deste tipo de aplicação (uma vez que não será necessário a introdução manual da localização do cidadão).

Foi desenvolvida ainda uma plataforma que armazena os pedidos dos clientes e com a qual a aplicação móvel comunica, quer para guardar os pedidos efetuados pelos clientes, quer para obter os dados que dizem respeito aos táxis. Nesta plataforma é ainda possível selecionar o táxi que irá realizar o serviço, tendo por base a distância e o tempo estimados que demora até chegar ao cliente.

Com o objetivo da aplicação móvel se aproximar do mais real possível, foi efetuado um levantamento dos preços cobrados pelos táxis, para que seja possível efetuar o cálculo do valor estimado a pagar pelo cliente, consoante a distância que irá percorrer, sendo mostrado um breve resumo com esta informação bem como um conjunto de opções que o cliente dispõe a fim de satisfazer o seu pedido.

Palavras-chave: Android, Táxi, Computação Móvel, Mobilidade Urbana, GPS

1. INTRODUÇÃO

A escolha do presente trabalho surgiu na verificação da necessidade de uma aplicação prática, económica e facilitadora no que respeita ao acesso de um meio de transporte, neste caso concreto, o táxi, mediante a utilização de dispositivos móveis.

Com o surgimento dos *smartphones* e *tablets* desenvolveram-se, consequentemente, aplicações para este tipo de equipamentos, as quais têm vindo a aumentar de forma bastante considerável. Este crescimento deve-se à grande evolução das características dos *smartphones*, à crescente popularidade e à mudança da forma como as pessoas interagem entre si e com o mundo. Assim, é de extrema importância o desenvolvimento de aplicações que sirvam de forma direta o homem, simplificando as suas tarefas diárias e que, desta maneira, possam melhorar a sua qualidade de vida.

Os dispositivos móveis fazem, assim, parte do nosso quotidiano e estão rapidamente a integrar-se nas atividades que realizamos nas mais diversas áreas. Estes equipamentos são cada vez mais sofisticados e fornecem diversas tecnologias que permitem ligações de banda larga, sensores e funcionalidades de geolocalização, que nos abrem inúmeras oportunidades de exploração e desenvolvimento.

No caso concreto da aplicação aqui realizada, Meu Táxi, tem a capacidade de comunicação e interação entre o cidadão e a oferta do serviço, podendo ser utilizada em qualquer lugar bastando para isso ter acesso à internet e, obviamente, o próprio dispositivo móvel.

Espera-se, com este trabalho, elucidar para a importância da utilização das novas tecnologias no âmbito da Mobilidade Urbana, em particular, na utilização dos meios de transporte disponíveis ao público num espaço urbano, em particular do táxi, bem como para o desenvolvimento de aplicações em dispositivos móveis.

1.1. Instituição de acolhimento

O Centro de Computação Gráfica é uma associação privada sem fins lucrativos com sede no *campus* da Universidade do Minho, em Guimarães, cuja principal missão é a transferência de tecnologia entre as Universidades e as Empresas. Os objetivos do CCG centram-se assim na aplicação da Investigação de base produzida nas Universidades, em novos desafios colocados à Industria de forma a contribuir quer para o reforço de competências das empresas, quer para a criação de novos produtos diferenciadores ao nível da Inovação.

O CCG encontra-se estruturado em cinco Departamentos de Investigação Aplicada (DIAs) entre os quais o DIA UMC – *Urban and Mobile Computing*, direcionado para as questões (ao nível da investigação e inovação) relacionados quer com a Computação Móvel, quer com a Computação Urbana; pelo que o desafio abordado neste trabalho é desta forma de interesse para a instituição.

1.2. Motivação

Mais do que na era da informação e da tecnologia, vivemos sobretudo na era do conhecimento. Por este motivo, considerámos essencial debruçarmo-nos sobre uma temática que nos desse, para além de capacidades técnicas, também a sabedoria para a podermos concretizar e desenvolver.

Assim, a principal motivação da realização deste trabalho prende-se, acima de tudo, com a possibilidade de explorarmos as novas tecnologias associadas à utilização de dispositivos ubíquos, como é o caso dos *smartphones*, neste caso em particular os que dispõem do sistema operativo *Android*, bem como facilitar o acesso e reforçar a utilização de um meio de transporte público de aluguer - o táxi - como uma solução possível de mobilidade a considerar em espaços urbanos com graves problemas ao nível da mobilidade.

Ainda no âmbito da motivação devemos realçar um outro aspeto que nos parece ser também ele de extrema relevância que tem a ver com o planeamento, quer do espaço urbano em si (tendo em consideração os comportamentos de mobilidade dos cidadãos), quer das soluções de mobilidade presentes no espaço urbano (tendo em consideração as necessidade dos cidadãos).

1.3. Solução

A solução proposta consiste no desenvolvimento de uma aplicação para dispositivos móveis com o sistema operativo *Android*, onde qualquer cidadão poderá "chamar" um táxi, através da internet, consoante a sua localização geográfica, obtida através das coordenadas GPS, e escolher, de um conjunto de opções, aquelas que mais se adequam às suas necessidades. A solução contempla ainda o desenvolvimento de um *BackOffice*, com o qual a aplicação móvel comunica através da internet, que faça a gestão dos pedidos dos cidadãos e ao mesmo tempo permita selecionar o táxi que melhor se adequa ao pedido, tendo em conta a distância e o tempo que demora a chegar junto de quem efetuou o pedido. O *BackOffice* é ainda responsável por (através da recolha de informação que será extraída da base de dados, tendo em conta os pedidos de serviço de táxi) possibilitar a realização de estudos da mobilidade dos cidadãos nos vários espaços urbanos.

1.4. Estrutura da tese

Este trabalho encontra-se estruturado em cinco capítulos. No segundo capítulo faz-se uma análise do estado da arte abordando conceitos como a acessibilidade, mobilidade urbana e estilos de vida, enumerando também os problemas socioeconómicos e problemática. É feito ainda um estudo e respetiva comparação de algumas aplicações que visam facilitar a mobilidade dos cidadãos no espaço urbano, no

que diz respeito ao transporte público de aluguer - o táxi. São também mostradas algumas aplicações que facilitam a mobilidade urbana dos cidadãos no que diz respeito a outro tipo de transportes públicos, como o caso dos autocarros e do metro.

De seguida, no terceiro capítulo, são definidos os problemas, objetivos previstos, metodologia e resultados esperados a atingir com a realização do presente trabalho.

No quarto capítulo são enumeradas as diversas tecnologias utilizadas no desenvolvimento da aplicação.

O quinto capítulo faz referência à descrição da implementação da solução, onde são dados a conhecer os diversos ecrãs que compõem o *FrontOffice*, bem como os vários *interfaces* que constituem o *BackOffice*.

Finalmente, no capítulo seis, são feitos alguns comentários e apresentadas algumas conclusões acerca do trabalho desenvolvido. São ainda apontadas possíveis melhorias e novas funcionalidades que poderão vir a ser adicionadas em termos de trabalho futuro.

2. ESTADO DA ARTE

2.1. Acessibilidade

Sendo este um trabalho em torno de uma aplicação móvel para a utilização de um transporte público de aluguer, será pertinente fazer um pequeno enquadramento no que respeita ao tema de acessibilidade. Este é um conceito bastante abrangente e complexo, o qual possui várias definições, sendo que estas variam dependendo tanto das aplicações como dos estudos realizados. Na verdade, este conceito flexível pode ser aplicado em diferentes áreas, tais como o planeamento urbano, o planeamento de transportes, a geografia e o marketing (Halden et al, 2005).

Segundo David Simmonds (Simmonds, 1998), acessibilidade é uma forma de medir a facilidade com que uma determinada categoria de pessoas pode chegar a um conjunto definido de destinos, a partir de uma origem (acessibilidade origem), ou a facilidade com que um determinado destino (acessibilidade destino) pode ser alcançado por um conjunto particular de indivíduos potenciais. Por sua vez, na opinião de Handy (Handy, 2004), acessibilidade é a capacidade de obter o que se necessita, com uma escolha de destinos e usando uma variedade de modos.

Após a explicação deste conteúdo tão versátil, é-nos possível identificar uma componente de acessibilidade que se revela importante para o tema em questão – a componente de transporte. Esta consiste numa descrição do sistema dos transportes (qualquer modo de transporte) bem como dos custos a ele associados, na ótica do indivíduo, relativos ao percurso efetuado entre a origem e o destino. De referir a importância de calcular estes valores e de ter acesso aos mesmos na medida em que revelam ser elementos essenciais para o próprio consumidor.

Ainda no que diz respeito ao conceito de acessibilidade, são identificadas quatro perspetivas básicas de abordagem: indicadores de acessibilidade para infraestruturas; indicadores de acessibilidade com base na localização; indicadores de acessibilidade baseados na pessoa; indicadores de acessibilidade com base na utilidade (Gama et al; 2011). Destas quatro perspetivas a que importa para o nosso estudo será única e exclusivamente os indicadores de acessibilidade com base na localização uma vez que a aplicação desenvolvida estará, também, assente neste pilar. Com efeito, esta aplicação aproveita o sensor de GPS presente nos dispositivos móveis, em particular neste caso, *smartphones* baseados em *Android*, o qual irá permitir ter a localização exata do ponto de origem bem como, após a aceitação do pedido, permitir o acesso à localização e deslocação do meio de transporte, neste caso concreto, o táxi. Tal ferramenta irá ajudar-nos a melhor compreender o conceito de mobilidade urbana e, até, alguns hábitos do quotidiano.

2.2. Mobilidade Urbana e estilos de vida

Os estilos de vida foram-se alterando ao longo dos tempos e estão em constante mudança, seja por serem influenciados pelo meio envolvente, pela cultura e tradição ou por fatores diretos e indiretos, como problemas socioeconómicos ou políticos. O mundo contemporâneo está marcado pelo acesso fácil à informação, pelas novas tecnologias e, sobretudo, pela globalização que se deve, em grande parte, à internet (Furtado e Oliveira, 2004).

Um fator que pode também condicionar os estilos de vida, tanto nos meios urbanos como rurais, diz respeito aos transportes e à mobilidade. No entanto este caso é mais flagrante em cidades, onde também a oferta é maior e mais diversificada, contando também com uma procura bastante superior e um público mais exigente e com diferentes necessidades. Uma delas é a falta de estacionamento nas grandes áreas urbanas e que é considerado como um dos principais fatores na satisfação dos

moradores (Paiva, 2006). Este é um problema cada vez maior e ao qual os transportes públicos, como é o caso do táxi, têm vindo a dar resposta.

Através das escolhas e de todo o processo que engloba mobilidade urbana podemos ter acesso ao estilo de vida que lhe está subjacente, sendo que será ainda importante conhecer e perceber quais as motivações e mecanismos que levam às decisões e opções dos cidadãos sobre as soluções que pretendem adotar no seu quotidiano. Algumas são reguladas pelo facilitismo (como por exemplo a proximidade), outras são condicionadas por fatores económicos (prioridade dos transportes públicos ao invés de viaturas próprias), outras ainda pela preocupação ambiental e, não em raros casos, pela inexistência de outros meios. A verdade é que, independentemente da opção escolhida, o homem urbano anda menos a pé, tendo-se "motorizado", ou seja, recorre bem mais aos transportes.

Como refere Santos (Santos, 1994) as deslocações não são somente de pessoas, mas também de produtos, bens ou serviços, sendo que essas deslocações são diárias e constantes, são complexas e, no caso da mobilidade urbana, envolvem igualmente urbanismo e modos de vida. O papel dos transportes (todo o tipo de transportes) tornou-se fundamental nos dias que correm, é parte integrante das nossas vidas e essencial para o bom funcionamento das mesmas, tendo de ser bem planeado e estruturado de forma a obtermos resultados positivos e que satisfaçam as necessidades de quem os utiliza.

Mobilidade é um conceito que surge da necessidade do ser humano ter de participar em atividades que, por norma, se desenrolam em locais fisicamente distantes, sendo para isso necessário ter de fazer deslocações de um local para o outro de quem pretende participar nessas atividades (Santos, 1994). Contudo, com os avanços tecnológicos, já nos é possível, em algumas ocasiões, participarmos em algumas atividades sem termos de estar fisicamente presentes. Na base deste conceito encontra-se uma palavra-chave, necessidade. Podemos afirmar que mobilidade é algo que nos permite e possibilita a satisfação de necessidades que o ser humano tem, sejam elas do tipo psicológico, fisiológico, social ou económico.

2.3. Problemas Socioeconómicos

Na base das inúmeras necessidades com que nos deparamos diariamente encontra-se uma que, cada vez mais, assume um papel bastante relevante e imprescindível nas nossas vidas, o fator económico. O dinheiro é o principal objeto no que respeita a trocas e, por isso mesmo, este é também muito relevante para a problemática que aqui tratamos na medida em que é necessário dinheiro para usufruir dos serviços de transporte, por norma há um serviço que é prestado pela "oferta" e pelo qual a "procura" irá pagar um determinado valor, quase sempre em dinheiro.

Derivado à instabilidade económica que se faz sentir, muitos portugueses deixaram de poder sustentar carro próprio. Deste modo, a opção pela utilização de transportes públicos poderia assumir-se como mais necessária do que nunca. Mas a verdade é que o número de passageiros tem vindo a diminuir, quer pelos sucessivos atrasos, quer pela falta de integração entre as várias empresas do setor – o que torna o sistema de bilhética mais complexo para os utilizadores, quer pelas tarifas elevadas ou pela inexistência de ligações que sirvam realmente as necessidades das pessoas (Nery e Fillol, 2013). O táxi, enquanto transporte público de aluguer, é o mais personalizado de todos.

Numa sociedade cada vez mais exigente, é necessário ter uma oferta capaz de satisfazer as necessidades dos clientes/consumidores sendo que, a nível de transportes públicos, o serviço de táxi é o que melhor se adequa e adapta a essas imposições. Apesar dos custos de utilização serem mais elevados que os de autocarros, metros e afins, a verdade é que os táxis são um serviço personalizado, estando disponíveis no horário e local que quisermos e com um atendimento individualizado e com comodidade e qualidade superior aos demais transportes. Na verdade, o táxi está disponível 24 horas diárias, durante todos os dias do ano, sendo uma alternativa e, não raras vezes, a única opção para nos deslocarmos (Vasconcellos, 2005).

Uma outra vantagem do táxi prende-se com o facto de ser um meio de transporte rápido e que não nos limita a parar em paragens pré-definidas, ou seja, somos nós (clientes) que escolhemos o destino do nosso trajeto e este não está limitado a um

horário. O táxi revela então ser uma opção mais simples na medida em que somos nós que escolhemos o trajeto, onde e quando se inicia e termina o serviço, se deve ou não fazer paragens bem como a escolha das mesmas. O táxi acaba então por influenciar o próprio estilo de vida de muitas pessoas uma vez que lhes dá a opção de se movimentarem da forma que mais lhes convém.

Como refere Pedro (Pedro, 2011), existem cerca de 3445 táxis na cidade de Lisboa e que todos os dias são realizadas em média 102 mil viagens diárias naquela cidade, o que se traduz numa média diária de 30 viagens por táxi. O número médio de passageiros é de 1,57, sendo a taxa de ocupação média dos táxis cerca de 63%. A procura de táxis com capacidade superior a 4 lugares tem aumentado, principalmente por grupo de turistas, empresas e grupos de jovens em saídas noturnas. Calcula-se que em média um táxi em Lisboa, cumprindo os turnos de dia e noite, percorra cerca de 8000 km por mês.

O aumento e manutenção da procura no setor tem-se mantido em parte pelo turismo, pela tomada de consciência por parte dos condutores em não conduzir quando ingerem bebidas alcoólicas, o incremento controlo policial em operações *stop* noturnas e pela disponibilidade crescente deste meio de transporte através dos meios de comunicação disponibilizados ao cidadão (Pedro, 2011).

Assim, o serviço personalizado, eficiente e cómodo acaba por se sobrepor, grande parte das vezes, ao fator preço.

2.4. Problemática

De acordo com os problemas identificados anteriormente surge a necessidade de se estruturar uma ação que permita utilizar meios de transporte de forma prática e personalizados, bem como efetuar a extração de perfis/padrões de mobilidade numa cidade. Um outro pilar desta problemática assenta no desenvolvimento de sistemas inteligentes de gestão de frotas e veículos, os quais possam fornecer dados

úteis aos gestores urbanos, informações essas muito relevantes mas pouco acessíveis na área da Computação Urbana.

No centro desta problemática está, principalmente, a resolução de uma dificuldade encontrada que se prende e pode ser resolvida com a criação de uma aplicação que nos seja útil no dia-a-dia no âmbito da utilização de um transporte público de aluguer (táxi) particularmente no perímetro urbano.

Também temáticas como o ordenamento do território e o planeamento de novas acessibilidades, tanto no contexto dos transportes públicos como com os principais *players* privados do sector, serão um enfoque importante neste trabalho devido à problematização dos mesmos e à sua possível resolução com a aplicação estudada/criada. Esta aplicação móvel será uma ferramenta deveras vantajosa na medida em que irá permitir perceber alguns comportamentos dos cidadãos no âmbito das suas trajetórias dentro do espaço urbano (quais os principais locais de origem e de destino).

Assim, de uma forma resumida e concreta, todo o trabalho se desenrola em torno de proporcionar uma maior facilidade, comodismo e rapidez no que respeita a mobilidade, mais concretamente na utilização do táxi. A aplicação a concretizar será uma das possíveis soluções desse problema, bem como uma mais-valia no traçar de rotas urbanas e pontos estratégicos para a criação de possíveis paragens e praças de táxis.

2.5. Aplicações

Para a elaboração deste trabalho e da aplicação em questão, importa também perceber qual o estado de arte de ferramentas idênticas para dispositivos móveis, de forma a percebermos qual o posicionamento e contexto em que queremos inserir o "Meu Táxi". Realizámos assim uma análise do mercado (interno e externo) de forma a conhecermos possíveis concorrentes e de forma a percebermos alguns pontos fortes (para os seguirmos) e encontrarmos algumas falhas (para as evitarmos). Esta pesquisa teve também o intuito de, ao conhecermos o mercado, nos distanciarmos do que é já

existente, tentando criar e desenvolver algo inovador e que possa satisfazer uma necessidade.

2.5.1. Aplicações para Táxis

2.5.1.1. *myTaxi*

A aplicação *myTaxi* permite ao utilizador chamar um táxi sem a necessidade de fazer uma chamada, basta pressionar um botão na aplicação e a localização do passageiro é automaticamente detetada através do GPS. Com a aplicação taxi radar é possível localizar os táxis mais próximos, podendo desta forma escolher um taxista, que após confirmação do mesmo, o utilizador acompanha o táxi no mapa, sendo-lhe facultada a distância e o tempo restante de chagada ao local onde este se encontra.

É ainda fornecido ao utilizador um recurso de avaliação dos taxistas com que se sentiu mais confortável e seguro, para que na próxima vez que chamar um táxi, esses taxistas sejam ordenados por ordem de perferência, ou seja classificação, servindo também para mostrar a avaliação obtida pelos outros passageiros, tornando-se assim mais confiável para os outros passageiros, uma vez que têm acesso aos dados do taxista, incluindo a classificação dada por outros passageiros.

Esta aplicação tem área de atuação em alguma cidades da Alemanha, Austria e Austrália (myTaxi, 2012).

2.5.1.2. *Moove Taxi*

O *Moove Taxi* foi criado para melhorar a mobilidade urbana das pessoas que usam o táxi como meio de transporte, uma vez que existem sérias dificuldades na utilização deste meio de transporte, como a disponibilidade, preço, qualidade e problemas no trânsito.

Com base na localização atual do utilizador, o *Moove Taxi* permite identificar e ligar para os pontos de táxi mais próximos do utilizador, na cidade de São Paulo.

Além de identificar e mostrar os pontos de táxi mais próximos do utilizador, é possível guardar os números preferidos ou mais usados e também ajudar a melhorar a aplicação sugerindo novos contactos de taxistas, pontos e cooperativas. Com a função de taxímetro, o utilizador pode calcular o valor da viagem e simular o percurso até ao destino, antes e durante o trajeto (MooveTaxy, 2012).

2.5.1.3. *TaxiMagic*

A aplicação permite chamar um táxi sem a necessidade de fazer chamadas, uma vez que a aplicação deteta a localização do utilizador pelas coordenadas GPS e fornece uma lista dos táxis mais próximos de si, podendo chamar o táxi diretamente na aplicação, ou então há também a possibilidade de reservar um táxi.

Quando o pedido é aceite, o utilizador recebe uma confirmação com o nome do taxista e tem a possibilidade de acompanhar a chegada do táxi através do mapa. No fim da viagem, o utilizador recebe um recibo eletrónico.

A aplicação encontra-se disponível para várias cidades dos Estados Unidos (TaxiMagic, 2012).

2.5.1.4. *Taksee*

É um sistema automatizado para reservar táxis, onde quer que o utilizador se encontre, apenas com um simples clique, uma vez que a sua localização passa a ser conhecida através das coordenadas GPS.

Uma vez selecionado o percurso que o utilizador pretende fazer, é mostrada uma estimativa do custo da viagem e ao submeter recebe uma mensagem com a confirmação do pedido. Há a possibilidade de reservar o táxi para uma data a definir pelo utilizador.

É possível encontrar esta aplicação em algumas cidades de Espanha, como Barcelona, Madrid, entre outras (Taksee, 2012).

2.5.1.5. *Etaxi Italy*

A aplicação permite detetar a posição do utilizador via GPS e configura automaticamente o endereço exato para fazer a solicitação do táxi, com a possibilidade de definir vários parâmetros para uma melhor filtragem, tais como pagamento com cartão de crédito, viaturas com um maior número de lugares, entre outros.

O *Etaxi Italy* está disponível para algumas cidades Italianas e é indicado para ser utilizado em restaurantes, hotéis, salas de exposições, convenções, conferências, para além do público em geral, possibilitando ainda o pagamento com cartão de crédito (ETaxi, 2012).

2.5.1.6. *WannaTaxi*

Wannataxi é um sistema totalmente automatizado que elimina a utilização de intermediários, uma vez que os utilizadores chamam um táxi diretamente do seu dispositivo móvel, graças à tecnologia GPS que permite saber a localização do utilizador, mostrando num mapa os táxis mais próximos do mesmo. Caso o utilizador prefira pode indicar a morada onde se encontra para que desta forma o táxi o consiga localizar.

Este sistema encontra-se disponível por toda a Espanha (Wannataxi, 2012).

2.5.1.7. *eTaxi*

eTaxi é um serviço que permite aos utilizadores solicitar táxis de forma autónoma e personalizada.

As solicitações são processadas pelos servidores do *eTaxi* de forma totalmente automática em que estabelece a posição do utilizador, localiza o táxi livre mais próximo do cliente, envia ao táxi a solicitação do serviço com o endereço do cliente, confirma ao cliente o pedido e por fim o cliente recebe a confirmação do pedido, o nome do taxista, tempo estimado de chegada e o número de telefone do taxista para que possa entrar em contacto com ele, se assim o desejar.

É ainda possível ao cliente optar por uma viatura maior, transportar animais, falar uma língua específica, aceitar pagamentos com cartão de crédito e definir a hora para que deseja o serviço.

Quando o táxi chega à morada de origem, o sistema notifica o cliente que já se encontra no local.

A aplicação encontra-se disponível em algumas cidades Espanholas (Cediant, 2012).

2.5.1.8. *DeinTaxi*

DeinTaxi é uma aplicação disponível em várias cidades Alemãs, e fornece aos seus utilizadores a possibilidade de chamarem um táxi através do seu dispositivo móvel. Para isso, o utilizador pode selecionar a morada de origem no mapa, ou então a aplicação deteta a sua posição através da tecnologia GPS, depois é só ligar para as centrais de táxi mais próximas de si (Gefos, 2012).

2.5.1.9. *TaxiButton2*

A aplicação *TaxiButton2* permite ao cliente, depois de detetada a sua localização via GPS ou entrada manual da sua localização, visualizar os táxis que se encontram mais próximos de si, escolhendo desta forma o táxi para fazer a sua viagem. Ao escolher o táxi, os dados da localização do cliente são transmitidos ao taxista para que este o possa ir buscar.

A aplicação fornece ainda a possibilidade de escolher o número de lugares necessários, indicar o tipo de pagamento, e fazer uma pré reserva.

A área de atuação do *TaxiButton2* é em Hamburgo (Taxi, 2012).

2.5.1.10. *Get-a-Taxi*

Com o *Get-a-Taxi* o cliente não precisa de saber nenhum contacto telefónico, uma vez que a aplicação deteta a sua localização via GPS, mostrando os táxis disponíveis num mapa. Ao selecionar um táxi, o cliente tem acesso ao perfil do taxista bem como à sua avaliação, atribuída por outros clientes. A aplicação coloca ao dispor do cliente mais opções, como pagamento com cartão de crédito ou dinheiro, viaturas com um maior número de lugares ou com cadeira de criança.

Estas funcionalidades encontram-se em algumas cidades Austríacas (Get-a-Taxi, 2012).

2.5.1.11. *GooTaxi*

GooTaxi é uma aplicação móvel que permite ao cliente pedir um táxi rapidamente, com segurança e conforto. Com apenas dois toques no ecrã o táxi mais próximo irá aparecer, tudo graças à localização automática via GPS. O cliente irá receber a confirmação do pedido na própria aplicação e quando o táxi estiver a chegar perto do local irá enviar uma mensagem a dizer que está a chegar ao local de origem, isto é, ao local onde foi feito o pedido.

Esta aplicação está disponível em algumas cidades Espanholas, tais como Barcelona, Madrid, Sevilla, Jaen, Córdoba, Toledo (Gootaxi, 2012).

2.5.1.12. *Hailo*

É o novo sistema disponível para se poder chamar táxis em Londres, bastando para isso ter um telemóvel com a plataforma *Android* ou *iOS*.

A aplicação veio colmatar a enorme dificuldade em se conseguir obter um táxi na capital inglesa. A aplicação localiza o utilizador via GPS do telemóvel, e todos os condutores de táxi que estejam perto vão poder ver o local onde se encontra. Depois o utilizador vai poder ver o nome do taxista, registo do veículo, a distância que estão e podem ainda pagar através de cartão de crédito (Hailo, 2013).

2.5.1.13. *Taxi-Link*

O *Taxi-Link* consiste numa aplicação para chamada de táxis, obtendo o endereço de recolha com base na localização GPS do *smartphone* e enviando o pedido diretamente para o táxi mais próximo. A aplicação encontra-se integrada com os sistemas eletrónicos de despacho dos táxis, garantindo a prioridade total nos pedidos feitos através da aplicação.

O serviço está disponível nas cidades do Porto, Odivelas, Loures, Póvoa de Varzim e Braga.

A aplicação sofreu alterações muito recentemente, permitindo agora o acompanhamento do táxi no percurso da recolha e a respetiva classificação do serviço (taxi-link, 2013).

De seguida é feita uma comparação entre as aplicações descritas anteriormente, podendo ver com maior detalhe algumas das suas funcionalidades, tendo em conta que a interação existente nas aplicações é feita com base na internet.

Nome	Área de Atuação	Morada de Origem			Morada de destino	Cálculo da despesa	Pedido			Confirmação	Estimativa da hora de chegada	Localização do táxi no mapa	SO Suportado		
		Coordenadas	Seleção Mapa	Morada			Aplicação	Chamada	SMS				Windows Phone	Android	iOS
MyTaxi	R	×	×	×		×	×	×		×	×	×	×	×	×
MooveTaxi	R					×		×						×	×
Taxi Magic	R	×		×	×	×	×			×		×		×	×
Taksee	R	×		×	×	×	×	×		×				×	×
Etaxi Italy	R	×		×			×			×				×	×
Wanna Taxi	NE	×		×			×			×				×	×
eTaxi	R			×	×		×			×	×			×	×
DeinTaxi	R	×	×	×				×						×	×
TaxiButton2	R	×		×				×		×		×		×	×
Get-a-Taxi	R	×				×	×			×		×		×	×
GooTaxi	R	×	×				×			×			×	×	×
Hailo	R	×					×	×		×	×	×		×	×
Taxi-Link	R	×					×			×	×	×		×	×

R- Regional
NE-Nacional-Espanha

Tabela 2-1 Comparação entre algumas aplicações existentes.

2.5.2. Aplicações para Autocarros

Ainda no contexto da mobilidade urbana e estilos de vida do cidadão, existem aplicações, de outros meios de transporte, que visam facilitar a mobilidade dos mesmos. Estas são apresentadas nas secções seguintes.

2.5.2.1. *IZI Carris*

A *IZI CARRIS* é uma aplicação que permite consultar, em plataformas *Android*, quanto tempo falta para chegar o próximo veículo da Carris à sua paragem.
Os dados apresentados na aplicação são fornecidos pela Carris, através do SAEIP (Sistema de Ajuda à Exploração e Informação aos Passageiros), que funciona em tempo real, com recurso aos equipamentos GPS instalados em cada veículo. A informação que é apresentada na aplicação é a mesma que consta em cada um dos painéis instalados em muitas das paragens da Carris em Lisboa (Carris, 2013).

2.5.2.2. MOVE-ME

O MOVE-ME é um protótipo de uma aplicação que permite o acesso móvel a um conjunto diversificado e completo de informação sobre os transportes públicos disponíveis na cidade do Porto.

Ao utilizar **esta aplicação, para *Android* ou *iOS*** os utilizadores beneficiam de informações atualizadas sobre as próximas partidas a partir do local onde se encontram (ou através de outra localização que indiquem), tempos de espera associados, assim como informação sobre a localização das paragens e principais pontos de interesse que estão ao seu alcance.

A aplicação permite igualmente, e em tempo real, construir rotas e planear viagens através da descrição de pontos de passagem definidos pelo utilizador e que, interligados, originam uma rota à sua escolha.

O **MOVE-ME** permite pesquisar por próximas partidas e consultar tempos de espera e os destinos associados às próximas viaturas a passar num determinado local, indicado pelo utilizador (Opt, 2013).

2.5.2.3. *SMTUC Mobile*

SMTUC Mobile disponibiliza todos os horários e paragens da rede de transportes públicos de Coimbra no *smartphone* com sistema operativo *Android*.

A aplicação permite ainda calcular as tarifas, ver os percursos num mapa e até saber os pontos de venda de passes com recurso ao sistema GPS do dispositivo (Google Play, 2013).

2.5.3. Aplicações para Metro

2.5.3.1. Metro Lisboa

A aplicação Metro Lisboa disponibiliza toda a informação relevante sobre o serviço do Metropolitano de Lisboa.

Saber o estado das linhas em tempo real, simular percursos entre estações ou pontos georreferenciados, navegar num módulo de realidade aumentada com pesquisa de estações de Metro e pontos de Interesse através de câmara de telefone e GPS, são algumas das funcionalidades que a aplicação oferece.

Através desta aplicação é ainda possível consultar o mapa da rede do Metro, informação sobre horários e tarifas, percorrer as últimas notícias e receber alertas em tempo real (Google Play, 2013a). Esta aplicação encontra-se disponível para os sistemas operativos *Android* e *iOS*.

2.5.3.2. *iMetroPorto*

O *iMetroPorto* disponibiliza mapas da rede e das seis linhas que a integram, com possibilidade de verificar as frequências, os horários, os lugares de estacionamento e os serviços de cada estação. Outra das vantagens desta aplicação é a possibilidade de efetuar o planeamento de viagens, ficando instantaneamente a par de tempos de

percursos e preço de títulos e assinaturas. Esta aplicação presta ainda informação em tempo real sobre o estado das linhas, bem como mantém atualizadas as últimas notícias do Metro e eventos (Metrodoporto, 2012).

A aplicação encontra-se disponível para os sistemas operativos *Android* e *iOS*.

2.6. Tice.Mobilidade

O TICE.Mobilidade (Sistema de Mobilidade Centrado no Utilizador) tem como finalidade disponibilizar serviços de mobilidade no mercado centrados no utilizador, usando para tal a infraestrutura internet e convergindo assim para a disponibilização deste tipo de soluções no mercado.

A concretização deste projeto passa pelo desenvolvimento de uma plataforma de comercialização de serviços de mobilidade, chamada *One.Stop.Transport*. Esta plataforma é uma plataforma digital de partilha de informação dirigida ao desenvolvimento de serviços de mobilidade urbana.

A complementaridade das origens de informação tratada na plataforma potencia o seu valor e é o motor de um ecossistema para o desenvolvimento de aplicações de mobilidade, onde a aplicação que desenvolvemos poderá vir a prestar um contributo de valor, permitindo ainda angariar informação que ficará disponível na plataforma de forma normalizada ao nível da representação, sincronizada no espaço e no tempo, numa arquitetura *publish/subscribe* (Tice.mobilidade, 2012).

3. DEFINIÇÃO DO PROBLEMA, OBJETIVOS PREVISTOS, METODOLOGIA E RESULTADOS ESPERADOS

3.1. Definição do problema

Cada vez mais com o recurso a tecnologias de informação é possível alterar os hábitos de mobilidade dos cidadãos num espaço urbano. São já vários os exemplos de *APPs* móveis que permitem a consulta de informação relativa a, por exemplo, transportes públicos, em particular os autocarros.

No entanto, ao contrário de outros países, em Portugal aplicações móveis destinadas a auxiliar o cidadão na utilização de Táxis são ainda pouco visíveis. Assim neste trabalho pretendeu-se desenvolver uma aplicação para dispositivos móveis, com a finalidade de um utilizador poder "chamar" um táxi, sem ter a necessidade de saber onde se encontra, ou simplesmente inserindo a morada de origem de forma manual, simplificando assim as suas tarefas diárias e aumentando a sua qualidade de vida. O utilizador poderá inserir o local de destino, sendo assim possível obter um custo estimado do serviço, da distância percorrida e poderá ainda selecionar, num conjunto de opções, as características do táxi que mais se adequam ao seu pedido (viatura com mais de 4 lugares, se permite transporte de animais, se suporta pagamento por multibanco, etc.)

É ainda objetivo deste projeto o desenvolvimento de uma plataforma *Web* que faça a gestão dos pedidos efetuados pelos utilizadores, e que ao mesmo tempo forneça mecanismos de seleção do veículo mais adequado ao pedido efetuado pelo cidadão, tendo em conta a distância e o tempo estimado que demora a chegar junto do mesmo.

Após o pedido ser aceite pela central, o utilizador poderá acompanhar, no seu dispositivo móvel, à chegada do táxi que o transportará até ao destino.

A solução final contempla três módulos, embora neste projeto só serão desenvolvidos os dois primeiros, ou seja, será desenvolvido o módulo que faz a comunicação entre o cidadão e a plataforma, aplicação móvel, e a própria plataforma, aplicação *Web*. O desenvolvimento do terceiro módulo, aquele que faz a comunicação entre o táxi e a plataforma, fica a cargo do Centro de Computação Gráfica, como mostra a figura 3.1.

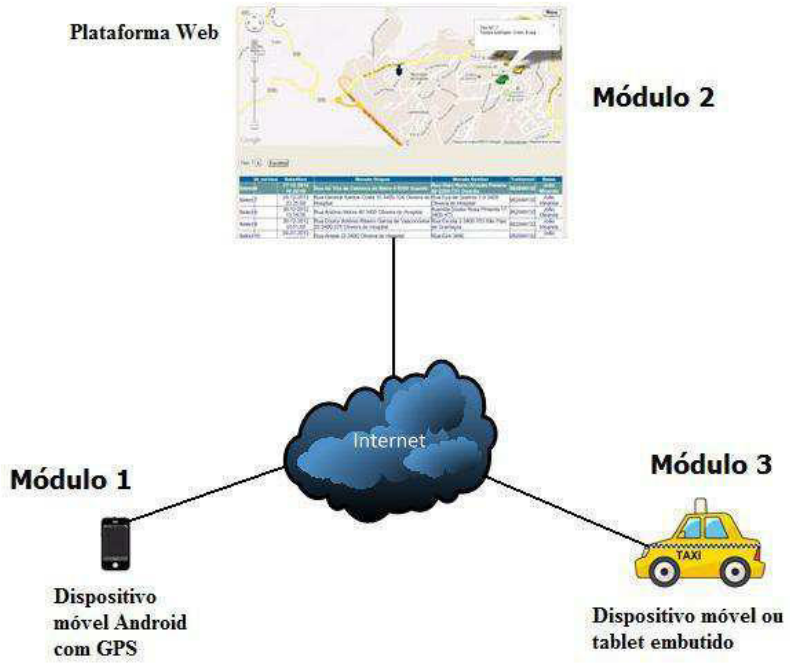


Figura 3.1 Solução final do projeto.

A solução final como um todo, além de pretender auxiliar o cidadão na sua mobilidade, fomentando a adoção de um meio de transporte público ao invés da utilização de viatura própria (originando diversos problemas para o espaço urbano), pretende ainda contribuir para a resolução de um outro problema, o completo desconhecimento, quer dos principais *players* de mobilidade, quer das autoridades responsáveis pela gestão da mobilidade, da real mobilidade urbana dos cidadãos, permitindo-lhes determinar, por exemplo, a quantidade de pessoas e as horas a que

viajam em táxis, a quantidade de táxis que circulam nas estradas, os percursos mais usados, os locais onde as pessoas mais solicitam os serviços de táxi e os locais para onde mais se deslocam.

Atualmente o planeamento de mobilidade é feito com recurso a métodos pouco fiáveis no que respeita à verdadeira mobilidade dentro dos espaços urbanos e fora deles. Não é de estranhar por isso ver autocarros vazios durante determinado período ou completamente lotados noutros períodos, fruto em parte, da falta de adequação da oferta de transportes públicos às reais necessidades dos cidadãos e respetivo espaço urbano.

Assim a solução final pretende utilizar de forma completamente anónima a informação de solicitação de serviços de táxi com o objetivo de contribuir para a caraterização da mobilidade urbana, em complementaridade com outra informação, inclusivamente dos restantes transportes públicos. O Centro de Computação Gráfica em colaboração com a Universidade do Minho pretende complementar a informação obtida atualmente de outras fontes de informação com esta nova fonte e assim conseguir cada vez mais caracterizar a mobilidade urbana o mais próximo da situação real.

Assim, os principais problemas aplicacionais a resolver são os seguintes:

- Definir os *layouts* dos ecrãs.
- Definir o sistema de navegação entre os ecrãs.
- Definir o sistema de gestão de base de dados a utilizar.
- Definir como se procede à troca de informação entre a aplicação móvel e a plataforma *Web*.
- Obter a localização do cidadão através do GPS do dispositivo móvel e apresentá-la num mapa.
- Definir como obter as informações relativas aos táxis, uma vez que não existe a componente que faz a comunicação entre os táxis e a plataforma.
- Definir como fazer a simulação, pelo mesmo motivo enumerado no ponto anterior.

- Definir qual o melhor método de cálculo da distância entre o cidadão e os vários táxis que se encontram nas melhores condições de satisfazer o pedido.

3.2. Objetivos previstos

Os principais objetivos do projeto aplicado são:

- Desenvolvimento de uma aplicação para dispositivos móveis baseados na plataforma *Android* que permita a um cidadão "chamar" um táxi de acordo com o local ou morada onde se encontra e respetivo destino, fazendo desta forma a aquisição de informação de mobilidade urbana.
- Desenvolvimento de uma aplicação *Web* que faça a gestão dos pedidos dos utilizadores e que ao mesmo tempo permita a seleção do táxi que melhor se adequa ao pedido.
- Recolha de um conjunto alargado de dados de mobilidade, que permitam caracterizar o comportamento do cidadão num espaço urbano.

3.3. Metodologia

Para o desenvolvimento deste projeto foi utilizada uma metodologia ágil, que consistia em apresentar, periodicamente, a evolução do trabalho realizado. Assim, a metodologia para desenvolver e implementar o projeto é a seguinte:

- Análise dos requisitos do projeto.
- Utilizar o *MSSQL Server* e o *Microsoft SQL Server Management Studio* para a criação da Base de Dados.

- Utilizar o *IDE Eclipse* e a plataforma *Android SDK* para o desenvolvimento da aplicação móvel.
- Utilizar o *Visual Studio* 2010 e a plataforma *Microsoft .NET* para o desenvolvimento da aplicação *Web* de gestão de pedidos.
- Recolher um conjunto de dados a fim de testar as aplicações, uma vez que não estará disponível a terceira componente do projeto, ou seja, a plataforma que comunica entre a plataforma *Web* e os táxis.
- Realização de testes e análise da eficiência e fiabilidade da aplicação.

3.4. Descrição das tarefas

As principais tarefas foram:

- Tarefa 1 – Estudo de soluções existentes dentro da área do projeto.
- Tarefa 2 – Desenhar a Base de Dados no *Microsoft SQL Server* 2012.
- Tarefa 3 – Especificação técnica da plataforma de gestão de solicitações da aplicação móvel e registo dos dados de mobilidade associados.
- Tarefa 4 – Desenvolver a aplicação móvel no *IDE Eclipse* e *Android SDK*.
- Tarefa 5 – Desenhar os ecrãs que compõem a aplicação móvel.
- Tarefa 6 – Obter os mapas através da plataforma *Google Maps API.*
- Tarefa 7 – Obter a localização do cidadão através das coordenadas GPS.
- Tarefa 8 – Desenhar as marcas no mapa quando este for clicado.
- Tarefa 9 – Desenhar um campo para a introdução de uma morada de forma manual.
- Tarefa 10 – Obter o valor estimado a pagar pelo cidadão.
- Tarefa 11 – Obter a distância da viagem através da *Google Directions API.*

- Tarefa 12 – Desenhar o ecrã que mostra as opções disponíveis ao cidadão.
- Tarefa 13 – Obter as informações do táxi que vai realizar o serviço.
- Tarefa 14 – Obter o tempo estimado que o táxi demora a chegar junto do cidadão recorrendo à *Google Directions API.*
- Tarefa 15 – Obter o histórico de viagens do cidadão, mostrando-os numa listagem, dando-lhe ainda a possibilidade de poder anular algum pedido que se encontre aberto.
- Tarefa 16 – Obter o possível percurso realizado e mostrá-lo no mapa.
- Tarefa 17 – Desenvolver a plataforma *Web* utilizando o *Visual Studio* 2010.
- Tarefa 18 – Determinar qual o táxi que se encontra na melhor posição para realizar o serviço, tendo em conta o tempo que demora e a distância a que se encontra do cidadão.
- Tarefa 19 – Definir como realizar uma simulação para que o funcionamento normal da aplicação seja o mais próximo da realidade possível, uma vez que não temos o terceiro módulo.
- Tarefa 20 – Extrair os dados de mobilidade e ilustrar, como exemplo, no mapa os locais onde as pessoas mais efetuam pedidos, bem como os locais para onde as pessoas mais se deslocam.
- Tarefa 21 – Testes das aplicações.
- Tarefa 22 – Elaboração do relatório.

3.5. Resultados esperados

Com o desenvolvimento deste projeto foram atingidos os seguintes resultados esperados:

- Permitir chamar um táxi consoante a localização geográfica do cidadão, via GPS, ou em alternativa com a introdução de uma morada de origem de forma manual ou selecionando no mapa.
- Permitir ao cidadão consultar o seu histórico de viagens efetuadas.
- Permitir ao cidadão anular um pedido em aberto.
- Permitir ao cidadão escolher de entre um conjunto de opções, aquelas que mais satisfaçam as suas necessidades.
- Permitir obter uma estimativa do custo da viagem.
- Permitir obter uma estimativa da distância entre a morada de origem e a morada de destino.
- Permitir ao cidadão acompanhar a chegada do táxi no mapa.

A plataforma de gestão de solicitações permite ao gestor:

- Selecionar um pedido e visualizar, no mapa, onde se encontra o cidadão que fez o pedido.
- Selecionar o táxi que se encontre mais perto e que demora menos tempo a chegar junto do cidadão.
- Ver dados sobre a mobilidade urbana dos cidadãos que utilizam este meio de transporte.

4. TECNOLOGIAS UTILIZADAS

Para o desenvolvimento da aplicação "Meu Táxi", foi necessário recorrer a diversas tecnologias. O principal objetivo deste capítulo é fazer uma descrição das tecnologias envolvidas no desenvolvimento de aplicações para dispositivos móveis, em particular para o Sistema Operativo *Android*.

4.1. Sistema Operativo *Android*

O *Android* é um sistema operativo baseado em Linux, para dispositivos móveis, desenvolvido pela *Open Handset Alliance*, liderada pela empresa *Google* (Android OS, 2012), sendo a sua primeira versão apresentada em setembro de 2008, encontrando-se atualmente na versão 4.2 designada por *Jelly Bean*, lançada em outubro de 2012.

Para este sistema operativo, as aplicações são desenvolvidas na linguagem de programação JAVA, utilizando o *Android SDK*, sendo executadas posteriormente numa máquina virtual designada de *Dalvik*.

As aplicações podem ser disponibilizadas e distribuídas através da loja de aplicações da *Google*, designada por *Google Play*, vindo este substituir o *Android Market* que, até há cerca de um ano atrás, vigorava como serviço de disponibilização de aplicações para os utilizadores do sistema operativo *Android*. Este serviço que conta já com mais de 500.000 aplicações, tanto pagas como gratuitas, e que, segundo dados de Lunden (2012), foram já descarregadas mais de 15 biliões destas aplicações.

Segundo o que Rubin (Rubin, 2012), vice presidente da Google, afirmou no *Mobile World Congress* 2012, em Barcelona, são ativados diariamente cerca de 850 mil dispositivos com o sistema operativo *Android*, representando num aumento anual de cerca de 250%, referindo ainda que já foram ativados, em todo o mundo, perto de 300 milhões de equipamentos com o referido sistema operativo, sendo que 12 milhões deles

correspondem a *tablets*, o que faz deste sistema operativo uma das plataformas mais interessantes para o desenvolvimento de aplicações para dispositivos móveis (Goasduff, 2012). De seguida é mostrado com algum detalhe o funcionamento dos principais componentes da arquitetura do sistema operativo *Android*.

4.1.1. Arquitetura

Segundo o que a própria *Google* afirma (Android Developers, 2013), o *Android* é mais do que um sistema operativo, sendo considerado pela mesma, uma pilha de *softwares*, composto por cinco camadas, em que cada camada da pilha agrupa vários programas que suportam funções específicas do sistema operativo, como mostra a figura 4.1.

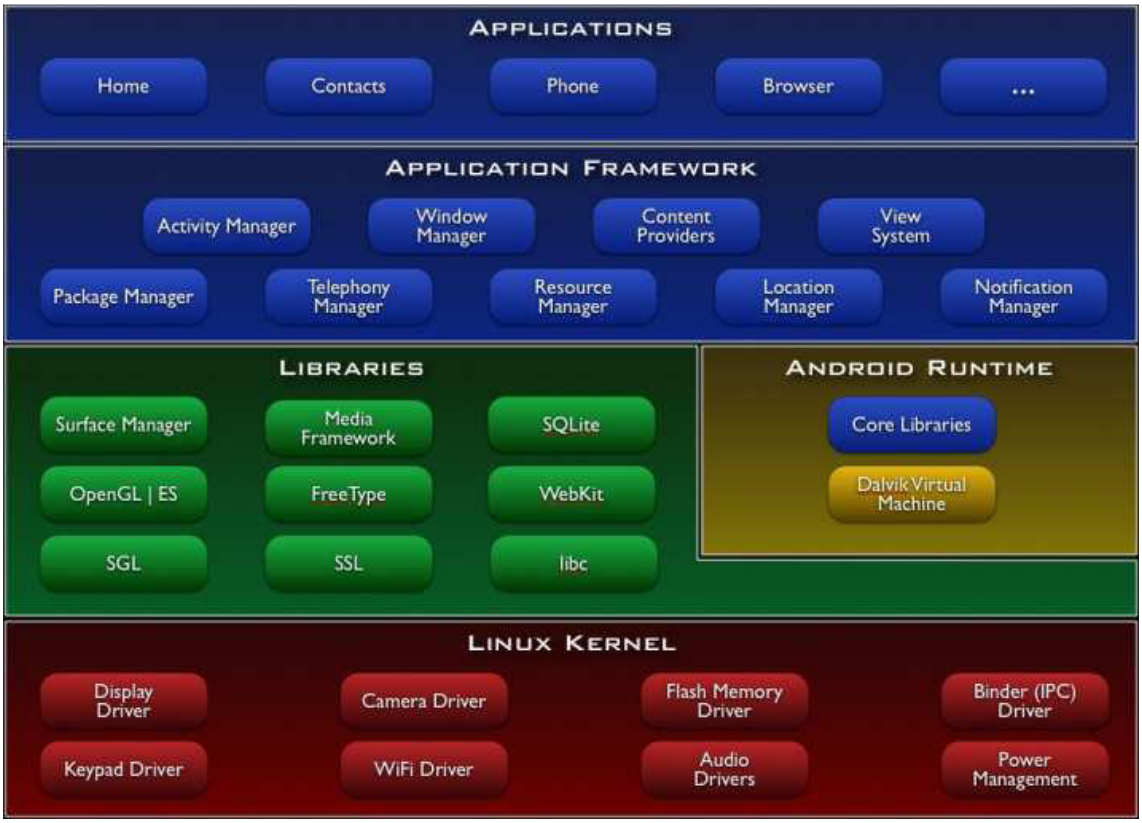


Figura 4.1 Arquitetura *Android* (Android Developers, 2013)

A base da pilha é o *kernel*, tendo sido usada a versão 2.6 do Linux para construir o *kernel* do *Android*, o que inclui os programas de gestão de memória, configurações de segurança, software de gestão de energia, controlo de processos, protocolos de rede e vários *drivers* de *hardware*.

Acima do *kernel* ficam as bibliotecas C/C++ utilizadas por diversos componentes do sistema, tais como, bibliotecas para suporte a formatos de áudio, vídeo e imagens, bibliotecas para gráficos 2D (SGL) e 3D (Open GL ES), para trabalhar com base de dados relacional como o *SQLite*, ou o motor para navegadores *WebKit*.

No *Android*, as aplicações escritas em *JAVA* são executas na sua própria máquina virtual, designada por *Dalvik*.

Uma máquina virtual é uma aplicação de *software* que se comporta como se fosse um dispositivo independente, ou seja, como se tivesse o seu próprio sistema operativo. O sistema operativo *Android* usa máquinas virtuais para executar cada aplicação, tornando-se importante por algumas razões, tais como, nenhuma aplicação é dependente de outra, se uma aplicação parar não afeta quaisquer outras aplicações a serem executadas no dispositivo, o que simplifica a gestão de memória.

Apesar das aplicações para *Android* serem escritas em *JAVA*, a *Dalvik VM* não pode ser considerada uma *Java Virtual Machine*, uma vez que não interpreta *Java bytecodes*. Em vez disso a ferramenta dx transforma os documentos .class compilados com um compilador *JAVA* normal, em documentos .dex, que são específicos para execução na *Dalvik VM*.

Na camada acima, quase toda ela escrita em Java, fica a *framework* de aplicações, que fornece todas as funcionalidades necessárias para a construção de aplicações, através das bibliotecas nativas, e faz a interface com as aplicações *Android*. Esta camada inclui os programas que fazem a gestão das funções básicas do *smartphone*, como gestão de recursos, aplicações de telefone, mudança entre processos ou programas de localização física do aparelho, como GPS. Os programadores têm total acesso ao *framework* de aplicações do *Android*, possibilitando com que tirem vantagem das capacidades de processamento e de recursos do mesmo quando estão a desenvolver uma aplicação *Android*.

No topo da pilha estão as aplicações em si, onde são encontradas funções básicas do dispositivo, como fazer chamadas telefónicas, aceder ao navegador *Web* ou

aceder à lista de contactos, calculadora, cliente de e-mail, mapas, cliente de SMS e MMS, entre outros (Android-App-Market, 2012).

4.2. *Google Maps* API

A Google criou uma API que visa facilitar a vida dos programadores que tenham a intenção de desenvolver aplicações, baseadas na localização, e que, ao mesmo tempo estejam integradas com o *Google Maps*, sem que para isso haja a necessidade das aplicações estarem alojadas nos servidores da própria Google.

A *Google Maps API for Android*, além de gratuita, desde que para uso não comercial, é formada por um conjunto de classes que fornecem uma interface ao utilizador que permite, entre outras, a criação de mapas com locais definidos, pesquisa de endereços, controlo de *zoom*, tipos de mapa, geração de rotas, transformar endereços em coordenadas.

De seguida é feita uma breve descrição das API's usadas no desenvolvimento deste projeto.

A API **android.location** contém classes que definem os serviços baseados no contexto da localização, fornecendo mecanismos para obter e manipular a posição geográfica do dispositivo móvel. São elas (Android Developers, 2012):

- **Address**: fornece mecanismos para obter a representação estruturada de um endereço, isto é, pode obter-se o nome da rua do endereço (*address.getThoroughfare()*), a localidade (*address.getLocality()*), o código postal (*address.getPostalCode()*), o nome do país (*address.getCountryName()*), entre outras informações.

- **Criteria:** classe responsável por indicar os critérios de aplicação para a seleção de um *provider* de localização, sendo estes *providers* ordenados de acordo com a

precisão, a utilização de energia, capacidade de informar a velocidade, altitude, direção, e custo monetário se aplicável.

- **Geocoder**: classe para geocodificação e geocodificação inversa. Geocodificação é o processo de transformar um endereço ou outra descrição de um local numa coordenada, isto é, latitude e longitude. Por outro lado, a geocodificação inversa é o processo de transformação de coordenadas, latitude e longitude, num endereço. A quantidade de informação numa descrição de localização inversa geocodificados pode variar, ou seja, pode conter o endereço completo do edifício mais próximo, enquanto outra pode conter apenas o nome de uma cidade e código postal. Os métodos de consulta *Geocoder* retornam uma lista vazia se não houver nenhuma informação na plataforma.

- **GpsSatellite:** classe que representa o estado atual de um satélite GPS. Esta classe é usada em conjunto com a classe *GpsStatus*.

- **GpsStatus:** esta classe representa o estado atual do mecanismo de GPS, sendo usada em conjunto com a interface *GpsStatus.Listener*, que é usada para receber notificações quando o estado do GPS é alterado.

- **Location**: classe que representa uma localização geográfica. Uma localização pode consistir em data/hora, latitude, longitude, e outras informações, tais como, altitude, direção e velocidade, sendo que, todos os locais gerados pelo *LocationManager*, são garantidos para ter uma latitude, longitude e data/hora válidas.

- **LocationManager**: fornece acesso aos serviços de localização do sistema. Estes serviços permitem que as aplicações obtenham atualizações periódicas de localização geográfica do dispositivo, ou para acionar um objetivo específico quando o dispositivo entra na proximidade de uma dada localização geográfica.

- **LocationProvider**: Um *provider* de localização proporciona relatórios periódicos sobre a localização geográfica do dispositivo. Cada *provider* tem um conjunto de critérios que podem ser usados, isto é, alguns *providers* requerem *hardware* de GPS, outros requerem a utilização do sinal da própria rede móvel ou ligação à internet. Com esta classe é possível obter informações relativas a altitude, velocidade, requisitos de energia, satélites, entre outras.

A API **android.maps** possui classes utilizadas no contexto de controlo e apresentação de mapas. São elas (Android Developers, 2012a):

- **MapView**: Componente gráfico (layout) do mapa. Quando destacado, irá capturar teclas pressionadas e gestos de toque para movimentar e aplicar *zoom* no mapa. Também pode ser controlado através de programação (getController()) e permite desenhar um número de camadas por cima do mapa (getOverlays()).

- **MapActivity**: Classe específica para gerir as necessidades de qualquer *Activity* que exiba uma *MapView*. Para usar a classe *MapActivity* é preciso estender a mesma na *Activity*, isto é, public class HelloGoogleMaps extends MapActivity. Depois é só criar o método *onCreate()*, tal como é ilustrado a seguir.

Listagem 1 Método *onCreate*

```
1  public void onCreate(Bundle savedInstanceState) {
2          super.onCreate(savedInstanceState);
3
4          // main.xml contém a MapView
5          setContentView(R.layout.main);
6
7          // extrai o MapView do layout e habilita os controlos de zoom
8          mapView = (MapView) findViewById(R.id.mapview);
9          mapView.setBuiltInZoomControls(true);
10
11         // cria um overlay e mostra a localização atual
12         myLocationOverlay = new MyLocationOverlay(this, mapView);
13
14         // adiciona o overlay ao MapView e atualiza-o
15         mapView.getOverlays().add(myLocationOverlay);
16         mapView.postInvalidate();
17 }
```

- **MapController:** Controla a movimentação e *zoom* no mapa.

- **Overlay**: Permite adicionar componentes ao mapa, tais como marcadores, balões, entre outros.

4.3. Google Directions API

A *Google Directions API* é um serviço que calcula rotas entre locais usando solicitações *HTTP*, podendo estas rotas serem procuradas de entre vários modos, ou seja, transporte (incluindo transporte público), vias (caminhos para andar a pé ou ciclovias). As rotas podem especificar origens, destinos e pontos de referência como *strings* de texto, como por exemplo o nome dos próprios locais, ou como coordenadas de latitude e longitude. A *Google Directions API* pode devolver partes de rotas usando uma série de pontos de referência (Android Developers, 2013a).

Uma solicitação da *Google Directions API* tem a seguinte forma:

http://maps.googleapis.com/maps/api/directions/*output*?*parameters*

em que o output pode ter um dos seguintes valores:
- **json:** indica a saída em *json*
- **xml:** indica a saída em *xml*

No que diz respeito aos parâmetros, há parâmetros que são obrigatórios, outros são opcionais.

Assim, os parâmetros obrigatórios são:

- **origin:** indica o endereço ou o valor textual da latitude/longitude a partir do qual se deseja calcular a rota. Se for introduzido um endereço como uma *string*, o serviço da *Google Directions API* geocodifica a *string* e converte-a numa coordenada de latitude/longitude para calcular as rotas.
- **destination:** indica o endereço ou o valor textual da latitude/longitude do destino.

- **sensor:** indica se a solicitação de rotas vem ou não de um dispositivo com um sensor de localização, como por exemplo *smartphones*. Esse valor deve ser *true* ou *false*.

Os parâmetros opcionais são mostrados a seguir:

- **mode** (por omissão é *driving*): especifica o meio de transporte a ser usado ao calcular a rota. Este meio de transporte pode ser *driving* (indica percursos de carro), *walking* (indica percursos a pé), *bicycling* (indica percursos de bicicleta) ou *transit* (indica percursos de transportes públicos).
- **waypoints:** especifica uma matriz de pontos de referência. Um ponto de referência é especificado como uma coordenada de latitude/longitude ou como um endereço que será geocodificado, servindo para que o utilizador possa definir locais de passagem no cálculo das suas rotas. Os pontos de referência só são usados em rotas de carro, a pé ou de bicicleta.
- **alternatives:** se definido como *true*, indica que o serviço da *Google Directions API* pode fornecer mais do que um percurso alternativo na resposta. O fornecimento de percursos alternativos pode aumentar o tempo de resposta do servidor.
- **avoid:** indica se os percursos calculados devem evitar os elementos indicados. Atualmente, esse parâmetro aceita estes dois argumentos:
  - **tolls:** indica que o percurso calculado deve evitar estradas/pontes com portagens.
  - **highways:** indica que o percurso calculado deve evitar autoestradas.
- **units:** especifica o sistema de medidas a ser usado ao exibir resultados, podendo ser *metric* ou *imperial* (unidade inglesa).
- **region:** indica o código da região, constituído por dois dígitos, como por exemplo PT, Portugal.
- **departure_time:** especifica a hora de partida desejada para percursos de transporte público. Este valor é definido em segundos.

- **arrival_time:** especifica a hora de chegada desejada para percursos de transporte público. Este valor é definido em segundos.

  De seguida é mostrado um exemplo de um *link* com alguns parâmetros, em que é definida como morada de origem a cidade do Porto e a morada de destino a cidade de Lisboa, com passagem pelas cidades de Aveiro e Coimbra respetivamente.

  http://maps.googleapis.com/maps/api/directions/json?origin=Porto&destination=Lisboa&waypoints=Aveiro|Coimbra&sensor=false

Ao ser feita uma solicitação *HTTP*, o resultado obtido, para o caso do output ser no formato *json*, vai ser idêntico ao da estrutura mostrada no Anexo A.

4.4. Web Services

Um *Web service* não é mais do que uma solução utilizada na integração de sistemas e na comunicação entre aplicações que podem ser desenvolvidas em plataformas distintas, permitindo assim, com esta tecnologia, que seja possível que novas aplicações possam interagir com as já existentes, independentemente das linguagens de programação onde foram desenvolvidas. Os *Web Services* são componentes que permitem às aplicações enviar e receber dados em formato *XML*.

Utilizando esta tecnologia, uma aplicação pode invocar outra para efetuar tarefas simples ou complexas, ou seja, o objetivo dos *Web Services* é a comunicação de aplicações através da Internet, permitindo a interoperabilidade entre a informação que circula numa organização nas diferentes aplicações como, por exemplo, o comércio eletrónico com os seus clientes e seus fornecedores, constituindo assim o sistema de informação de uma empresa. Para além da interoperabilidade entre as aplicações, a EAI (*Enterprise Application Integration*) permite definir um *workflow* entre as aplicações e pode constituir uma alternativa aos ERP (*Enterprise Resource Planning*), permitindo

otimizar e controlar processos e tarefas de uma determinada organização. (Pamplona, 2010)

4.5. SQL Server 2012

Para implementar a Base de Dados necessária à realização do projeto utilizou-se o *Microsoft SQL Server*. Este é o principal Sistema de Gestão de Base de Dados (SGBD) relacionais da Microsoft e assenta num modelo Cliente/Servidor, pois envolve diferentes tipos de plataformas e possui funcionalidades divididas entre clientes e servidores, onde o cliente fornece uma ou mais interfaces que serão usadas para requerer um pedido ao servidor, e este por sua vez, processa o pedido e devolve o resultado ao cliente (Macoratti, 2013).

O *SQL Server* possui uma linguagem relacional designada *Transact-SQL,* e foi elaborada para ser independente do *hardware* ou do *software*. Tudo o que é necessário é utilizar os comandos/instruções SQL padrão para realizar as consultas à base de dados.

A linguagem SQL tem como grandes características a capacidade de gerir índices, construir vistas, e cancelar uma série de atualizações ou de as gravar antes de iniciar uma sequência de atualizações através da utilização dos vários comandos que o SQL possui, sendo esta linguagem dividida em subconjuntos de acordo com as operações que queremos executar, tais como (Goldschmidt, 2012):

- DML – Linguagem de manipulação de Dados
  É utilizado para realizar inserções, consultas, alterações e exclusões de dados presentes em registos, podendo estas tarefas ser executadas em vários registos de diversas tabelas ao mesmo tempo. Os comandos que realizam as funções acima referidas são *Insert*, *Select*, *Update* e *Delete*, respetivamente.

- DDL – Linguagem de Definição de Dados
  Uma DDL permite ao utilizador definir a estrutura da base de dados. A maioria das bases de dados SQL comerciais tem extensões proprietárias na DDL.

- DCL – Linguagem de Controlo de Dados

  O DCL é utilizado para controlar os aspetos de autorização de dados e licenças de utilizadores de forma a controlar quem tem acesso para ver ou manipular os dados dentro da base de dados. Os comandos utilizados são o "*Grant*" e o "*Revoke*", que servem, respetivamente, para autorizar a execução de operações ou para remover essa mesma capacidade de executar operações na base de dados.

- DTL – Linguagem de Transação de Dados

  A DTL serve para proceder a transações na base de dados. Os comandos utilizados nesta linguagem são o "*Begin Work*" para dar início à transação, o "*Commit*" para enviar todas as alterações dos dados de forma permanente e o "*Rollback*" que faz com que todas as alterações nos dados existentes, desde o último "*Commit*" ou "*Rollback*", sejam anuladas.

- DQL – Linguagem de Consulta de dados

  O comando "SELECT" permite ao utilizador especificar uma consulta ("*query*") como uma descrição do resultado desejado. Esse comando é composto por várias cláusulas e opções, possibilitando elaborar consultas das mais simples às mais elaboradas.

  Para este projeto apenas foram usadas as linguagens DDL e DML.

4.6. GPS – Sistema de Posicionamento Global

O GPS é um sistema de posicionamento geográfico que nos dá as coordenadas de um lugar na Terra. Este sistema foi desenvolvido pelo Departamento de Defesa Americano para ser utilizado com fins civis e militares.

A nossa posição na Terra é referenciada em relação ao equador e ao meridiano de Greenwich e é traduzido por três números: a latitude, a longitude e a altitude. Assim, para saber a nossa posição sobre a Terra basta saber a latitude, a longitude e a altitude.

Para que sejam calculadas estas variáveis, isto é, latitude, longitude e altitude, existem 24 satélites que dão uma volta à terra em cada 12 horas e que enviam de forma contínua, sinais de rádio, o que faz com que em cada ponto da terra estão sempre visíveis quatro satélites e, com os diferentes sinais destes satélites o recetor GPS consegue calcular a latitude, longitude e altitude do local onde se encontra (Dilão, 2013).

- Latitude: é a distância ao Equador medida ao longo do meridiano *Greenwich*, sendo esta distância medida em graus, podendo variar entre 0º e 90º para Norte ou Sul.

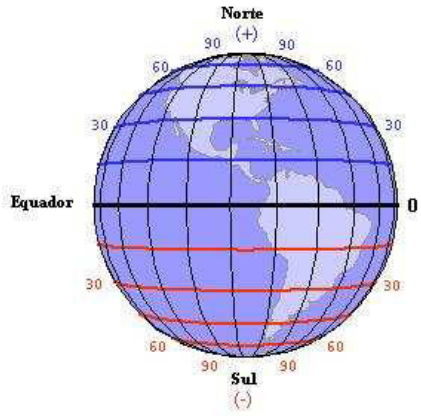


Figura 4.2 Cálculo da Latitude.

- Longitude: é a distância ao meridiano de *Greenwich* medida ao longo do Equador. Esta distância mede-se em graus, podendo variar entre 0º e 180º para Este ou para Oeste.

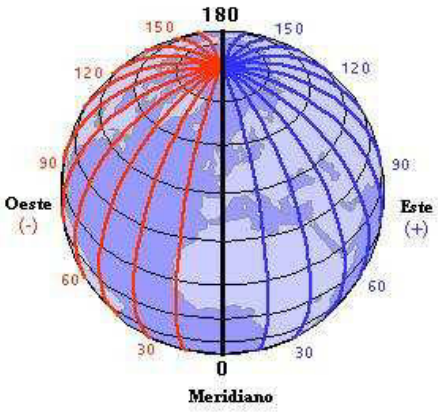


Figura 4.3 – Cálculo da Longitude.

- Altitude: a Terra é aproximadamente esférica, com um ligeiro achatamento nos polos. Para se definir a altitude de um ponto sobre a Terra define-se uma esfera, designada de geoide, com um raio de 6378 km. A altitude num ponto da Terra é a distância na vertical à superfície deste geoide.

4.7. *.Net Framework*

A *.NET Framework* é uma plataforma inovadora e revolucionária, construída especificamente para a *Web* que fornece uma programação consistente e orientada aos objetos, onde o código é guardado e executado localmente, sendo ao mesmo tempo possível distribuir pela Internet ou ser executado remotamente, oferecendo um ambiente de código executável que minimiza a distribuição física de *software*, assim como o conflito de versões. Possui uma grande compatibilidade com diversos dispositivos e as suas bibliotecas são facilmente atualizadas e instaladas (Vieira, 2002).

Esta plataforma possui, entre outras coisas, a BCL (*Base Class Library*) e a chamada CLR (*Common Language Runtime*), tal como se pode observar na figura 4.4. A BCL vem substituir a antiga *Windows API* e a MFC (*Microsoft Foundation Classes*), trazendo consigo uma organização conseguida com a verdadeira estruturação orientada

a objetos que não tinha sido conseguida pela MFC. A CLR é a origem de toda a *Framework,* é como um agente que gere o código em tempo real, fornecendo serviços como gestão de memória, gestão de *threads* e gestão remota, que também força tipos de segurança e outras formas de correção de código que garantam robustez e segurança (Vieira, 2002). A CLR, vem permitir agora, não só aos programadores de *Java,* desenvolverem para qualquer ambiente, graças ao J# que corre na *.Net Framework*, como a qualquer outro tipo de programador fazer o mesmo. Por exemplo os programadores que usam *VB* usarão o *VB.NET*, os seguidores do *COBOL* o *COBOL.NET*, os de C/C++ poderão já usar o *C#* ou o *Visual C++* com *Managed Code*, isto é, todo o código que é executado sobre a CLR em vez de correr diretamente no *Windows* atual na forma de *Intel x86 code*. O suporte para múltiplas linguagens é conseguido graças à CLR que funciona sobre a MSIL (*Microsoft Intermediate Language*). O MSIL é o código executável que vai correr dentro da CLR e é obtido depois da compilação de uma linguagem qualquer para a *.NET Framework*.

Com a plataforma *.NET Framework* é possível desenvolver aplicações *Web* usando qualquer linguagem *.NET* e o mesmo paradigma de programação usado nas aplicações *Windows*. A esta tecnologia a Microsoft chamou de ASP.NET e é mais do que uma evolução do ASP. De facto é possível agora em ASP.NET criar formulários *Web* da mesma forma que se fazem *interfaces* em *Visual Basic* graças ao IDE (*Integrated Development Environment*), isto sem a perda de restrição da linguagem como o HTML. Qualquer aplicação *Web* desenvolvida, hoje em dia, em ASP.NET, pode ser executada em qualquer *browser* com a mesma facilidade que uma aplicação desenvolvida em *Dreamweaver, FrontPage* ou *Notepad*. É de realçar que o desenvolvimento de páginas *Web* com recurso à tecnologia ASP.NET pode considerar-se como programação, pois agora o código final é compilado para .DLL e corre diretamente na CLR como outro programa qualquer, ao contrário de outras linguagens para a *Web* onde a atividade principal é ainda criar *scripts* que irão ser interpretados pelo *Web Server* (Vieira, 2002).

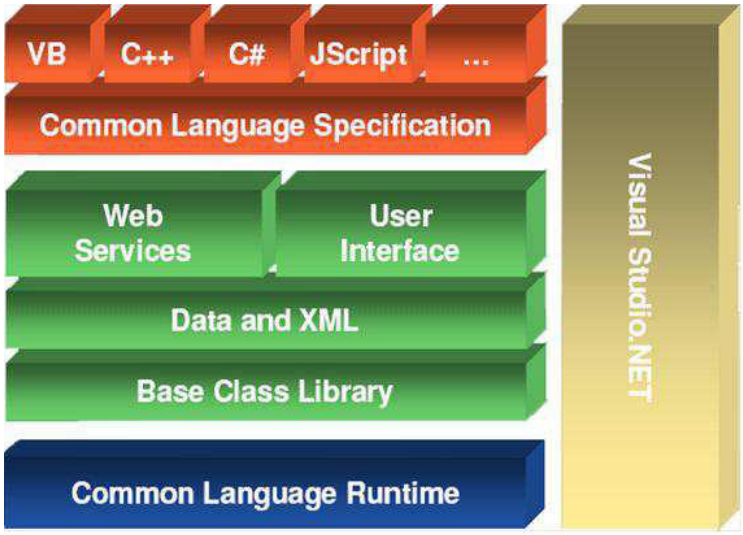


Figura 4.4 Arquitetura *.NET Framework* (Haddad, 2012).

Tal como ilustra a figura 4.4, no topo da arquitetura temos a linguagem a ser usada, como o *Visual Basic*, o C++, o C#, entre outras 36 linguagens que se encontram disponíveis para a plataforma .NET. Em seguida temos o CLS (*Commom Language Specification*), que contém o compilador de cada uma das linguagens acima e é também responsável pela geração do *Intermediate Language*. A seguir temos os *Web Services*, *User Interface*, *Data and XML* e o *Base Class Library*, que contém todas as bibliotecas de classes do .NET. Por fim temos o CLR (*Common Language Runtime*) que é o produto final da aplicação e que será executada no cliente ou servidor (Haddad, 2012).

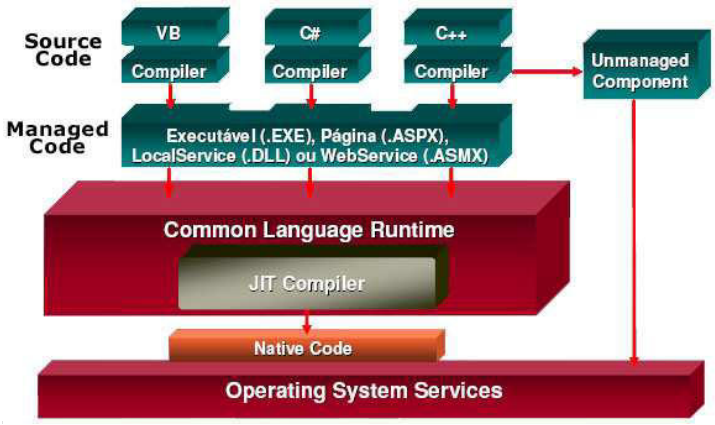


Figura 4.5 Modelo de execução do *.NET Framework* (Haddad, 2012).

O processamento do *Framework* ocorre da seguinte forma: Cada linguagem tem um compilador para .NET. Com isso é possível desenvolver um programa em C#, uma DLL em VB.NET e outra parte em C++. Depois do código compilado será gerado um EXE, ASPX, DLL ou ASMX conforme o tipo de projeto, mas o mais importante é que será gerado um código *Intermediate Language*, sendo por isso possível fazer com que as linguagens interajam entre si. Este *Intermediate Language* será gerado no ambiente *Managed Code*, ou seja, no ambiente do .NET. Em seguida, o CLR passa por um JIT (*Just In Time Compiler*) e o código fica pronto (nativo) para ser executado. É também possível usar componentes externos ao .NET, no entanto, será executado num ambiente *Unmanaged Code*, o que significa dizer que se ocorrer algum erro no componente, pode bloquear o sistema operativo, enquanto que no ambiente .NET isso não acontece (Haddad, 2012).

5. IMPLEMENTAÇÃO DA SOLUÇÃO

A aplicação *Android* foi desenvolvida no ambiente de programação *Eclipse*, com recurso ao *SDK Android*. O *Eclipse* é um IDE desenvolvido em *JAVA*, seguindo o modelo *Open Source* de desenvolvimento de *software*.

A plataforma que recebe as solicitações da aplicação móvel e regista os dados de mobilidade associados foi desenvolvida na plataforma ASP.NET, com recurso à linguagem de programação C#.

Quer a aplicação *Android* quer a plataforma que recebe as solicitações dos utilizadores guardam os dados numa Base de Dados *Microsoft SQL Server*.

5.1. Ciclo de Vida de uma Aplicação *Android*

Para se começar a desenvolver aplicações baseadas no sistema operativo *Android* é necessário compreender o ciclo de vida de uma *Activity*, que não é mais do que uma tarefa daquilo que um utilizador pode fazer, sendo esta classe, *Activity*, responsável pelo ciclo de vida de uma aplicação.

A classe *Activity* é quem faz a gestão da *Interface* com o utilizador, recebe as requisições, trata-as e processa-as.

As *Activities* no sistema operativo são geridas como uma pilha de *Activities*, ou seja, quando uma *Activity* é iniciada é colocada no topo da pilha e torna-se na *Activity* em execução, o que faz com que a *Activity* anterior permanece mais em baixo na pilha, não sendo mostrada enquanto a *Activity* em execução não terminar.

Uma *Activity* pode assumir vários estados (Android Developers, 2012b):

- **Em execução**: se uma *Activity* está a ser executada e ao mesmo tempo está a ser mostrada no ecrã (que é o topo da pilha), está em modo *active* ou *running*.

- **Interrompida**: se uma *Activity* perdeu o foco mas ainda assim está visível, ou seja, uma nova *Activity* está a ser mostrada mas não ocupar o ecrã por completo, então está em modo *paused*. Uma *Activity* neste modo pode dizer-se que está completamente viva, uma vez que mantém todos os estados e informações ativos, podendo ser encerrada pelo sistema em caso de situações extremas, como por exemplo memória baixa.

- **Parada**: se uma *Activity* ficar em segundo plano, então passa a estar em modo *stopped*, mantendo ainda assim o seu estado e informações, não sendo no entanto visível ao utilizador, estando a sua janela escondida e podendo ser encerrada pelo sistema quando houver a necessidade de libertar memória.

- **Destruída**: se uma *Activity* está em modo *paused* ou *stopped*, o sistema pode retirá-la da memória, pedindo que a mesma seja finalizada ou simplesmente terminando o seu processo. Quando é mostrada novamente ao utilizador, terá de ser novamente reiniciada e restaurada para o seu estado anterior.

5.1.1. Máquina de Estados

Uma *Activity* possui métodos que ajudam o programador a controlar o estado da aplicação. De seguida é feita uma descrição de cada método (Android Developers, 2012b):

- ***onCreate()*** - É o primeiro método a ser executado numa *Activity*. Geralmente é responsável por carregar os *layouts* XML e outras operações de inicialização. É executada apenas uma vez.

- ***onStart()*** - É chamado imediatamente após o *onCreate()* e também quando uma *Activity* que estava em segundo plano volta a ter foco.

- **o*nResume()*** - Assim como o método *onStart()*, é chamado na inicialização da *Activity* e também quando uma *Activity* volta a ter foco. A diferença entre o método *onStart()* e o *onResume()* é que o primeiro só é chamado quando a *Activity* que não era mais visível volta a ter o foco, o segundo é chamado nas "retomadas de foco".

- ***onPause()*** - É o primeiro método a ser invocado quando a *Activity* perde o foco, ocorrendo quando uma nova *Activity* é iniciada.

- ***onStop()*** - É chamado quando a *Activity* fica completamente encoberta por outra *Activity*.

- ***onDestroy()*** – É o último método a ser executado antes da *Activity* ser destruida ou finalizada, podendo acontecer porque a *Activity* está realmente a ser encerrada a pedido do utilizador, ou porque o sistema operativo está a destruir a sua instância de forma temporária para aumentar espaço na memória. Se o utilizador voltar a requisitar essa *Activity*, um novo objeto será contruído.

- ***onRestart()*** – É chamado imediatamente antes do método *onStart()*, ou seja, quando uma *Activity* volta a ter o foco depois de estar em segundo plano.

  A imagem 5.1 ilustra os métodos de uma *Activity*.

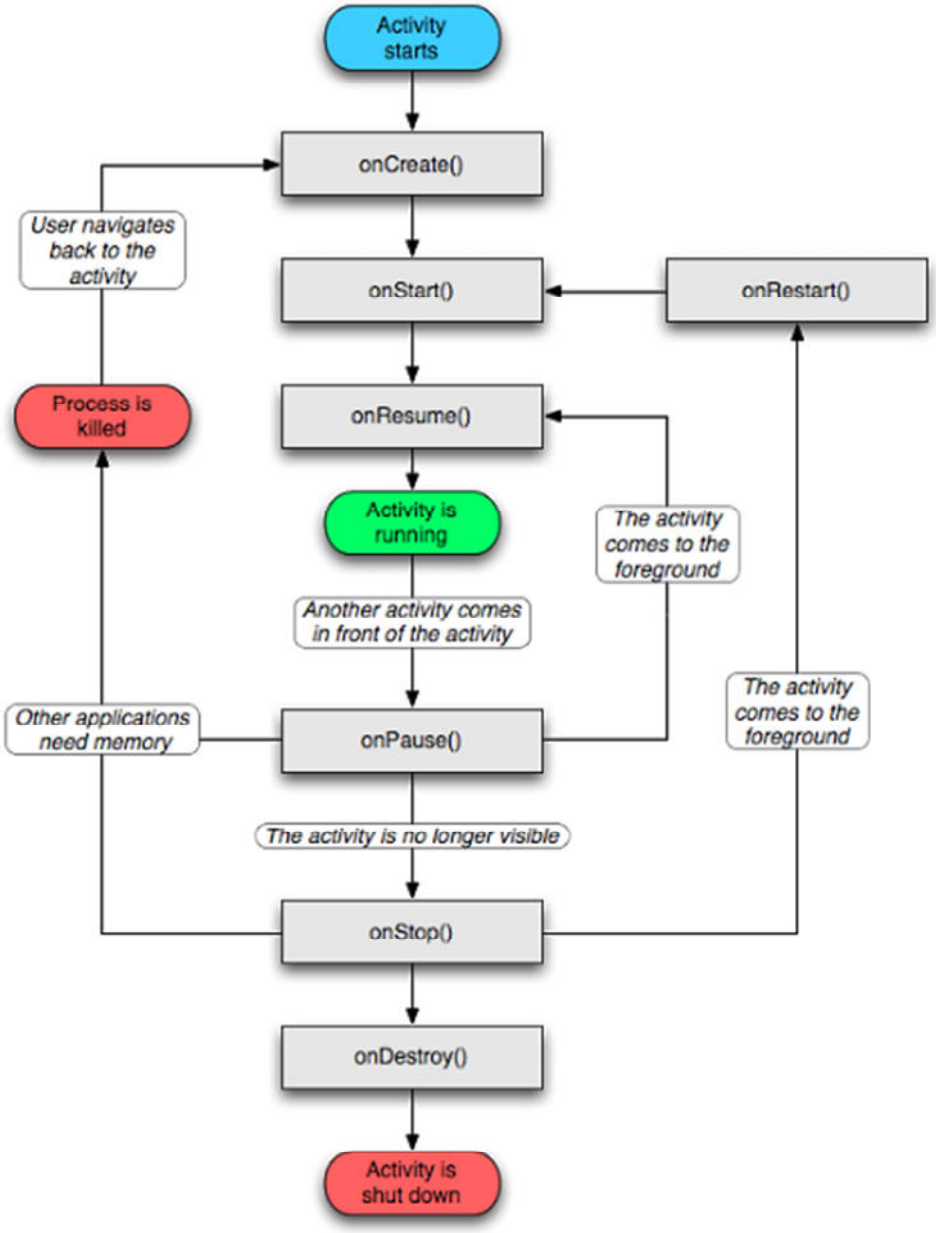


Figura 5.1 Ciclo de vida de uma aplicação *Android* (Android Developers, 2012b).

5.2. Arquitetura

A figura 5.2 mostra a estrutura da arquitetura da aplicação desenvolvida. Inicialmente o utilizador faz uma solicitação de um táxi, através do seu dispositivo

móvel, que comunica, através da internet, com o servidor *Web*, onde estão alojados os *Web Services*. Estes *Web Services*, por sua vez, vão comunicar com o servidor de Base de Dados, no qual está presente a Base de Dados do serviço, com o nome MeuTaxi, que armazena os dados dos pedidos dos utilizadores.

Os próprios Táxis vão também eles comunicar da mesma forma com o mesmo serviço, isto é, através de *Web Services*, embora este módulo não tenha sido desenvolvido, uma vez que não faz parte do âmbito do trabalho, estando previsto o desenvolvimento no Centro de Computação Gráfica.

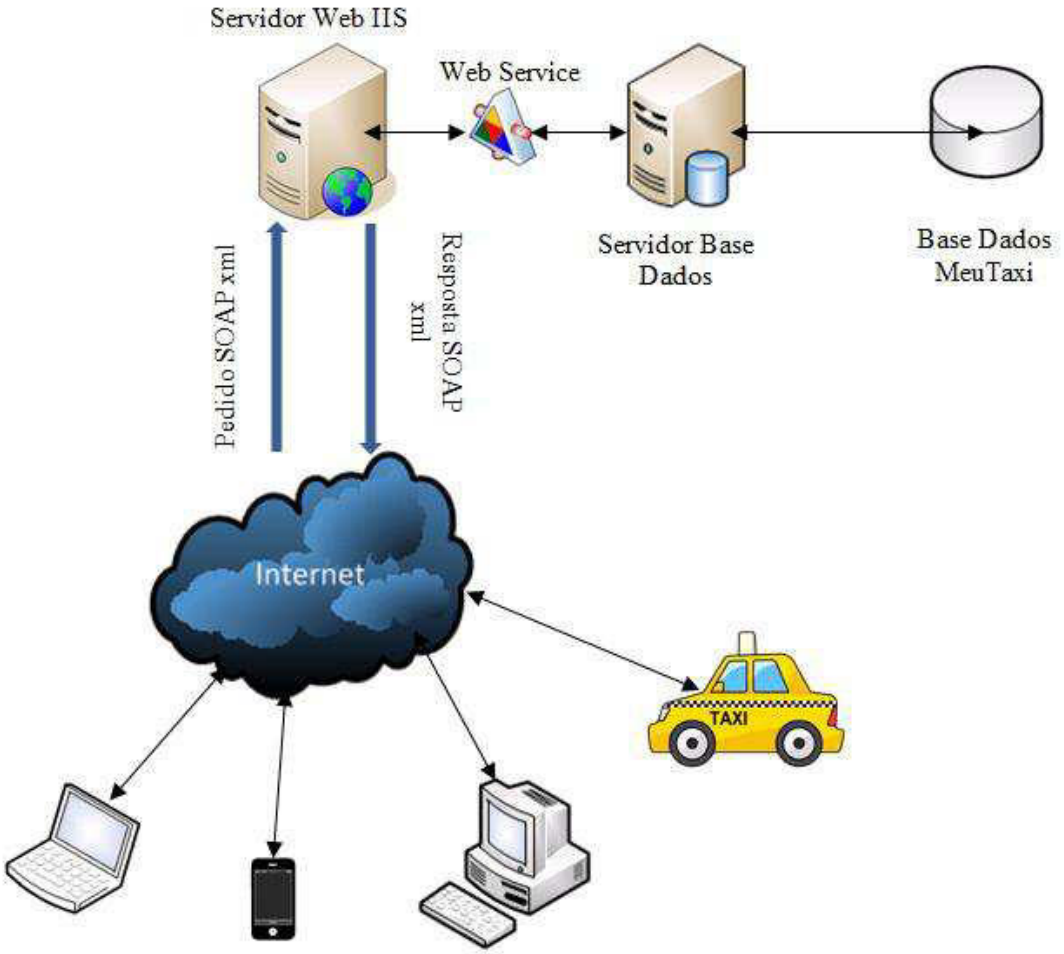


Figura 5.2 Arquitetura da solução.

5.3. Base de Dados

As solicitações dos utilizadores são guardadas numa Base de Dados desenvolvida na plataforma *Microsoft SQL Server*. Na figura 5.3 é apresentado o modelo relacional da Base de Dados.

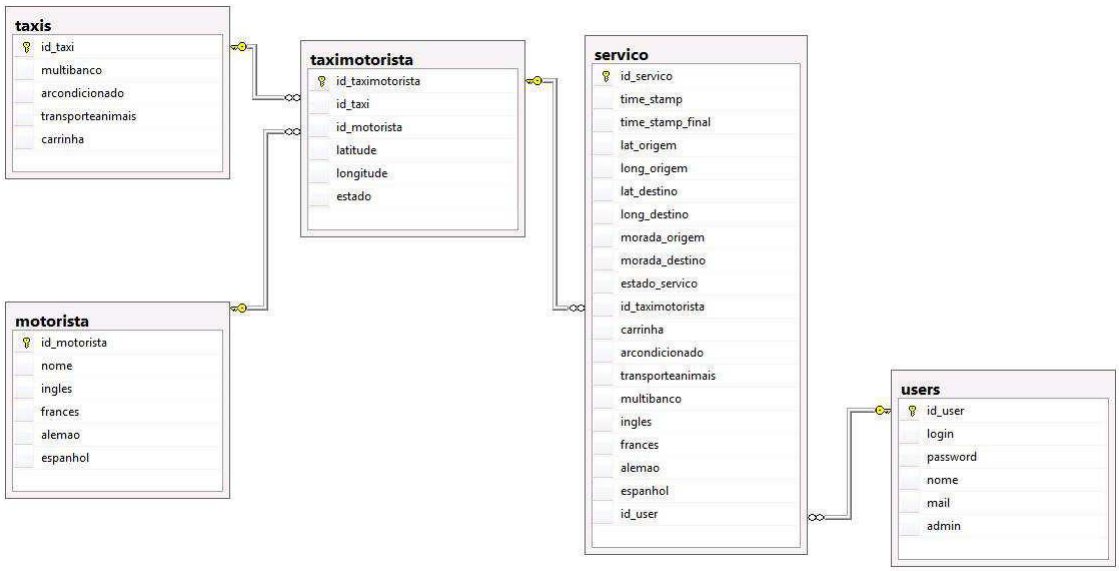


Figura 5.3 Modelo relacional da Base de Dados *Microsoft SQL Server.*

5.3.1. Dicionário de Dados

O dicionário de dados consiste numa lista organizada de todos os elementos de dados que são pertinentes para o sistema.

Assim, é apresentada, de seguida, uma descrição da estrutura das tabelas da Base de Dados, bem como o dicionário de dados do sistema a desenvolver, apresentando os respetivos atributos.

5.3.1.1. Tabela Users

A tabela **Users** guarda a informação que diz respeito aos utilizadores do sistema.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_User (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada utilizador [1, 99999999]
Login	nchar	50	Sim	login do utilizador para entrar no sistema [AZ az]
Password	nchar	50	Sim	Password do utilizador para entrar no sistema [AZ az]
Nome	varchar	50	Sim	Nome do utilizador [AZ az]
Mail	varchar	50	Sim	E-Mail do utilizador [AZ az]
Admin	tinyint	1	Sim	Privilégios do utilizador 0 - Utilizador 1- Administrador

Tabela 5-1 Estrutura da tabela Users.

5.3.1.2. Tabela Motorista

Na tabela **Motorista** são guardadas as informações que dizem respeito aos motoristas dos táxis, sendo mostrada na tabela 5.2 a sua estrutura e respetiva descrição dos seus atributos.

Campo	Tipo	Tamanho	Obrigatório	Descrição
id_motorista (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada motorista [1,99999999]
Nome	varchar	50	Sim	Atributo que representa o nome do motorista [AZ az]
Ingles	tinyint	1	Sim	O motorista tem conhecimentos da língua Inglesa 0 - Não 1- Sim
Frances	tinyint	1	Sim	O motorista tem

				conhecimentos da língua francesa 0 - Não 1- Sim
Alemao	tinyint	1	Sim	O motorista tem conhecimentos da língua alemã 0 - Não 1- Sim
Espanhol	tinyint	1	Sim	O motorista tem conhecimentos da língua espanhola 0 - Não 1- Sim

Tabela 5-2 Estrutura da tabela Motorista.

5.3.1.3. Tabela Taxis

A tabela **Taxis** guarda as informações que dizem respeito aos táxis e suas características, sendo mostrada na tabela 5.3 a sua estrutura e uma breve descrição dos seus atributos.

Campo	Tipo	Tamanho	Obrigatório	Descrição
id_taxi (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada táxi [1,99999999]
Multibanco	Tinyint	1	Sim	O táxi é detentor de pagamentos por multibanco 0 - Não 1- Sim
Arcondicionado	Tinyint	1	Sim	O táxi possui sistema de ar condicionado 0 - Não 1- Sim
Transporteanimais	Tinyint	1	Sim	O táxi faz transporte de animais 0 - Não 1- Sim
Carrinha	Tinyint	1	Sim	Atributo que representa se o táxi em questão possui mais de quatro lugares

				0 - Não 1- Sim

Tabela 5-3 Estrutura da tabela Taxis.

5.3.1.4. Tabela TaxiMotorista

A tabela **TaxiMotorista** resulta de uma divisão da tabela Taxis e da tabela Motorista, uma vez que tínhamos uma relação de N para N. Nesta tabela são guardadas as informações relativas a cada motorista e a cada táxi que estão ao serviço, sendo mostradas na tabela 5.4 a descrição dos seus atributos bem como a sua estrutura.

Campo	Tipo	Tamanho	Obrigatório	Descrição
id_taximotorista (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada táxi e motorista ao serviço [1,99999999]
id_taxi (FK)	Inteiro	8	Sim	Identifica o táxi que se encontra ao serviço [1,99999999]
id_motorista (FK)	Inteiro	8	Sim	Identifica o motorista que se encontra ao serviço [1,99999999]
Latitude	Decimal	(12, 9)	Sim	Identifica a latitude do táxi. Este atributo será atualizado de forma dinâmica
Longitude	Decimal	(12, 9)	Sim	Identifica a longitude do táxi. Este atributo será atualizado de forma dinâmica
Estado	Tinyint	1	Sim	Identifica o estado atual. Este atributo pode assumir os seguintes valores: 0 – Ocupado 1 – Livre 2 – Indisponível

Tabela 5-4 Estrutura da tabela TaxiMotorista.

5.3.1.5. Tabela Servico

Um utilizador ao submeter um pedido de um táxi, o sistema regista, na tabela **Servico**, as informações do serviço solicitado pelo mesmo, desde morada de origem, morada de destino, pagamentos por multibanco, transporte de mais de quatro passageiros ou ainda selecionar um motorista que fale mais do que a sua língua materna (entenda-se português).

Na tabela 5.5 é mostrada uma descrição dos atributos da tabela Servico, bem como a sua estrutura.

Campo	Tipo	Tamanho	Obrigatório	Descrição
id_servico (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada serviço [1,99999999]
time_stamp	smalldatetime		Sim	Data e hora do pedido
time_stamp_final	smalldatetime		Não	Data e hora em que o serviço é finalizado
lat_origem	Decimal	(12, 9)	Sim	Latitude de origem do utilizador
long_origem	Decimal	(12, 9)	Sim	Longitude do utilizador
lat_destino	Decimal	(12, 9)	Não	Latitude do local de destino do utilizador
long_destino	Decimal	(12, 9)	Não	Longitude do local de destino do utilizador
morada_origem	Varchar	100	Sim	Morada de origem do utilizador [AZ az]
morada_destino	Varchar	100	Não	Morada de destino do utilizador [AZ az]
estado_servico	Tinyint	1	Sim	Atributo que identifica o estado atual do serviço, podendo assumir vários valores: 0 – Foi solicitado 1 – A ser executado 2 – Já foi executado 3 – Anulado
id_taximotorista (FK)	Inteiro	8	Não	Atributo que identifica o táxi e o motorista ao serviço
Carrinha	Tinyint	1	Não	Táxi com mais de

				quatro lugares 0 - Não 1- Sim
Arcondicionado	Tinyint	1	Não	Táxi com sistema de ar condicionado 0 - Não 1- Sim
Transporteanimais	Tinyint	1	Não	Táxi com a possibilidade de transportar animais 0 - Não 1- Sim
Multibanco	Tinyint	1	Não	Táxi que suporte pagamentos com multibanco 0 - Não 1- Sim
Ingles	Tinyint	1	Não	Motorista com conhecimentos da língua inglesa 0 - Não 1- Sim
Francês	Tinyint	1	Não	Motorista com conhecimentos da língua francesa 0 - Não 1- Sim
Alemão	Tinyint	1	Não	Motorista com conhecimentos da língua alemã 0 - Não 1- Sim
Espanhol	Tinyint	1	Não	Motorista com conhecimentos da língua espanhola 0 - Não 1- Sim
id_user (FK)	Inteiro	8	Sim	Atributo que identifica o utilizador que solicitou um táxi

Tabela 5-5 Estrutura da tabela Servico.

5.4. *FrontOffice*

O *FrontOffice* do projeto diz respeito à aplicação móvel, ou seja, tudo o que o utilizador final tem acesso, desde a possibilidade de poder "chamar" um táxi, como poder consultar o seu histórico de pedidos e seus trajetos. Entenda-se por utilizador final, todo aquele que, depois de instalar a aplicação no seu dispositivo móvel e efetuar o registo para poder ter acesso à aplicação, pretende "chamar" um táxi através de um dispositivo móvel.

5.4.1. Sistema de Navegação da Aplicação Móvel

O utilizador da aplicação móvel tem acesso a diversos ecrãs, designados de *Activities,* por onde pode navegar livremente a fim de satisfazer as suas necessidades, podendo consultar um histórico dos seus pedidos, bem como efetuar o pedido de um serviço de táxi. Assim, na figura 5.4, é mostrado o esquema de navegação das diversas *Activities* do projeto.

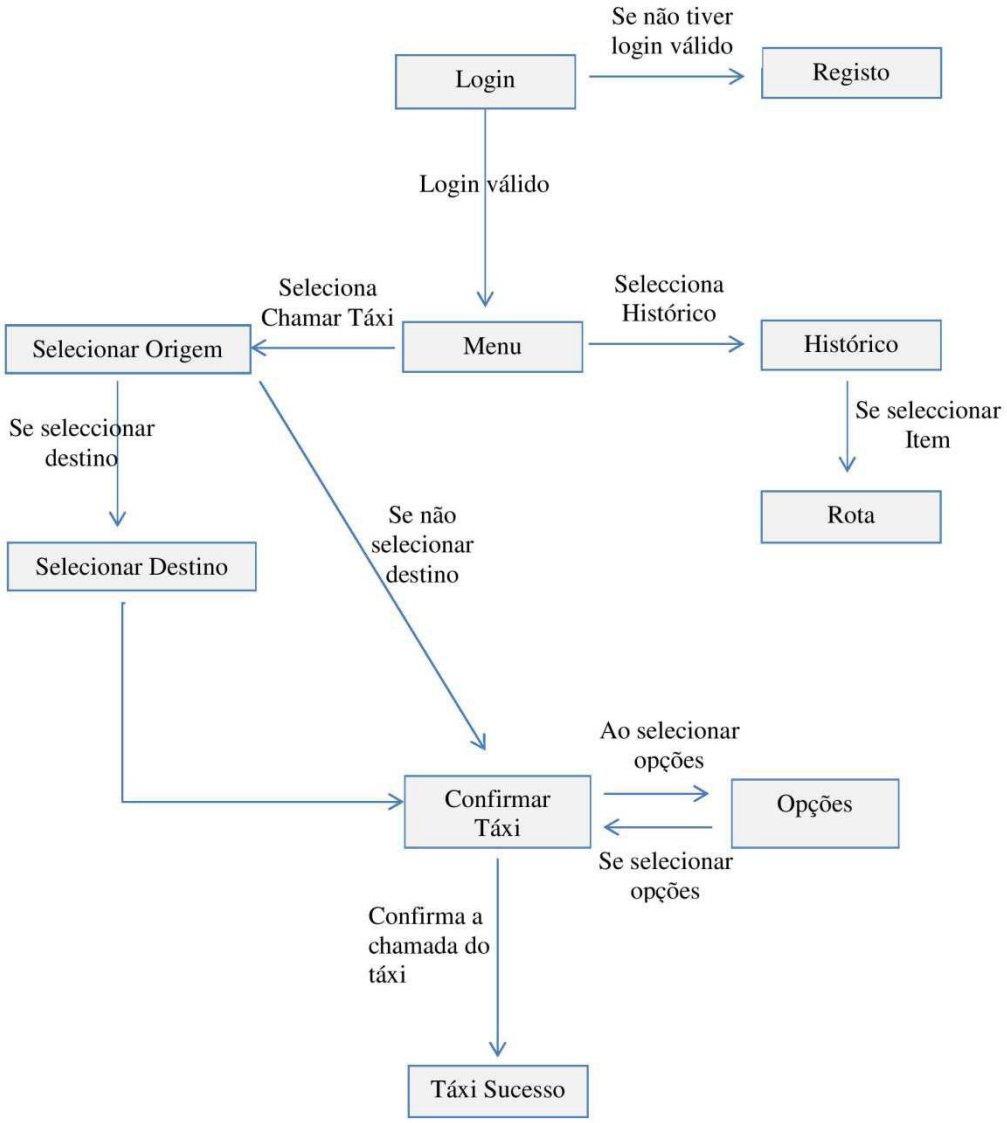


Figura 5.4 Esquema de navegação pelas *Activities* do projeto.

5.4.2. Acesso aos Dados

A comunicação entre a aplicação móvel e o servidor de Base de Dados é feita através de *Web Services*. Estes *Web Services* foram desenvolvidos na linguagem de programação C#, fazendo portanto sentido referir como é feito o acesso aos dados a partir desta linguagem. O acesso aos dados é feito através de métodos específicos, que passam a ser descritos a seguir:

- ***SQLConnection***

  Objeto responsável por estabelecer a ligação a uma Base de Dados *SQL Server*. Este método está sempre acompanhado de um prefixo estabelecido pela classe *Connection* do *Namespace* em que trabalhamos, sendo neste caso o *Namespace System.Data.SqlClient*.

- ***SQLDataAdapter***

  Objeto utilizado para preencher ou manipular um *DataSet*, ou seja, é um conjunto de dados e ligações que preenchem um *DataSet*.

- ***DataSet***

  Este objeto funciona como uma cópia de uma base de dados que é criada temporariamente em memória, permitindo assim que o utilizador possa inserir, alterar e eliminar os dados pretendidos.

- ***SQLCommand***

  Este objeto é utilizado para executar comandos de SQL nas bases de dados, executando as operações de consulta, inserção, atualização e eliminação de dados.

5.4.3. *Activity* Principal – *Login*

Na *Activity Login*, o utilizador insere os dados de autenticação, número de telemóvel e *password*, nos respetivos campos, tal como mostra a figura 5.5. Ao serem submetidos, a password é encriptada em MD5 e os dados vão ser comparados com os dados que constam na Base de Dados e, no caso de serem válidos, o utilizador passa então a ter acesso à aplicação.

O processo de autenticação é efetuado com base em *Web Services*, ou seja, a aplicação comunica com o *Web Service*, que se encontra alojado no servidor, ficando

este responsável por comunicar com a Base de Dados e, assim que seja executada a tarefa, retorna um *feedback* do resultado obtido.

O *Web Service* responsável pela autenticação é mostrado no Anexo C.

Figura 5.5 *Activity Login*.

Para a aplicação comunicar com o *Web Service*, foi desenvolvido um método responsável por esta comunicação, permitindo passar os parâmetros necessários para que a autenticação tenha sucesso. O método responsável por fazer a comunicação com o *Web Service* é mostrado no Anexo D.

Caso o utilizador introduza as credenciais corretas, é então concedida autorização para poder aceder à aplicação e daí usufruir dos recursos que são permitidos, tais como, chamar um táxi e consultar o seu histórico.

Para ajudar ao processo de autenticação, a aplicação fornece, de forma automática, no campo do login, o número de telemóvel que se encontra no aparelho, como é mostrado de seguida.

Listagem 2 Método que fornece o número de telemóvel do *smartphone*

```
1  private String returnPhoneNumber()
2  {
3      TelephonyManager mTelephonyMgr;
4      mTelephonyMgr = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
5      return mTelephonyMgr.getLine1Number();
6  }
```

5.4.4. *Activity* Registo

Todo o utilizador não registado, tem nesta *Activity* a possibilidade de o fazer. Esta *Activity* encontra-se acessível a partir da *Activity Login*. Para se registar, o utilizador apenas tem de preencher os campos necessários que se apresentam no *layout* da *Activity*, como mostra a figura 5.6.

O princípio de funcionamento da comunicação da aplicação com o *Web Service* é o mesmo descrito no anexo D – Método Responsável por fazer a comunicação com o *Web Service*, isto é, vão ser passados os dados introduzidos pelo utilizador ao *Web Service*, ficando este responsável pela introdução desses mesmos dados na Base de Dados.

Figura 5.6 *Activity* Registo.

Para que os dados sejam introduzidos na base de dados é feita uma verificação a fim de determinar se o utilizador com aquele *login* já existe. No caso de se verificar a existência de um *login* igual, o *Web Service* devolve uma mensagem alertando o utilizador que o *login* já existe.

Por fim, caso o *login* introduzido seja validado com sucesso, é então feito o registo do utilizador na base de dados. No Anexo E - é mostrado o *Web Service* responsável por introduzir os dados que dizem respeito ao registo do utilizador na aplicação.

5.4.5. *Activity* Menu

Quando o utilizador se autentica com sucesso, a *Activity* Menu é a *Activity* que se torna visível ao utilizador. Esta *Activity* possibilita ao utilizador chamar um táxi

(se selecionar a opção Chamar Táxi) ou consultar o seu histórico de serviços efetuados (se selecionar a opção Histórico), como mostra a figura 5.7.

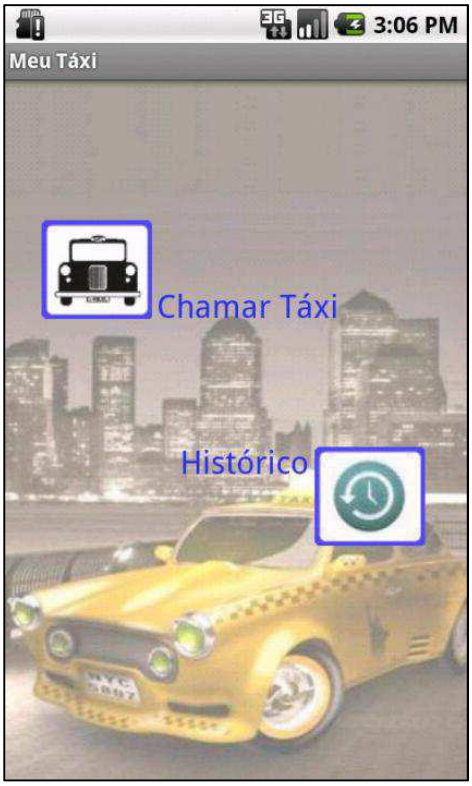


Figura 5.7 *Activity* Menu.

O utilizador ao selecionar a opção Chamar Táxi, é feita uma verificação na base de dados de modo a averiguar se existem pedidos por concluir referentes ao utilizador em questão. No caso de existirem pedidos por concluir é mostrado um alerta ao utilizador.

5.4.6. *Activity* Chamar Táxi

A *Activity* Chamar Táxi, mostrada na figura 5.8, é a *Activity* que aparece ao utilizador quando escolhe a opção Chamar Táxi na *Activity* Menu.

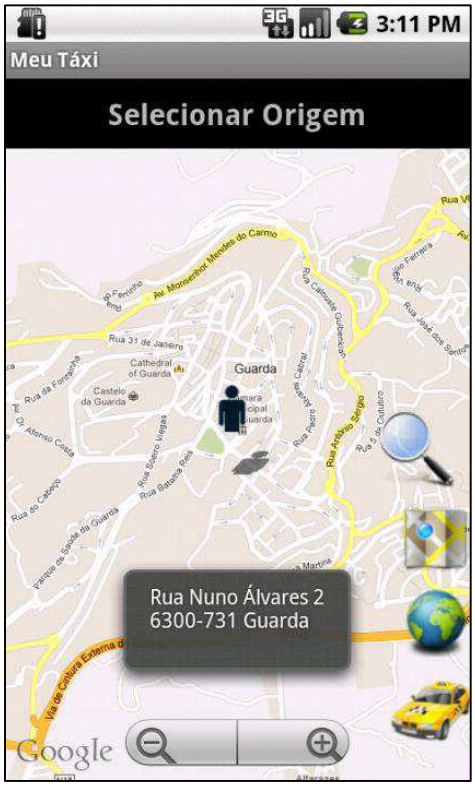


Figura 5.8 *Activity* Chamar Táxi.

Quando o utilizador entra nesta *Activity*, é de imediato ativado o GPS (caso o aparelho seja detentor da respetiva antena), que fornece e assinala a localização no mapa através de uma marca.

No Anexo F – Método Localização, pode consultar-se o código que determina a localização do utilizador.

O utilizador, ao clicar no *icon* Lupa, tem a possibilidade de poder fazer uma pesquisa pelo nome de uma localidade ou morada, de onde pretende chamar o táxi. Para que tal aconteça, apenas tem de introduzir o texto a pesquisar no campo "Escreva o Endereço" e de seguida clicar no botão OK. Ao fazer este procedimento, o mapa é deslocado para o local que diz respeito à pesquisa que o utilizador introduziu, posicionando-se no centro do ecrã do *smartphone*.

Quando for selecionado o *icon* Globo, a visualização do mapa é alterada para vista satélite, permitindo ao utilizador ter uma maior perceção do mapa. Se por outro lado for selecionado o *icon* Mapa, a visualização do mapa volta para a vista mapa.

O comando *map.setSatellite(true);* é o método responsável por permitir ao utilizador alternar entre a vista satélite e a vista normal.

Para que o utilizador possa selecionar uma localização de forma manual, apenas tem de tocar no mapa. Ao tocar no mapa, o utilizador define o seu local de origem, como sendo a morada de origem, morada esta onde o táxi irá "buscar" o utilizador. Para que esta escolha aconteça, o utilizador apenas tem de ir manobrando o mapa, isto é, fazendo *zoom*, arrastando (entre outros movimentos), até encontrar o local pretendido para que o táxi o possa ir "buscar". Ao tocar no mapa, é desenhada uma marca e consequentemente aparece uma pequena mensagem, designada de *Toast*, ao fundo do ecrã do *smartphone*, que mostra a morada do local selecionado, para que desta forma possa ter uma maior perceção do local que está a escolher.

O evento *onTouchEvent(MotionEvent event, MapView mapview1)* permite realizar o que foi descrito anteriormente, sendo o seu código mostrado no Anexo H – Evento *onTouch.* Este recebe como parâmetros um objeto do tipo *MotionEvent* e outro do tipo *MapView*. A classe *MotionEvent* possui informações relativas ao toque do utilizador, tais como (Android On Board, 2012):

- **A acção**: valor do tipo inteiro que pode ser acedido através do método *getAction()*, sendo o *onTouchEvent* acionado devido a três ações básicas, ***ACTION_DOWN*****,** ***ACTION_UP*** e ***ACTION_MOVE*****, que** significam clicar, soltar e mover o dedo pelo ecrã respetivamente.

- **Coordenada X e Y:** São dois valores do tipo *float* que indicam a coordenada do ponto do ecrã que foi clicado.

Uma vez selecionado o local que corresponde à morada de origem, o utilizador pode avançar com o pedido, bastando para isso que clique no *icon* táxi. Ao selecionar esta opção, será mostrada uma caixa de texto que questiona o utilizador se pretende introduzir uma morada de destino e, para o caso de responder afirmativamente, será reencaminhado para a *Activity* Destino, mostrada de seguida.

5.4.7. *Activity* Destino

A *Activity* Destino, mostrada na figura 5.9, é a *Activity* onde o utilizador pode introduzir a morada para onde se pretende deslocar, correspondendo à morada de destino. Esta *Activity* é em tudo semelhante à já apresentada no ponto anterior (*Activity* Chamar Táxi), permitindo o mesmo leque de opções que a anterior. Esta *Activity* apenas vai diferir da anterior num ponto, isto é, vai ser apresentada uma marca, que vai permanecer sempre estática, representando a localização original (morada de origem) do utilizador.

Quando o utilizador tiver selecionado um ponto correspondente ao local para onde se pretende deslocar (morada de destino), pode prosseguir com o pedido, clicando no *icon* táxi.

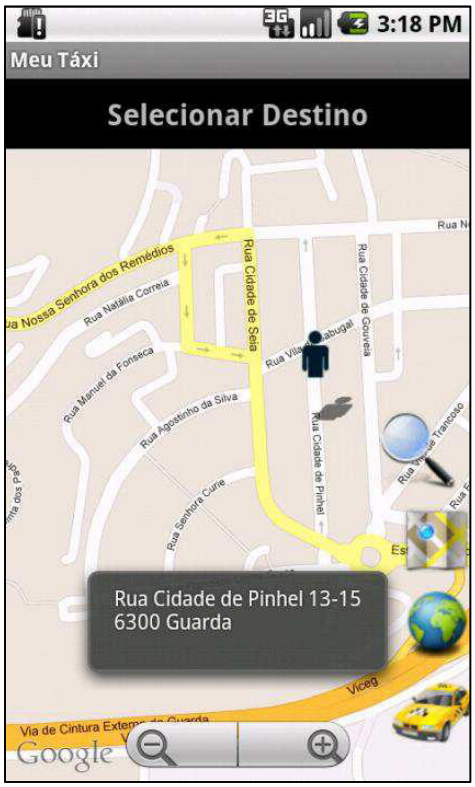


Figura 5.9 *Activity* Destino.

5.4.8. *Activity* Confirmar Táxi

A *Activity* da figura 5.10 mostra um resumo dos dados solicitados pelo utilizador, desde a morada de origem, morada de destino (caso tenha sido introduzida), a distância aproximada entre a morada de origem e a de destino e um valor estimado do custo da viagem. Esta informação é mostrada ao utilizador a fim de verificar com detalhe as preferências selecionadas e para que não haja erro algum antes de submeter o pedido.

O utilizador tem ainda acesso a um conjunto de opções que podem satisfazer melhor as suas necessidades, tais como, escolher uma viatura com um maior número de lugares, com possibilidade de pagamentos por multibanco, com possibilidade de transporte de animais, com ar condicionado, ou também ter a possibilidade de escolher um motorista que fale outras línguas, como o caso do inglês, alemão, espanhol ou francês. Estas opções estão descritas na figura 5.11 e explicadas mais à frente.

Figura 5.10 *Activity* Confirmar Táxi.

Para o cálculo da distância entre a morada de origem e a morada de destino, inicialmente optou-se pelo método disponibilizado pela *class* da *Google Maps API*, o método *distanceTo*(), mas este método não se mostrou totalmente eficaz no cálculo da distância, uma vez que não é possível determinar por que ruas o táxi pode ou não passar, podendo a distância ser maior ou menor conforme os casos. Após uma análise ao serviço da *Google Directions API*, concluiu-se que este serviço se mostrou mais rigoroso na obtenção da distância entre as moradas de origem e de destino, uma vez que tem em atenção os locais por onde o táxi pode passar, sendo rigoroso ao ponto de perceber que há dadas ruas que apenas têm um sentido e que o táxi terá de dar uma volta maior para poder alcançá-las.

Este serviço disponibiliza um documento *Json* com todas as informações do percurso que o táxi irá efetuar, muito usado nos aparelhos de GPS convencionais, mostrando os percursos que os automobilistas podem percorrer até chegarem ao seu destino.

O código do Anexo I – Método Responsável por Ler um Documento *Json*, mostra como foi feita a leitura da distância a partir do documento disponibilizado pelos serviços da *Google*. A leitura do documento *Json* é feita respeitando a sua estrutura hierárquica, como mostra o Anexa A – Estrutura do Resultado *Json*. Após a leitura deste documento é possível obter a distância entre os dois pontos (morada de origem e morada de destino).

O cálculo do custo aproximado da viagem foi efetuado com base nas tarifas da tabela disponibilizada pela Federação Portuguesa do Táxi, mostrada no **Anexo B**. De seguida é mostrado o algoritmo usado no cálculo do custo da viagem.

Listagem 3 Algoritmo para o cálculo do custo da viagem

```
1   Algoritmo: CalculaTarifaQuilometro(distancia, ida_volta, np, hora)
2   Objetivo: Permite calcular um valor estimado da tarifa ao quilómetro a pagar por uma
3   viagem de táxi.
4   Constantes:
5     P4T3 [2] (Real T5.2) - 4 Passageiros - Tarifa 3 retorno em vazio - diurna / noturna
6   (€) (Valor: 0.94,1.13)
7     P4T5 [2] (Real T5.2) - 4 Passageiros - Tarifa 5 retorno ocupado - diurna / noturna
8   (€) (Valor: 0.47,0.56)
9     P5T3 [2] (Real T5.2) - + 4 Passageiros - Tarifa 3 retorno em vazio - diurna /
10  noturna (€) (Valor: 1.21,1.45)
```

```
  P5T5 [2] (Real T5.2) - + 4 Passageiros - Tarifa 5 retorno ocupado - diurna /
noturna (€) (Valor: 0.61,0.73)
Parâmetros
  Entrada:
    distancia (Real T6.3) - Distância (Km) (>= 0.0, <= 999.999)
    ida_volta (Inteiro T1) - Viagem de ida e volta (>= 0, <= 1)
    np (Inteiro T2) - Número de passageiros (>= 1, <= 12)
    hora ( T2) - Hora (>= 0, <= 23)
  Saída:
    tarifa (Real T6.2) - Valor da tarifa (€) (>= 0.0, <= 9999.99)
Data: 2013-5-28
Autor: João Miranda
Versão: 1.0
Início:
  SE np <= 4 ENTÃO
    SE ida_volta = 0 ENTÃO
      SE hora <= 21 ENTÃO
        T = P4T3[1]
      SENÃO
        T = P4T3[2]
      FIMSE
    SENÃO
     SE hora <= 21 ENTÃO
        T = P4T5[1]
      SENÃO
        T = P4T5[2]
      FIMSE
    FIMSE
  SENÃO
    SE ida_volta = 0 ENTÃO
      SE hora <= 21 ENTÃO
        T = P5T3[1]
      SENÃO
        T = P5T3[2]
      FIMSE
    SENÃO
     SE hora <= 21 ENTÃO
        T = P5T5[1]
      SENÃO
        T = P5T5[2]
      FIMSE
    FIMSE
  FIMSE
  RETORNA tarifa
Fim.
```

Não sendo fácil determinar as zonas de mudança de tarifa, tarifa 1, para o cálculo do valor nos serviços urbanos (serviço muito utilizado nos grandes centros urbanos), optou-se então pelo cálculo baseado nas tarifas 3 e 5, retorno em vazio e retorno com cliente respetivamente, sendo este cálculo uma aproximação do custo da viagem. Na tabela seguinte são apresentados os valores das tarifas 3 e 5.

Tarifa ao Quilómetro

Nº Lugares	Bandeirada		Km	Hora	Frações			
Tarifas	Metros	€	€	€	Metros	€	Seg.	€
4 passageiros								
Tarifa 3 retorno em vazio								
- diurna	1800	3,25	0,94	14,80	106,38	0,10	24,0	0,10
- noturna	1800	3,90	1,13	14,80	88,50	0,10	24,0	0,10
Tarifa 5 retorno ocupado								
- diurna	3600	3,25	0,47	14,80	212,77	0,10	24,0	0,10
- noturna	3600	3,90	0,56	14,80	178,57	0,10	24,0	0,10
+4 passageiros								
Tarifa 3 retorno em vazio								
- diurna	1400	3,25	1,21	14,80	82,65	0,10	24,0	0,10
- noturna	1400	3,90	1,45	14,80	68,97	0,10	24,0	0,10
Tarifa 5 retorno ocupado								
- diurna	2800	3,25	0,61	14,80	163,93	0,10	24,0	0,10
- noturna	2800	3,90	0,73	14,80	136,99	0,10	24,0	0,10

Tabela 5-6 Tabela das tarifas 3 e 5.

O utilizador, ao selecionar o botão "Opções", tem acesso a um conjunto de opções que lhe permitem escolher os requisitos do seu pedido, como mostra a figura 5.11.

Figura 5.11 *Activity* Opções.

Para submeter o pedido o utilizador deve carregar no botão Ok. O *Web Service* apresentado no Anexo J permite armazenar os dados do pedido do utilizador na base de dados.

5.4.9. *Activity* Táxi Sucesso

O utilizador ao submeter o pedido passa a ter acesso à *Activity* da figura 5.12. Após a atribuição de um táxi ao seu pedido é colocado o *ícon* de um táxi no mapa, de acordo com as suas coordenadas, e no topo do ecrã a identificação do táxi e o tempo estimado de chegada.

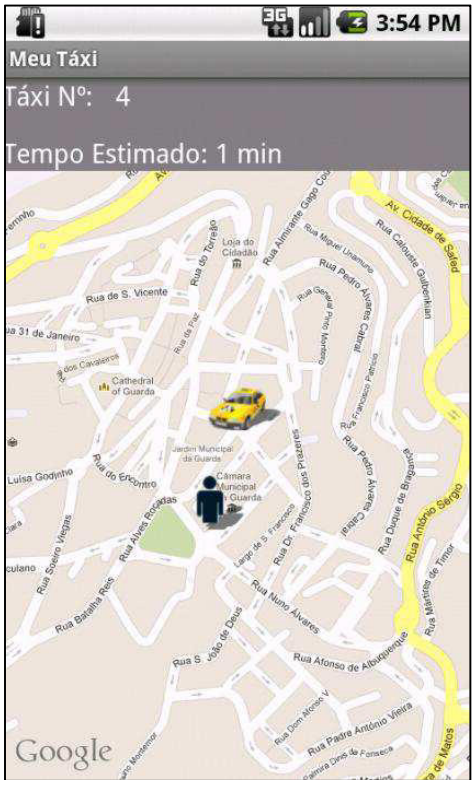


Figura 5.12 *Activity* Táxi Sucesso.

O utilizador pode visualizar também a deslocação do táxi no próprio mapa, de forma a ter uma melhor perceção da distância e do tempo que este pode demorar a chegar até si. Esta informação é atualizada de 30 em 30 segundos, fazendo deslocar o

mapa e centrando a posição do táxi para que desta forma pareça mais realista ao utilizador.

A obtenção dos dados referidos anteriormente é feita da seguinte forma: a aplicação passa como parâmetro o ID do utilizador para o *Web Service* responsável por devolver os dados do táxi (Anexo K). O serviço *Web* devolve então a posição geográfica do táxi. Uma vez obtidas as coordenadas latitude e longitude do táxi, é possível então determinar a distância aproximada que falta ao táxi percorrer para chegar junto do utilizador, como mostra o Anexo I.

Neste ecrã é ainda possível saber o tempo estimado que o táxi demora para chegar junto do utilizador, (correspondendo à morada de origem do mesmo). O cálculo deste tempo estimado é obtido da mesma forma que é determinada a distância, ou seja, é necessário extrair do documento *Json* a *tag* referente ao tempo, designada por "*duration*", como é mostrado no Anexo I, bastando apenas substituir a *tag* "*distance*" pela tag "*duration*".

5.4.10. *Activity* Histórico

No ecrã correspondente ao Menu, o utilizador, para além de poder chamar um táxi, como foi demonstrado nas secções anteriores, tem ainda a possibilidade de consultar o histórico dos seus pedidos. Ao selecionar a opção "Histórico", o utilizador é encaminhado para a *Activity* da figura 5.13. Neste histórico o utilizador tem a possibilidade de visualizar os dados relativos aos serviços que efetuou, tais como a data e hora do serviço, bem como a morada de origem e destino do pedido.

Neste histórico, é dada a possibilidade do utilizador poder anular um pedido, desde que este ainda se encontre em aberto, bastando para isso clicar no botão "Anular" do respetivo pedido.

Figura 5.13 *Activity* Histórico.

Ao clicar no botão "Anular", o utilizador cancela o pedido de táxi para aquela morada, ou seja, o motorista de táxi não vai mais ao seu encontro.

A aplicação passa o ID do serviço a ser anulado como parâmetro para o *Web Service* efetuar a anulação do pedido do cliente na Base de Dados, como mostra o código do Anexo L – *Web Service* Responsável por Anular um Pedido.

5.4.11. *Activity* Map

Quando o utilizador clicar num item do *ListView*, mostrado na figura 5.13, que não seja anulado nem esteja aberto, ou seja, que já tenha sido efetuado, vai ser mostrado um ecrã com uma rota desenhada, bem como os pontos de partida (que representa a morada de origem da viagem) e o ponto de chegada (que representa a morada de destino da viagem) como mostra a figura 5.14.

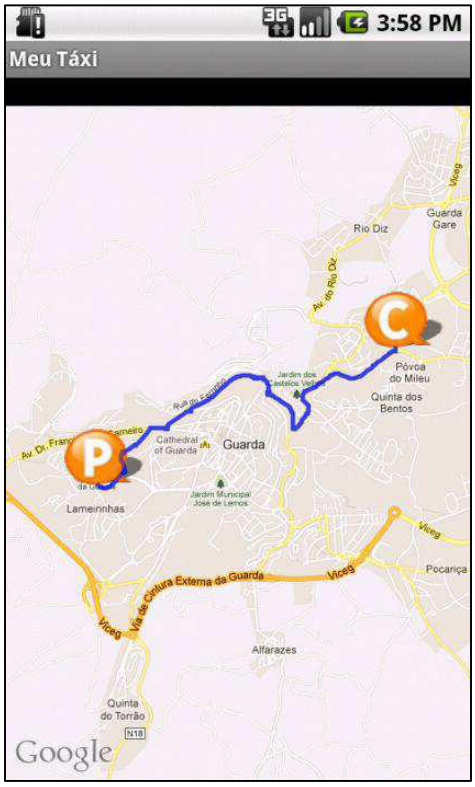


Figura 5.14 Rota realizada na viagem.

Esta rota é apenas uma ilustração do possível caminho que o taxista podia ter tomado para ir da morada de origem para a morada de destino, não sendo por isso a rota verdadeira, uma vez que, como não existe o módulo que contempla a comunicação do táxi com a plataforma, não é possível a leitura constante das coordenadas geográficas do táxi para determinar o verdadeiro trajeto da viagem.

Para desenhar a rota, houve a necessidade de ler o documento *Json*, obtido através da *Google Directions API*, para extrair as coordenadas do trajeto e posteriormente converter de base64 para *Geopoint*, uma vez que a rota com as coordenadas mais precisas estão codificadas em base64 no atributo *polyline*, como mostra o código do Anexo M – Desenha Rota (Leal, 2012).

5.5. *BackOffice*

O *backOffice* é o espaço reservado onde os utilizadores podem consultar o seu histórico, mas é também um espaço reservado para administradores, onde podem fazer toda a gestão dos pedidos dos utilizadores, ou seja, é a partir desta plataforma que o administrador pode atribuir um táxi a um utilizador, libertar um táxi (uma vez que não temos o terceiro módulo), e ainda obter um conjunto de dados que nos permite visualizar, num aglomerado de pontos, as zonas onde as pessoas efetuam mais pedidos e ao mesmo tempo as zonas para onde as pessoas mais se deslocam.

A atribuição do táxi a um cidadão é feita de forma manual por um administrador, uma vez que se encontra numa fase de testes, embora seja intenção de no futuro passar a ser de forma autónoma, em que o pedido chegue à plataforma, seja processado, e ao mesmo tempo seja atribuído um táxi ao utilizador.

5.5.1. Histórico

Um utilizador, ao aceder à plataforma *Web*, tem a possibilidade de poder consultar o histórico referente às suas viagens, podendo visualizar a data e hora do pedido, data e hora em que o pedido foi efetuado, morada de origem e morada de destino, como mostra a figura 5.15.

Para o efeito é feita uma consulta à base de dados onde apenas vão ser mostrados os pedidos já realizados para aquele utilizador, ou seja, os pedidos que foram realizados com sucesso. Posteriormente é preenchida uma tabela, *GridView*, onde são mostrados os resultados que correspondem à pesquisa efetuada, como mostra o código do Anexo N – Método Carrega *Grid*.

Figura 5.15 Página Histórico do utilizador.

5.5.2. Alterar *Password*

Esta página é responsável por oferecer ao utilizador a possibilidade de alterar a sua *password*, como mostra a figura 5.16.

Figura 5.16 Página Alterar *Password*.

O utilizador, para alterar a *password*, tem de se certificar que os dados que está a introduzir nos campos "*Password*" e "Conf. *Password*" correspondem, caso contrário a aplicação não deixa o utilizador prosseguir na alteração da password. Após o

utilizador clicar no botão "Alterar", e os dados dos campos "*Password*" e "Conf. *Password*" coincidirem, a nova password é encriptada e armazenada na Base de Dados.

5.5.3. Administração

Nesta página o administrador pode fazer a gestão dos pedidos efetuados pelos utilizadores, ou seja, visualiza os pedidos e atribui o táxi que se encontra mais próximo e aquele que demora menos tempo a chegar ao utilizador. O administrador recebe estas informações (descritas anteriormente) num mapa, onde pode visualizar a posição exata do utilizador e ao mesmo tempo visualiza também a posição dos cinco táxis que se encontram mais próximos e que ao mesmo tempo demorem o menor tempo possível a chegar ao utilizador. O táxi que corresponder a estes critérios, destaca-se com uma cor diferente dos restantes (passando a mostrar a cor verde), sendo ainda visível um pequeno balão por cima do táxi que indica a informação relativa ao mesmo, bem como o tempo estimado que demora a chegar ao utilizador, como mostra a figura 5.17.

	id_servico	Data/Hora	Morada Origem	Morada Destino	Telémovel	Nome
Select	6	27-12-2012 18:26:00	Rua da Vila de Celorico da Beira 4 6300 Guarda	Rua Dom Nuno Alvares Pereira 48 6300-721 Guarda	962848132	João Miranda
Select	7	28-12-2012 23:25:00	Rua General Santos Costa 10 3400-124 Oliveira do Hospital	Rua Eça de Queirós 7-9 3400 Oliveira do Hospital	962848132	João Miranda
Select	8	30-12-2012 13:14:00	Rua António Nobre 40 3400 Oliveira do Hospital	Avenida Doutor Rosa Pimenta 17 3400-473	962848132	João Miranda
Select	9	30-12-2012 18:51:00	Rua Doutor António Ribeiro Garcia de Vasconcelos 20 3400-075 Oliveira do Hospital	Rua Escola 3 3400-703 São Paio de Gramaços	962848132	João Miranda
Select	10	04-01-2013	Rua Ameal 22 3400 Oliveira do Hospital	Rua Eiró 3400	962848132	João

Figura 5.17 Gestão dos pedidos efectuados.

Para que o administrador tenha acesso a esta informação, basta selecionar um pedido na tabela. Ao fazê-lo, o mapa é atualizado de forma automática, mostrando então as informações dos táxis mais próximos, obedecendo ao mesmo tempo às preferências do utilizador.

Para o cálculo da distância foi efetuada uma pequena análise de qual seria o melhor método para o cálculo da distância entre o táxi e o utilizador. Foi utilizada a fórmula de *Haversine* que é largamente utilizada na navegação para calcular a distância entre dois pontos de uma esfera a partir das suas latitudes e longitudes (Veness, 2012). Quando aplicada à terra, a fórmula de *Haversine* representa apenas uma aproximação, já que o nosso planeta não é uma esfera perfeita, uma vez que é achatado nos polos. O raio da Terra é variável, ou seja, nos polos é na ordem dos 6357km enquanto que no equador é na ordem dos 6378km. Para o cálculo desta fórmula é utilizado um valor de raio médio, geralmente aceite, de 6371km.

A fórmula de *Haversine* é dada por dois pontos de uma esfera, de raio *R*, com latitudes $\varnothing_1$ e $\varnothing_2$, diferença de latitudes $\Delta_\varnothing = \varnothing_1 - \varnothing_2$ e de longitudes $\Delta\lambda = \lambda_1 - \lambda_2$, onde os ângulos são em radianos, a distância *d* entre dois pontos é relacionada nas suas localizações pela fórmula:

$$haversine\left(\frac{d}{R}\right) = haversine\,(\Delta\varnothing) + \cos(\varnothing 1)\cos(\varnothing 2)\,haversine\,(\Delta\lambda) \quad (1)$$

A distância *d* pode ser obtida pela aplicação da *haversine* inversa

$$d = R\,haversin^{-1}(h) = 2\,R\arcsin\left(\sqrt{h}\right) \quad (2)$$

onde *h* denota *haversine* (d/R), isto é (Rubin, 2006),

$$d = 2\,R\arcsin\left(\sqrt{sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat1) * \cos(lat2) * sin^2\left(\frac{\Delta long}{2}\right)}\right) \quad (3)$$

É então aplicada a fórmula *Haversine* à consulta que será efetuada para encontrar, num raio de 10 km, os cinco táxis que estejam mais próximos do utilizador, como mostra o código do Anexo O – Consulta *Haversine*.

Após serem encontrados os cinco táxis que se encontrem mais próximos, é então determinado o tempo que cada um deles demora até chegar junto do utilizador.

Tal como acontecia na aplicação *Android*, também aqui na plataforma *Web* é extraído o tempo que o táxi demora a chegar junto do utilizador recorrendo à *Google Directions API*, isto é, através da leitura do documento *Json*, onde são passadas quer as latitudes e longitudes do táxi quer do utilizador ao endereço *Web* do *Google Directions API*. Posteriormente é feita a leitura integral do documento e de onde é extraído o tempo, que se encontra na *tag* "*duration*" do mesmo documento. Este valor é armazenado num vetor para posteriormente ser comparado com os restantes valores obtidos dos outros táxis. De seguida é feita uma comparação por todos os valores armazenados no *array* a fim de determinar o táxi com o menor tempo, sendo esse apresentado como melhor opção para realizar o serviço, como é mostrado no Anexo P.

Determinados os táxis mais próximos do utilizador e sabendo o tempo que cada um demora a chegar até junto do mesmo, são então desenhadas no mapa as marcas que dizem respeito quer ao utilizador quer aos táxis, destacando-se aquele que tem o menor tempo de chegada, como é mostrado no Anexo P.

O mapa utilizado na plataforma *Web*, é proveniente de uma biblioteca, desenvolvida pela Subgurium.net, que permite, entre outras funcionalidades, fazer uso do mapa da *Google*, bem como desenhar marcas no próprio mapa.

5.5.3.1. Simulação

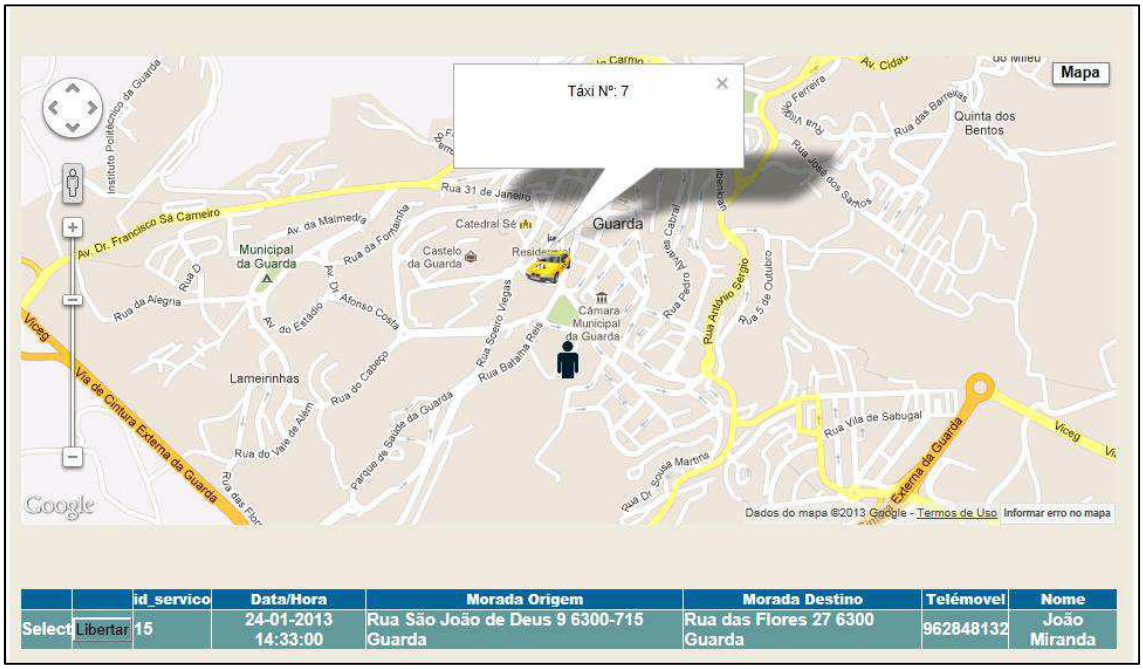
A simulação foi construída com o objetivo de simular o normal funcionamento do terceiro módulo, ou seja, o módulo do táxi, uma vez que o seu desenvolvimento ficou a cargo do CCG. Com esta simulação é-nos então permitido obter a latitude e longitude do táxi que se encontra a efetuar o serviço, como se a aplicação (terceiro módulo) estivesse a operar diretamente no táxi. Para que esta simulação se aproximasse o mais possível da realidade, foram adicionadas coordenadas de latitude e longitude, de forma manual, de um percurso previamente definido. Foi ainda implementado um *timer*, fazendo com que a cada 30 segundos a posição atual do

táxi seja atualizada na Base de Dados, podendo desta forma o utilizador acompanhar a chegada do táxi do lado da aplicação móvel.

5.5.4. Libertar Táxi

Uma vez que o módulo dos táxis não faz parte do âmbito deste projeto, foi elaborado um módulo que permite ao administrador libertar um táxi, finalizando assim o serviço. Permite ainda visualizar os táxis que se encontram a efetuar os serviços.

Para que o administrador tenha acesso a esta informação, tem apenas de selecionar um serviço na tabela, e de imediato as marcas, quer do táxi quer do utilizador, são mostradas no mapa, como mostra a figura 5.18.



		id_servico	Data/Hora	Morada Origem	Morada Destino	Telémovel	Nome
Select	Libertar	15	24-01-2013 14:33:00	Rua São João de Deus 9 6300-715 Guarda	Rua das Flores 27 6300 Guarda	962848132	João Miranda

Figura 5.18 Possibilidade do administrador libertar um táxi.

O administrador ao libertar um táxi assume que este já terminou de efetuar o serviço, fazendo com que fique disponível para efetuar outros serviços.

O modo como é feito o processamento para libertar um táxi é feita da seguinte forma:

1. Atualiza o campo estado_servico da tabela Servico para 2 (executado).
2. Atualiza o campo estado da tabela Taximotorista para 1 (disponível).
3. Atualiza a GridView com os pedidos.
4. Limpa o mapa até o administrador voltar a selecionar novamente um serviço na tabela.

5.5.5. Ver Histórico

Esta página é responsável por mostrar um resumo dos pedidos efetuados, sendo possível visualizar num mapa, os locais com um maior número de pedidos, bem como os locais para onde as pessoas mais se deslocam. Estes dados podem ajudar os decisores urbanos a tomarem decisões importantes, tais como, se num determinado local existe um elevado número de pedidos, pode justificar-se a instalação de uma praça de táxis próximo do local.

Embora o âmbito deste trabalho não seja o tratamento dos dados recolhidos para fins estatísticos, na figura 5.19 é mostrada uma possível ilustração daquilo que é possível fazer com os dados recolhidos da Base de Dados.

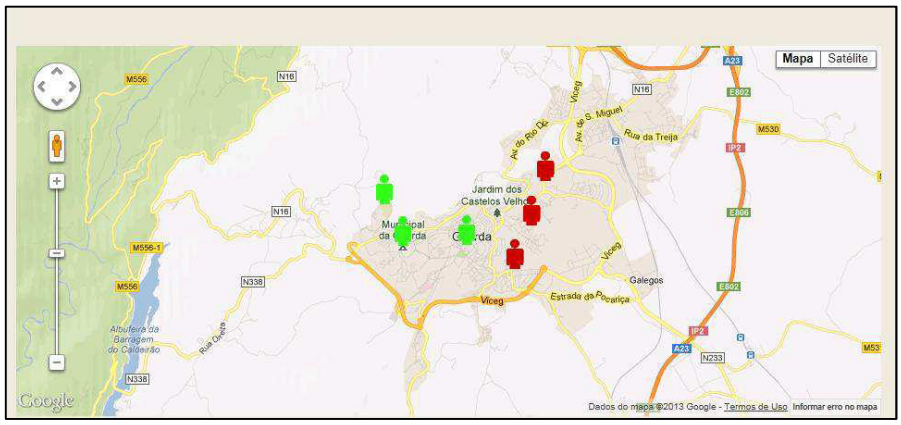


Figura 5.19 – Histórico de pedidos.

5.6. Avaliação de Desempenho

O primeiro teste da aplicação Meu Táxi, realizado no exterior, consistiu na avaliação de desempenho do sistema GPS, tendo demorado cerca de 15 segundos a obter a localização do cidadão. Já para testes efetuados no interior de edifícios, a aplicação mostrou ter mais dificuldade em obter a localização do cidadão, demorando mais tempo a conseguir encontrá-la, e em casos extremos, com ausência de sinal GPS não conseguiu mesmo encontrar a localização do cidadão.

O segundo teste realizado foi verificar se a aplicação, após ter sido selecionado e atribuído um táxi na plataforma, recebe uma notificação com a confirmação de que o pedido foi aceite e que já se encontra um táxi atribuído ao serviço. Este teste foi realizado com sucesso, uma vez que foi recebida a confirmação na aplicação móvel de que um táxi já se encontra a caminho a fim de efetuar o serviço solicitado.

O terceiro teste à aplicação Meu Táxi foi confirmar se o cidadão acompanhava a movimentação do táxi diretamente no mapa da aplicação bem como a informação relativa ao tempo estimado que demora a chegar junto do cidadão. O teste foi concluído com sucesso, tendo-se verificado que a atualização da informação, quer do tempo estimado quer da aproximação do táxi, é efetuada de 30 em 30 segundos, como era de esperar.

6. CONCLUSÕES E TRABALHO FUTURO

6.1. Conclusões

Este trabalho tinha como objetivo o desenvolvimento de uma solução para dispositivos móveis, com o intuito de dar a possibilidade a um cidadão de poder requisitar um serviço de táxi, de forma simples e intuitiva, e seguir todo o processo de prestação do serviço (desde o envio do táxi até à finalização do serviço), contribuindo desta forma para a melhoria dos estilos de vida do cidadão. Era ainda objetivo deste trabalho construir uma plataforma que fizesse a gestão dos pedidos efetuados pelos cidadãos e que ao mesmo tempo fornecesse mecanismos de seleção de um táxi que se encontrasse nas melhores condições de realizar o serviço, tendo em conta a distância e o tempo. A plataforma permite, através de um pequeno histórico, ter uma ilustração possível do que pode ser feito com os dados recolhidos, uma vez que a estrutura de dados está pensada nesse sentido, ou seja, o foco não é a visualização, mas sim como a informação está estruturada e o potencial aplicacional que no futuro poderá ser explorado.

A necessidade que tínhamos do terceiro módulo, em que o seu desenvolvimento ficou a cargo do Centro de Computação Gráfica, não nos permitiu obter a localização dos táxis, o que se tornou uma dificuldade, uma vez que não conseguíamos fazer com que aplicação funcionasse da maneira mais realista possível. Para ultrapassar esta necessidade, foram adicionados os dados relativos aos táxis de forma manual, elaborando uma pequena simulação a fim de ilustrar o funcionamento normal da aplicação.

Uma das vantagens oferecidas pelos dois módulos desenvolvidos é que um cidadão pode requisitar um serviço de táxi, da forma mais cómoda possível, contribuindo ao mesmo tempo para uma recolha de informação que permitirá aos

decisores urbanos tomar decisões mais eficientes no planeamento de mobilidade de uma cidade.

O desenvolvimento da aplicação móvel permitiu-nos um maior contacto com o Sistema Operativo *Android*, sendo revelador de uma eficiência notável, mostrando-nos desta forma a crescente adoção por parte dos utilizadores na escolha deste Sistema Operativo.

Com a elaboração destes dois módulos temos consciência de que foram cumpridos os objetivos a que nos propusemos.

6.2. Trabalho Futuro

Como qualquer aplicação, esta é passível de melhorias em questões de trabalhos futuros. Assim, o sucesso da aplicação como um todo dependerá da elaboração do terceiro módulo, que permite determinar a posição do táxi, estando o Centro de Computação Gráfica responsável pelo seu desenvolvimento e respetiva divulgação junto da Antral, a fim de recolhermos o máximo de dados de mobilidade, para posterior análise de comportamentos e comparação com a restante oferta de outros meios de transporte públicos.

A próxima etapa será a implementação da seleção automática do táxi a realizar o serviço, uma vez que com a obtenção da posição do táxi em tempo real, é possível determinar o funcionamento da aplicação de forma mais realista.

Uma outra funcionalidade a implementar é a obtenção da localização do cidadão com recurso à triangulação das antenas dos operadores móveis, para dispositivos que não possuam antenas de GPS.

No futuro deverá também ser implementado um método de recuperação da *password* por parte de um cidadão que se tenha esquecido da mesma.

O registo do utilizador poderá também ser melhorado, de maneira a que o cidadão ao efetuar o seu registo identifique as letras de uma imagem distorcida, mais conhecido como *CAPTCHA*, e posteriormente receba um e-mail com um *link* de confirmação de registo de conta para que a mesma fique ativa.

Seria de valor a disponibilização desta aplicação ou sua variante ser considerada no TICE.Mobilidade, como forma de divulgação.

REFERÊNCIAS BIBLIOGRÁFICAS

Android-App-Market (2012); "Android Architecture – The Key Concepts of Android OS" Acedido em janeiro de 2013, em: http://www.android-app-market.com/android-architecture.html

Android Developers (2013); "Android Architecture". Acedido em janeiro de 2013, em: http://developer.android.com/about/versions/index.html

Android Developers (2013a); "The Google Directions API". Acedido em janeiro de 2013, em: https://developers.google. com/maps/documentation/directions

Android Developers (2012). Acedido em maio de 2012, em: http://developer.android.com/reference/android/location/package-summary.html

Android Developers (2012a). Acedido em maio de 2012, em: https://developers.google.com/maps/documentation/android

Android Developers (2012b); "Activity lifecycle". Acedido em dezembro de 2012, em: http://developer.android.com/reference/android/app/Activity.html

Android On Board (2012); "OnTouchEvent! Interceptando eventos TouchScreen na sua Activity". Acedido em novembro de 2012, em: http://androiddevbr.wordpress.com/2012/08/30/ontouchevent-interceptando-eventos-touchscreen-na-sua-activity

Android OS (2012); "Philosophy and goals.". Acedido em dezembro de 2012, em: http://source.android.com/about/philosophy.html

Carris (2013); "Carris e IzziMoove lançam Izi Carris". Acedido em abril de 2013, em: http://www.carris.pt/pt/noticias/2012/carris-e-a-izimoove-lancam-izi-carris

Cediant (2012); Acedido em abril de 2012, em: http://www.cediant.es/index.php/joomla/mobility/etaxi

Dilão, Rui (2013); "GPS (Global Positioning System)". Acedido em janeiro de 2013, em: http://www.cienciaviva.pt/latlong/anterior/gps.asp

ETaxi (2012); Acedido em novembro de 2012, em http://www.etaxi.it/radiotaxis.htm

Ferreira, José Manuel Rodrigues (2011); "Mobilidade e Desenvolvimento Local: o Caso do Concelho da Lousã"; Universidade de Aveiro

Furtado, Gonçalo; Oliveira, Miguel (2004); "Reflexão sobre os Novos Modelos Comunicacionais de Mobilidade Urbana"

Gama; Melissa; Tralhão, Lino; Rodrigues, João Coutinho (2011); "Um Estudo de Acessibilidade a Zonas Verdes em Meio Urbano com Tecnologia SIG e Redes Multimodais": Aplicação a Coimbra.

Gefos (2012); Acedido em novembro de 2012, em: http://www.gefos.net/index.php?id=163

Get-a-Taxi (2012); Acedido em novembro de 2012, em http://www.get-a-taxi.at

Goasduff, Laurence (2012); "Gartner Says Worldwide Sales of Mobile Phones Declined 2 Percent in First Quarter of 2012; Previous Year-over-Year Decline Occurred in Second Quarter of 2009". Acedido em dezembro de 2012, em: http://www.gartner.com/it/page.jsp ?id=2017015

Goldschmidt, Julio (2012); "SQL Definições". Acedido em janeiro de 2013, em: http://www.dotclass.com.br/portal/?p= 386

Google Play (2013); "Transportes Urbanos de Coimbra". Acedido em abril de 2013, em: https://play.google.com/store/apps/details?id=com.enso.smtuc&feature= search_result

Google Play (2013a); "Metro Lisboa". Acedido em abril de 2013, em: https://play.google.com/store/apps/details?id=com.seara.metrolisboa&feature=search_result;

Gootaxi (2012); Acedido em novembro de 2012, em: http://www.gootaxi.com

Haddad, Renato (2012); "Net Framework 2.0 Introduction Visual C# 2005". Acedido em dezembro de 2012, em: http://www.doc88.com/p-280601839001.html

Hailo (2013); Acedido em abril de 2013, em: https://hailocab.com/london

Halden, Derek; Peter, Jones and Wixey, Sara (2005); "Accessibility Analysis Literature Review"; University of Westminster.

Handy, Susan (2004); "Driving by Choice or Necessity?".

Leal, Nelson Glauber de Vasconcelos (2012); "Google Maps: Traçando Rotas". Acedido em novembro de 2012, em: http://nglauber.blogspot.pt/2011/10/google-maps-tracando-rotas.html

Lunden, Ingrid (2012), "Google Play About To Pass 15 Billion App Downloads? Pssht! It Did That Weeks Ago". Acedido em dezembro de 2012, em: http://techcrunch.com/2012/05/07/google-play-about-to-pass-15-billion-downloads-pssht-it-did-that-weeks-ago

Macoratti, José Carlos (2013); "SQL SERVER - Usando a linguagem de definição de dados (DDL)". Acedido em janeiro de 2013, em: http://www.macoratti.net/sql_ddl.htm

Metrodoporto (2012); "Todo o Metro em todo o lado". Acedido em abril de 2013, em: http://www.metrodoporto.pt/PageGen.aspx?WMCM_PaginaId=16779&noticiaId=24322&pastaNoticiasReqId=15503

MooveTaxy (2012); Acedido em abril de 2012, em: http://jurema.la/trabalhos/moove-taxi

myTaxi (2012); Acedido em novembro de 2012, em: http://www.mytaxi.net

Nery, Isabel e Fillol, Joana (2013); "Transportes: Chamam-lhes Públicos". Acedido em março de 2013, em: http://visao. sapo.pt/transportes-chamam-lhes-publicos=f706936

Opt (2013); "Move-me, os transportes todos ligados". Acedido em abril de 2013, em: http://www.opt.pt/produto.asp?codProduto=14

Paiva, J. V., J. Aguiar, et al (2006); "Guia Técnico de Reabilitação Habitacional; Lisboa"; LNEC.

Pamplona, Vitor Fernando (2010); "Web Services. Construindo, disponibilizando e acessando Web Services via J2SE e J2ME". Acedido em janeiro de 2013, em: http://javafree.uol.com.br/artigo/871485/Web-Services-Construindo-disponibilizando-e-acessando-Web-Services-via-J2SE-e-J2ME.html

Pedro, Henrique (2011); "Panorama dos Táxis em Lisboa". Acedido em maio de 2013, em: http://start-upportugal.blogspot.pt/2011/12/quantos-taxis-existem-em-lisboa.html

Rubin, Alexander (2006); "Geo/Spatial Search with MySQL". Acedido em novembro de 2012, em: http://pt.scribd.com/doc /2569355/Geo-Distance-Search-with-MySQL

Rubin, Andy (2012); "Android@Mobile World Congress: It's all about the ecosystem". Acedido em janeiro de 2013, em: http://googlemobile.blogspot.pt/2012/02/androidmobile-world-congress-its-all.html

Santos, P. A. G. (1994); "A mobilidade urbana em Lisboa e Porto. Interpretação das principais cadeias de viagens."; Dissertação apresentada à Universidade Técnica de Lisboa para obtenção do grau de mestre em Transportes.

Simmonds, David (Consultancy) (1998); "Developing land-use/transport economic efficiency appraisal"; Cambridge.

Taksee (2012); Acedido em dezembro de 2012, em: http://www.taksee.net

Taxi (2012); Acedido em dezembro de 2012, em: http://taxibutton.de/index.html

Taxi-link (2013); Acedido em abril de 2013, em: https://play.google.com/store/apps/details?id=com.geolink.taxilink&feature=search_result#?t=W251bGwsMSwyLDEsImNvbS5nZW9saW5rLnRheGlsaW5rIl0.

TaxiMagic (2012); Acedido em dezembro de 2012, em: http://taximagic.com

Tice.mobilidade (2012); "One.Stop.Transport". Acedido em março de 2013, em: http://tice.mobilidade.ipn.pt/index.php/single-article-pt

Vasconcellos, Eduardo (2005); "Mobilidade, equidade e sustentabilidade". I Curso Internacional de Transporte e Sustentabilidade. ANTP/Movimento. São Paulo.

Wannataxi (2012); Acedido em dezembro de 2012, em: http://www.wannataxi.com/user

Veness, Chris (2012); "Calculate distance, bearing and more between Latitude/Longitude points". Acedido em novembro de 2012, em: http://www.movable-type.co.uk/scripts/latlong.html

Vieira, João (2002); "Programação com ASP.NET", Volume I; Editora FCA; Lisboa.

ANEXO A

Listagem – Estrutura do Resultado JSON

```
{
  "status": "OK",
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "steps": [ {
        "travel_mode": "DRIVING",
        "start_location": {
          "lat": 41.8507300,
          "lng": -87.6512600
        },
        "end_location": {
          "lat": 41.8525800,
          "lng": -87.6514100
        },
        "polyline": {
          "points": "a~l~Fjk~uOwHJy@P"
        },
        "duration": {
          "value": 19,
          "text": "1 min"
        },
        "html_instructions": "Head \u003cb\u003enorth\u003c/b\u003e on \u003cb\u003eS Morgan
St\u003c/b\u003e toward \u003cb\u003eW Cermak Rd\u003c/b\u003e",
        "distance": {
          "value": 207,
          "text": "0.1 mi"
        }
      },
      ...
      ... additional steps of this leg
    ...
    ... additional legs of this route
      "duration": {
        "value": 74384,
        "text": "20 hours 40 mins"
      },
      "distance": {
        "value": 2137146,
        "text": "1,328 mi"
      },
      "start_location": {
        "lat": 35.4675602,
        "lng": -97.5164276
      },
      "end_location": {
        "lat": 34.0522342,
        "lng": -118.2436849
      },
      "start_address": "Oklahoma City, OK, USA",
      "end_address": "Los Angeles, CA, USA"
    } ],
    "copyrights": "Map data ©2010 Google, Sanborn",
    "overview_polyline": {
      "points":
"a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\\|tNtqf@l{Yd_Fblh@rxo@b}@xxSfytAblk@xxaBeJxlcBb~t@zbh@j
c|Bx}C`rv@rw|@rlhA~dVzeo@vrSnc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nurDznmFfwMbwz@b
bl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_Wr|Efc\\x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg@|kHnt
```

```
yArpb@bijCk_Kv~eGyqTj_|@`uV`k|DcsNdwxAott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy@`omFvaErfo@igQx
nlApqGze~AsyRzrjAb__@ftyB}pIlo_BflmA~yQftNboWzoAlzp@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdmY_hL
xiZd~XtaBndgC"
    },
    "warnings": [ ],
    "waypoint_order": [ 0, 1 ],
    "bounds": {
      "southwest": {
        "lat": 34.0523600,
        "lng": -118.2435600
      },
      "northeast": {
        "lat": 41.8781100,
        "lng": -87.6297900
      }
    }
  } ]
}
```

ANEXO B – TABELA DE PREÇOS

F.P.T. **Federação Portuguesa do Táxi**

TABELA DE PREÇOS

PARA ENTRAR EM VIGOR A PARTIR DE 1 DE JANEIRO DE 2013
DE HARMONIA COM A CONVENÇÃO CELEBRADA EM 27/12/2012

ANEXO

1. TIPOLOGIA DE TARIFAS E PRINCÍPIOS DE APLICAÇÃO

Os preços a pagar pelos serviços de transporte em táxi são determinados, consoante o tipo de tarifa, da seguinte forma:

TARIFA URBANA, – Identificada pelo algarismo 1

- **Diurna -** em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, aplicada nos dias úteis entre as 6 e as 21 horas;
- **Nocturna -** em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, aplicada nos dias úteis entre as 21 horas de um dia e as 6 horas do dia seguinte e aos sábados domingos e feriados nacionais.

Por despacho do Presidente do IMT - Instituto da Mobilidade e dos Transportes, I.P., ouvida a Direcção-Geral das Actividades Económicas e as Associações do sector, poderá ser autorizada a prática da tarifa urbana em freguesias ou grupos de freguesias (coroas) de um concelho, a pedido da respectiva Câmara Municipal. Nas freguesias ou grupos de freguesias (coroas) onde se aplica a tarifa urbana haverá mudança para a tarifa ao quilómetro quando os táxis realizarem serviços para fora da área a que estão afetos.

TARIFA AO QUILÓMETRO COM RETORNO EM VAZIO – identificada pelo algarismo 3

- **Diurna -** em função de um valor inicial (bandeirada), de fracções de distâncias percorrida incluindo o retorno em vazio e de tempos de espera, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 6 e as 21 horas;
- **Nocturna -** em função de um valor inicial (bandeirada), de fracções de distância percorrida incluindo o retorno em vazio e de tempos de espera, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 21 horas de um dia e as 6 horas do dia seguinte e aos sábados domingos e feriados nacionais.

TARIFA AO QUILÓMETRO COM RETORNO OCUPADO - identificada, pelo algarismo 5

- **Diurna -** em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, quando o cliente regresse à localidade de início do serviço, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 6 e as 21 horas;
- **Nocturna -** em função de um valor inicial (bandeirada), de fracções de distância percorrida e de tempos de espera, quando o cliente regresse à localidade de início do serviço, aplicada onde não esteja autorizada a tarifa urbana, nos dias úteis entre as 21 horas de um dia e as 6 horas do dia seguinte e aos sábados domingos e feriados nacionais.

TARIFA DO SERVIÇO À HORA – identificada com o algarismo 6

Tarifa em função da duração do serviço, que só pode ser adoptada com o acordo do cliente, podendo aplicar-se, nomeadamente, em serviços por ocasião de casamentos, baptizados, funerais e outros eventos sociais e culturais.

TARIFA A CONTRATO – identificada com a letra C

Tarifa em função de acordo, reduzido a escrito, estabelecido por prazo não inferior a trinta dias, onde constem obrigatoriamente o respectivo prazo, a identificação das partes e o preço acordado.

TARIFA A PERCURSO – identificada com a letra P

Tarifa em função dos preços estabelecidos para determinados itinerários, em adenda à convenção de preços.

2. TARIFAS A APLICAR

TARIFA URBANA

N.º lugares / Tarifas	Bandeirada metros		Km	hora	Fracções metros		seg.	
4 passageiros								
Tarifa 1								
- diurna	1800	3,25	**0,47**	**14,80**	212,77	0,10	24,0	0,10
- noturna	1440	3,90	**0,56**	**14,80**	178,57	0,10	24,0	0,10
+ 4 passageiros								
Tarifa 1								
- diurna	1800	3,25	**0,61**	**14,80**	163,93	0,10	24,0	0,10
- noturna	1440	3,90	**0,73**	**14,80**	136,99	0,10	24,0	0,10
VEÍCULOS S/ DISTINTIVO								
4 passageiros	1440	3,90	**0,56**	**14,80**	178,57	0,10	24,0	0,10
+ 4 passageiros	1440	3,90	**0,67**	**14,80**	149,25	0,10	24,0	0,10

TARIFA AO QUILÓMETRO

N.º lugares / Tarifas	Bandeirada metros		Km	hora	Fracções Metros		seg.	
4 passageiros								
Tarifa 3 retorno em vazio								
- diurna	1800	3,25	**0,94**	**14,80**	106,38	0,10	24,0	0,10
- noturna	1800	3,90	**1,13**	**14,80**	88,50	0,10	24,0	0,10
Tarifa 5 retorno ocupado								
- diurna	3600	3,25	**0,47**	**14,80**	212,77	0,10	24,0	0,10
- noturna	3600	3,90	**0,56**	**14,80**	178,57	0,10	24,0	0,10
+ 4 passageiros								
Tarifa 3 retorno em vazio								
- diurna	1400	3,25	**1,21**	**14,80**	82,65	0,10	24,0	0,10
- noturna	1400	3,90	**1,45**	**14,80**	68,97	0,10	24,0	0,10
Tarifa 5 retorno ocupado								
- diurna	2800	3,25	**0,61**	**14,80**	163,93	0,10	24,0	0,10
- noturna	2800	3,90	**0,73**	**14,80**	136,99	0,10	24,0	0,10
VEÍCULOS S/ DISTINTIVO								
4 passageiros Tarifa c/ retorno em vazio	1440	3,90	**1,14**	**14,80**	87,72	0,10	24,0	0,10
4 passageiros Tarifa c/ retorno em ocupado	2880	3,90	**0,57**	**14,80**	175,43	0,10	24,0	0,10
+ 4 passageiros Tarifa c/ retorno em vazio	1400	3,90	**1,30**	**14,80**	76,92	0,10	24,0	0,10
+ 4 passageiros Tarifa c/ retorno em ocupado	2880	3,90	**0,65**	**14,80**	153,84	0,10	24,0	0,10

TARIFA DO SERVIÇO À HORA (Tarifa 6)

TIPO DE VEÍCULO	1.ª Hora	1/2 Hora
4 passageiros	8,35	4,18
+ 4 passageiros	9,80	4,90
VEÍCULOS S/ DISTINTIVO		
4 passageiros	11,70	5,85
+ 4 passageiros	13,55	6,78

SUPLEMENTOS

TIPO	VALOR
Bagagem	1,60
Animais Domésticos	1,60
Suplemento de chamada (estacionamento livre ou condicionado)	0,80

ANEXO C

Listagem – *Web Service* que verifica o *Login* (C#)

```
using System.Data.SqlClient;
using System.Configuration;

public string Verificalogin(String login, String pass)
{
   //verifica se o login e a password não são vazios
   if (login == "" || pass == "")
   {
      return "false";
   }
   else
   {
       //faz a ligação à base de dados
       SqlConnection con = new SqlConnection("workstation
        id=testetaxi.mssql.somee.com;packet size=4096;user id=vuk_SQLLogin_1;pwd=n2558p1gwp;data
        source=testetaxi.mssql.somee.com;
        persist security info=False;initial catalog=testetaxi");

        //string da consulta à base de dados
        string sql = "select login from users where login = " + login + " and
        password = '" + pass + "'";

        SqlCommand cmd = new SqlCommand(sql, con);
       SqlDataReader dr;
        try
       {
           con.Open();
           dr = cmd.ExecuteReader();

           if (dr.Read())//se existir então devolve verdadeiro
           {
               return "true";
           }
           else
           {
               return "false";
           }
       }
       finally
       {
           con.Close();
       }
   }
}
```

ANEXO D

Listagem – Método responsável por fazer a comunicação com o *Web Service* (Java)

```
//constantes com os dados necessários para a comunicação com o Web Service
private static final String SOAP_ACTION = "http://tempuri.org/Verificalogin";
private static final String METHOD_NAME = "Verificalogin";
private static final String NAMESPACE = "http://tempuri.org/";
private static final String URL = "http://10.0.2.2:54691/verificalogin.asmx";


private String doLogin(String user, String password)
{
   //parâmetros que vão ser passados ao Web Service
   SoapObject request = new SoapObject(NAMESPACE, METHOD_NAME);
   request.addProperty("login", user);
   request.addProperty("pass", password);

   //versão do protocol SOAP
   SoapSerializationEnvelope soapenvelop =
        new SoapSerializationEnvelope(SoapEnvelope.VER11);
   soapenvelop.dotNet = true; //vai comunicar com um Web Service .Net
   soapenvelop.setOutputSoapObject(request);
   AndroidHttpTransport transport = new AndroidHttpTransport(URL);

   SoapPrimitive resultString = null;
   try{
        //envia o pedido ao Web Service
        transport.call(SOAP_ACTION, soapenvelop);
        //recebe a resposta do Web Service
        resultString = (SoapPrimitive) soapenvelop.getResponse();

   }
   catch (Exception e) {
        e.printStackTrace();
   }
        return resultString + "";
}
```

ANEXO E

Listagem – *Web Service* responsável pelo registo do utilizador (C#)

```
public string insereregisto(long login, string password, string nome, string mail)
{
      //String de ligação à base de dados
      SqlConnection con = new SqlConnection("workstation id=testetaxi.mssql.somee.com;packet
      size=4096;user id=vuk_SQLLogin_1;pwd=n2558p1gwp;data
      source=testetaxi.mssql.somee.com;persist security info=False;initial catalog=testetaxi");
      string sql = "Insert Into users (login, password, nome, mail, admin) Values (" + login +
      ", '" + password + "', '" + nome + "', '" + mail + "', 0)";
     SqlCommand cmd = new SqlCommand(sql, con);

     con.Open();

     try
     {
        //verifica se já existe algum registo com aquele Nº de telémovel
        string sql_verifica = "Select * from users where login = " + login;
       SqlCommand cmd_verifica = new SqlCommand(sql_verifica, con);
       SqlDataReader dr;

       dr = cmd_verifica.ExecuteReader();
       if (dr.Read())
       {
           return "Já Existe"; //utilizador já existe
       }
       else
       {
           dr.Close();
          cmd.ExecuteNonQuery(); //regista o utilizador

          return "true";
       }

    }
    catch (SqlException e)
    {
       return e.ToString();
    }
    finally
    {
        con.Close();
    }
 }
```

ANEXO F

Listagem – Método Localização (Java)

```
private void localizacao()
{
   //criação do overlay para mostrar a localização actual
   final MyLocationOverlay mylocationOverlay = new MyLocationOverlay(this, map);

   map.getOverlays().add(mylocationOverlay);

   //se for encontrado um provider então o runnable é executado uma vez já que existe
   //uma localização fixa
   mylocationOverlay.enableMyLocation();

   mylocationOverlay.runOnFirstFix(new Runnable() {

   public void run()
   {
        //foca a localização e movimenta o mapa para essa localização

        map.getController().animateTo(mylocationOverlay.getMyLocation());

        //faz zoom para que a localização possa ser vista com maior detalhe
       map.getController().setZoom(20);

       get_Latitude_origem =
           (float) mylocationOverlay.getMyLocation().getLatitudeE6();

       get_Longitude_origem =
          (float) mylocationOverlay.getMyLocation().getLongitudeE6();

        get_morada_origem = getAddress(get_Latitude_origem, get_Longitude_origem);

        //desenha a marca e posiciona o mapa no centro da interface
        CenterLocation(mylocationOverlay.getMyLocation());

   }

   });
}
```

ANEXO G

Listagem – Método responsável pela pesquisa (Java)

```
private GeoPoint getAddressbyLocation(String address)
{
   GeoPoint geopoint = null;
   Geocoder geocoder = new Geocoder(getApplicationContext(),Locale.getDefault());

   try{
        List<Address> listAddress = geocoder.getFromLocationName(address, 2);
        if(listAddress.size() > 0)
        {

            //devolve as coordenadas geográficas latitude e longitude
            geopoint = new GeoPoint((int) (listAddress.get(0).getLatitude() * 1E6) ,
               (int) (listAddress.get(0).getLongitude() * 1E6));

            return geopoint;
        }
   }
   catch (IOException e) {
                // TODO: handle exception
                e.printStackTrace();
   }

   return null;
}
```

ANEXO H

Listagem – Evento *onTouch* (Java)

```
public boolean onTouchEvent(MotionEvent event, MapView mapview1)
{
    int Action = event.getAction();

    if (Action == MotionEvent.ACTION_UP) //se o mapa é clicado
    {
        if(!MoveMap)
        {
           //valores x e y que indicam a coordenada do ponto do ecrã que foi clicado
           GeoPoint geoPoint = map.getProjection().fromPixels((int)event.getX(),
              (int)event.getY());

           Geocoder geocoder = new Geocoder(getBaseContext(),Locale.getDefault());

           get_Latitude_origem = (float) (geoPoint.getLatitudeE6() / 1E6);
           get_Longitude_origem = (float) (geoPoint.getLongitudeE6() / 1E6);

           try{
                //estrutura de dados da morada
                List<Address> addresses = geocoder.getFromLocation(geoPoint.getLatitudeE6() /
                    1E6, geoPoint.getLongitudeE6() / 1E6, 1);

                String add = "";
                if(addresses.size() > 0){
                   for(int i=0;i<addresses.get(0).getMaxAddressLineIndex();i++)
                   add += addresses.get(0).getAddressLine(i) + "\n";

                }

                //mostra a mensagem com os dados da morada
                Toast.makeText(getBaseContext(),add,Toast.LENGTH_SHORT).show();

                //remove a última marca
                map.getOverlays().clear();

                //desenha a marca
                CenterLocation(geoPoint);

           }
           catch (IOException e) {
                e.printStackTrace();
                               }
           }
        }
        else if (Action == MotionEvent.ACTION_DOWN){ //ação pressionar o mapa
           MoveMap = false;
        }
        else if (Action == MotionEvent.ACTION_MOVE){ //ação mover o mapa
           MoveMap = true;
        }
        return super.onTouchEvent(event, mapview1);
}
```

ANEXO I

Listagem – Método responsável por ler um documento *JSON* (Java)

```
private String doUrl(){

   final String result = convertStreamToString(this.getInputStream());
   String distancia = null;

   try {

        final JSONObject json = new JSONObject(result);
        final JSONObject jsonRoute = json.getJSONArray("routes").getJSONObject(0);
        final JSONObject leg = jsonRoute.getJSONArray("legs").getJSONObject(0);

        final int legs = leg.length();

        for (int i = 0; i < legs; i++)
        {
           //procura pela tag “distance” e “text” e retorna o valor dessa mesma tag
           //“text”

           final JSONObject distance = leg.getJSONObject("distance");
           distancia = distance.getString("text");
        }
   }
   catch (JSONException e)
   {
        Log.e(e.getMessage(),"");
   }
   return distancia;
}

protected InputStream getInputStream()
{
   try {
        //é passado o Url com as latitudes e longitudes de origem e destino

        URL urlRota = new URL(
                "http://maps.googleapis.com/maps/api/"+
                "directions/json?origin=" + get_Latitude_origem + "," + get_Longitude_origem +
                "&"+
                "destination=" + get_Latitude_destino + "," + get_Longitude_destino + "&" +
                "sensor=true&mode=driving");

        return urlRota.openConnection().getInputStream();
   }
   catch (IOException e) {

        return null;
   }
}


private static String convertStreamToString(final InputStream input)
{
   //lê o documento Json do Google Directions API

   final BufferedReader reader = new BufferedReader(new InputStreamReader(input));
   final StringBuilder sBuf = new StringBuilder();
```

```
   String line = null;
   try {
        while ((line = reader.readLine()) != null)
        {
           //armazena o resultado na stringbuilder
           sBuf.append(line);
        }
   } catch (IOException e) {
        Log.e(e.getMessage(), "");
   } finally {
        try {
           input.close();
        } catch (IOException e) {
           Log.e(e.getMessage(), "");
        }
   }

   return sBuf.toString();
}
```

ANEXO J

Listagem – *Web Service* responsável por guardar o pedido do utilizador (C#)

```
public String Insereservico(string data_hora, string lat_origem, string long_origem, string
lat_destino, string long_destino, string morada_origem, string morada_destino, int carrinha, int
arcondicionado, int transporteanimais, int multibanco, int ingles, int frances, int alemao, int
espanhol, int iduser)
{
   //string de ligação á base de dados
   SqlConnection con = new SqlConnection("workstation id=testetaxi.mssql.somee.com;
     packet size=4096;user id=vuk_SQLLogin_1;pwd=n2558p1gwp;
     data source=testetaxi.mssql.somee.com;persist security info=False;
     initial catalog=testetaxi");

   //string sql para inserir os dados na base de dados
   string sql = "Insert Into servico (time_stamp,lat_origem,long_origem,lat_destino,
     long_destino,morada_origem,morada_destino,estado_servico,carrinha,
     arcondicionado,transporteanimais,multibanco,ingles,frances,
     alemao,espanhol,id_user) Values ('" + data_hora + "', " + lat_origem + ",
     " + long_origem + ", " + lat_destino + ", " + long_destino + ",
     '" + morada_origem + "' ,'" + morada_destino + "',0," + carrinha + "," +
     arcondicionado + ", " + transporteanimais + ", " + multibanco + ",
     " + ingles + ", " + frances + ", " + alemao + ", " + espanhol + ","+iduser+")";

   SqlCommand cmd = new SqlCommand(sql, con);

   con.Open();

   try
   {
        cmd.ExecuteNonQuery();

       return "true";

   }
   catch(SqlException e)
   {
      return e.ToString();
   }
   finally
   {
      con.Close();
   }
}
```

ANEXO K

Listagem – *Web Service* responsável por devolver os dados do táxi (C#)

```
public string acompanha_motorista(int iduser)
{
   //string de ligação à base de dados
   SqlConnection con = new SqlConnection("workstation id=testetaxi.mssql.somee.com;
     packet size=4096;user id=vuk_SQLLogin_1;pwd=n2558p1gwp;
     data source=testetaxi.mssql.somee.com;persist security info=False;
     initial catalog=testetaxi");

   //string sql que pesquisa pelos dados do táxi
   string sql = "select top 1 taximotorista.id_taxi, latitude,longitude
     from taximotorista, servico
     where servico.id_taximotorista = taximotorista.id_taximotorista and
     id_user = " + iduser + " order by id_servico desc";

   SqlCommand cmd = new SqlCommand(sql, con);
   SqlDataReader dr;

   try
   {
      con.Open();
      dr = cmd.ExecuteReader();

      if (dr.Read())
      {
           //devolde os dados do taxi para aquele serviço
           return dr["id_taxi"].ToString() + "&" + dr["latitude"].ToString() + "&" +
              dr["longitude"];
          dr.Close();
      }
      else
      {
         return "false";
      }

   }
   finally
   {
      con.Close();
   }
}
```

ANEXO L

Listagem – *Web Service* responsável por anular um pedido (C#)

```
public string anular_servico(int id_servico)
{
   string controlo;

   //string de ligação à base de dados
   SqlConnection con = new SqlConnection("workstation id=testetaxi.mssql.somee.com;
     packet size=4096;user id=vuk_SQLLogin_1;pwd=n2558p1gwp;
     data source=testetaxi.mssql.somee.com;persist security info=False;
     initial catalog=testetaxi");

   //string sql que altera o estado do service para 3 - anulado
   string sql = "Update Servico set estado_servico = 3
     where id_servico = " + id_servico;

   SqlCommand cmd = new SqlCommand(sql, con);

   con.Open();
   try
   {
      cmd.ExecuteNonQuery();
      controlo = "true";
   }
   catch
   {
      controlo = "false";
   }
   finally
   {
      con.Close();
   }

   return controlo;
}
```

ANEXO M

Listagem – Desenha Rota (Java)

```
public Rota parse()
{
    // Cria uma rota vazia
    final Rota route = new Rota();
    try {
      // Obtém a String do JSON
      final String result =
        convertStreamToString(feedUrl.openConnection().getInputStream());

      // Transforma a string em JSON
      JSONObject json = new JSONObject(result);
      // Pega a primeira rota retornada
      JSONObject jsonRoute = json.getJSONArray("routes").getJSONObject(0);
      JSONObject leg = jsonRoute.getJSONArray("legs").getJSONObject(0);

      // Obtém os passos do caminho
      JSONArray steps = leg.getJSONArray("steps");

      final int numSteps = steps.length();
      /*
       * Itera através dos passos, decodificando
       * a polyline e adicionando à rota.
       */
      JSONObject step;

      for (int i = 0; i < numSteps; i++) {
        // Obtém o passo corrente
        step = steps.getJSONObject(i);
        // Decodifica a polyline e adiciona à rota
        route.addPoints(decodePolyLine(step.getJSONObject("polyline")
              .getString("points")));
      }
    } catch (Exception e) {
    }
    return route;
  }

private String convertStreamToString(final InputStream input)
{
    final BufferedReader reader =
      new BufferedReader(
        new InputStreamReader(input));

    final StringBuilder sBuf = new StringBuilder();

    String line = null;
    try {
      while ((line = reader.readLine()) != null) {
        sBuf.append(line);
      }

    } catch (IOException e) {
    } finally {

    try {
        input.close();
    } catch (IOException e) {
```

```
    }
    }
    return sBuf.toString();
}

private List<GeoPoint> decodePolyLine(final String poly)
{
    int len = poly.length();
    int index = 0;
    List<GeoPoint> decoded = new ArrayList<GeoPoint>();

    int lat = 0;
    int lng = 0;

    while (index < len) {
      int b;
      int shift = 0;
      int result = 0;
      do {
        b = poly.charAt(index++) - 63;
        result |= (b & 0x1f) << shift;
        shift += 5;
      } while (b >= 0x20);
      int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
      lat += dlat;

      shift = 0;
      result = 0;
      do {
        b = poly.charAt(index++) - 63;
        result |= (b & 0x1f) << shift;
        shift += 5;
      } while (b >= 0x20);

      int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
      lng += dlng;

      decoded.add(
                new GeoPoint((int)(lat * 1E6 / 1E5), (int)(lng * 1E6 / 1E5)));
    }
    return decoded;
}
```

ANEXO N

Listagem – Método Carrega Grid (C#)

```
public void carrega_grid()
{
   DataTable dt = new DataTable();

   //string de ligação à base de dados
   SqlConnection conn =
        new SqlConnection(
        ConfigurationManager.ConnectionStrings["connection"].ConnectionString);

   //pesquisa apenas os pedidos que foram realizados
   String sql = "select id_servico, time_stamp, time_stamp_final, morada_origem,
     morada_destino from servico, users where estado_servico = 2
     and servico.id_user = users.id_user and login = " + Session["user"] + "
     order by time_stamp desc";

   SqlCommand command = new SqlCommand(sql, conn);
   DataSet ds = new DataSet();
   SqlDataAdapter da = new SqlDataAdapter(sql, conn);
   conn.Open();

   try
   {
      //o dataAdapter preenche o DataSet
      da.Fill(ds);

      //carrega a Grid
      GridView1.DataSource = ds;
      GridView1.DataBind();

      GridView1.HeaderRow.Cells[1].Text = "Pedido";
      GridView1.HeaderRow.Cells[2].Text = "Realizado";
      GridView1.HeaderRow.Cells[3].Text = "Morada Origem";
      GridView1.HeaderRow.Cells[4].Text = "Morada Destino";

   }
   finally
   {
      conn.Close();
   }
}
```

ANEXO O

Listagem – Consulta *Haversine* (C#)

```
String sql = "SELECT top 5 * ";
sql += "FROM ";
sql += "(";
sql += "select taximotorista.id_taximotorista,taximotorista.id_taxi, latitude, longitude, (6371
* 2 * ASIN(SQRT( POWER(SIN((" + lat_cliente + " - latitude) * pi()/180 / 2), 2) + COS(" +
lat_cliente + " * pi()/180) * COS(latitude * pi()/180) * POWER(SIN((" + long_cliente + " -
longitude) * pi()/180 / 2), 2) ))) as distancia ";
sql += "from taximotorista, servico, taxis, motorista ";
sql += "where estado = 1 and estado_servico = 0 and taxis.id_taxi=taximotorista.id_taxi and
id_servico = " + id_servico + " and taximotorista.id_motorista = motorista.id_motorista " +
continuacao + " ";
sql += ") devolvedistancia ";
sql += "where distancia < 10 ";
sql += "order by distancia"
```

ANEXO P

Listagem – Método que devolve o tempo do táxi e método que desenha as marcas do táxi no mapa (C#)

```
//método que retorna o tempo que o táxi demora a chegar
private string get_tempo(string lat_taxi, string longi_taxi, string lat_cliente, string
longi_cliente)
{
        string url = "http://maps.googleapis.com/maps/api/directions/json?origin=" + lat_taxi +
        "," + longi_taxi + "&destination=" + lat_cliente + "," + longi_cliente +
        "&sensor=false&mode=driving";

        WebResponse response = null;
       string tempo = "";

       try
       {
           HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
          request.Method = "GET";
          response = request.GetResponse();
          if (response != null)
          {
             string str = null;
             using (Stream stream = response.GetResponseStream())
             {
                using (StreamReader streamReader = new StreamReader(stream))
                {
                    str = streamReader.ReadToEnd();
                }
             }

             JObject o = JObject.Parse(str);

             tempo = o.SelectToken("routes[0].legs[0].duration.value").ToString();

          }
       }
       catch (JsonReaderException m)
       {
             //throw new Exception("JSON response failed.");
             texto.Text = "" + m;
       }
       finally
       {
          if (response != null)
          {
             response.Close();
             response = null;
          }
       }
       return tempo;
}

//método que determina o táxi com o menor tempo
private int procura_menor()
{
   int menor;
   int controlo;
```

```
   menor = Convert.ToInt32(tempo[0]);
   controlo = -1;

   for (int i = 0; i < tempo.Length; i++)
   {
      if (tempo[i] != null)
      {
         if (menor > Convert.ToInt32(tempo[i]))
         {
            menor = Convert.ToInt32(tempo[i]);
            controlo = i;
         }
         else
         {
            if (i == 0)
            {
               controlo = 0;
            }
         }
      }
   }
   return controlo;
}

//método que desenha as marcas to táxi no mapa
private void desenha(int controlo)
{
   for (int i = 0; i < tempo.Length; i++)
   {
      if (tempo[i] != null)
      {
         if (controlo == i) //é o menor
         {
            GIcon icon = new GIcon();
            icon.image = "../images/taxi_1.png";

            GMarkerOptions mOpts = new GMarkerOptions();
            mOpts.clickable = true;
            mOpts.icon = icon;

            GMarker marca = new GMarker(
               new GLatLng(Convert.ToDouble(lat_[i].ToString().Replace(".", ",")),
               Convert.ToDouble(longi_[i].ToString().Replace(".", ","))), mOpts);

           GInfoWindow window = new GInfoWindow(marca, "<p>Táxi Nº: "+ idtaxi[i] +"</p>
               <p>Tempo estimado: "+ converteminutos(tempo[i]) +"</p>", true);

             GMap1.Add(window);
         }
         else
         {
            GIcon icon1 = new GIcon();
            icon1.image = "../images/taxi_.png";

            GMarkerOptions mOpts1 = new GMarkerOptions();
            mOpts1.clickable = true;
            mOpts1.icon = icon1;

           GMarker marca1 = new GMarker(
              new   GLatLng(Convert.ToDouble(lat_[i].ToString().Replace(".", ",")),
              Convert.ToDouble(longi_[i].ToString().Replace(".", ","))), mOpts1);

           GInfoWindow window = new GInfoWindow(marca1, "<p>Táxi Nº: " + idtaxi[i] +
              "</p><p>Tempo estimado: " + converteminutos(tempo[i]) + "</p>", false);

              GMap1.Add(window);
         }
      }
   }
}
```