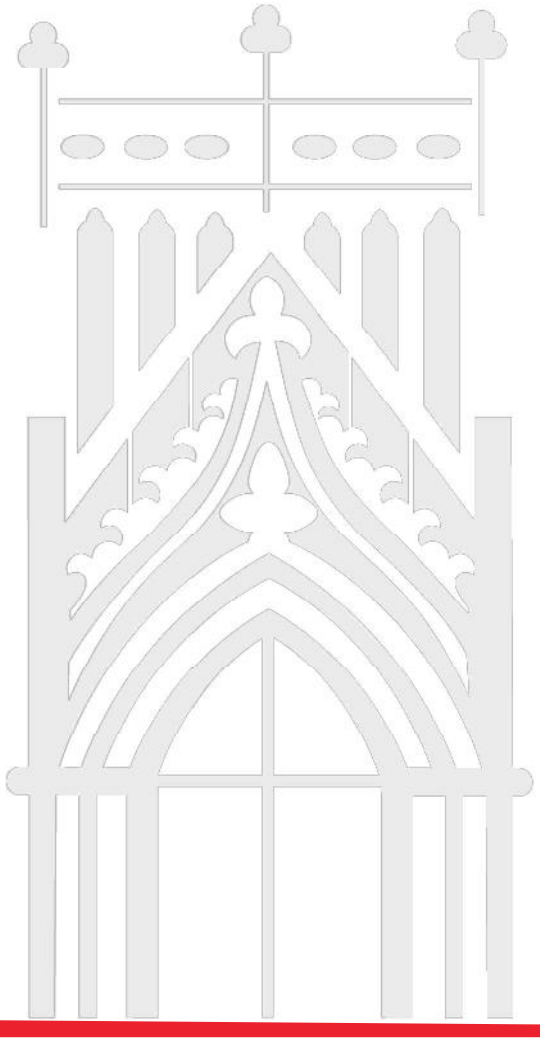
IPG Politécnico da Guarda
Polytechnic of Guarda

Mestrado em Computação Móvel

RouteMe - Uma Aplicação para Dispositivos Móveis Android para Resolução de Problemas de Rotas de Veículos



Ricardo Miguel Dias Antunes

junho | 2015

Escola Superior
de Tecnologia
e Gestão

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

RouteMe

Uma Aplicação para Dispositivos Móveis Android para Resolução de Problemas de Rotas de Veículos

Relatório de Projeto Aplicado submetido como requisito parcial para obtenção do grau de Mestre em Computação Móvel

Orientador: Professor Doutor Carlos Carreto

Ricardo Miguel Dias Antunes

Junho | 2015

"Para ser grande, sê inteiro: nada
Teu exagera ou exclui.

Sê todo em cada coisa. Põe quanto és
No mínimo que fazes.

Assim em cada lago a lua toda
Brilha, porque alta vive."

(Fernando Pessoa)

Agradecimentos

O projeto aplicado que aqui se apresenta é fruto de muito tempo de esforço e dedicação, que representa o investimento feito para a sua realização. Tudo isto só foi possível graças à colaboração e apoio dados por várias pessoas, às quais não posso deixar de prestar o meu reconhecimento.

Em primeiro lugar quero agradecer ao Professor Carlos Carreto pela possibilidade de realizar este trabalho e todo o conhecimento e supervisão que prestou ao longo da sua elaboração.

Um especial agradecimento à Professora Maria Clara Silveira, Diretora da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda por todo o apoio prestado.

Quero também deixar um agradecimento aos meus colegas de trabalho da empresa onde trabalho, Altran, por toda a motivação, experiência e apoio transmitidos.

À minha família, pela preocupação e pela coragem que sempre me transmitiram. A eles, que sempre incentivaram o meu sucesso, um enorme bem-haja.

Um enorme obrigado ao colega de Mestrado, António Barbas, por todo o apoio, força e tempo dedicado a ajudar na realização deste trabalho.

Quero igualmente deixar um grande agradecimento à Joana Sousa, que foi quem me ajudou na construção de todo o design da aplicação.

Finalmente, mas não menos importante, um enorme agradecimento aos meus verdadeiros amigos, aqueles que nunca me deixaram desanimar, aqueles que sempre me entenderam, que me encorajaram e me deram força quando mais precisei.

Resumo

Os Problemas de Rotas de Veículos (PRV) são estudados há dezenas de anos, existindo nos dias de hoje um grande número de algoritmos que lhes dão resposta de forma muito satisfatória. Paralelamente à evolução dos métodos de resolução desses mesmos problemas, tem-se igualmente assistido a uma enorme evolução dos dispositivos móveis que estão cada vez mais presentes no nosso quotidiano, tanto a nível pessoal como profissional.

O presente documento descreve o trabalho realizado no âmbito do Mestrado em Computação Móvel do Instituto Politécnico da Guarda, que visou aliar ambos os pontos referidos, numa aplicação para dispositivos móveis *Android* de modo a permitir a qualquer utilizador proceder à resolução de PRV através desse tipo de dispositivos.

A aplicação desenvolvida tira partido das características e serviços da plataforma *Android*, em especial o serviço *Google Maps*, para aquisição de dados georreferenciados usados na definição dos PRV, assim como as características de interatividade com o utilizador que permitiram o desenvolvimento de ferramentas interativas para o utilizador poder construir e resolver PRV com base no conhecimento existente em cada uma das situações.

Para além da aplicação móvel, foi também desenvolvida uma plataforma web que serve de intermediária entre a aplicação e o serviço *Google Maps*, que ficou responsável por processar todos os pedidos entre estes.

Acredita-se que a aplicação desenvolvida permite ao utilizador uma maior mobilidade, possibilitando a que em qualquer local possa proceder à construção de um novo PRV, bastando para isso a existência de uma ligação à internet. Acredita-se igualmente nas potencialidades da mesma para a resolução desses problemas da forma o mais otimizada possível, bem como no potencial crescimento e adaptabilidade para casos mais específicos nesta vasta área.

Palavras-chave: Problemas de Rotas de Veículos, Android, Google Maps, GRASP, Métodos Visuais Interativos.

Abstract

Vehicle Routing Problems (VRP) are known from decades, and in our days there exist several algorithms which meet these problems with the most possible optimal way. Alongside the evolution of the methods of resolution of these same problems, we have also witnessed a tremendous evolution of the mobile devices, which are more and more presents both in our personal and professional life.

The present document describes the work carried out in the Master degree in Mobile Computing at Polytechnic Institute of Guarda, which aimed essentially to combine both mentioned points in an application that allows any user to carry out the resolution of those VRP through a mobile device.

The developed application take advantage of the services and features of Android platform, especially the Google Maps service, for data collection between all customers, as well as interactivity features which allows the developmentand of interactive tools so the user can build their routes based on self-knowledge in each situation.

In addition to the mobile application, it has also developed a web platform, which serves as intermediary between the application and Google Maps, and it is responsible for processing all requests between them.

It's believed that the constructed tools allows the user increased mobility, being able to proceed to the construction of a new VRP anywhere, by simply existing an internet connection. It's also believed in the potentialities of it to solve these problems in an optimal way, as well as growth potential and adaptability to more specific cases in this huge area.

Keywords Vehicle Routing Problems, Android, Google Maps, GRAPS, Visual Interactive Methods.

Índice

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

SIGLAS

AJAX – Asynchronous JavaScript And XML

API – Application Programming Interface

ASP – Active Server Pages

CLR – Common Language Runtime

CLS – Common Language Specification

FCL – Framework Class Library

FK – Foreign Key

GPS – Global Positioning System

GRASP – Greedy Randomised Adaptative Search Procedure

IDE – Integrated Development Environment

JIT – Just-In-Time

Json – Javascript Object Notation

MD5 – Message-Digest Algorithm 5

MIC – Metaheuristics International Conference

MSIL – Microsoft Intermediate Language

OpenGEL ES – OpenGL for Embedded Systems

PK – Primary Key

SDK – Software Development Kit

SGL – Scalable Graphics Library

SHA – Secure Hash Algorithm

SMS – Short Message Service

SOAP – Simple Object Access Protocol

SQL – Structured Query Language

URL – Uniform Resource Locator

VRP – Vehicle Routing Problem

W3C – World Wide Web Consortium

WSDL – Web Service Description Language

XML – Extensible Markup Language

1. INTRODUÇÃO

1.1. Enquadramento e Motivação

A disponibilidade dos *smartphones* e *tablets* tem vindo a proporcionar um constante desenvolvimento de aplicações para este tipo de equipamentos que, ano após ano tem vindo a crescer, muito devido à evolução tecnológica dos mesmos. Este crescimento é também devido à mudança da forma como as pessoas comunicam e interagem entre si e com o mundo. Esta evolução da utilização destes dispositivos tem que ser igualmente acompanhada pela evolução das suas aplicações, uma vez que as exigências dos utilizadores são cada vez mais elevadas em todas as áreas. O aumento e melhoria das características destes dispositivos possibilitam a sua utilização para as mais diversas atividades, desde o seu uso como objetos de lazer até a ferramentas de trabalho.

Neste projeto pretendeu-se utilizar estes dispositivos no âmbito da resolução de problemas complexos de logística, tirando partido das caraterísticas e serviços que estes proporcionam. O projeto consistiu no desenvolvimento de uma aplicação para dispositivos móveis *Android*, designada por RouteMe, que permite a um utilizador comum resolver problemas de rotas de veículos que são um tipo de problemas de logística muito comuns A aplicação tira partido das características únicas de interatividade dos atuais dispositivos móveis e dos serviços que estes proporcionam, nomeadamente na área dos mapas e dados georreferenciados, permitindo que os dispositivos móveis comuns tipo *smartphone* e *tablet* possam ser usados para resolver este tipo de problemas.

O Problema de Rotas de Veículos (PRV) é um problema logístico muito conhecido e estudado que consiste no atendimento de um conjunto de "clientes" por intermédio de uma frota de veículos que partem de um ou mais pontos denominados depósitos [1]. A resolução do PRV pode estar sujeita a diversas restrições conforme o contexto do problema [2][3][4][5]. Por exemplo, é comum considerar que cada veículo da frota possui uma dada capacidade e o somatório de todas as demandas dos clientes atendidos por um dado veículo não pode ultrapassar a sua capacidade. A resolução do

problema consiste em encontrar o conjunto de rotas que satisfazem a demanda dos clientes, assim como outras restrições do problema, e minimizam o custo total da operação. Dependendo da dimensão e das restrições do problema, este torna-se impossível de resolver de forma ótima, em tempo útil, pelo que tipicamente se recorre a métodos heurísticos que embora não garantam a resolução ótima, encontram soluções muito próximas da solução ótima [6][7][8][9].

O PRV tem sido estudado com muito interesse nas últimas décadas devido à necessidade de diminuição de custos em todo o processo que engloba desde a produção de uma mercadoria até a sua distribuição e venda. É estimado que 10% a 15% do preço final dos produtos derive dos gastos com a distribuição [10]. O objetivo dos estudos em torno dos PRV é reduzir esta percentagem de custos com a distribuição possibilitando assim reduções de preço nos produtos finais [2][8].

Atendendo a esta problemática, a principal motivação para a realização deste projeto foi a possibilidade de poder criar uma aplicação móvel, ainda pouco explorada neste mercado, que pode ajudar na resolução destes frequentes problemas

Outra grande motivação para a realização deste projeto foi a possibilidade de trabalhar com diferentes tecnologias como dispositivos móveis *Android*, ferramentas Microsoft, integração da aplicação móvel com a API do *Google Maps*, entre outras.

1.2. Solução proposta

Tal como referido anteriormente, o PRV é um tipo de problema que tem vindo a ser estudado com bastante ênfase nos últimos anos. Embora o número de trabalhos, estudos e aplicações desenvolvidos neste âmbito esteja constantemente a crescer, a disponibilidade destas aplicações no mercado dos dispositivos móveis continua a ter diversos obstáculos, principalmente devido às limitações existentes tanto a nível do software como do hardware. As aplicações existentes neste mercado são neste momento pouco amigas do utilizador, tal como se poderá ver mais à frente no capítulo Estado de Arte. Para colmatar essa falha de usabilidade e a não confiança nestes pequenos equipamentos a solução proposta passa pela criação de uma aplicação que utilizará apenas os recursos do dispositivo para a resolução de PRV.

Para ajudar à resolução destes problemas, a aplicação irá recorrer a métodos visuais interativos que tiram partido dos recursos dos dispositivos, como por exemplo, os sensores e serviços de localização geográfica e a interação visual com o utilizador [11]. Este último será um ponto importante da aplicação. Todo o PRV será construído pelo utilizador (desde a introdução dos clientes, à criação e edição de rotas), tirando o maior partido do seu conhecimento no que a resolução do PRV diz respeito, através de ferramentas disponibilizadas pela aplicação para seu auxílio.

Assim sendo, no dispositivo móvel será possível fazer a gestão do PRV e geração de rotas. A solução contempla ainda o desenvolvimento de um *BackOffice*, com o qual a aplicação irá comunicar e partilhar dados através da internet, possibilitando uma posterior análise das rotas geradas.

As contribuições deste trabalho dividem-se em vários níveis que vão desde o nível aplicacional, ao nível da investigação. A nível aplicacional oferece uma nova solução, diferente do já existente, na medida em que a aplicação permite ao utilizador fazer tomadas de decisão de forma mais intuitiva, rápida e eficazmente, com uma interface mais acessível. A nível da investigação aplicada, são aqui apresentadas formas de utilização dos recursos disponibilizados pelos dispositivos móveis que ajudam na resolução de problemas da complexidade do PRV.

A solução final irá contemplar três módulos principais: a aplicação móvel, o *WebService* que receberá os pedidos da aplicação e os reencaminhará para o terceiro módulo, correspondente ao *Google Maps*. Neste último módulo não haverá qualquer tipo de desenvolvimento neste projeto pois serão aproveitadas todas as ferramentas que o Google nos fornece hoje para obter informações de rotas as direções, distâncias e tempo, informações essas retornadas novamente para a aplicação móvel. A Figura 3 mostra os módulos referidos e o fluxo de informação entre eles.

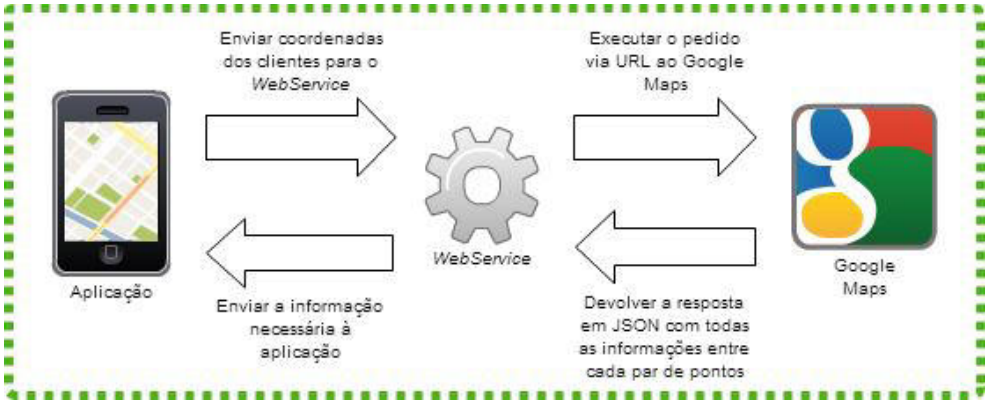


Figura 1 – Os principais módulos do projeto e o fluxo de dados entre os mesmos

O projeto tem os seguintes objetivos:

- Implementação de métodos visuais interativos para o PRV com base nos recursos dos dispositivos móveis tipo *tablet/smartphone*;
- Desenvolvimento da aplicação referida para a plataforma *Android*;
- Implementação de uma interface simples e intuitiva que permita o uso da aplicação por utilizadores comuns;
- Avaliação da eficácia da aplicação recorrendo a dados reais e data sets conhecidos.

Ao longo do processo de desenvolvimento deste projeto foi utilizada uma metodologia ágil, em que eram apresentados, periodicamente, resultados e evoluções do trabalho realizado, podendo assim uma fácil e eficaz atualização dos objetivos e requisitos finais pretendidos.

Posto isto, a metodologia utilizada para o desenvolvimento e implementação do presente projeto foi o seguinte:

- Implementação da aplicação na plataforma *Android*;
- O desenvolvimento foi realizado recorrendo ao IDE Eclipse e ao SDK da plataforma *Android*;
- A resolução do PRV foi realizada com base em métodos visuais interativos. Este tipo de métodos baseiam-se na combinação do conhecimento empírico que o utilizador possa ter do problema em questão, com o poder computacional das heurísticas, o que possibilita ao utilizador ter controlo das rotas finais geradas pelos métodos. Esta

combinação foi implementada com base nos recursos dos dispositivos móveis tipo *tablet/smartphone*.

- A eficácia da aplicação foi avaliada recorrendo a um estudo comparativo, recorrendo a dados reais.

De uma forma sucinta, as tarefas realizadas foram as seguintes:

- Análise dos requisitos do projeto, com base nos objetivos propostos;
- Estudo e desenvolvimento dos algoritmos a implementar para resolver o PRV com base em métodos visuais interativos;
- Implementação da aplicação com base dois primeiros pontos;
- Realização de testes e análise da eficácia da aplicação na resolução do PRV;
- Documentação do projeto no presente relatório.

1.3. Estrutura do relatório

Este documento encontra-se estruturado em sete capítulos. No capítulo **Problema de Rotas de Veículos** é feito um enquadramento sobre o PRV, fazendo-se uma abordagem geral das características deste tipo de problema, conceitos básicos, objetivos gerais, das suas principais variantes e dos principais algoritmos utilizados na resolução deste tipo de problemas.

No capítulo **Estado de Arte** é apresentado um estudo do trabalho relacionado no que se refere a aplicações para resolver o PRV, fazendo-se um estudo aos níveis quer do mercado móvel, abrangendo os Sistemas Operativos *Android* e *Windows Phone*, quer de aplicações para computadores. No fim deste capítulo é apresentada uma tabela comparativa, mostrando as potencialidades de cada aplicação estudada de uma forma simples, e apresentado aquilo que se pretende fazer com a aplicação RouteMe, produto deste trabalho.

O capítulo **Tecnologias Utilizadas** é a secção do documento reservada para a descrição das tecnologias utilizadas no desenvolvimento da aplicação.

No capítulo **Implementação da Solução** é apresentada a descrição da implementação da solução, arquitetura da aplicação, base de dados, ecrãs constituintes da aplicação, bem como as soluções apresentadas pelo *FrontOffice* e *BackOffice*.

Por fim, no capítulo **Conclusões e Trabalho Futuro**, são apresentadas as conclusões sobre o trabalho desenvolvido, e comentários sobre o que foi feito e que ficou por fazer. São igualmente apontadas possíveis melhorias para um trabalho futuro.

2. PROBLEMA DE ROTAS DE VEÍCULOS

2.1. Enquadramento

O principal objetivo do PRV é a obtenção de melhores soluções, que diminuam os custos desses problemas.

Neste âmbito, surgem as soluções para os PRV que se propõem encontrar rotas otimizadas tendo em conta variáveis como os veículos disponíveis, localização do(s) depósito(s) e dos clientes.

A representação normal de um PRV é semelhante à apresentada na Figura 2. O objetivo é obter um conjunto de rotas que satisfaçam todos os clientes, reduzindo ao máximo os custos de distribuição, seja a nível de distância total de viagem, tempo total de viagem, ou número de veículos utilizados.

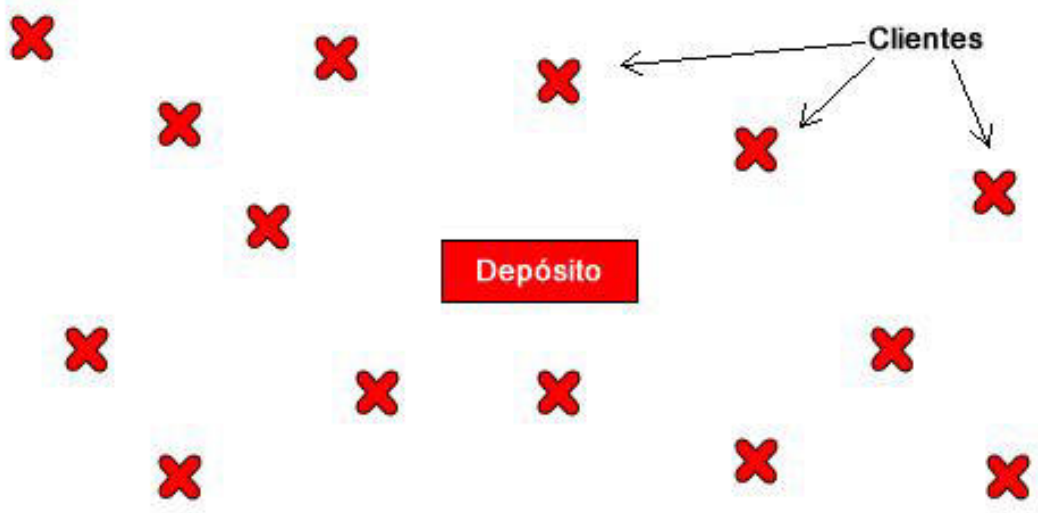


Figura 2 – Representação típica de um PRV

Pretende-se, partindo de um PRV com uma estrutura semelhante ao da apresentada na Figura 2, chegar à melhor solução possível de rotas, tal como o exemplo da Figura 3 mostra (neste caso considerando 4 veículos).

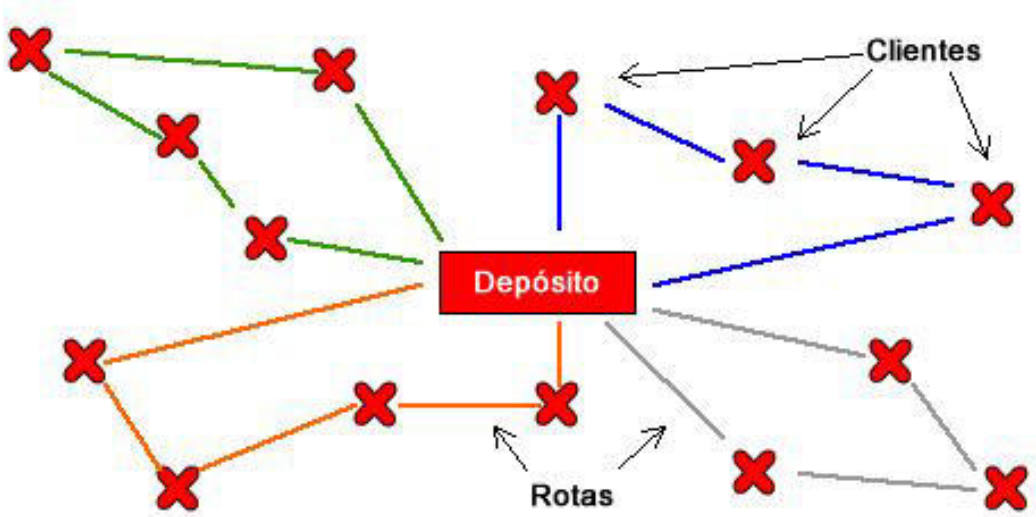


Figura 3 – Exemplo de solução de um PRV

2.2. Conceitos básicos

Tal como referido anteriormente, um PRV passa por um conjunto de veículos que vão sair de um ou mais depósitos, conduzidos por um conjunto de condutores, que irão proceder à recolha e/ou entrega de carga por um conjunto definido de clientes, através da rede rodoviária. Com todas estas variáveis e atendendo a que todos os veículos têm que voltar ao depósito de onde saíram, são determinadas as rotas de cada veículo de forma a minimizar ao máximo os custos globais do transporte.

Tendo em conta todos estes elementos, proceder-se-á de seguida à descrição de cada um deles.

2.2.1. Armazém

O armazém é o local de onde todos os veículos saem e para onde regressam. Além disso é aqui que está armazenada toda a carga a transportar. Os armazéns são caracterizados pelo número de veículos disponíveis. Em algumas variantes do VRP pode existir mais do que um armazém. Para esses casos cada veículo volta para o respetivo armazém.

Uma vez que os armazéns são definidos como ponto inicial e final de cada rota, estes são os vértices inicial e final do grafo.

2.2.2. Clientes

Como cliente entende-se um ponto de entrega e/ou recolha de produtos. Estes são caracterizados como os vértices de um grafo, à semelhança dos depósitos. Para além da quantidade de produtos a entregar e/ou recolher, os clientes podem ainda ser caracterizados por um dado período de tempo em que devem ser servidos, devido a horários ou outras restrições temporais.

2.2.3. Veículos

São os veículos que fazem as entregas e recolhas dos produtos, sendo caracterizados pelo armazém a que estão associados e para onde regressam no final de cada rota, pela sua capacidade de transporte, pelo tipo de carga que transportam, se têm ou não estradas (arcos) em que não podem passar, e os custos associados à sua utilização (sejam esses por rota, tempo, ou distância).

2.2.4. Condutores

Os condutores estão sujeitos a várias restrições, que vão desde fatores pessoais, contratuais, ao tipo de veículos que podem conduzir, ao número de horas diárias de trabalho, ao seu horário de trabalho e ao tempo máximo de condução consecutivo.

2.2.5. Rede Rodoviária

As ligações entre clientes e entre estes e os depósitos são efetuadas através da rede rodoviária, que pode ser descrita como um grafo onde clientes e depósitos representam os vértices e as estradas os arcos. A cada arco está sempre associado um custo, que pode corresponder ao tempo de viagem, à distância, ou à quantidade de carga a transportar entre cada vértice.

2.2.6. Rotas

Uma rota é composta por um depósito, por um conjunto de clientes (vértices do grafo), por um conjunto de ruas (arcos) que estabelecem o trajeto a percorrer para passar pelo conjunto de clientes, e o veículo que irá realizar essa rota. Tipicamente existem duas restrições associadas a cada rota, o facto de o total de carga dos clientes a

dar resposta não ultrapassarem a capacidade máxima do veículo, bem como a distância total de viagem não ultrapassar a distância máxima associada com o veículo.

2.3. Objetivos do PRV

Tal como até aqui referido, o principal objetivo do PRV é encontrar o conjunto de rotas que respeitando todas as restrições de um dado problema, satisfaçam a demanda dos clientes e minimizem os custos de operação da frota de veículos. No entanto, nem todos os casos requerem uma redução de custos. Há certos casos em que o objetivo passa por reduzir o número de veículos necessários para servir todos os clientes ou equilibrar as rotas em termos de tempo e distância de viagem.

2.4. Variantes do PRV

Uma vez que os PRV têm uma grande abrangência, têm sido estudadas várias variantes deste tipo de problemas [12]. As diferenças entre as várias variantes estão relacionadas com as restrições associadas a estes problemas. Esta secção apresenta um levantamento das variantes mais conhecidas do problema.

2.4.1. Traveling Salesman Problem (TSP)

O princípio básico de um TSP é que existe um único veículo, e esse veículo tem que visitar todos os seus clientes, e regressar ao ponto de origem. O objetivo geral é determinar a rota que sirva todos os clientes, apenas uma vez, em que a distância total percorrida pelo veículo seja menor. Esta rota é calculada tendo em conta que o veículo regressa para o ponto de onde iniciou a rota.

2.4.2. Multiple Traveling Salesman Problem (MTSP)

O princípio aplicado é o mesmo que na variante anterior. Neste tipo de problemas existem múltiplos veículos e um único depósito, e cada um dos veículos passa, no mínimo, por um cliente. Cada cliente é visitado apenas uma vez.

2.4.3. Capacited VRP (CVRP)

Esta variante é considerada o PRV básico onde os veículos têm associada apenas uma restrição de capacidade máxima, o que implica que a soma das demandas dos clientes associados a uma determinada rota não pode ultrapassar a capacidade máxima do veículo associado a essa mesma rota.

2.4.4. Distance Constrained VRP (DCVRP)

DCVRP é uma variante do CVRP, onde, neste caso, os veículos têm associada uma restrição de distância máxima que podem viajar, o que implica que as rotas da solução do PRV não podem ultrapassar a distância máxima associada aos veículos.

2.4.5. DC-CVRP

Esta variante associa nela as restrições das variantes CVRP e DCVRP onde os veículos têm associadas restrições de capacidade máxima e de distância máxima.

2.4.6. VRP with Time Windows (VRPTW)

A diferença desta variante em relação ao PRV básico é que os clientes têm associada uma restrição na forma de janela temporal que define o intervalo de tempo em que estes devem ser servidos pelo veículo da rota a que pertencem. Poderão haver aplicações desta variante em que a solução tem que respeitar obrigatoriamente todas as janelas temporais, ou em outras aplicações poderá existir uma penalização sempre que uma restrição de janela temporal não seja respeitada. Estas penalizações serão somadas ao custo total da solução.

2.4.7. VRP with Pick – up and Delivery (VRPPD)

A diferença desta variante em relação ao PRV básico é que neste caso pode haver clientes que para além de terem uma determinada demanda de produto que deve ser entregue, têm também uma determinada quantidade de produto que deve ser recolhida. Nesta variante é possível que um cliente seja servido por dois veículos (o veiculo que assegura a entrega e o veiculo que assegura a recolha). Em algumas aplicações, a sequência de recolhas e entregas realizadas por um dado veiculo pode ser aleatória, enquanto que noutras, as recolhas devem suceder as entregas.

2.4.8. Multiple Depot VRP (MDVRP)

No MDVRP, os veículos, em vez de partirem de um único depósito, partem de vários e, no final das suas rotas, regressam para o seu depósito de origem. Este tipo de instância é usado para minimizar os custos de distância, uma vez que, havendo vários depósitos, é possível haver uma maior abrangência de área a responder.

2.4.9. Split Delivery VRP (SDVRP)

Neste tipo de variante é possível que um cliente seja servido por mais do que um veículo. Isto faz com que o somatório da demanda dos clientes de uma dada rota não tenha que ser inferior ou igual à capacidade do veículo que assegura essa rota, tal como acontece no PRV básico.

2.5. Resolução do PRV

Para além das variantes do PRV apresentadas anteriormente, é igualmente importante referir os principais tipos de algoritmos utilizados para resolver o PRV.

O PRV pertence a uma classe de problemas em relação aos quais não se conhecem algoritmos capazes de os resolver de forma ótima em tempo útil. Isto significa que o tempo que os algoritmos necessitam para encontrar a solução ótima destes problemas, é exponencialmente proporcional à dimensão dos mesmos, o que implica que para instâncias de problemas a partir de uma certa dimensão, os algoritmos demoram tanto tempo que se torna inviável o seu uso em aplicações reais. Assim, embora existam algoritmos exatos para o PRV que garantem a solução ótima, apenas é possível usá-los em casos especiais ou em instâncias de pequena dimensão.

Para além dos algoritmos exatos, um outro tipo de algoritmos muito mais interessante para resolver o PRV são os algoritmos heurísticos que embora não garantam que seja encontrada a solução ótima, encontram soluções muito próximas desta e em tempo útil. O tempo que este tipo de algoritmos necessita para encontrar boas soluções é polinómiamente proporcional à dimensão do problema, o que torna viável o seu uso em aplicações reais.

Os algoritmos heurísticos que têm vindo a ser desenvolvidos para o PRV são normalmente de dois tipos, Algoritmos Construtivos ou Algoritmos de Duas Fases.

Os Algoritmos Construtivos resolvem o PRV construindo uma solução passo a passo adicionando cliente após cliente, respeitando todas as restrições do problema e tendo em conta a minimização do custo total dessa mesma solução.

Os Algoritmos de Duas Fases encontram a solução do PRV agrupando numa primeira fase os clientes que devem pertencer a cada rota e depois numa segunda fase definindo a sequência pela qual os clientes de cada rota devem ser visitados pelos respetivos veículos. Há também algoritmos deste tipo em que as fazes são invertidas. Começam por primeiro estabelecer a sequência pelas quais os clientes devem ser visitados e depois agrupam estes em diferentes rotas.

Para além dos algoritmos exatos e dos algoritmos heurísticos, o PRV também pode ser resolvido recorrendo a Metaheuristicas. As Metaheuristicas podem ser consideradas algoritmos heurísticos, pois tal como estes, encontram soluções próximas da solução ótima em tempo útil. A diferença é que são algoritmos genéricos que podem ser facilmente adaptados para resolver problemas diferentes, normalmente problemas de otimização combinatória como o PRV. O princípio de funcionamento das Metaheuristicas é o de pesquisar de forma inteligente o vasto espaço de soluções possíveis de modo a encontrar as melhores. Para o conseguir usam normalmente dois mecanismos de pesquisa, Pesquisa Local e Pesquisa com Base numa População. No caso da Pesquisa Local, a pesquisa do espaço de soluções é realizada passando, a cada passo, da solução corrente para uma nova solução promissora na vizinhança. A Pesquisa com Base numa População consiste em manter uma população de boas soluções e recombinar as mesmas para obter uma nova geração de soluções. Ao repetir este processo, a população de soluções converge para soluções do problema cada vez melhores.

3. ESTADO DE ARTE

3.1. Introdução

Para a elaboração deste trabalho foi importante perceber o estado de arte de aplicações que se assemelhassem ao pretendido. Foi possível averiguar que, para dispositivos móveis, não existem ainda muitas aplicações para o PRV, apesar de existirem muitas para computadores desktop. É no entanto importante referir a análise feita ao mercado de forma a conhecermos possíveis concorrentes e a percebermos os pontos fortes e fracos deste mercado. Esta pesquisa teve também como objetivo, através do conhecimento do mercado, permitir distanciar a aplicação desenvolvida de semelhanças em aplicações existentes para dispositivos móveis e, ao mesmo tempo, tentar a aproximação a soluções ótimas existentes para computadores, tentando alcançar um produto inovador na área dos dispositivos móveis.

As secções que se seguem refletem o estudo feito às lojas online das plataformas de dispositivos móveis *Android* e *Windows Phone*, onde foi possível recolher informações referentes às aplicações existentes.

3.2. Aplicações para a plataforma Android

3.2.1. Route & Go

Aplicação paga (3,99€), que permite ao utilizador gravar rotas ou viagens.

O utilizador marca os pontos por onde quer passar (num máximo de 50) e é automaticamente calculada uma rota (otimizada) que passa por todos esses pontos. São-lhe apresentadas informações da rota relativas a tempo e distância totais. Só é possível criar uma rota por mapa; se o utilizador pretender criar uma nova rota tem que gravar (se o pretender) a corrente rota e começar uma nova num novo mapa.

Para a geração da rota o utilizador pode identificar se quer um percurso a pé, de bicicleta ou de carro, e se quer evitar ou não portagens e autoestradas. Toda esta informação é adquirida a partir do *Google Maps*.

3.2.2. Speedy Route

Aplicação gratuita, onde o principal objetivo é oferecer ao utilizador o caminho mais rápido e com menores custos de transporte para um determinado trajeto. Permite ao utilizador identificar diversos pontos a incluir na rota mas é possível criar uma única rota. Todas as informações recolhidas pela aplicação são fornecidas pelo *Google Maps* e é possível a qualquer momento ver informações sobre um determinado ponto. Só é possível adicionar novos pontos através da morada (escrita), e não através de cliques no mapa.

3.2.3. Route4Me Route Planner

Aplicação gratuita que permite ao utilizador gerar rotas com um número indeterminado de pontos. Através de algoritmos de otimização, oferece ao utilizador uma rota otimizada. Só é possível criar uma única rota.

3.2.4. MySmartRoute Route Planner

Aplicação gratuita, que permite ao utilizador gerar uma única rota de cada vez, com um número indeterminado de pontos. Apesar disso, a otimização de rotas funciona apenas em rotas com, no máximo, 30 pontos. É possível guardar um número ilimitado de rotas. Permite ainda definir se a rota será de carro, de bicicleta ou a pé.

3.2.5. Road Warrior Route Planner

Aplicação com versão gratuita e paga. O utilizador tem permissão para gerar apenas uma única rota. A quantidade de pontos que pode inserir depende da versão instalada; a versão gratuita pode inserir apenas 8 pontos enquanto que na versão paga pode inserir até 120. É possível editar a rota através de um simples *drag & drop* e também rastrear a rota através de "check-ins", assim que se passa por cada um dos pontos.

3.2.6. Route Navigation

Aplicação gratuita. É uma aplicação pouco intuitiva, com fraca usabilidade e bastante lenta. Permite adicionar vários pontos, através de moradas, pesquisa por locais (como museus, pizzarias, escolas, etc...), ou através de cliques no mapa, mas permite a criação de apenas uma rota. Quanto à geração da rota, recorre ao *Google Maps*, em que o utilizador é transferido da aplicação para o próprio *Google Maps*.

3.2.7. Voyager: Route Planner

Aplicação gratuita. Permite gerar apenas uma rota, com tantos pontos quantos o utilizador queira. Apesar disso, o criador da aplicação alerta que usa a versão free do *Google Maps*, em que o limite de pontos é o estipulado pela Google. Utiliza dois algoritmos diferentes para a otimização da rota: no caso de haver poucos pontos, um algoritmo de força bruta; no caso de haver vários pontos, utiliza algoritmos de proximidade (algoritmos esses não especificados na aplicação), sendo que o que estes algoritmos fazem é adicionar à rota o ponto mais próximo. Os resultados são apresentados num mapa do *Google Maps*.

3.2.8. BestRoute Pro

Aplicação paga (4,99€). Disponibiliza uma versão trial que permite criação e otimização de rotas com até 30 pontos, enquanto que a versão paga permite até um máximo de 85. Permite controlo da rota através de "check-ins" para que o utilizador se possa manter atualizado sobre a sua rota. Permite apenas a criação de uma única rota.

3.2.9. trucker

Aplicação com versões gratuita e paga (6,39€ por mês). Aplicação pouco intuitiva que no caso de se querer gerar rotas é preciso subscrever a versão paga. A versão gratuita permite apenas obter uma rota entre dois pontos, sendo um ponto a indicar pelo utilizador e o segundo a posição atual.

3.2.10. Route Optimizer

Aplicação gratuita, que permite ao utilizador inserir até um máximo de 8 pontos, para além do ponto inicial e final da rota. A inserção dos pontos é pouco intuitiva.

Sempre que é acrescentado um novo ponto é regenerada toda a rota. Apresenta ainda informação sobre tempo e distância totais.

3.3. Aplicações para a plataforma Windows Phone

3.3.1. Triplexer

Aplicação gratuita, onde o utilizador pode desenhar apenas uma rota de cada vez, indicando o ponto seguinte através da morada. É possível adicionar novos pontos no mapa através de clique, mas não é possível inseri-los na rota.

3.3.2. OptiRoute

Aplicação gratuita que permite ao utilizador marcar vários pontos, gerando de seguida uma rota otimizada através de um algoritmo de proximidade (algoritmo esse não especificado na aplicação), sendo que o que este algoritmo faz é adicionar à rota o ponto mais próximo. Permite apenas a geração de uma rota de cada vez.

3.4. Aplicações para computadores desktop

3.4.1. The Wall – Vehicle Routing Problem

Este projeto, criado por um aluno [13], foi feito com recurso aos algoritmos Clarcks & Wrights [14] e Sweep [15], e desenvolvido com recurso à tecnologia Java e API's como GWT-MAPS, GWT-INCUBATOR e *Google Maps*.

Esta aplicação permite apresentar os resultados em mapas reais (caso com dados georreferenciados) ou em grafos (no caso de instâncias com dados definidos no plano).

Neste projeto foram tidos em conta os seguintes fatores: localização do depósito; frota de veículos, cada veículo com uma determinada capacidade de transporte; clientes, com a sua localização e uma demanda.

No caso da representação das rotas por mapas reais, a aplicação parece devolver rotas de boa qualidade. No entanto, no caso da representação das rotas por

grafos, as rotas geradas apresentam por vezes cruzamentos, o que significa que teria sido sempre possível encontrar uma rota melhor onde não ocorressem cruzamentos.

3.4.2. Logvrp

Logvrp é uma aplicação "*web based*", o que permite que esta seja acedida em qualquer lado e em qualquer altura, e todos as atualizações são livres de pagamentos extras [16].

Esta aplicação permite fazer uma gestão exata dos custos, quer antes da aquisição da licença, quer durante a utilização, não tendo contrato de longo termo.

Utiliza a API do *Google Maps*, o que faz com que não seja necessário adquirir qualquer software extra para a criação dos mapas e das rotas. Através desta mesma API permite determinar a posição atual, bem como a procura de localizações.

Logvrp está desenhado para contemplar um vasto tipo de requisitos dos VRP's, tais como depósitos, clientes, viagens com entrega e recolha, etc, e permite a utilização de vários algoritmos para resolver estes mesmos problemas. Permite inclusive a utilização de diferentes algoritmos em simultâneo para o mesmo problema, permitindo uma comparação de custos, distâncias, rotas e mapas das soluções geradas pelos vários algoritmos.

As rotas geradas são visualizadas a cores diferentes e podem ser vistas como estradas reais, fornecendo vários tipos de dados, tais como a carga do veículo antes e depois de cada ponto de entrega/recolha, as respetivas quantidades de entrega/recolha, datas de entrega e recolha, duração de cada troço de viagem, entre outros. É ainda possível definir a localização de começo e término de toda a frota, veículo a veículo.

As rotas, assim que geradas (com recurso a vários tipos de algoritmos existentes para a resolução do PRV [7][8]), para além de poderem ser postas em prática pelos condutores dos veículos de forma a que estes tenham sempre presente a estrutura da rota e os tempos que devem seguir, podem também ser exportadas para Excel com todos os dados da rota ou enviar os dados da rota por email. Podem também ser importados alguns dados, tais como dados de veículos e clientes.

É possível editar manualmente as rotas, guardá-las com as alterações feitas e reedita-las novamente.

3.4.3. A Web Spatial Decision Support System for Vehicle Routing using Google Maps

Esta aplicação [17] permite gerar soluções com algoritmos heurísticos embebidos no sistema, com uma utilização transparente para o utilizador; gerar mapas e instruções para rotas individuais de cada veículo; mostrar informação detalhada de cada rota e representar as rotas num browser. Para representar essas rotas, recorreram à API do *Google Maps*. Os dados de cada problema são guardados em base de dados acessíveis também via internet.

A solução foi desenvolvida com recurso à framewoork ASP.Net e às linguagens de programação C# e Javascript, sendo utilizada a base de dados Microsoft SQL Server Express.

A aplicação permite editar todos os dados dos veículos (desde custos, a capacidades e tempos máximos, tipo de veículo), pontos de entrega. Permite também editar toda a rota, alterando-a diretamente no mapa.

Uma vez que a aplicação comunica diretamente com a API do *Google Maps*, consegue saber automaticamente se as ruas por onde a rota passa são de sentido único ou não. É possível também marcar um ponto terminal para cada rota, sendo que esse ponto final pode não ser o depósito mas sim outro local.

Quando se está a visualizar a rota no mapa é disponibilizada um vasto número de detalhes, desde direções, tempos entre cada nó, carga no momento, entre outros. Tudo isto serve para saber se a dada altura o veículo está a cumprir com os parâmetros (quer de tempo, quer de carga).

3.5. Análise comparativa das aplicações estudadas

A partir da análise das aplicações até aqui apresentadas, foi possível obter a seguinte tabela, que nos permite comparar as características de cada uma das aplicações.

Características / Aplicações	Versão			Dados em:				Inserir Pontos:			Dados		Rotas									
	Paga	Gratuita	Google Maps	Mapas Reais	Grafos	Veículos com Propriedades	Clientes com Propriedades	Através de Moradas	Através de Cliques no Mapa	Através de Coordenadas	Exportar Dados	Importar Dados	Editar Manualmente as Rotas	Iniciar Rotas Singulares Manualmente	Múltiplos Algoritmos para Comparação	Viagens com Entregas e Recolhas	Não Permitir Cruzamento de Rotas	Pontos de Chegada/Partida de cada Veículo	Quantidade de Armazéns	Número Máximo de Clientes	Número Máximo de Rotas	Informações das Rotas
The Wall			X	X	X	X	X										X		1	ND	ND	
Logvrp			X	X		X	X				X	X	X		X	X		X	>1	ND	ND	
WSD *			X	X		X	X						X					X	1	ND	ND	
Route & Go	X		X	X					X										0	50	1	X
Speedy Route		X	X	X				X											0	ND	1	X
Route4 Me Route Planner		X		X					X										ND	N	1	
MySmartRoute Route Planner		X		X					X				X						ND	30	1	
Road Warrior Route Planner	X	X		X					X				X						ND	8/120	1	
Route Navigation		X	X					X	X										ND	N	1	
Voyager: Route Planner		X	X	X					X										ND	G	1	
BestRoute Pro	X	X		X					X										ND	30/85	1	
trucker	X	X		X					X										ND	ND	1	
Route Optimizer		X	X	X					X										ND	10	1	X
Triplexer		X		X				X											ND	N	1	
OptiRoute		X		X															ND	N	1	
Route Me			X	X		X	X		X		X		X	X			X		1	N	N	X

Tabela 1 – Comparação das características das aplicações estudadas

* A Web Spatial Decision Support System for Vehicle Routing using Google Maps
* N – tantos pontos quantos o utilizador quiser;
* G – tantos pontos quanto a versão free do Google Maps permita;
* ND – Não Definido

As aplicações estudadas apresentam vários pontos fortes, tal como a utilização de mapas reais do *Google Maps*, veículos e clientes com propriedades, permitir exportar rotas, utilização dos componentes interativos com os dispositivos móveis através de cliques no ecrã, entre outros, os quais foram considerados como objetivos na implementação do projeto. Algumas características, como por exemplo a exportação e importação de dados, definição de pontos diferentes de partida e chegada dos veículos e definição de pontos/clientes com entrega e recolha, bem como a interligação da aplicação com uma plataforma web, foram tidas em conta numa fase mais avançada do projeto. É também possível observar alguns pontos fracos, tais como permitir ao utilizador iniciar as rotas sementes e as ferramentas interativas (existentes em poucas aplicações), os quais se tentaram inverter, implementando-se como sendo pontos fortes da solução final.

Genericamente, a aplicação teve em conta todos os aspetos relevantes de um PRV (clientes, depósitos, frota de veículos) e a interatividade com o utilizador (em que este terá hipótese de definir o inicio de cada rota, bem como editar manualmente a rota depois de gerada).

Olhando mais atentamente para as aplicações para dispositivos móveis que foram estudadas, nota-se que nenhuma delas tem a capacidade de gerar mais do que uma rota, nem permite definir as características de cada cliente ou veículo, nem importar ou exportar dados, nem sequer permite ao utilizador desenhar, ele próprio, a sua rota.

4. TECNOLOGIAS UTILIZADAS

Para desenvolver a aplicação RouteMe recorreu-se a diversas tecnologias. Neste capítulo será feita uma descrição das tecnologias utilizadas no desenvolvimento da presente aplicação.

4.1. Sistema Operativo Android

Em outubro de 2003, Andy Rubin, Rich Miner, Nick Sears e Chris White fundaram, na Califórnia, a *Android Inc*, cujo principal objetivo era, segundo as palavras de um dos fundadores, desenvolver dispositivos móveis inteligentes cientes da localização e preferências do utilizador [18].

A Google viria a adquirir a *Android* Inc. em 2005 e, em 2007, após a criação da Open Handset Alliance, uma aliança de várias empresas como Google, eBay, HTC, LG, Samsung, entre outras, foi apresentada a primeira versão do Android SDK, *Android alfa*, que foi a primeira versão de pré-lançamento.

O primeiro dispositivo móvel a utilizar o sistema operativo *Android* (versão 1.0) comercializado foi apresentado em outubro de 2008. Neste momento encontramo-nos já na versão 5.0, sendo que as percentagens de utilização de cada versão serão apresentadas de seguida.

Versão	Nome de Distribuição	API	Utilização (%)
2.2	Froyo	8	0.4
2.3.3 – 2.3.7	Gingerbread	10	6.9
4.0.3 – 4.0.4	Ice Cream Sandwich	15	5.9
4.1.x	Jelly Bean	16	17.3
4.2.x		17	19.4
4.3		18	5.9
4.4	KitKat	19	40.9
5.0	Lolipop	21	3.3

Tabela 2 – As diferentes versões do sistema operativo *Android* e a respetiva percentagem de utilização atual [19]

O mercado dos dispositivos móveis tem tido um enorme crescimento nos últimos anos e, apesar do *Android* não ser o único sistema operativo para dispositivos móveis, é, sem dúvida, detentor de uma grande parcela deste mercado como se pode ver pelos dados da tabela 3 [20].

Ano	Android	IOS	Windows Phone	BlackBerry OS	Others
2014	76,6%	19,7%	2,8%	0,4%	0,5%
2013	78,2%	17,5%	3.0%	0,6%	0,8%
2012	70,4%	20,9%	2,6%	3,2%	2,9%
2011	52,8%	23,0%	1,5%	8,1%	14,6%

Tabela 3 – Utilização de Sistemas Operativos móveis nos últimos 4 anos

4.1.1. Arquitetura

A Arquitetura do sistema operativo *Android* está organizada em pilha, composta por cinco secções e agrupada em quatro camadas. Cada uma destas camadas agrupa um conjunto de programas que suportam funções específicas do Sistema Operativo, tal como mostra a seguinte figura.

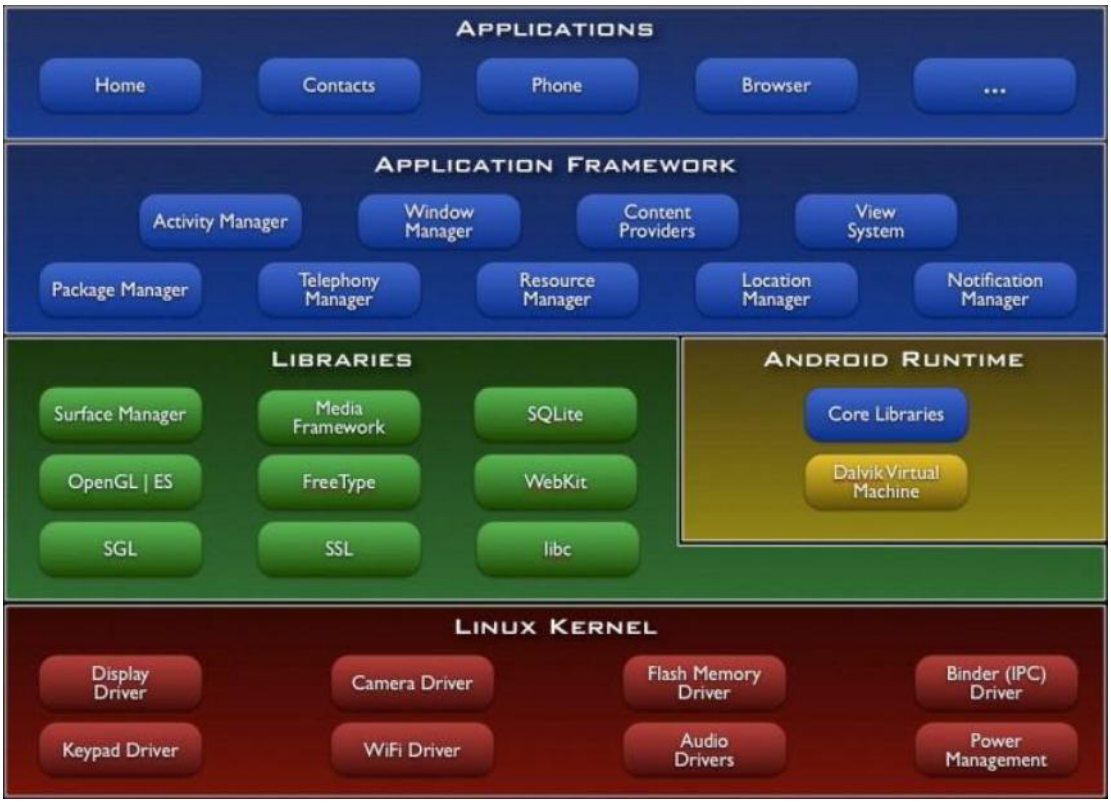


Figura 4 – Arquitetura do sistema operativo Android

- Linux Kernel – nível mais baixo do sistema operativo, baseado na versão 2.6 do Kernel do sistema operativo Linux, que contém os drivers para os componentes de hardware existentes num dispositivo *Android* e programas de gestão de memória, de gestão de energia, configurações de segurança, controlo de processos, entre outros;
- Libraries – camada superior ao Kernel, que contém as bibliotecas C/C++ utilizadas por diversos componentes do sistema, como por exemplo as bibliotecas SQLite (para base de dados), WebKit (para navegação Web), bibliotecas para suporte a formatos de media (áudio, imagens e vídeo), ou para gráficos 2D (SGL) e 3D (OpenGL ES);
- Android Runtime – disponibiliza um conjunto de bibliotecas que permitam aos programadores o desenvolvimento de aplicações para o sistema operativo *Android* com recursos à linguagem de programação Java. É também aqui que se encontra a máquina virtual Dalvik onde as aplicações escritas em Java e compiladas em *bytecodes* (formato binário) Dalvik são executadas. Desta forma as aplicações podem ser distribuídas em *bytecodes* e executadas em qualquer dispositivo *Android*, independentemente do processador utilizado.
- Application Framework – responsável por fornecer todas as funcionalidades necessárias para a construção de aplicações, com recurso a bibliotecas nativas, e faz a interface com as aplicações *Android*. Esta camada é maioritariamente escrita em Java.
- Applications – esta é a camada mais alta; é aqui que estão todas as aplicações usadas pelo utilizador do dispositivo, onde podem ser encontradas funcionalidades como gestor de contactos, calculadora, correio eletrónico, browser, mapas, cliente de mensagens (SMS), entre muitos outros.

4.2. Google Maps API

O *Google Maps* é um serviço público e gratuito que qualquer pessoa pode utilizar nas suas aplicações e sites [21]. À parte deste serviço, existe a API *Google Maps*

que fornece ao programador um conjunto de classes e ferramentas para aceder a esse serviço.

Esta API é, na verdade, constituída por várias API's:

- **Google Maps JavaScript API –** permite inserir um mapa do Google num site através de JavaScript;
- **Google Maps API for Flash –** com esta API ActionScript é possível inserir um mapa num site ou aplicação baseados em Flash;
- **Google Earth API –** permite adicionar um globo digital terrestre 3D nos sites;
- **Google Static Maps API –** permite adicionar uma imagem simples de um mapa do *Google Maps* numa aplicação móvel sem ter que recorrer a API's baseadas em JavaScript nem a carregamentos dinâmicos;
- **Serviços Web –** permite obter dados, através de um pedido URL, relativos a georreferenciação, tais como rotas entre pontos, distâncias entre pontos, elevações do terreno, entre outros. Estes dados podem ser obtidos em formato JSON ou XML;
- **Google Maps Data API –** permite ver, guardar e atualizar dados de mapas através de feeds do Google Data API.

4.3. WebService

A W3C (World Wide Web Consortium) [22] é um dos principais responsáveis pela padronização dos WebService, que criaram o padrão WSDL (Web Services Description Language), que permite descrever os WebServices no formato XML.

Um WebService é um método de comunicação utilizado para garantir a interoperabilidade entre vários serviços ou aplicações sem a necessidade de interação humana, onde o principal objetivo destes sistemas passa por garantir a comunicação entre diversos sistemas, mesmo que estes sejam desenvolvidos com diferentes tecnologias. Estes WebServices ficam alojados em servidores Web, por forma a poderem ser acedidos por todos em qualquer altura.

As comunicações entre um WebService e as outras aplicações são feitas através da Internet, permitindo às diversas aplicações invocar métodos do WebService para efetuar determinada tarefa, que podem, por sua vez, comunicar com outras aplicações. Por norma o transporte de dados é realizado através dos protocolos HTTP ou HTTPS (no caso de ligações seguras), e estes são transferidos no formato XML, encapsulados pelo protocolo SOAP (Simple Object Access Protocol) [23].

4.4. SQLite

Para a implementação da base de dados da aplicação do projeto recorreu-se ao SQLite, que é um motor de Base de Dados Transacional, autossuficiente, e que não necessita qualquer tipo de configuração [24].

O código do SQLite é Open Source, sendo livre a sua utilização para quaisquer fins, pessoais ou empresariais e é, segundo o próprio SQLite, utilizado em inúmeras aplicações. O SQLite armazena os dados de forma compacta em que as informações de todas as tabelas, índices, triggers e views são escritas num único documento, que pode ser partilhado e lido em qualquer sistema de qualquer arquitetura (32 ou 64 bits), o que faz deste um sistema de base de dados multiplataforma.

A forma de funcionamento do SQLite que recorre a um único ficheiro para armazenar os dados, faz com que ocupe pouco espaço e seja muito rápido, e portanto ideal para ser usado em dispositivos móveis caracterizados por terem memória e capacidade de processamento reduzidos. Este é o motor de base de dados mais utilizado a nível de plataformas móveis.

4.5. Microsoft .Net Framework

A plataforma Microsoft .Net Framework é uma plataforma de desenvolvimento e execução de aplicações e sistemas, fazendo assim com que tudo o que é desenvolvido utilizando a tecnologia .Net possa ser executado em qualquer dispositivo que possua essa plataforma [25].

O Microsoft .Net Framework é constituída por dois componentes:

- **Common Language Runtime (CLR)** [26], que é o núcleo da tecnologia .Net, visto ser o ambiente de execução das diversas aplicações desenvolvidas em várias linguagens de programação. A ferramenta de desenvolvimento compila o código fonte da aplicação numa das linguagens suportadas pelo .Net (MSIL – Microsoft Intermediate Language), baseando-se no Common Language Specification (CLS) que determina as regras necessárias para a criação do código MSIL compatível com o CLR. Este código MSIL é posteriormente convertido em código Assembly, por intermédio do compilador JIT (Just-In-Time). É desta forma que o .Net Framework implementa a independencia da plataforma, pois cada uma tem o seu compilador JIT e cria o seu próprio código Assembly a partir do código MSIL.
- **Framework Class Library (FCL)** é uma biblioteca de classes, interfaces, e tipos de dados que fornece acesso às funcionalidades do sistema. É recorrendo a esta biblioteca que todas as aplicações, componentes e controladores baseados no .Net Framework são construídos [27].

O CLR comunica com o FCL, e é capaz de executar diferentes linguagens de programação, como o C#, C++, F#, COBOL, Java, Javascript, Perl, Visual Basic, entre outros, e que permite que todas estas diferentes linguagens possam comunicar entre si como se de uma só se tratassem.

Esta tecnologia foi utilizada para a criação do WebService, desenvolvido com recurso à linguagem de programação Visual Basic.

4.6. Google Docs

O Google Docs é um pacote de aplicações baseadas em AJAX, que funcionam totalmente online e apenas com necessidade de utilizar um browser [28]. Atualmente é composto por um editor de texto, um editor de apresentações, um editor de formulários e um editor de folhas de cálculo, e todos eles são compatíveis com as ferramentas Microsoft Office. Uma grande particularidade desde documentos é que permitem que vários

utilizadores estejam a editar simultaneamente o mesmo documento, vendo todas as alterações em tempo real.

O Google Docs foi utilizado no desenvolvimento do RouteMe para exportar dados derivados de pedidos ao WebService, guardando aqui os dados de todos os clientes e o tempo que o WebService necessitou na comunicação com o *Google Maps*.

5. IMPLEMENTAÇÃO DA APLICAÇÃO ROUTEME

A aplicação móvel RouteMe foi totalmente desenvolvida com recurso ao ambiente de programação Eclipse, utilizando o Android SDK e a biblioteca Google Maps API.

Quanto ao WebService, todo o desenvolvimento foi feito com recurso à plataforma ASP.NET, através do ambiente de desenvolvimento Visual Studio e com recurso à linguagem de programação Visual Basic.

A base de dados, existente apenas do lado do cliente, foi criada com recurso às bibliotecas SQLite disponibilizadas pelo Android.

5.1. Arquitetura do Sistema Desenvolvido

A Figura 5 mostra a arquitetura do sistema desenvolvido, constituída por diferentes módulos. O módulo Interface com o Utilizador comunica com o módulo Base de Dados e com o módulo Resolução de Problemas Este, por sua vez, irá comunicar com a Base de Dados e Algoritmos necessários, obtendo destes uma resposta que, depois da resolução do problema, irá apresentar a solução encontrada ao utilizador através do módulo de Interface com o Utilizador.

Figura 5 – Arquitetura do sistema desenvolvido

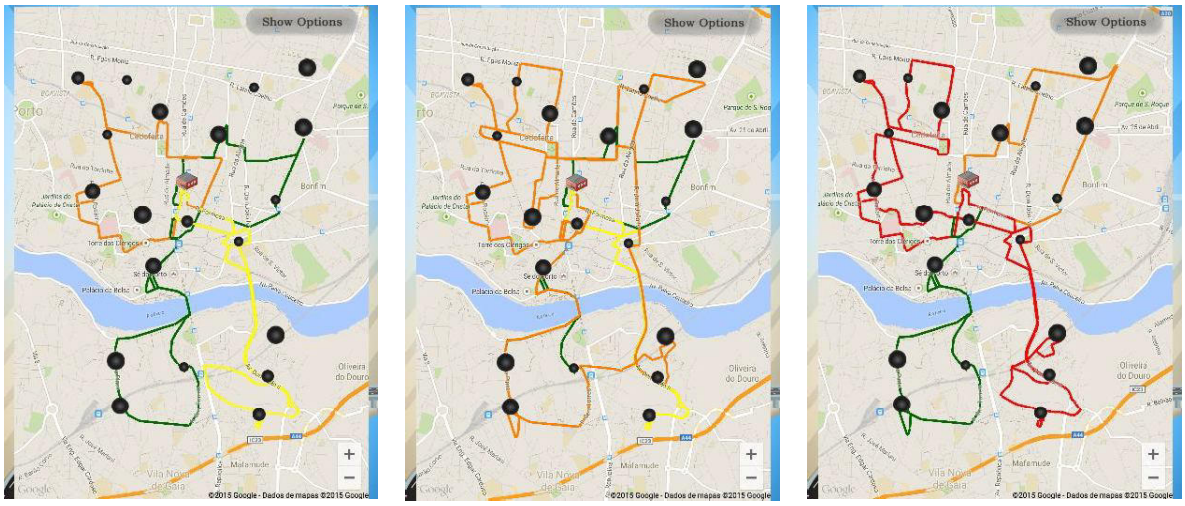
5.2. Método visual interativo para a resolução do PRV

A aplicação desenvolvida, RouteMe, permite resolver *Capacited Vehicle Routing Problems* da vida real [1], baseada num método de solução visual interativa implementada através de um Greedy Randomized Adaptative Search Procedure (GRASP) [29]. O foco principal da aplicação é tirar partido das características e dos recursos dos dispositivos móveis, tal como mapas geográficos e serviços de localização e as capacidades interativas que permitem o desenvolvimento de uma aplicação avançada que pode ser usada por utilizadores comuns, capaz de resolver problemas da vida real.

A motivação para utilizar um método visual interativo foi a de integrar a experiência e conhecimento do utilizador, e o poder e precisão das heurísticas num ambiente interativo. Através dessa interação o utilizador é capaz de controlar a solução, selecionando os parâmetros iniciais, bem como de ajustar as soluções. O utilizador pode

também utilizar os seus conhecimentos sobre o problema da vida real, guiando as heurísticas para áreas promissoras da solução. Uma solução interativa facilita também a inclusão de restrições e torna as soluções mais aceitáveis porque o utilizador participa ativamente na criação da solução.

Trabalhos anteriores [11] mostraram que é possível obter excelentes resultados para o PRV através de métodos visuais interativos e do GRAPS. Esse método foi mais tarde estendido para resolver diferentes variantes do PRV [30][31], mostrando que o método é flexível e adaptável a diferentes tipos de problemas. A aplicação desenvolvida usa um método adaptado do método referido, que tem em conta interessantes características tal como mencionado anteriormente, mas também as limitações e restrições dos dispositivos móveis, especialmente no que tem a ver com poder de processamento e capacidade de memória. Para superar estas limitações é necessária uma implementação cuidadosa dos métodos visuais interativos.



a) b) c)

Figura 6 – Ilustração das três fases do método implementado; a) esboço das rotas após seleção das sementes; b) solução obtida depois da construção e melhoramento das rotas; c) solução obtida após aperfeiçoamento da solução

O método visual interativo tem três fases diferentes: *Seleção das Sementes*, *Construção e Melhoria das Rotas*, e *Aperfeiçoamento da Solução*. A Figura 6 ilustra as três fases que são descritas com mais detalhe de seguida.

O método começa com a definição do problema onde o utilizador cria uma nova instância e seleciona as localizações dos clientes e do armazém diretamente num mapa real, através de um simples toque no ecrã. Para cada cliente o utilizador indica também a respetiva demanda. Os clientes são representados no mapa como círculos com diferentes diâmetros, proporcionais à sua demanda. Depois de selecionar os clientes, a aplicação utiliza os serviços do Google Maps para obter as distâncias e as rotas entre os clientes e o armazém, utilizados pelo método implementado.

Na fase da *Seleção das Sementes*, com a localização dos clientes representada num mapa real, o utilizador seleciona um ou mais clientes semente na ordem em que o utilizador espera que sejam visitados, para criar um esboço de uma rota. Os clientes semente são os clientes que por alguma razão o utilizador força a serem servidos por veículos específicos. Estes clientes semente são selecionados baseados no conhecimento e experiência do utilizador no problema da vida real, ou então através da identificação de determinados padrões, como grupos de clientes, clientes com demandas muito elevadas ou clientes isolados. Durante o processo de seleção o utilizador é ajudado pela aplicação, que verifica se não existem violações das limitações de capacidade de transporte dos veículos. A Figura 6 a) mostra um esboço típico das rotas após a seleção dos clientes semente.

A fase de *Construção e Melhoria das Rotas* constrói as rotas agrupando os clientes restantes de acordo com os veículos definidos nas sementes. Este agrupamento é feito sem a intervenção do utilizador e é implementado na fase de construção do algoritmo GRASP. A heurística pode encontrar uma solução viável onde todos os clientes são alocados aos veículos, ou pode encontrar uma solução inviável, devido à violação de restrições.

A heurística de agrupamento aloca cada cliente numa rota cuja posição represente um menor custo, dando prioridade de alocação a clientes com uma rota de inserção mais óbvia (dado pela diferença entre o segundo menor e o menor custos de inserção), e a clientes cujo número de rotas onde podem ser inseridos é menor. Durante este processo é aplicada a heurística *2-optimal* [32] de forma a reduzir a distância total de viagem de cada veículo. A heurística de agrupamento é seguida por uma fase de pesquisa local que tenta melhorar a solução obtida pelo agrupamento, se viável. A fase de pesquisa local GRASP implementa um procedimento de troca de um nó, que aceita imediatamente

a primeira troca que resultar numa melhor solução. De forma a acelerar este processo, o algoritmo implementa uma estratégia *p-Neighbourhood*, semelhante à apresentada por Gendreau et al [33], que reduz significativamente o número de trocas a considerar, selecionando apenas as rotas a ter em consideração que se encontrarem "perto" do cliente que se pretende reposicionar. Esse conceito de "perto" é definido por uma medida que corresponde à distância entre clientes no mapa. A Figura 6 b) mostra um exemplo de uma solução viável obtida através do conjunto inicial de sementes. A solução obtida na fase de construção pode ser igualmente viável ou inviável. O método implementado guarda a melhor solução viável, e o utilizador a pode tentar melhorar na fase de aperfeiçoamento da solução. A melhor solução inviável é definida como a solução inviável com um total de demanda não alocado menor, uma vez que desta forma torna-se mais provável a possibilidade de encontrar uma solução viável durante a fase de aperfeiçoamento da solução.

Na fase de *Aperfeiçoamento da Solução*, o utilizador pode tentar aperfeiçoar a solução obtida de forma a melhorá-la, especialmente se esta for inviável, utilizando as ferramentas interativas. Sem regras formais a seguir, o sistema interativo desenvolvido é suficientemente flexível para permitir a criatividade e imaginação do utilizador, e existem várias técnicas de melhoria disponíveis através destas mesmas ferramentas. As ferramentas interativas foram implementadas baseadas no simples toque no ecrã. Os exemplos mais simples consistem na remoção de clientes das suas rotas. Este passo é seguido por uma nova chamada às heurísticas de construção e melhoria das rotas, permitindo que esses clientes sejam inseridos em diferentes rotas. Este é um procedimento muito simples mas muito eficaz quando há sobreposição de rotas, e pode levar a diferentes soluções viáveis num curto período de tempo. A Figura 6 c) mostra uma possível solução final após a utilização das ferramentas interativas.

O utilizador tem diversas ferramentas interativas que pode escolher: remover clientes de uma rota, inserir clientes numa rota, regressar às rotas sementes. Todas as ferramentas interativas foram implementadas baseadas no toque no ecrã dos dispositivos móveis.

5.3. Diagrama de Casos de Uso da aplicação

A aplicação terá apenas um Actor, denominado "User", que representa o utilizador que irá utilizar a aplicação. O utilizador terá acesso a duas grandes secções dentro da aplicação que são a Gestão de Frota e a Gestão de Rotas. No caso da Gestão de Frota, poderá ver uma lista de veículos, onde terá a função de inserir, editar e remover qualquer veículo na sua lista. Quanto à Gestão de Rotas, esta será o ponto crucial da aplicação. Em primeiro lugar poderá ver todas as rotas anteriormente geradas, podendo, para cada uma delas, proceder à sua remoção, edição e recriação. A edição da rota consistirá essencialmente em carregar a rota e proceder à edição da mesma manualmente. No caso da recriação da rota, o utilizador terá a possibilidade de gerar a rota a partir do ponto zero (semelhante à criação de rotas, explicada a seguir).

A criação da rota englobará todos os pontos necessários para a criação de uma rota. O utilizador poderá inserir a posição do armazém e inserir, editar e remover clientes. Depois disso o utilizador poderá proceder de duas formas diferentes: iniciar a criação de rotas sementes ou permitir que os algoritmos gerem uma primeira rota sozinhos. Se o utilizador optar por deixar os algoritmos gerarem as rotas por si, a aplicação irá mostrar ao utilizador aquilo que será uma primeira solução viável do problema. Por outro lado, se o utilizador optar pela criação de rotas sementes, terá a possibilidade de inserir os clientes que pretende que cada rota individual englobe, indicando também a ordenação dos mesmos. Depois da criação das rotas semente, a aplicação irá inserir os clientes que não foram inseridos pelo utilizador nas rotas cujo custo seja o mais barato.

Depois destes passos, o utilizador terá a possibilidade de editar todas as rotas geradas, movendo clientes de uma rota para outra, ou simplesmente de os retirar de uma rota, e, por fim, pedir novamente à aplicação de gere uma nova rota. Este processo poderá ser feito tantas vezes quantas o utilizador queira.

O passo que concluirá o processo de criação de rotas será feito pela aplicação, sem contar com a ajuda do utilizador. A aplicação irá, através de algoritmos, proceder à otimização das rotas, movimentando clientes de rota para rota se os custos das mesmas diminuírem com essas trocas. O resultado será apresentado ao utilizador, que poderá, novamente, editar manualmente a rota, procedendo novamente a este ponto final da criação das rotas.

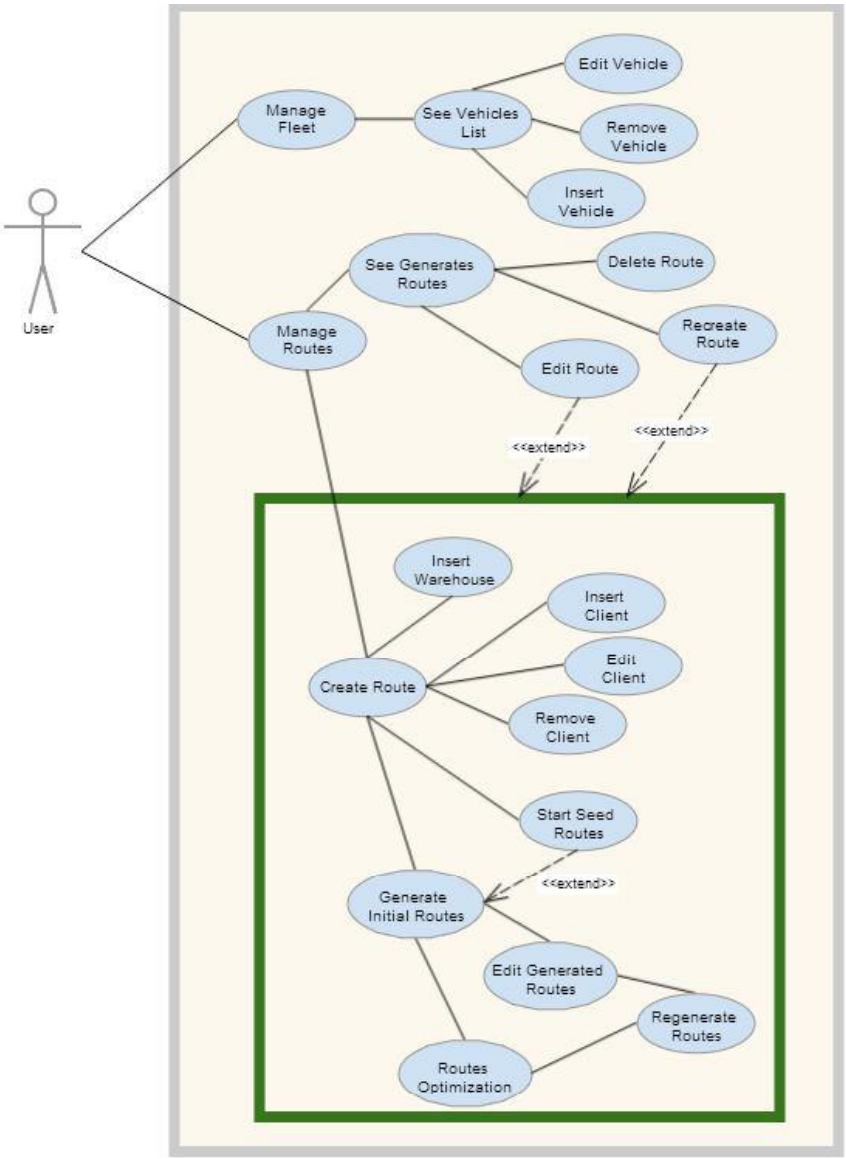


Figura 7 – Diagrama de Casos de Uso da aplicação

5.4. Diagrama de Classes da aplicação

Tendo em conta o princípio de funcionamento do método visual interativo e os casos de uso identificados, foi definido o Diagrama de Classes da aplicação, apresentado na figura 8, o qual representa os objetos a considerar e as relações entre os mesmos.

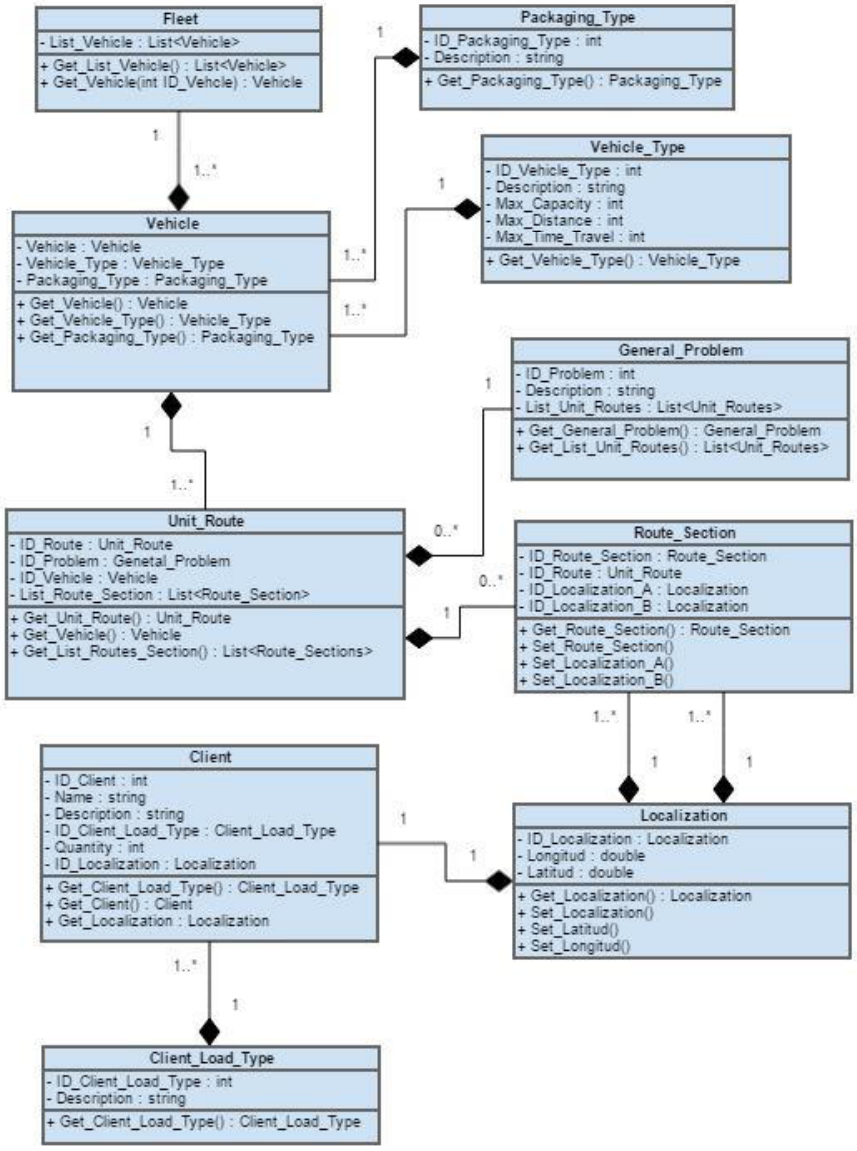


Figura 8 – Diagrama de Classes

Por um lado, teremos a classe Fleet, que conterá uma lista de veículos que terá dois métodos, um que devolverá essa mesma lista e outro que devolverá apenas informação de um veículo pedido. A informação desses mesmos veículos estará guardada na classe Vehicle, que terá todos os atributos de cada veículo, bem como informação do tipo de acondicionamento (relacionado com a classe Packaging_Type), e do tipo de veículo (classe Vehicle_Type, que conterá informações da capacidade, distância e tempo máximos de cada tipo de veículo). A classe Vehicle possuirá métodos que permitirão

obter toda a informação de um dado veículo, bem como do respetivo tipo de veículo e de acondicionamento.

Por outro lado teremos os clientes, identificados pela classe Client, que terá atributos referentes ao cliente (desde nome, à quantidade de produtos do cliente) e métodos que permitirão obter toda a informação do cliente, do tipo de carga do produto do cliente (relacionado com a classe Cliente_Load_Type que conterá essa informação) e da sua localização, que estará relacionado com a classe Localization, responsável por armazenar os dados da localização do cliente com métodos que permitirão obter tanto a latitude como a longitude da posição de cada cliente.

Por fim, teremos as rotas. Cada problema será representado pela classe General_Problem, que consistirá numa lista de rotas unitárias (classe Unit_Route) e conterá um método que devolverá a lista de todas essas rotas unitárias. Uma rota unitária será o caminho que um determinado veículo irá fazer para responder aos pedidos dos clientes (inseridos nessa rota). Assim sendo, a classe Unit_Route será constituída por atributos relativos ao veículo, ao problema geral, e ainda por uma lista de secções de rotas (classe Route_Section). Estas secções de rotas consistem em dois pontos, referentes a dois clientes diferentes, que representam o percurso que um veículo irá fazer, que irá de um ponto A para um ponto B. Nesta classe existirão métodos que irão fornecer dados referentes à localização de cada cliente e da rota unitária da qual faz parte.

5.5. Base de Dados da aplicação

5.5.1. Diagrama do Modelo Entidade-Relacionamento

A figura 9 apresenta o Diagram do Modelo Entidade-Relacionamento da base de dados da aplicação.

Para cada Veículo (Vehicle), será preciso ter em conta o Tipo de Veículo (Vehicle_Type) a que pertence, que pode condicionar variáveis como a distância máxima que poderá percorrer, ou a capacidade máxima do mesmo. Para além disso, é importante também ter em conta o tipo de acondicionamento da carga do veículo, que poderá ser usada para separar os veículos em determinados problemas.

Cada cliente terá um Tipo de Carga (Client_Load_Type), que se referirá, tal como o nome indica, ao tipo de carga que será entregue nesse mesmo cliente (relacionado, assim, com o Tipo de Acondicionamento (Packaging_Type) de cada veículo), bem como uma quantidade (referente à quantidade de carga para o cliente) e uma localização (Localization), onde irá constar a longitude e latitude do mesmo.

A solução final (que será a composição de rotas geradas) irá representar uma solução do Problema (General_Problem), para o qual existirão Rotas Unitárias (Unit_Route) que representam a rota definida para cada veículo. Uma Rota Unitária será composta por uma sequência de Troços de Rota (Route_Section), que representam a ligação entre os clientes que serão englobados em cada Rota Unitária, clientes esses que aqui serão representados pelo identificador da sua localização.

Uma vez que o módulo de gestão de frotas não foi implementado na solução final, a classe Fleet não é visível no diagrama da Figura 9.

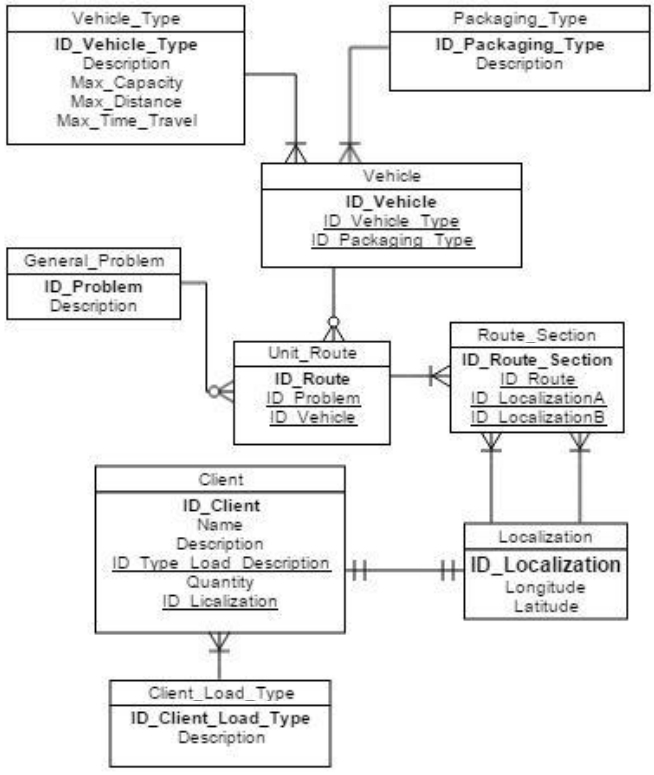


Figura 9 – Diagrama do Modelo de Entidade Relacionamento

5.5.2. Dicionário de Dados

Serão de seguida descritas todas as tabelas da base de dados e o dicionário da mesma, com os respetivos atributos.

A tabela **Vehicle** guarda toda a informação que diz respeito aos veículos utilizados nas rotas. Através desta tabela é possível saber o tipo de veículo e o tipo de acondicionamento de carga do mesmo.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Vehicle (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada veículo [1, 99999999]
ID_Vehicle_Type (FK)	Inteiro	8	Sim	Identifica o tipo de veículo [1, 99999999]
ID_Packaging_Type (FK)	Inteiro	8	Sim	Identifica o tipo e acondicionamento de carga do veículo [1, 99999999]

Tabela 4 – Dicionário de Dados – Vehicle

A tabela **Vehicle_Type** contém informações sobre os possíveis tipos de veículo, tal como capacidade, tempo de viagem e distância máximas para cada viagem.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Vehicle_Type (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada tipo de veículo [1, 99999999]
Description	varchar	50	Sim	Descrição do tipo de veículo [AZ az]
Max_Capacity	Inteiro	8	Sim	Identifica a capacidade máxima dos veículos deste tipo [1, 99999999]
Max_Distance	Inteiro	8	Sim	Identifica a distância máxima que veículos deste tipo podem percorrer numa viagem [1, 99999999]
Max_Time_Travel	Inteiro	8	Sim	Identifica o tempo máximo de viagem durante o qual veículos deste tipo podem circular [1, 99999999]

Tabela 5 – Dicionário de Dados – Vehicle Type

Na tabela **Packaging_Type** ficam guardadas as informações do tipo de acondicionamento de produtos de cada veículo.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Packaging_Type (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada tipo de acondicionamento [1, 99999999]
Description	varchar	50	Sim	Descrição do tipo de acondicionamento [AZ az]

Tabela 6 – Dicionário de Dados – Packaging Type

A tabela **General_Problem** é a responsável por guardar cada problema que o utilizador iniciar para que mais tarde e através da descrição, possa visualizar novamente as rotas geradas.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Problem (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada problema [1, 99999999]
Description	varchar	50	Sim	Descrição do problema [AZ az]

Tabela 7 – Dicionário de Dados – General Problem

A tabela **Cliente_Load_Type**, à semelhança da tabela **Packaging_Type**, guarda o tipo de acondicionamento da carga a transportar, mas, nesta tabela, para o cliente.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Client_Load_Type (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada tipo de carga dos clientes [1, 99999999]
Description	varchar	50	Sim	Descrição do tipo de carga dos clientes [AZ az]

Tabela 8 – Dicionário de Dados – Client Load Type

A tabela **Localization** guarda a informação das coordenadas (latitude e longitude) de cada cliente.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Localization (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada localização de clientes [1, 99999999]
Longitude	Decimal	(12,9)	Sim	Identifica a longitude do cliente.
Latitude	Decimal	(12,9)	Sim	Identifica a longitude do cliente.

Tabela 9 – Dicionário de Dados – Localization

A tabela **Client** é a tabela responsável por guardar toda a informação relativa aos clientes. Essa informação vai desde nome e descrição até ao tipo de carga a transportar, a sua localização, e a demanda a transportar.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Client (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada cliente [1, 99999999]
Name	varchar	50	Sim	Identifica o nome do cliente [AZ az]
Description	varchar	250	Não	Descrição individual do cliente [AZ az]
ID_Load_Type_Description (FK)	Inteiro	8	Sim	Identifica o tipo de carga do cliente [1, 99999999]
Quantity	Inteiro	8	Sim	Identifica a quantidade de carga a transportar [1, 99999999]
ID_Localization (FK)	Inteiro	8	Sim	Identifica a localização do cliente [1, 99999999]

Tabela 10 – Dicionário de Dados – Client

A tabela responsável por guardar informações de cada rota constituinte do problema é a tabela **Unit_Route**. A partir desta é também possível obter informações do veículo que irá servir a rota.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Route (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada rota [1, 99999999]
ID_Problem (FK)	Inteiro	8	Sim	Identifica o problema a que a rota pertence [1, 99999999]
ID_Vehicle (FK)	Inteiro	8	Sim	Identifica o veículo que irá responder à rota [1, 99999999]

Tabela 11 – Dicionário de Dados – Unit ROute

A tabela **Route_Section**, à semelhança da tabela **Unit_Route**, guarda também informações sobre as rotas, mas aqui é guardada a informação da rota entre cada par de pontos constituintes da rota final, isto é, cada trajeto entre dois diferentes clientes.

Campo	Tipo	Tamanho	Obrigatório	Descrição
ID_Route_Section (PK)	Inteiro	8	Sim	Número sequencial que identifica univocamente cada secção da rota [1, 99999999]
ID_Rota (FK)	Inteiro	8	Sim	Identifica a rota a que a secção de rota pertence [1, 99999999]
ID_LocalizationA (FK)	Inteiro	8	Sim	Identifica a localização do primeiro ponto da secção de rota [1, 99999999]
ID_LocalizationB (FK)	Inteiro	8	Sim	Identifica a localização do segundo ponto da secção de rota [1, 99999999]

Tabela 12 – Dicionário de Dados – Route Section

5.6. Implementação das funcionalidades da Aplicação

Nas próximas seções será feita uma descrição de todas as funcionalidades da aplicação móvel, bem como a descrição de cada uma das Activities e da navegação entre elas. Uma vez que não existe qualquer tipo de perfil de utilizador, todas as funcionalidades descritas são acessíveis ao utilizador do dispositivo onde a aplicação possa estar instalada. De forma a melhorar a usabilidade e interatividade entre utilizador e aplicação, sempre que o utilizador procede a uma operação na criação / gestão das suas rotas, são-lhe apresentadas descrições de qual o passo que deve executar de seguida.

5.6.1. Sistema de navegação da aplicação

A aplicação é constituída por três Activities, que serão pormenorizadamente descritas mais à frente. Nesta secção será apenas feita uma pequena descrição do sistema de navegação entre essas mesmas Activities.

A primeira, definida como página principal, tem as seguintes opções:

- Criar um novo problema – opção que reencaminha o utilizador para a página de criação de um novo problema;
- Ver lista de problemas – opção que reencaminha o utilizador para uma página onde este poderá ver a lista de problemas já criados.

Na Activity que mostra a lista de problemas, o utilizador terá informação do nome de cada problema, bem como da quantidade de clientes em cada um deles. A partir desta mesma lista poderá seguir para a edição de cada um dos problemas.

Independentemente de o utilizador pretender fazer a edição de um problema, ou proceder à criação de um novo, essas operações serão executadas numa única Activity. É nela que se encontram todas as funcionalidades propriamente ditas que deram origem à criação deste projeto.

5.6.2. Integração com o serviço Google Maps

No que se refere ao uso de dados georreferenciados, a aplicação foi desenvolvida com base no serviço Google Maps e respetiva API, que disponibiliza a cada aplicação um número máximo de pedidos diários. Após a realização de vários testes, chegou-se à conclusão que no caso de utilização da aplicação em casos reais, o número

de pedidos disponibilizados pelo serviço não seriam suficientes. Assim, foi necessário proceder ao desenvolvimento de um WebService que ficaria encarregue por processar os pedidos de todos os utilizadores. Com utilização deste WebService as restrições de pedidos ao Google Maps diminuíam, uma vez que se deixava de fazer pedidos padrão através da API (que tem as restrições a cima referidas) e passava a fazer pedidos diretamente ao Google Maps através de um URL construído com as coordenadas dos clientes, que será descrito mais à frente. Independentemente disso, a própria estrutura da aplicação é fortemente baseada nas funcionalidades do serviço Google Maps, mais precisamente na versão 2 da Google Maps API, que era a utilizada na altura do início do desenvolvimento da presente aplicação. Atualmente a mais recente versão da API é a versão 3.

De seguida será feita uma breve descrição da API v2 e dos passos seguidos para a sua integração na aplicação desenvolvida.

A API Google Maps v2 permite ao programador disponibilizar mapas, marcas, procurar localidades por nome ou coordenadas, encontrar locais dentro de uma cidade (restauração, cultura, hotéis), entre outros.

Esta versão acrescentou essencialmente, em relação à versão anterior, a divisão dos mapas em "vetor de azulejos", de forma a aumentar a velocidade de representação dos mapas com uma menor largura de banda, acrescentando ainda a visão dos mapas em 3D.

Para poder usufruir das funcionalidades do Google Maps na aplicação foi necessário obter um Certificado Fingerprint (MD5) do computador onde a aplicação seria desenvolvida, sendo único para cada computador, tal como é possível ver na seguinte imagem.

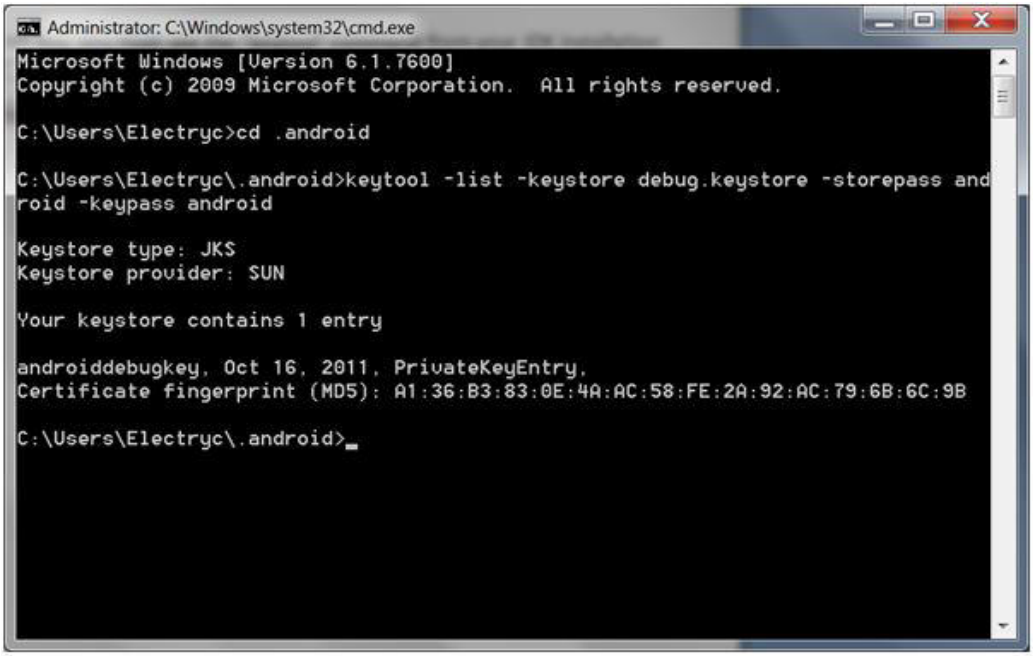


Figura 10 – Obtenção do certificado MD5 do computador

Após obter o certificado foi necessário aceder à plataforma Google Developers Console para fazer o registo do certificado e obter a chave a ser utilizada para autenticar a API Google Maps. Aqui foi necessário ir a "APIs e autenticação – Credenciais", tal como se pode ver na Figura 12 e de seguida criar uma nova chave, introduzindo o certificado FingerPrint, seguido do namespace da aplicação onde a chave será utilizada, como é visível na Figura 13.

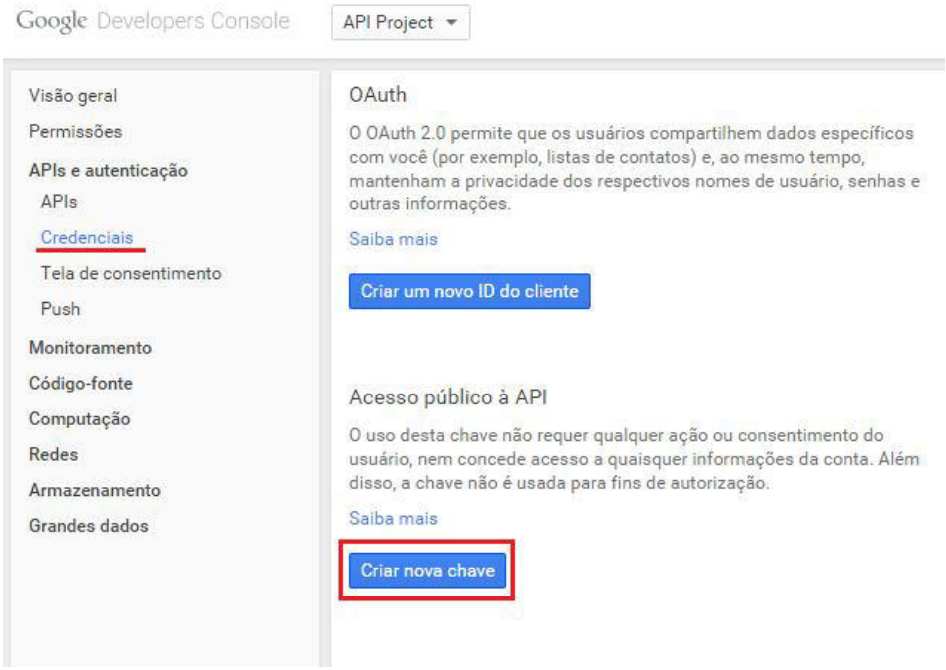


Figura 11 – Google Developers Console – Credenciais

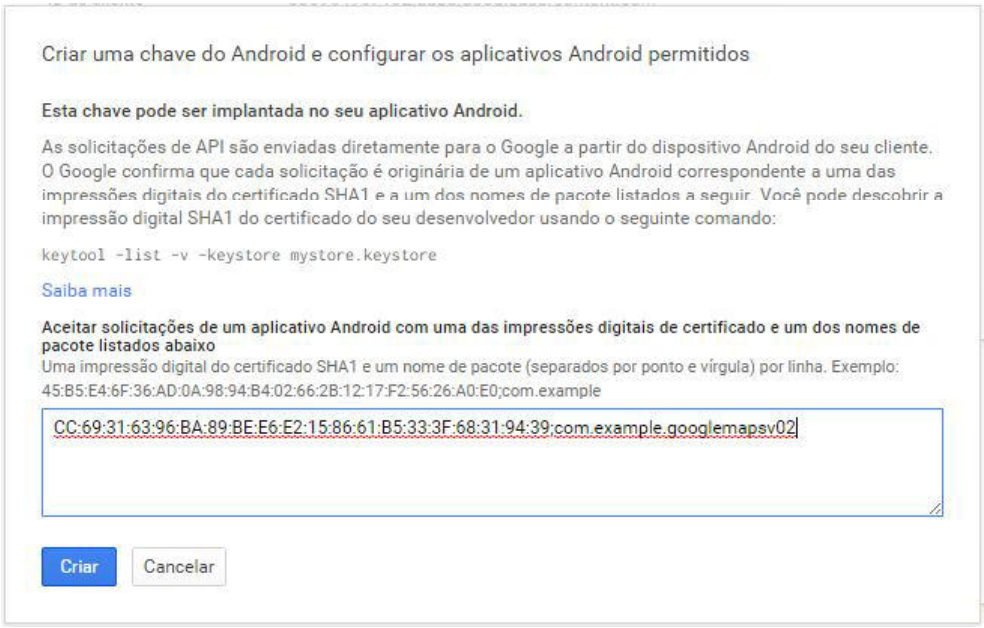


Figura 12 – Google Developers Console – Obter nova key

A chave atribuída foi a que se pode ver na Figura 14.

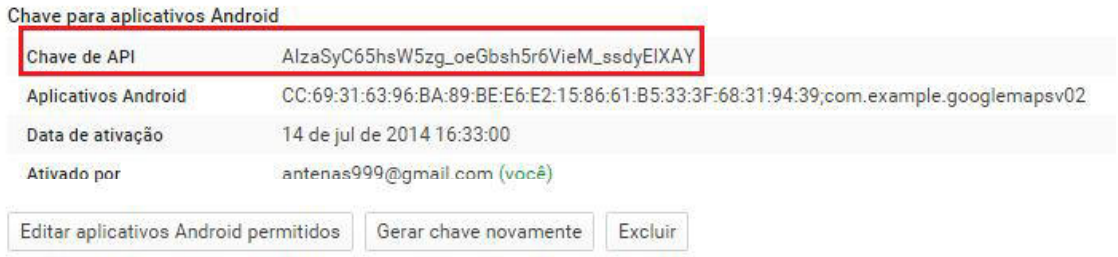


Figura 13 – Google Developers Console – Key atribuída

Para poder integrar os mapas da Google Maps na aplicação foi necessário criar um projeto Android com a Build SDK Google API's, na altura com a versão 14 da API.

Depois disso foi apenas necessário inserir um componente do tipo Fragment, relativo ao mapa, dentro do layout da Activity onde se pretende visualizar o mapa. O código do componente é apresentado a seguir.

```
<fragment
   android:id="@+id/mapview"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   class="com.google.android.gms.maps.MapFragment" />
```

Para que a aplicação possa aceder ao serviço Google Maps teve ainda que se colocar a chave anteriormente gerada no ficheiro Manifest do projeto da aplicação., Isso pode ser feito através de um componente do tipo Meta-data, como ilustrado no código a seguir.

```
<application
   ...
   <meta-data
        android:name="com.google.android.maps.v2.API_KEY"
        android:value="AIzaSyC65hsW5zg_oeGbsh5r6VieM_ssdyEIXAY" />
</application>
```

Todas as funcionalidades usufruídas da API Google Maps serão descritas em secções posteriores. Será igualmente feita uma descrição detalhada do ficheiro Manifest da aplicação.

5.6.3. Integração com o WebService

Em traços gerais, o Web Service recebe todos os pares de pontos para realizar o pedido ao Google Maps através de um URL pré-construído, e recebe do mesmo uma resposta em formato JSON com a informação de direções, tempo e distância de viagem

(para além de outras informações), entre esses mesmos pares de pontos. Este processo é repetido para todos os pares de pontos enviados pela aplicação móvel, e é posteriormente construído um objeto com as informações necessárias que é enviado como resposta para a aplicação.

Ao contrário do número máximo de pedidos possíveis de realizar através da API do Google (que é possível de aumentar através de pagamentos mensais), os pedidos feitos deste modo através do Web Service têm um limite superior em relação aos da API, o que permite fazer testes com um maior número de pontos.

Para além disto, o Web Service vai ser ainda responsável pelo envio de dados e rotas da aplicação móvel, sendo que esta comunica igualmente com o Web Service e este reencaminha os dados para um ficheiro de folha de cálculo do Google Docs, por forma a ter os dados de todos os clientes e rotas para possíveis comparações e análises.

5.6.3.1. Métodos do WebService

Nesta secção será feita uma descrição dos métodos desenvolvidos do lado do WebService.

Método *GetStringWSPoints*

A função deste método é receber os pontos através de uma string concatenada, separando os pontos por um indicador ":" e cada um dos conjuntos de coordenadas pelo indicador ";",como exemplificdo a seguir:

41.1234;7.1234:42.4567;8.4567.

Neste método é feita uma separação de todos os pontos executando a seguinte instrução:

```
Dim listAlPoints As String() = listAlPointss.Split(":")
```

De de seguida é construída uma matriz com todas as coordenadas através do seguinte código:

```
For i As Integer = 0 To listAlPoints.Count - 2
    Dim listLatLong As String() = listAlPoints.ElementAt(i).Split(";")
```

```
            matrizPoints(i, 0) = listLatLong(0)
            matrizPoints(i, 1) = listLatLong(1)
        Next
```

Após a criação da matriz de coordenadas, procede-se aos pedidos ao serviço Google Maps para receção da informação de cada par de pontos. Esse pedido é feito através da criação de um URL, no seguinte formato:

"http://maps.googleapis.com/maps/api/directions/json?origin=41,1234, 7.1234&destination=42.4567, 8.4567&sensor=false"

A resposta a este pedido é obtida em formato JSon, que contém todas as informações do caminho entre cada par de pontos, das quais, para o nosso caso, apenas interessam as informações relativas a distância e tempo de viagem e o conjunto de coordenadas necessários para traçar a rota entre ambos os pontos. Apresenta-se a seguir a resposta JSon (resposta completa no Anexo A).


```
{
  "status": "OK",
  "routes": [ {
    "summary": "I-40 W",
    "legs": [ {
      "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"
    } ],
    "overview_polyline": {
      "points":
"a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\\ltNtqf@l{Yd_Fblh@rxo@b}@xxSfytAblk@xxaB
eJxlcBb~t@zbh@jc|Bx}C`rv@rw|@rlhA~dVzeo@vrSnc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo
@`cFp~xBc`Hk@nurDznmFfwMbwz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_
Wr|Efc\\x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg@|kHntyArpb@bijCk_Kv~eGyqTj_|@`uV`k|DcsNdwx
```

```
Aott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy@`omFvaErfo@igQxnlApqGze~AsyRzrjAb__@
ftyB}pIlo_BflmA~yQftNboWzoAlzp@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdmY_hLxiZd~Xt
aBndgC"
    },
    }
  } ]
}
```

Para obter cada um dos nós desejados, faz-se uma desserialização do objeto JSon com o código que se segue.

```
Dim distance As Integer = deserialize("routes")(0)("legs")(0)("distance")("value")
```

O conjunto de coordenadas necessárias para traçar a rota, correspondente ao nó "overview_polyline", é enviada para a aplicação, no formato recebido, sendo posteriormente convertido na aplicação para pontos de coordenadas. Todos estes dados são concatenados numa string e devolvidos para a aplicação.

O código completo implementado neste método pode ser encontrado no Anexo B.

Método *exportRoutes*

Para exportar os dados das rotas foi criado um formulário no Google Docs, sendo que a aplicação móvel envia o conjunto de dados para o WebService no formato string, e este procede à sua separação e posterior reencaminhamento para esse mesmo formulário através deste método. Este formulário recebe como parâmetros de entrada a data de envio, o tempo total do pedido (em segundos), e uma string concatenada que pode conter ou o conjunto de todos os pontos, ou os pontos constituintes de cada rota. Para os dados serem enviados para o formulário é igualmente construído um URL, no seguinte formato.

"docs.google.com/forms/d/14Yd0NfGR7efa8DQAOLhqGHAOyZTPwBR3Caj3eo7p0Qg/formResponse?entry.2054938567=2015/04/18&entry.1827975684=30&entry.1852957370=[Conjunto de pontos ou pontos de cada rota]"

O método completo implementado encontra-se no Anexo C.

5.6.4. Acesso aos Dados

A parte do acesso aos dados e utilização dos mesmos na aplicação foi estruturada em duas diferentes camadas. Essas duas camadas são as seguintes:

- ***Entity*** – definem os objetos, essencialmente da base de dados, que contêm todos os parâmetros de cada objeto, bem como os respetivos métodos Get e Set. Para além dos objetos da base de dados, foram ainda criadas outras entidades para facilitar o acesso e a manipulação dos dados, bem como a respetiva apresentação dos mesmos ao utilizador.
- ***Data*** – camada que define o acesso à base de dados, onde estão os métodos para cada objeto da mesma que permitem a inserção de novos dados, atualização e remoção dos mesmos, entre outras funcionalidades.

De seguida é feita uma breve descrição destas duas camadas.

Entity

Nesta camada, tal como referido anteriormente, foram criadas entidades para acesso direto aos dados da base de dados, tendo em conta o diagrama de classes anteriormente descrito, e outras para ajudar a manipulação e apresentação dos mesmos ao utilizador.

Para o acesso à base de dados foram criadas as seguintes entidades, que contêm os parâmetros indicados do modelo de dados:

- ClientEntity;
- ClientLoadTypeEntity;
- GeneralProblemEntity;
- LocalizationEntity;
- PackagingTypeEntity;
- RouteSectionEntity;
- UnitRouteEntity;
- VehicleEntity;
- VehicleTypeEntity;

Ainda para o acesso à base de dados foi criada uma outra entidade, objectToRoute, que não faz parte do modelo de dados inicialmente desenhado. Esta entidade permite guardar informações referentes a cada par de pontos, tais como a localização de cada um desses pontos (com referência à entidade LocalizationEntity), distância e tempo de viagem entre o par de pontos, o polígono (que é uma string codificada que representa o caminho a percorrer, que será descrito numa secção mais à frente), entre outros.

Para além de todas estas entidades, e tal como anteriormente referido, foram criadas outras para facilitar a manipulação dos dados, que são:

- ClientsToEachRoute;
- CostRoute;
- routeOfTheMap;

A estrutura de todas entidades é muito semelhante. A principal diferença é encontrada nas entidades cujo objetivo não é o acesso aos dados da base de dados, como é o caso da ClientsToEachRoute. De seguida será feita uma breve descrição de uma das entidades da base de dados (tendo sido selecionada para a demonstração a entidade ClientEntity) e da ClientsToEachRoute.

Entity ClientEntity

No início de cada uma das entidades existe uma secção para a inicialização dos parâmetros.

Logo a seguir, existe uma secção para o construtor da entidade. É normal nesta secção existir mais do que um construtor, dependendo daquilo que programador pretende. No caso da presente aplicação, existem pelo menos dois construtores diferentes, sendo um utilizado para inicializar um objeto vazio, e outro que recebe os valores de cada um dos parâmetros.

Por fim podemos encontrar os métodos Get e Set, que são utilizados para obter o valor do parâmetro e atribuir valores a um parâmetro, respetivamente. O código completo desta classe pode ser visto no Anexo D.

Entity ClientsToEachRoute

Este objeto foi criado com o intuito de ajudar a manipulação de dados na aplicação de alguns algoritmos, sendo que aqui são guardados dados relativos à posição do cliente na lista de clientes do problema atual, e uma lista de objetos que contêm informações sobre a rota e o respetivo custo de adição do atual cliente nessa mesma rota.

Este objeto é em tudo semelhante à entidade anteriormente descrita, sendo que contém a declaração das variáveis e os respetivos Get e Set, mas não contém um construtor. Uma outra diferença entre as duas entidades é que esta, no início da classe, contem a implementação do método Comparable. Este método permite, de uma forma simplificada, fazer a ordenação de uma lista. Para isso é apenas preciso, após preencher a lista, fazer um sort, da seguinte forma:

```
Collections.sort(listClientsToEachRoute);
```

Na classe do objeto está a implementação do método que é invocado por esta chamada, que a única coisa que faz é comparar o elemento atual com o elemento seguinte. No presente caso, existem duas formas de comparação:

- Na primeira, o objetivo passa por comparar o número de elementos de cada lista, devolvendo "1" no caso de o tamanho da primeira lista ser maior que o da segunda, "0" em caso contrário, "-1" no caso de ambos terem o mesmo tamanho;
- Na segunda, o objetivo passa por ordenar em relação à diferença dos dois menores custos de cada elemento da lista. Esta segunda opção de ordenação é utilizada num dos algoritmos de otimização e construção das rotas, que será descrito mais à frente.

De seguida pode-se ver o método implementado, em pseudocódigo, onde é possível ver as duas diferentes formas de ordenação (código completo no Anexo E). No segundo caso, é invocado um método, com o nome getLowerCostDiff, cujo objetivo é devolver a diferença dos dois menores custos.

```
Algoritmo: compareTo(another)
Objetivo: Fazer dois tipos de comparação.
Comparação 1: Comparar dois objetos do tipo ClientsToEachRoute, tendo em conta o número de rotas
onde pode ser adicionado.
Comparação 2: Comparar dois objetos do tipo ClientsToEachRoute tendo em conta a diferença entre os
dois menores custos de inserção de cada um dos clientes.
Constantes:
        orderBySize (Booleano) - indica se é para aplicar a Comparação 1 ou a Comparação 2 (0 ou 1)
```

```
        NRC1 (Inteiro T2) - indica o número de rotas onde o cliente atual pode ser adicionado
        NRC2 (Inteiro T2) - indica o número de rotas onde o cliente a ser comparado pode ser
adicionado
        DCC1 (Real T6.3) - indica a diferença dos dois menores custos das rotas onde o cliente atual
pode ser adicionado
        DCC2 (Real T6.3) - indica a diferença dos dois menores custos das rotas onde o cliente a ser
comparado pode ser adicionado
Parâmetro de saída:
        resultado (Inteiro T2) - valor utilizado para a ordenação (-1, 0 ou 1)
                1 - indica que o objeto atual é maior que o objeto de comparação
                0 - indica que o objeto atual é igual ao objeto de comparação
                -1 - indica que o objeto atual é menor que o objeto de comparação
Data: 2015-05-12
Autor: Ricardo Antunes
Versão: 1.0
Início
        SE orderBySize = 1 ENTÃO
                SE NRC1 > NRC2 ENTÃO
                        RETORNA 1
                SENÃO SE NRC1 = NRC2
                        RETORNA 0
                SENÃO
                        RETORNA -1
                FIM SE
        SENÃO
                SE DCC1 > DCC2 ENTÃO
                        RETORNA 1
                SENÃO SE DCC1 = DCC2
                        RETORNA 0
                SENÃO
                        RETORNA -1
                FIM SE
        FIM SE
FIM.
```

Data

Nesta camada foram criadas todas as classes com os respetivos métodos que permitem obter e inserir dados na base de dados. Para isso foram criadas as seguintes classes:

- ClientData;
- ClientLoadTypeData;
- GeneralProblemData;
- LocalizationData;
- ObjectToRouteData;
- PackagingTypeData;

- RouteSectionData;
- UnitRouteData;
- VehicleData;
- VehicleTypeData;

Á semelhança das entidades, a estrutura de todas estas classes é muito semelhante. Umas têm mais métodos do que outras, consoante aquilo que foi necessário fazer com cada um dos objetos, mas, no geral, todas elas seguem o mesmo padrão. Para demonstração, foi selecionada a classe ClientData.

Data ClientDate

No início de cada classe, a primeira secção é, à semelhança das entidades, a secção de inicialização das variáveis necessárias para o uso da classe. Neste caso, existe a inicialização do nome da base de dados, da tabela que se irá aceder, dos respetivos campos dessa tabela, e do respetivo objeto.

Tal como referido, cada classe da camada Data tem um número de métodos variáveis, consoante aquilo que for necessário fazer com cada objeto em relação à base de dados e manipulação de dados. No caso desta classe, existem os seguintes métodos:

- addClient – que recebe como parâmetro uma entidade do tipo ClientEntity e o insere na base de dados.
- addClientWithId – em tudo igual ao método anterior, mas com a diferença de que enquanto o anterior apenas insere o cliente na base de dados, este devolve o id do cliente inserido.
- getClient – recebe como parâmetro o ID do cliente que se pretende obter todas as informações da base de dados. Este método devolve um objeto do tipo ClientEntity.
- getAllClients – devolve uma lista com todos os clientes existentes na base de dados.
- deleteClient – recebe um objeto do tipo ClientEntity como parâmetro, e elimina esse cliente da base de dados.

Para efeitos de demonstração do acesso aos dados, de seguida encontra-se o método addClientWithId (todo o código desta classe no Anexo G). Detalhadamente, temos o seguinte:

- Inicialização da conexão à base de dados (SQLiteDatabase).
- Mapeamento dos campos do objeto recebido por parâmetro para as respetivas colunas da tabela, atribuídos ao ContentValues.
- Inserção do novo cliente na respetiva tabela. O método insert recebe três parâmetros, sendo eles o nome da tabela (em formato string), uma lista com os nomes das colunas que é permitido serem inseridos com valor nulo (no nosso caso, não é apresentada qualquer lista) e, por fim,

os valores a inserir. É possível ainda ver que a inserção deste novo cliente é igualada a uma variável do tipo long, sendo que será nela guardado o ID do último cliente adicionado.

- Por fim, a conexão à base de dados é fechada, e é devolvido o valor do ID.

```
public int addClientWithId(ClientEntity client) {
        SQLiteDatabase db = this.getWritableDatabase();

        ContentValues values = new ContentValues();
        values.put(C_NAME, client.getName());
        values.put(C_DESCRIPTION, client.getDescription());
        values.put(C_ID_TYPE_LOAD, client.getIdTypeLoad());
        values.put(C_QUANTITY, client.getQuantity());

        long idClient = db.insert(TABLE_CLIENT, null, values);

        db.close();

        return (int)idClient;
}
```

5.6.5. As *Activities* da aplicação

Descrevem-se a seguir as Activities que compõem a aplicação desenvolvida.

5.6.5.1. Activity Menu

Esta é a Activity que o utilizador vê assim que abre a aplicação. Foram projetados três menus, sendo que o último (Options) não foi, por motivos de tempo, totalmente desenvolvido tal como esperado inicialmente. O menu principal pode ser visto na Figura 14.

Figura 14 – Route Me – Menu Principal

A partir daqui o utilizador pode ter acesso à Activity da lista de problemas já criados (descrita no seguinte subcapítulo), ou ao pop-up para criação de um novo problema, onde tem apenas que inserir o nome do problema que quer criar, tal como pode ser observado na Figura 15.

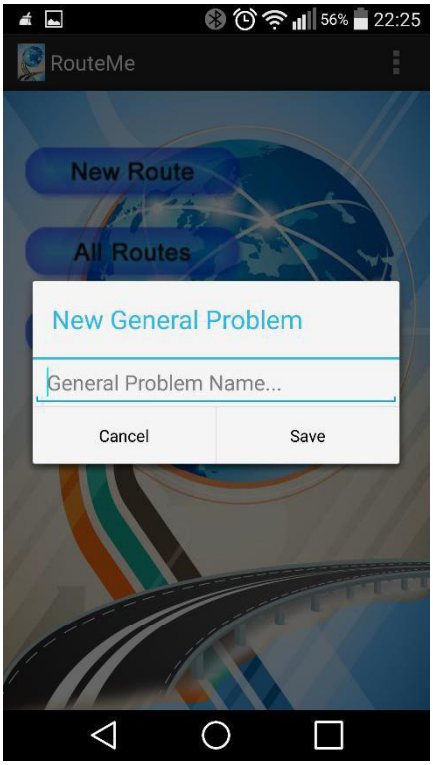


Figura 15 – Route Me – Criar novo problema

O nome do novo Problema tem que ter no mínimo quatro letras, sendo que sempre que o utilizador inserir um nome inválido, é-lhe devolvido o erro apresentado na Figura 16.

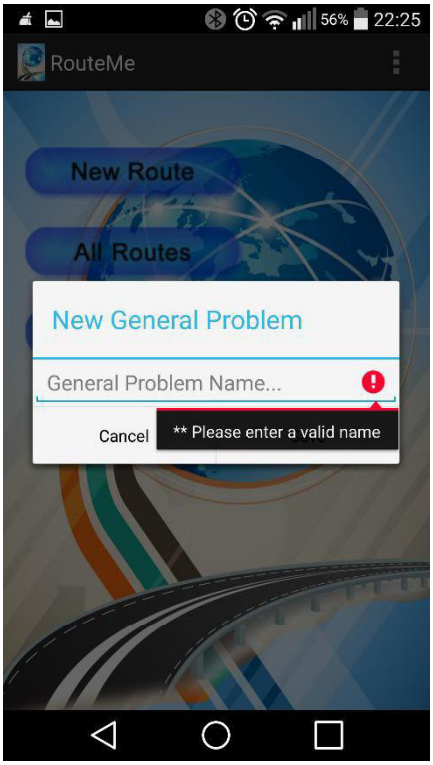


Figura 16 – Route Me – Validação do nome do novo problema

Assim que for inserido um nome válido, o Problema é criado e o utilizador é redirecionado para a Activity da criação e manipulação de rotas.

5.6.5.2. Activity Options

O objetivo principal desta Activity passou por permitir ao utilizador definir dois parâmetros, que são:

- Em primeiro lugar, e sendo o único a refletir alterações no aspeto visual da aplicação, uma opção para definir cores das rotas, oferecendo as seguintes possibilidades:
  - Todas as rotas com cores diferentes;
  - Cor da rota relativa à capacidade total utilizada da carga do veículo atribuído à rota;
- Em segundo lugar, uma opção de apoio à utilização da aplicação. Trata-se de uma simples caixa de texto onde é possível atualizar o endereço (URL) referente ao Webservice com o qual a aplicação irá comunicar.

Ambas as definições são muito simples e pouco desenvolvidas, mas permitem um grande suporte, evitando que sempre que alguma delas tenha que ser modificada seja necessária uma nova instalação da aplicação. Essas definições são visíveis na Figura 17.

Figura 17 – Route Me – Opções

5.6.5.3. Activity Lista de Problemas

É nesta Activity que o utilizador tem acesso a todos os problemas criados, através de uma lista, tal como mostra a Figura 18, onde é possível ver o nome de cada problema, acompanhado pelo número total de clientes existentes no mesmo. Selecionando um problema e clicando no botão que aparece no topo da página "Load", o utilizador é redirecionado para a Activity da criação e manipulação de rotas, onde continuará a partir do passo onde ficou da última vez que editou esse mesmo problema.

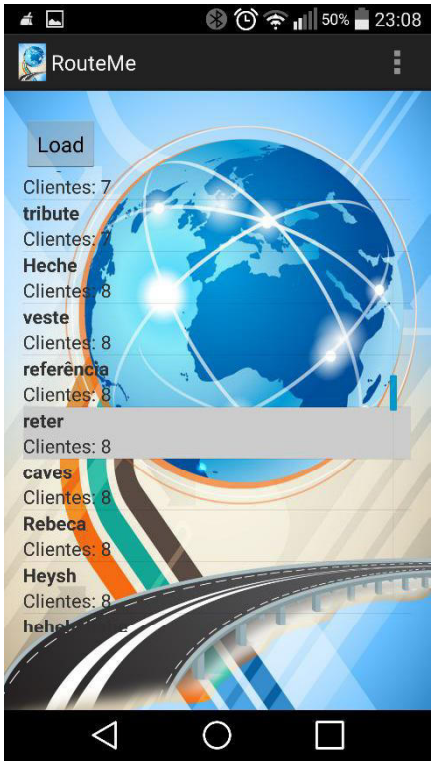


Figura 18 – Route Me – Lista de problemas criados

5.6.5.4. Activity Criação e Manipulação de Rotas

Esta é a Activity principal de toda a aplicação. É esta a responsável por todas as operações possíveis de executar com clientes e rotas, comunicação com o WebService, tratamento de dados, aplicação de algoritmos, entre outros que serão detalhadamente descritos ao longo deste subcapítulo.

A Figura 19 mostra o aspeto inicial desta Activity, um mapa centrado na posição atual do utilizador. Existe um botão no canto superior direito do mapa, que é utilizado para expandir o menu das operações possíveis de realizar nesta Activity.

Ao longo de toda a manipulação das rotas vão sendo apresentadas mensagens ao utilizador no fundo da Activity, de forma a auxiliar o utilizador nessa mesma manipulação, indicando qual o passo seguinte que deve executar. No caso da criação de um novo problema, a primeira mensagem que a parece é uma pequena informação com as coordenadas da posição atual, seguida de uma mensagem indicando que a criação do problema foi iniciada, e informando que a partir dali o utilizador pode começar com a inserção dos clientes através de simples cliques no mapa. Acrescenta ainda a informação de que o primeiro ponto inserido não será um cliente, mas sim o armazém de onde partirão

(e para onde regressarão) todos os veículos, o que, por outras palavras, corresponde ao ponto inicial e final de todas as rotas.

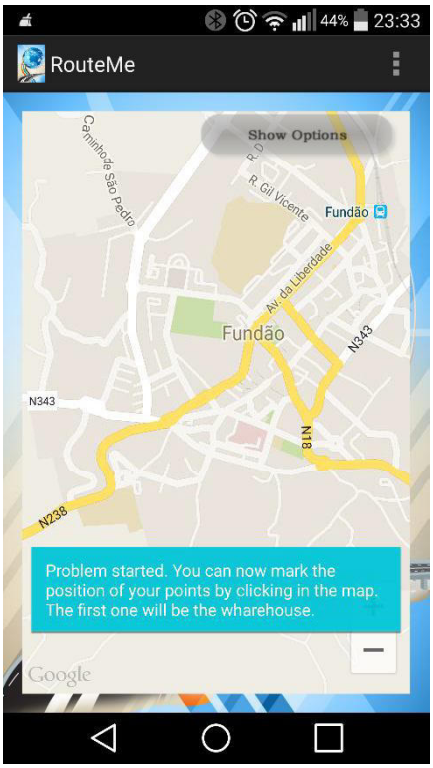


Figura 19 – Route Me – Visão inicial de um novo problema

Ao clicar no botão do menu, podemos ver todas as operações possíveis de exercer sobre as rotas, clientes e dados, tal como se pode ver na Figura 20. De uma forma muito resumida, as opções disponíveis e respetivos significados são as seguintes:

- Start Route – esta opção deve ser utilizada assim que o utilizador tiver inserido todos os clientes. É aqui que a aplicação contacta com o WebService, enviando-lhe os pares de coordenadas e recebendo, tal como referido anteriormente na explicação do WebService, os dados relativos a distância e tempo de viagem e a rota entre os dois pontos.
- Edit Route – esta operação só pode ser utilizada a partir do momento em que existem rotas desenhadas. O utilizador seleciona um cliente que ainda não tenha sido adicionado a qualquer rota e, de seguida, o cliente da rota desejada, a seguir ao qual será adicionado.

- Remove From Route – esta opção serve para remover um ponto da sua presente rota.
- Add New Points – ao clicar aqui, o utilizador regressa, de certo modo, ao inicio do seu problema, sendo que volta a poder adicionar clientes e terá que clicar novamente em "Start Route" para poder continuar a manipulação das rotas.
- Remove Points – através desta opção o utilizador pode, selecionando um cliente, eliminá-lo do mapa e da solução.
- Complete Route – esta opção ira finalizar as rotas pelo utilizador criadas, aplicando para isso os algoritmos desenvolvidos.
- Remove Multi Markers – o utilizador seleciona no mapa as duas extremidades de um retângulo, sendo que todos os clientes dentro dos limites desse mesmo retângulo serão eliminados das respetivas rotas.
- Check Client Info – ao acionar esta opção, e ao clicar em qualquer cliente, o utilizador poderá ver as informações relativas a esse mesmo cliente.
- Export Routes – envia os dados das rotas para o WebService para serem posteriormente enviados para um documento do Google.
- Back To Seed Routes – opção que permite ao utilizador desfazer tudo aquilo que a opção "Complete Route" fez e regressar às rotas inicialmente por ele desenhadas.
- Hide Options – volta a esconder todo o menu.

Figura 20 – Route Me – Menu de opções e ferramentas disponíveis

Após a criação do novo problema o utilizador deve, tal como anteriormente referido, começar a introduzir os clientes no mapa, bastando para isso um simples clique no mapa na localização desse mesmo cliente. A Figura 21 mostra o pop-up que é aberto após clicar no mapa, onde deverão ser introduzidos os dados do cliente, que são:

- Nome do cliente;
- Descrição;
- Tipo de acondicionamento da carga a transportar;
- Quantidade de carga a transportar.

Numa primeira fase, o tipo de carga não terá qualquer relevância para o desencadear das ações durante a manipulação dos clientes e rotas.

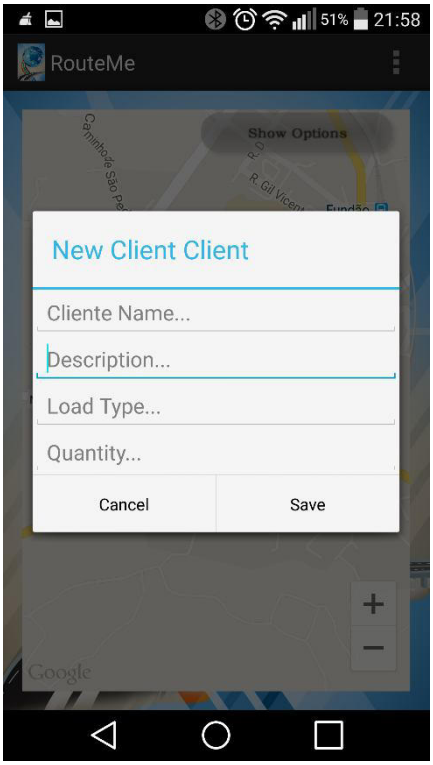


Figura 21 – Route Me – Pop-up de novo cliente

Por definição, os marcadores inseridos nos mapas são os típicos marcadores, à semelhança do que é possível ver na seguinte imagem.

Figura 22 – Marcadores genéricos do Google Maps

Por forma a ajudar o utilizador na compreensão de todos os marcadores inseridos no mapa, esses mesmos marcadores foram substituídos por outras imagens. No caso do armazém foi utilizada uma imagem apelativa ao efeito e, no caso dos clientes, foi

utilizado um círculo. Este círculo tem tamanho variável consoante os clientes já inseridos, sendo que é calculada a capacidade média de carga de todos os clientes e é aplicado um fator ao tamanho original da imagem por forma a ajudar o utilizador a identificar quais os clientes que têm uma menor quantidade de produtos a transportar. Essas diferenças entre tamanhos da imagem representativa dos clientes podem ser observadas na Figura 23.

Figura 23 – Route Me – Clientes inseridos no mapa

Sempre que um novo cliente é inserido é apresentada uma mensagem informando o utilizador que assim que terminar a inserção de clientes deve clicar no botão "Start Route" para dar início à comunicação com o WebService de forma a obter os dados necessários, tal como representado na Figura 24 e, após isso, tornar-se possível a iniciação do desenho das rotas sementes.

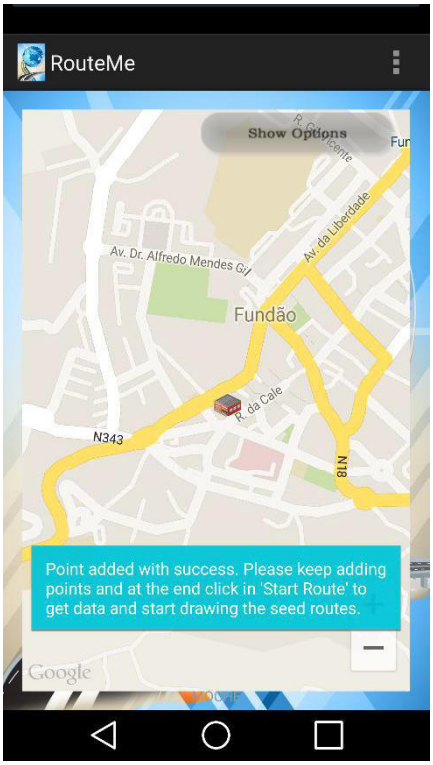


Figura 24 – Route Me – informações fornecidas pela aplicação ao utilizador

A qualquer altura o utilizador pode clicar em "Check Client Info" e, ao clicar em qualquer um dos clientes, poderá ver informação relativa ao mesmo, tal como apresentado na Figura 25.

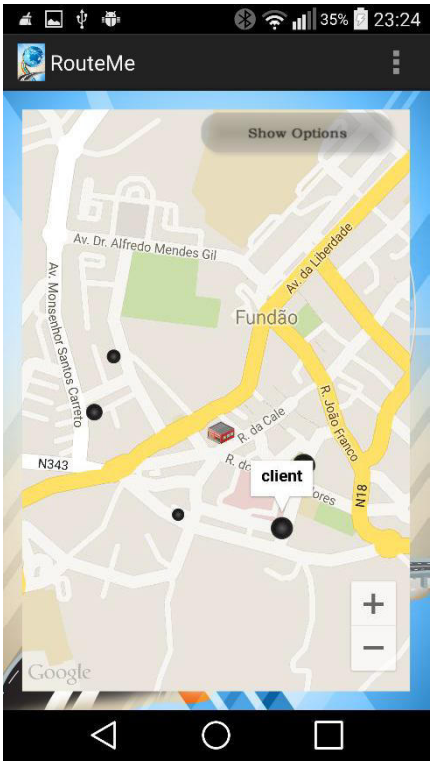


Figura 25 – Route Me – Informação de clientes inseridos

Após o utilizador inserir todos os pontos deve, tal como indicado, proceder a iniciação das rotas, contactando o Webservice para adquirir todos os dados necessários para poder traçar as rotas. Uma vez que se trata de um acesso a um Webservice, esta tarefa será uma tarefa assíncrona.

Os métodos assíncronos possuem quatro propriedades:

- Construtor: recebe as variáveis que se deseja utilizar. No caso da presente operação recebe a lista de marcadores do mapa e concatena-os numa string, separado pelo indicador ";";
- onPreExecute: referente ao método executado antes da tarefa principal. Normalmente é neste método que se instancia a mensagem que é mostrada ao utilizador enquanto a tarefa principal é executada. A Figura 26 mostra essa mensagem;

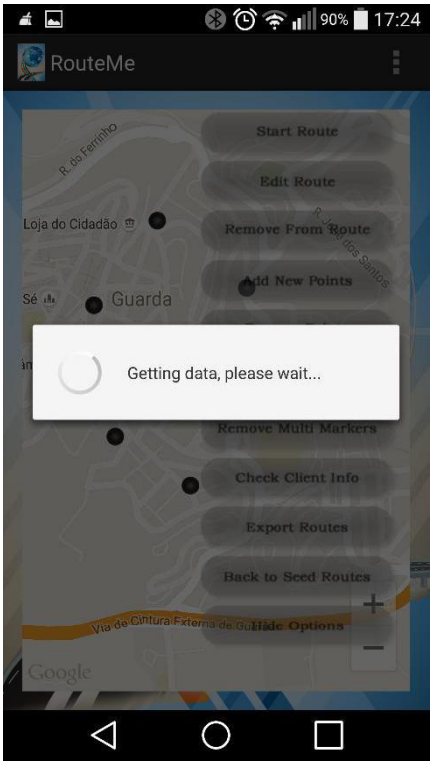


Figura 26 – Route Me – Obtenção de dados para a construção das rotas

- doInBackground: este método é executado exatamente a seguir ao onPreExecute, e é utilizado para executar processos em background que podem demorar bastante tempo. No caso concreto deste caso, é aqui que a aplicação comunica com o Webservice.
- onPostExecute: método executado assim que o anteriormente referido, doInBackground, termina. O resultado dos processos executados em background são passados para este método por parâmetro. Neste caso, o que é feito neste método é inserir nas várias listas e matrizes utilizadas para a criação das rotas os dados recebidos do Webservice e, no caso de existirem rotas já previamente traçadas, voltar a desenhá-las.

Este processo é repetido sempre que o utilizador clicar na operação de início das rotas ou sempre que mude um cliente de localização, uma vez que neste ultimo caso as rotas têm que ser redesenhadas.

Para mudar a localização de um cliente o utilizador tem apenas que clicar permanentemente no mesmo e arrastar para a posição desejada.

Após todos os dados dos clientes estarem disponíveis o utilizador pode começar a criação das suas rotas semente. Estas rotas semente são rotas que o utilizador cria, utilizando o seu conhecimento no tipo de problemas que está a tentar resolver. Por norma, nessas rotas são inseridos pontos críticos, seja pela sua localização, seja pelo tipo de demanda, ou até mesmo pela quantidade da demanda a transportar. Todas estas rotas têm início e fim no armazém, sendo este portanto o primeiro e ultimo ponto que o utilizador seleciona para cada rota. Sempre que inicia uma nova rota, a aplicação apresenta um pop-up (Figura 27) onde são introduzidos os dados do veículo que irá servir aquela rota, que são eles:

- Descrição do tipo de veículo;
- Capacidade máxima de transporte, com o mesmo significado que a demanda do cliente;
- Distância máxima, em metros, que o veículo pode percorrer;
- Tempo máximo, em segundos, que o veículo pode circular.

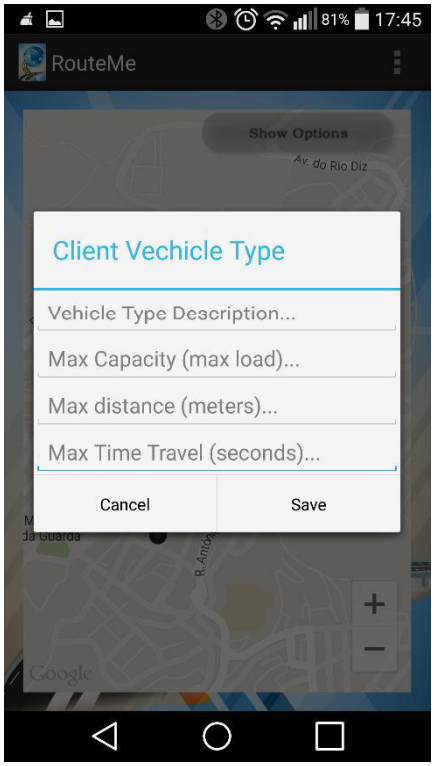


Figura 27 – Route Me – Criação de novo veículo para iniciar uma nova rota

Estas variáveis são utilizadas ao longo da resolução do problema, podendo interferir com a possibilidade (ou impossibilidade) de adição de mais clientes em determinada rota, bem como na aplicação dos algoritmos.

Por forma a facilitar o processo de criação de rotas semente, a apresentação dos clientes é diferente, aplicando para isso uma pequena transparência nos pontos, sendo que podem apenas ser selecionados os pontos opacos e não os transparentes, tal como visível na Figura 28.

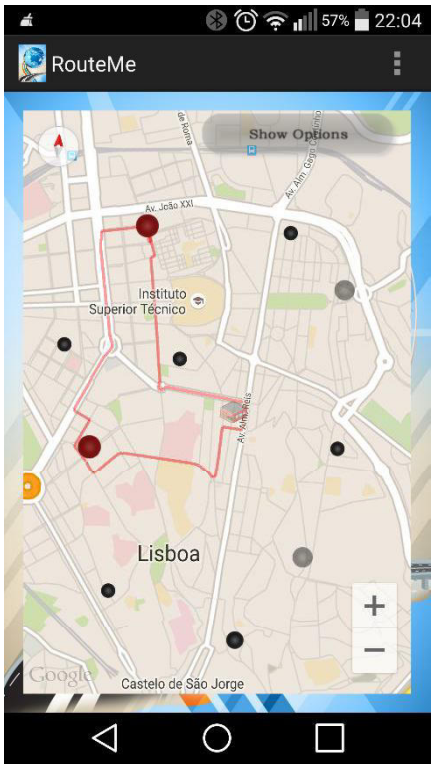


Figura 28 – Route Me – Distinção de clientes já inseridos em alguma rota

Após o utilizador selecionar o armazém, a rota é desenhada, utilizando a variável definida na Activity Opções – Dif Colors – para definir as cores de cada uma das rotas até então desenhadas. A Figura 29 mostra duas rotas com diferentes cores, tal como explicado.

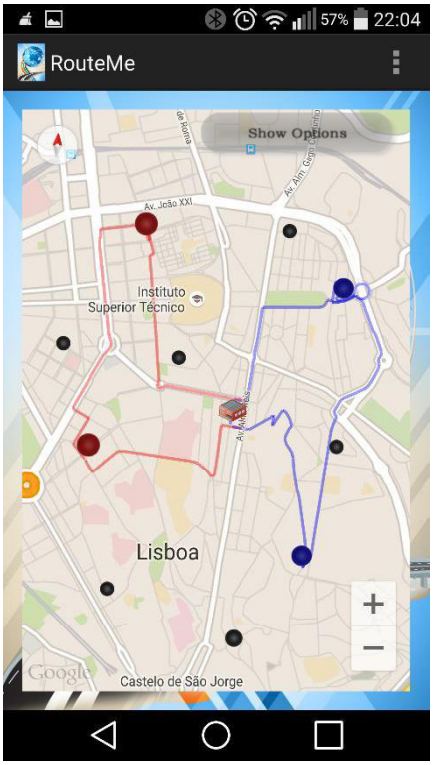


Figura 29 – Route Me – Exemplo de rotas sementes

Assim que todas as rotas semente estejam definidas, o utilizador pode completar o problema, aplicando os algoritmos, ou fazer modificações nas rotas com recurso às ferramentas interativas disponíveis.

Tal como referido anteriormente, existem várias ferramentas disponíveis para fazer a edição e melhoramento das rotas.

Começando pelo topo da lista, temos a opção "Edit Route" que permite inserir um ponto que ainda não foi incluído em nenhuma rota numa rota em que o veículo ainda tenha capacidade de carga. Na Figura 30 é possível ver a mensagem apresentada ao utilizador após selecionar a opção referida. O utilizador seleciona um ponto possível, sendo estes distinguidos pela sua opacidade, visível na Figura 30Figura 31 e, de seguida, o cliente de uma rota a seguir ao qual o anteriormente selecionado será inserido, tal como representado na Figura 31. Assim que o processo seja completo, a aplicação recalcula e redesenha a rota. Na Figura 32 é possível ver a nova rota desenhada.

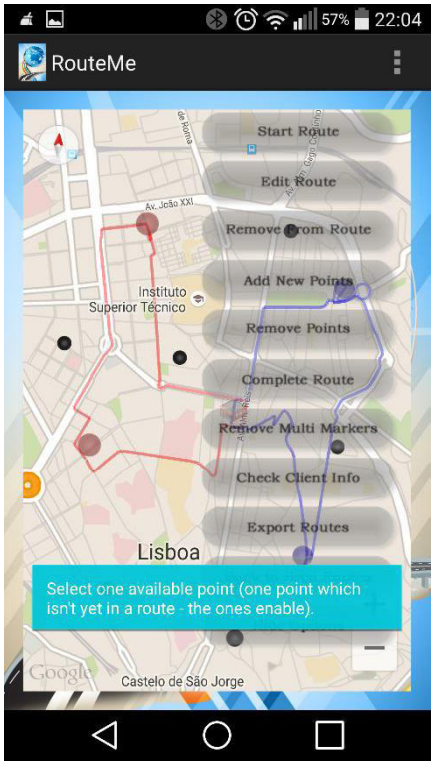


Figura 30 – Route M e – Seleção da opção "Edit Route"

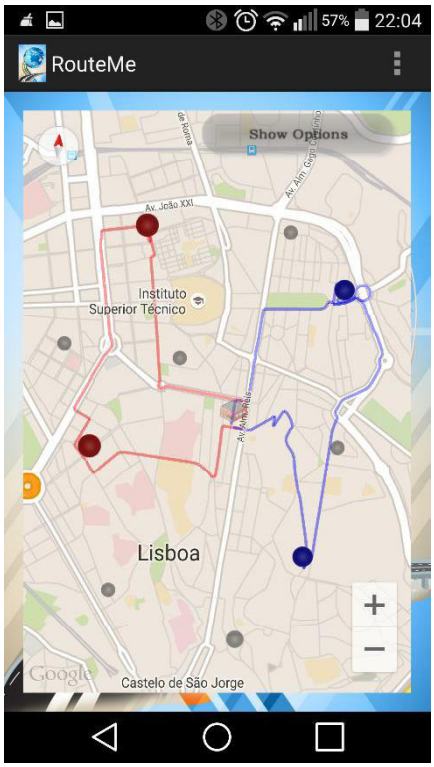


Figura 31 – Route M e – Seleção da rota onde o utilizador pretende inserir o ponto selecionado

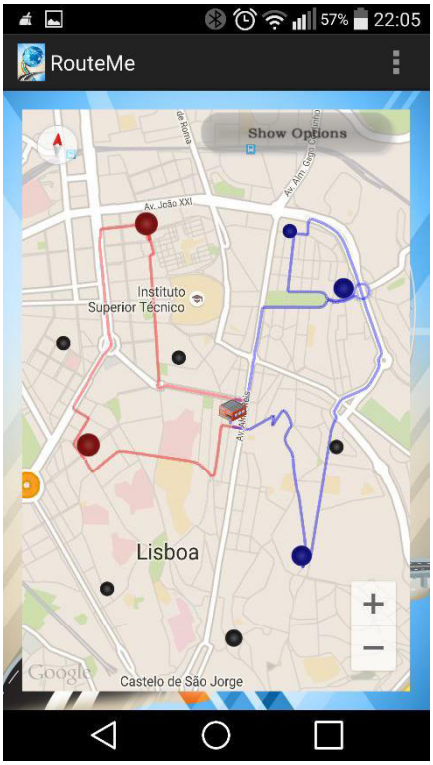


Figura 32 – Route Me – Resultado após edição da rota semente

A opção que se segue da lista é a que permite ao utilizador remover um ponto de uma das rotas – Remove From Route – sendo para isso apenas necessário clicar no cliente desejado. Os clientes possíveis de remover são igualmente destacados pela sua opacidade (Figura 33). Mais uma vez, no final desta operação, as rotas voltam a ser recalculadas e redesenhadas. O resultado desta operação pode ser visto na Figura 34.

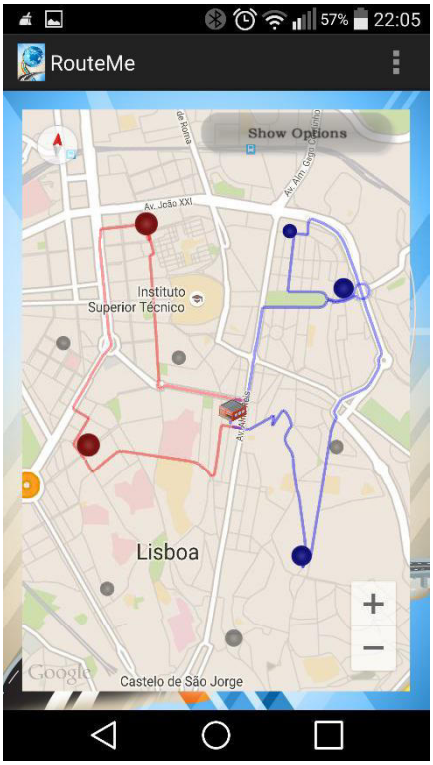


Figura 33 – Route Me – Seleção do ponto de uma rota que o utilizador pretende remover da mesma

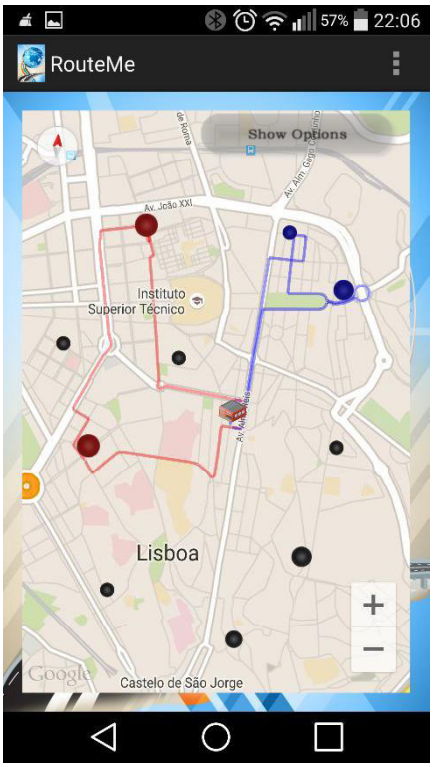


Figura 34 – Route Me – Resultado após remoção do ponto selecionado

A opção "Remove Multi Markers" permite, como referido, remover um conjunto de pontos, traçando um retângulo (Figura 35) em que todos os pontos apanhados pela área desse retângulo serão removidos das respetivas rotas. Para isso o utilizador tem que marcar as 2 extremidades do retângulo e, após confirmar a seleção, os pontos são removidos e as rotas recalculadas. O resultado desta remoção de vários pontos em simultâneo é visível na Figura 36.

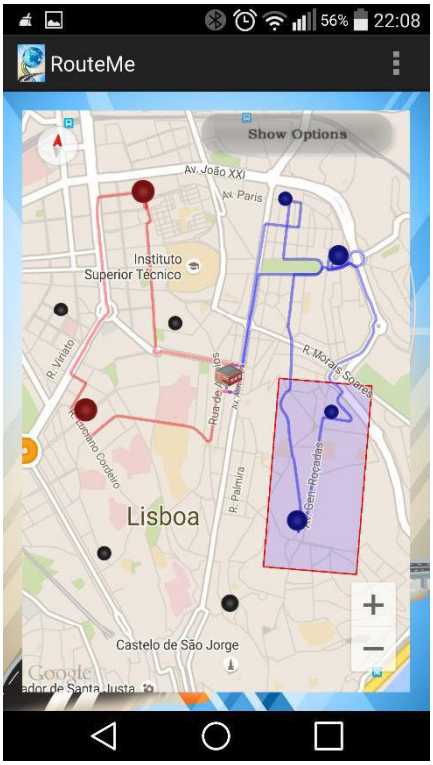


Figura 35 – Route Me – Ferramenta de multi-seleção de pontos a remover das rotas

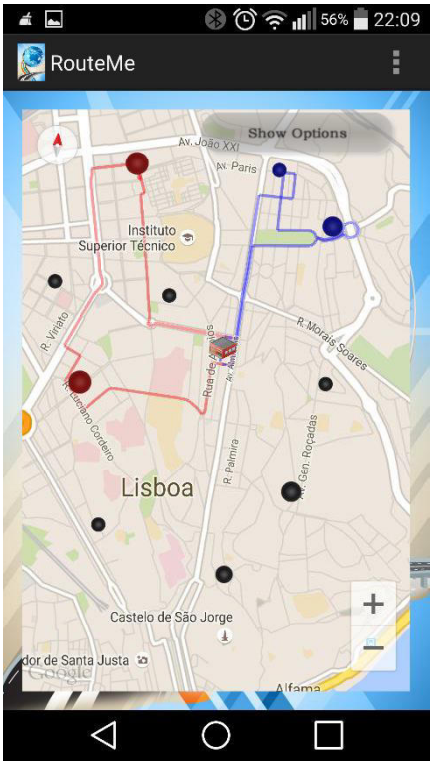


Figura 36 – Route Me – Resultado após remoção de pontos da multi-seleção

O utilizador pode ainda mudar a localização de qualquer um dos pontos, através de um clique longo em qualquer um dos pontos e arrastando para a posição desejada. Caso esse ponto pertença a alguma rota, essa rota é recalculada após nova ligação ao WebService para atualizar os dados. A Figura 37 mostra a solução antes de o utilizador arrastar um ponto e, na Figura 38 é possível ver o resultado dessa movimentação de um ponto, recalculando a rota na qual está inserido.

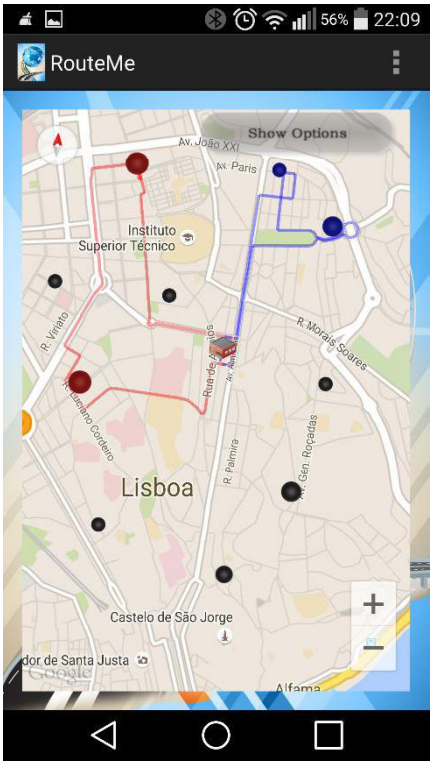


Figura 37 – Route Me – Rotas andes da movimentação de um ponto

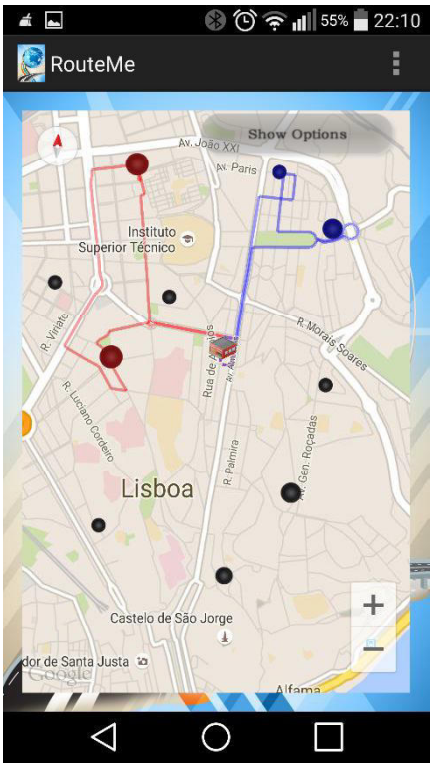


Figura 38 – Route Me – Rotas após movimentação de um ponto

Assim que as rotas sementes estejam concluídas, o utilizador pode aplicar os algoritmos para completar as rotas na opção "Complete Routes". Mesmo após a aplicação dos algoritmos, o utilizador tem sempre a possibilidade de melhorar a rota com recurso às ferramentas interativas até aqui descritas.

Sempre que o utilizador requer a aplicação dos algoritmos, no final desta é apresentado um pequeno relatório ao utilizador, com informação de todas as rotas, contendo dados tais como os pontos incluídos em cada rota, a capacidade do veículo utilizada e a distância total percorrida pelo veículo. A Figura 39 mostra um exemplo de relatório de rotas exibido ao utilizador.

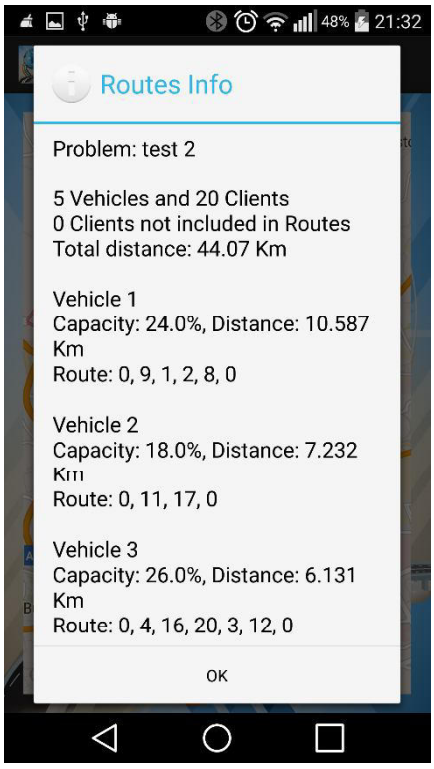


Figura 39 – Route Me – Relatório de rotas após aplicação de algoritmos

5.7. Testes da aplicação

5.7.1. Desempenho do WebService

Nesta secção será apresentado um breve estudo feito ao desempenho do Web Service no que se refere aos tempos de espera nos pedidos realizados ao serviço Google Maps. É de salientar que este tempo estudado corresponde apenas ao tempo gasto na comunicação entre o Web Service e o serviço Google Maps. Considerou-se que os tempos de comunicação entre a aplicação e o Web Service são desprezáveis.

Os resultados aqui apresentados foram obtidos através de dados recolhidos aquando de testes na aplicação. Foram utilizadas 150 amostras de dados, que representam exemplos de resolução de PRV distintos, onde o número médio de pedidos por problema foi de 74 e o tempo médio de resposta para cada conjunto de pedidos (de um determinado problema) foi de 27 segundos. Estes resultados são apresentados na seguinte figura.

É possível verificar que para o mesmo número de pedidos realizados, nem sempre a comunicação demorava o mesmo tempo, o que poderá ter sido causada por oscilações na carga do próprio serviço e no trafego da Intenet.

Tendo em conta os dados recolhidos, podemos concluir que foi possível obter um desempenho satisfatório, tendo em conta que conseguiu processar em média 3 pedidos por segundo.

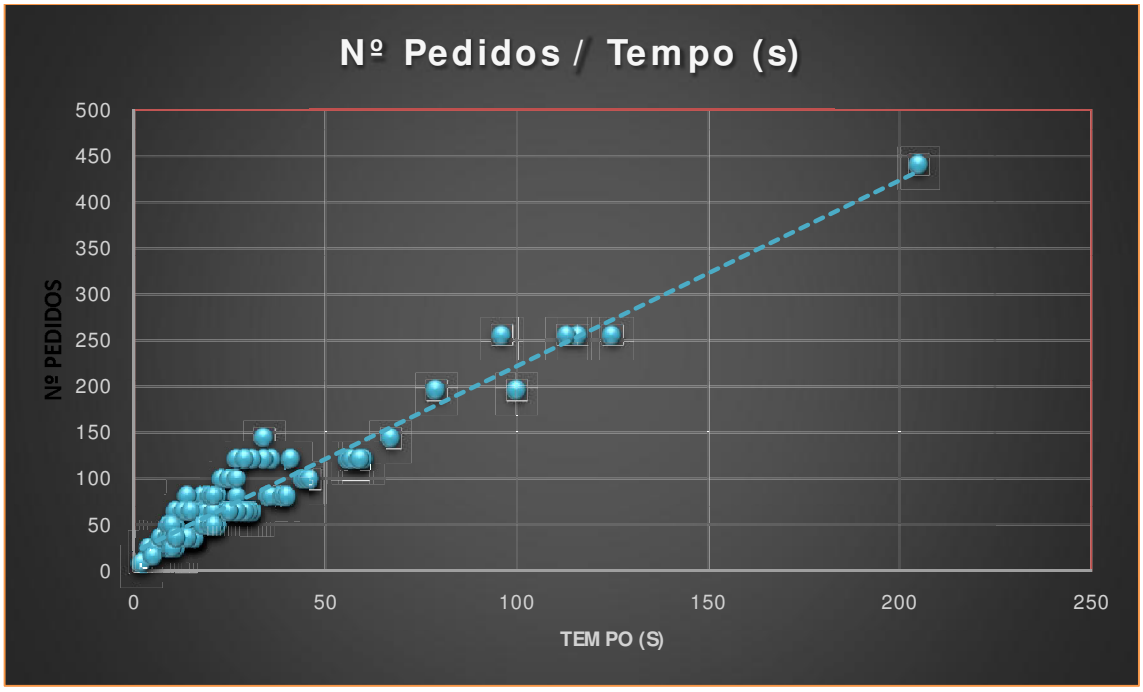


Figura 40 – Resultados do WebService

5.7.2. Eficácia na resolução do PRV

Para testar o funcionamento da aplicação e o seu desempenho na resolução do PRV, foram criados 2 problemas de teste com 13 e 20 clientes. Os problemas simularam situações de rotas de veículos em ambientes urbanos de grandes cidades. Em todos os problemas foi considerado que os veículos das frotas de um dado problema tinham todos a mesma capacidade, sendo a sua totalidade igual a cerca de 110% do total da demanda dos clientes e que a distância máxima que podiam viajar era ilimitada. A demanda de cada cliente foi escolhida aleatoriamente, assim como a localização dos mesmos.

Os testes foram efetuados num smartphone LG G3, com 2 Gb RAM, processador Snapdragon 801, 2.5 GHz QCore e Android 5.0.

Cada problema foi resolvido em seções interativas com duração máxima de 5 minutos, e considerando 2 clientes por rota semente. A fase de construção e melhoramento do método visual interativo foi configurado com os valores standard de 100 iterações e uma lista restrita de candidatos com tamanho igual a 5.

Para além destes exemplos, foi ainda criado um terceiro caso de testes, sendo que o seu único objetivo é de exemplificar a utilização das ferramentas interativas implementadas.

5.7.2.1. Teste 1

Na figura 42 podemos ver a disposição de todos os clientes e do armazém. É também possível distinguir a diferença de demandas através do tamanho de cada um dos clientes, sendo que quanto maior for o icon que o representa, maior demanda este tem.

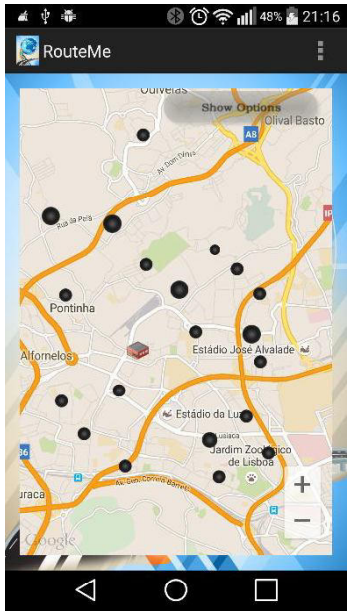


Figura 41 – Teste 1: Disposição dos clientes e armazém

Na figura 43 é possível ver já as rotas sementes desenhadas, utilizando apenas 2 clientes em cada uma das rotas, com um total de 5 rotas distintas.

Figura 42 – Teste 1: Desenho das rotas semente

O relatório final das rotas, que contém informação sobre todas as rotas após aplicar os algoritmos, é apresentado na figura 44.

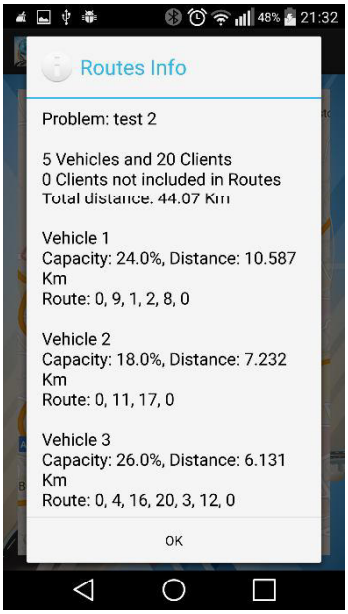


Figura 43 – Teste 1: Relatório após aplicação dos algoritmos

O resultado obtido pelos algoritmos, correspondente às rotas finais, pode ser visto na figura 45.

Figura 44 – Teste 1: Solução final obtida

5.7.2.2. Teste 2

Na figura 46 podemos ver a disposição de todos os clientes e do armazém, à semelhança do exemplo anterior. Mais uma vez, foi respeitado o número de clientes por cada rota semente, tal como se pode verificar na figura 47.

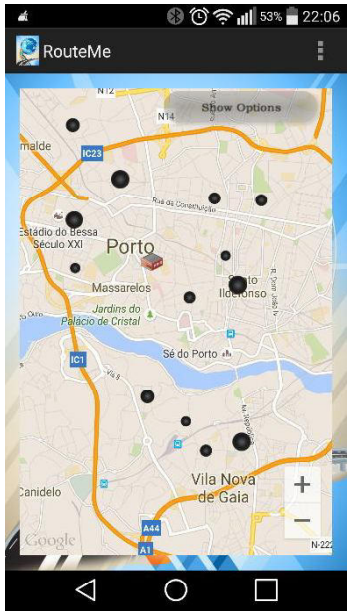


Figura 45 – Teste 2: Disposição dos clientes e armazém

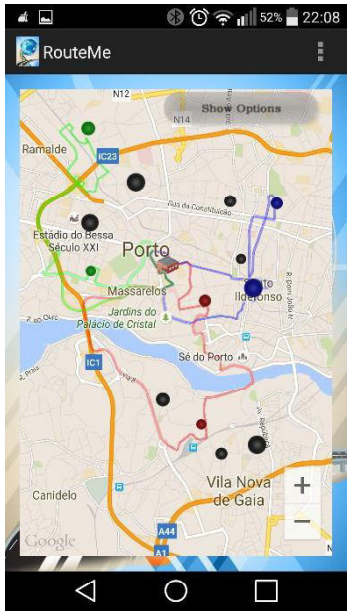


Figura 46 – Teste 2: Desenho das rotas semente

O relatório final das rotas é apresentado na figura 48 e, de seguida, na figura 49, podemos ver a solução final apresentada pela aplicação ao utilizador.

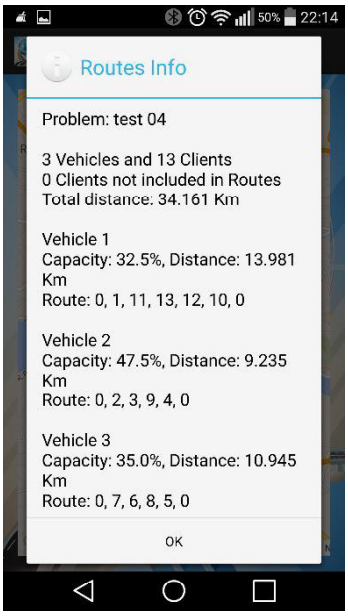


Figura 47 – Teste 2: Relatório após aplicação dos algoritmos

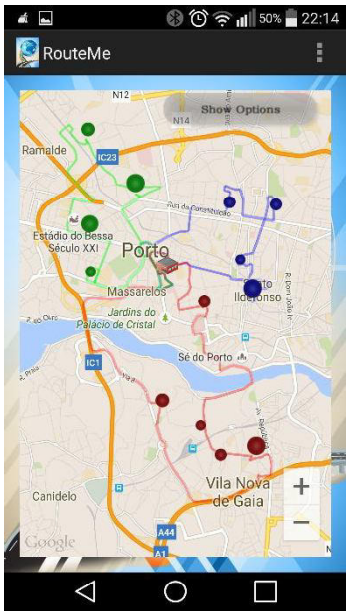


Figura 48 – Teste 2: Solução apresentada pela aplicação

Neste teste foram utilizadas algumas das ferramentas interativas implementadas, ferramentas essas apresentadas mais detalhadamente no teste 3. Na figura 50 podemos ver o menu com todas as ferramentas disponibilizadas ao utilizador e na figura 51 é apresentada a solução final, após a utilização dessas mesmas ferramentas pelo utilizador.

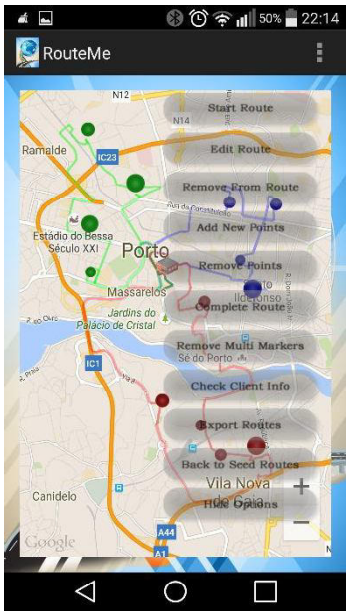


Figura 49 – Teste 2: Menu com todas as ferramentas interativas disponíveis

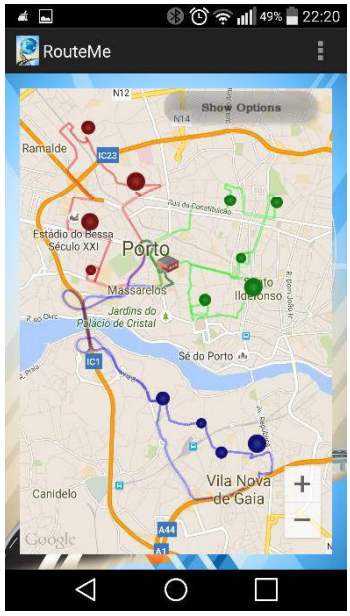


Figura 50 Teste 2: Solução final após utilização das ferramentas interativas

5.7.2.3. Teste 3

Este teste foi realizado com o objetivo de, tal como anteriormente referido, exemplificar a utilização das ferramentas interativas implementadas. Para isso foram desenhados apenas dois segmentos de rotas, visíveis na figura 52.

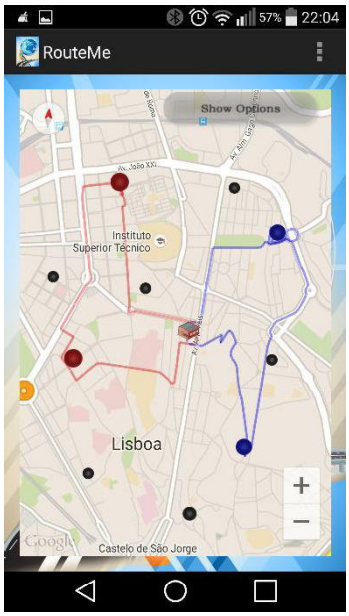


Figura 51 – Teste 3: Segmentos de rotas iniciais

Após a utilização da ferramenta que permite adicionar pontos a uma rota já existente, obteve-se o resultado visível na figura 53. Por outro lado, utilizando a ferramenta que permite ao utilizador remover clientes de uma rota existente, obteve-se o resultado da figura 54.

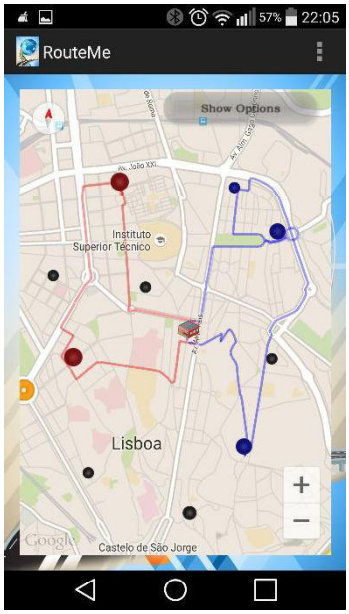


Figura 52 – Teste 3: Resultado após adicionar um cliente a uma rota existente

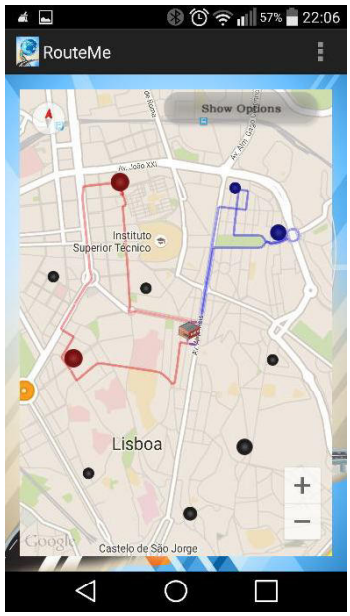


Figura 53 – Teste 3: Resultado após remover um cliente de uma rota existente

Após utilizar novamente a ferramenta que permite adicionar pontos a uma rota já existente, obteve-se as rotas visíveis na figura 55.

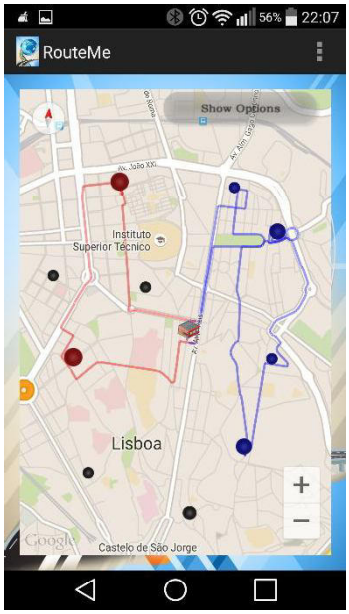


Figura 54 – Teste 3: Resultado após adicionar vários clientes a uma rota

Na figura 56 podemos observar a utilização da ferramenta que permite remover multiplos pontos de uma só vez. São traçadas as extremidade superior esquerda e inferior direita por forma a desenhar um rectângulo, sendo que todos os pontos incluídos nessa área serão removidos, tal como se pode verificar na figura 57.

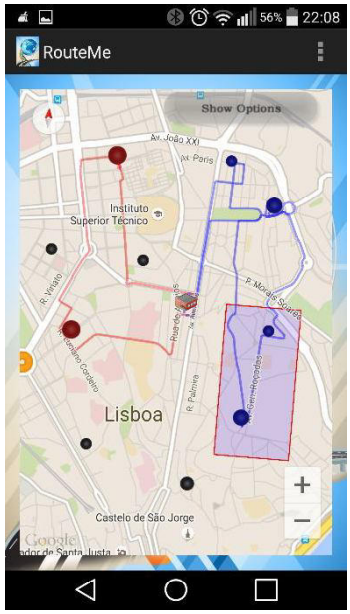


Figura 55 – Teste 3: Utilização da ferramenta para remover múltiplos pontos

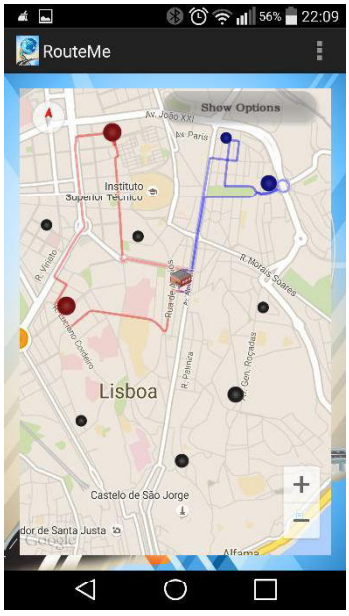


Figura 56 – Teste 3: Resultado obtido após utilização da ferramenta para remover múltiplos pontos

Por fim, na figura 58, podemos observar o resultado obtido após mover um dos pontos. Neste caso, o ponto movido foi o ponto mais a baixo da rota vermelha.

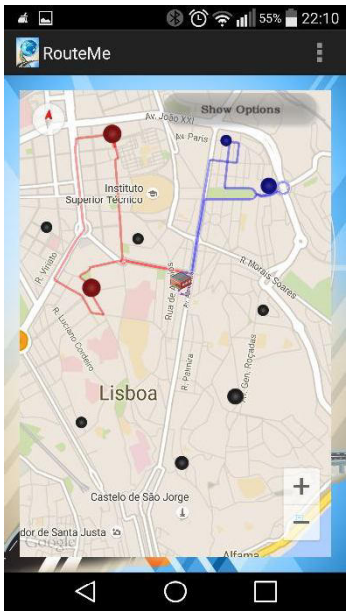


Figura 57 – Teste 3: Resultado após mover um dos pontos de uma rota

6. CONCLUSÃO

Nos dias que correm a utilização de dispositivos móveis é cada vez mais comum, sendo que os utilizadores procuram cada vez mais aplicações que lhes permitam fazer grande parte das operações realizadas nos típicos computadores pessoais de secretária. À parte disso, os smartphones têm vindo a evoluir consideravelmente de ano para ano, principalmente a nível de capacidade de processamento e armazenamento de dados.

O objetivo geral deste projeto foi de, utilizando as capacidades atuais dos smartphones e tendo em conta que parece não haver aplicações semelhante à desenvolvida, construir uma ferramenta que permitisse a qualquer utilizador criar e resolver um PRV básico de forma simples, rápida e recorrendo a métodos visuais interativos.

Apesar da caraterística principal do projeto estar focada na utilização do conhecimento do tipo de problemas a resolver por parte do utilizador e na disponibilização de ferramentas visuais e interativas que permitam ao mesmo, um controlo e gestão sem qualquer obstáculo nem complicação, foi igualmente necessário proceder a um estudo aprofundado no que a Problemas de Rotas de Veículos diz respeito. Foi necessário conhecer a fundo as metodologias utilizadas para a obtenção de soluções de boa qualidade, bem como dos algoritmos existentes para a construção e melhoramento das mesmas. Paralelamente a esse estudo, de forma a fazer da aplicação criada, uma aplicação que se destaque das já existentes pelas ferramentas desenvolvidas, foi necessário fazer uma análise cuidada das aplicações já existentes no mercado dos dispositivos móveis.

Apesar de terem ficado alguns pontos por melhorar, principalmente a nível de performance, uma vez que a implementação de todas as ferramentas interativas e de todos os algoritmos representam um enorme peso a nível de processador, diminuindo assim a performance da aplicação, pode-se considerar, através dos testes realizados, que com a ferramenta construída é possível obter soluções que se aproximem a uma boa solução. Esta permite ao utilizador fazer uma gestão das rotas mesmo fora do escritório, sendo que é possível com ela proceder à resolução de problemas de casos reais. É ainda possível concluir que os objetivos iniciais foram, na sua grande maioria, cumpridos.

Devido a restrições de tempo, ficaram por alcançar alguns objetivos, tais como a implementação de uma ferramenta que permitisse ao utilizador fazer uma gestão da frota e do tipo de veículos.

Foi recentemente elaborado um artigo científico sobre o projeto, o qual foi apresentado na conferência Metaheuristics International Conference 2015 [34] que decorreu em Agadir, Marrocos. Este artigo pode ser encontrado no anexo I.

6.1. Trabalho Futuro

Conforme anteriormente referido, ficaram vários pontos por fechar no desenvolvimento da aplicação, principalmente no que a performance diz respeito. Onde se notou um maior défice de performance foi no desenho de rotas e na aplicação dos algoritmos quando se trata de resolver problemas com um elevado número de pontos. Este problema acontece devido ao número de iterações feitas em ambas as fases de construção e melhoramento da solução, sendo possível melhorar este processo através de uma análise cuidada ao código, melhorando a implementação dos algoritmos. Há ainda lugar para melhorar a comunicação com os dados dos mapas, feito através da Google Maps API, procedendo à troca do fornecedor de mapas utilizando para isso por exemplo o OpenStreetMap [35]. Acredita-se que a utilização desta alternativa poderá ajudar a melhorar a performance das comunicações e a diminuir os problemas atualmente existentes com a versão grátis da Google Maps API.

Ficaram ainda outros pontos em aberto, tal como a utilização e divisão de rotas de acordo com o tipo de carga dos veículos e dos clientes, que será igualmente uma funcionalidade útil a implementar, principalmente para utilizadores em que o tipo de carga a transportar nem sempre é do mesmo tipo. Seria interessante implementar mecanismos que permitam ao utilizador fazer a gestão das suas frotas e do tipo de carga que cada veículo poderá transportar. Desse modo, na construção das rotas semente o utilizador teria a possibilidade de selecionar da sua lista um determinado veículo, para dar resposta a uma possível restrição de acondicionamento de carga de um certo cliente.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] P. Toth and D. Vigo, *The Vehicle Routing Problem*. SIAM, 2002, p. 367.

[2] C. Archetti, "Complexity and Reducibility of the Skip Delivery Problem," pp. 1–20, 2003.

[3] N. Bianchessi and G. Righini, "Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery," 2003.

[4] C. Jacobs-blecha, "The Vehicle Routing Problem with Backhauls : Properties and Solution Algorithms," 1998.

[5] J. Carlsson, D. Ge, A. Wu, Y. Yinyu, and A. Subramaniam, "Solving Min-Max Multi-Depot Vehicle Routing Problem ∗," pp. 1–17, 2007.

[6] S. Coene, A. Arnout, F. Spieksma, and S. Coene, "Faculty of Business and Economics The periodic vehicle routing problem : a case study The Periodic Vehicle Routing Problem : A Case," 2008.

[7] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems Part I : Transportation between customers and depot," pp. 1–28, 2008.

[8] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems Part II : Transportation between pickup and delivery locations," pp. 1–35, 2008.

[9] F. Louveaux and G. Laporte, "Stochastic Vehicle Routing Problems. SVRP," in in *Encyclopedia of Optimization*, C. A. Floudas and P. M. Pardalos, Eds. Springer US, 2009, pp. 3829–3832.

[10] C. Comtois, B. Slack, and J.-P. Rodrigue, *The geography of transport systems*, Third. London, 2013.

[11] B. Baker and C. Carreto, "A visual interactive approach to vehicle routing," *Comput. Oper. Res.*, vol. 30, no. 3, pp. 321–337, Mar. 2003.

[12] Auren and NEO, "VRP," 2006. [Online]. Available: http://www.bernabe.dorronsoro.es/vrp/.

[13] J. Fajfr, "The Wall - Vehicle Routing Problem," 2010. [Online]. Available: http://hoonzis.blogspot.pt/2010/05/vehicle-routing-problem.html.

[14] J. Lysgaard, "Clarke & Wright ' s Savings Algorithm," no. September, 1997.

[15] N. Suthikarnnarunai, "A Sweep Algorithm for the Mix Fleet Vehicle Routing Problem," vol. II, pp. 19–21, 2008.

[16] E. X. Bonn, "CONFERENCE Thrilling speed , Superb performance , Astonishing engine . More information at :," 2009.

[17] L. Santos, J. Coutinho-Rodrigues, and C. H. Antunes, "A web spatial decision support system for vehicle routing using Google Maps," *Decis. Support Syst.*, vol. 51, no. 1, pp. 1–9, Apr. 2011.

[18] XClubeLABS, "The Android Story." [Online]. Available: http://www.xcubelabs.com/infographic-android-story/.

[19] Android, "Android Platform Versions," 2015. [Online]. Available: http://developer.android.com/intl/zh-CN/about/dashboards/index.html.

[20] IDC, "Smartphone OS Market Share, Q1 2015," 2015. [Online]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[21] Google, "Google Maps API." [Online]. Available: https://developers.google.com/maps/.

[22] W3C, "World Wide Web Consortium." [Online]. Available: http://www.w3.org/.

[23] W3schools, "SOAP Introduction." [Online]. Available: http://www.w3schools.com/webservices/ws_soap_intro.asp.

[24] SQLite, "SQLite." [Online]. Available: https://www.sqlite.org/.

[25] Microsoft, "Framework .Net." [Online]. Available: http://www.microsoft.com/net.

[26] Microsoft, "Common Language Runtime." [Online]. Available: https://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.110).aspx.

[27] Microsoft, "Framework Class Library." [Online]. Available: https://msdn.microsoft.com/en-us/library/gg145045(v=vs.110).aspx.

[28] Google, "Google Docs." [Online]. Available: https://www.google.com/docs/about/.

[29] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *J. Glob. Optim.*, vol. 62, pp. 109–133, 1995.

[30] Y. Tutuncu, C. Carreto, and B. Baker, "A visual interactive approach to classical and mixed vehicle routing problems with backhauls," *Omega*, vol. 37.1, pp. 138–154, 2009.

[31] C. Carreto and B. Baker, "A GRASP interactive approach to the vehicle routing problem with Backhauls," in in *Essays and Surveys on Metaheuristics*, C. Ribeiro and P. Hansen, Eds. Kluwer Academic Publishers, 2002, pp. 185–200.

[32] S. Lin, "Computer solutions of the traveling salesman problem," *Bell Syst. Tech. J.*, vol. 44, pp. 2245–2269, 1965.

[33] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and post optimization procedures for the traveling salesman problem," *Oper. Res.*, vol. 40, no. 6, pp. 1086–1094, 1992.

[34] "MIC 2015," 2015. [Online]. Available: http://www.lifl.fr/MIC2015/.

[35] "OpenStreetMap." [Online]. Available: https://www.openstreetmap.org/about.

ANEXOS

Anexo A – Resposta JSon completa


```
{
 "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"
    }
   },
   "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@xxaB
eJxlcBb~t@zbh@jc|Bx}C`rv@rw|@rlhA~dVzeo@vrSnc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo
@`cFp~xBc`Hk@nurDznmFfwMbwz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_
Wr|Efc\\x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg@|kHntyArpb@bijCk_Kv~eGyqTj_|@`uV`k|DcsNdwx
Aott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy@`omFvaErfo@igQxnlApqGze~AsyRzrjAb__@
ftyB}pIlo_BflmA~yQftNboWzoAlzp@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdmY_hLxiZd~Xt
aBndgC"
    },
    "warnings": [ ],
    "waypoint_order": [ 0, 1 ],
    "bounds": {
      "southwest": {
        "lat": 34.0523600,
        "lng": -118.2435600
      },
      "northeast": {
        "lat": 41.8781100,
        "lng": -87.6297900
      }
    }
  } ]
}
```

Anexo B – Método GetStringWSPoints

```
<WebMethod()> _
Public Function GetStringWSPoints(listAlPointss As String) As String

    Dim startDate As Date
    startDate = Date.Now
    Dim ok As Integer = 0
    Dim listAlPoints As String() = listAlPointss.Split(":")
    Dim matrizPoints(listAlPoints.Count - 2, 2) As String
    Dim listJsons(listAlPoints.Count - 2, listAlPoints.Count - 2) As String
    Dim pointsString As String = String.Empty
    Dim cnt As Integer = 1

    For i As Integer = 0 To listAlPoints.Count - 2
        Dim listLatLong As String() = listAlPoints.ElementAt(i).Split(";")
        matrizPoints(i, 0) = listLatLong(0)
        matrizPoints(i, 1) = listLatLong(1)
        pointsString += String.Format("Point {0}: {1};{2}---", cnt,
listLatLong(0), listLatLong(1))
        cnt += 1
    Next

    Dim initialDate As Date = Date.Now

    For i As Integer = 0 To listAlPoints.Count - 2
        For j As Integer = 0 To listAlPoints.Count - 2
            Dim urlString As StringBuilder
            urlString = New StringBuilder()

urlString.Append("http://maps.googleapis.com/maps/api/directions/json")
            urlString.Append("?origin=")
            urlString.Append(matrizPoints(i, 0))
            urlString.Append(",")
            urlString.Append(matrizPoints(i, 1))
            urlString.Append("&destination=")
            urlString.Append(matrizPoints(j, 0))
            urlString.Append(matrizPoints(j, 1))
            urlString.Append(",")
            urlString.Append("&sensor=false")

            Dim reader As StreamReader
            Dim request As WebRequest
            Dim response As WebResponse
            Dim data As String = ""
            Dim hasDone As Boolean = False

            While (hasDone = False)
                Try
                    ok += 1
                    request = WebRequest.Create(urlString.ToString())
                    request.Timeout = 10000
                    response = request.GetResponse()
                    reader = New StreamReader(response.GetResponseStream())
                    data = reader.ReadToEnd()
                    Dim deserialize As Object = New
JavaScriptSerializer().Deserialize(Of Object)(data)
```

```
                    Dim distance As Integer =
deserialize("routes")(0)("legs")(0)("distance")("value")
                    Dim duration As Integer =
deserialize("routes")(0)("legs")(0)("duration")("value")
                    Dim polilyne As String =
deserialize("routes")(0)("overview_polyline")("points")

                    listJsons(i, j) = String.Format("{0};{1};{2}", distance,
duration, polilyne)

                    hasDone = True
                Catch ex As Exception
                End Try
            End While
        Next
    Next

    Dim finalDate As Date = Date.Now
    Dim stringFinal As String

    For i As Integer = 0 To listAlPoints.Count - 2
        For j As Integer = 0 To listAlPoints.Count - 2
            stringFinal += listJsons(i, j) + ":"
        Next
    Next

    Dim endDate As Date
    endDate = Date.Now
    Dim totalSeconds As TimeSpan
    totalSeconds = endDate - startDate
    Dim url As String = String.Empty
    url =
"docs.google.com/forms/d/14Yd0NfGR7efa8DQACLhqGHAOyZTPwBR3Caj3eo7p0Qg/formRespons
e?entry.2054938567="
    url += endDate.ToShortDateString
    url += "&entry.1827975684=" + totalSeconds.ToString + "&entry.1852957370=" +
pointsString

    Dim webStream As Stream
    Dim webResponse = ""
    Dim req As HttpWebRequest
    Dim res As HttpWebResponse
    req = CType(WebRequest.Create("http://" + url), HttpWebRequest)
    res = CType(req.GetResponse(), HttpWebResponse)
    webStream = res.GetResponseStream()

    Return stringFinal 'ok.ToString()
End Function
```

Anexo C – Método exportRoutes

```
<WebMethod()> _
Public Function exportRoutes(listStrings As String) As String
    Dim listAllStrings As String() = listStrings.Split("&")
    Dim DateExport As Date
    DateExport = Date.Now
    Dim varTime As String
    varTime = 0
    Dim webStream As Stream
    Dim webResponse = ""
    Dim req As HttpWebRequest
    Dim res As HttpWebResponse
    Dim url As String = String.Empty

    For i As Integer = 0 To listAllStrings.Count - 2
        url = String.Empty
        url =
"docs.google.com/forms/d/14Yd0NfGR7efa8DQACLhqGHAOyZTPwBR3Caj3eo7p0Qg/formRespons
e?entry.2054938567="
        url += DateExport.ToShortDateString
        url += "&entry.1827975684=" + varTime.ToString + "&entry.1852957370="
        If (i = 0) Then
            url += "Points: "
        Else
            url += "Rote: "
        End If

        url += listAllStrings.ElementAt(i)
        webResponse = ""
        req = CType(WebRequest.Create("http://" + url), HttpWebRequest)
        res = CType(req.GetResponse(), HttpWebResponse)
        webStream = res.GetResponseStream()
    Next
    Return True
End Function
```

Anexo D – Classe ClientEntity

```
public class ClientEntity {
	private int idClient;
    private String name;
    private String description;
    private int idTypeLoad;
    private double quantity;
    private ClientLoadTypeEntity clientLoad;

	public ClientEntity(){}

	public ClientEntity(String name, String description, int
idTypeLoad,
			double quantity, ClientLoadTypeEntity clientLoad) {
		super();
		this.name = name;
		this.description = description;
		this.idTypeLoad = idTypeLoad;
		this.quantity = quantity;
		this.clientLoad = clientLoad;
    }

    public ClientEntity(String name, String description, int
idTypeLoad,
			double quantity) {
		super();
		this.name = name;
		this.description = description;
		this.idTypeLoad = idTypeLoad;
		this.quantity = quantity;
    }

    @Override
    public String toString() {
	return String.format("Client [id=%d, Name=%s, Description=%s, "
+
			"TypeLoad=%d, Quantity=%2f]", idClient,
			name, description, idTypeLoad, quantity);
    }

	public int getIdClient() {
		return idClient;
	}

	public void setIdClient(int idClient) {
		this.idClient = idClient;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}
```

```
	public String getDescription() {
		return description;
	}

	public void setDescription(String description) {
		this.description = description;
	}

	public int getIdTypeLoad() {
		return idTypeLoad;
	}

	public void setIdTypeLoad(int idTypeLoad) {
		this.idTypeLoad = idTypeLoad;
	}

	public double getQuantity() {
		return quantity;
	}

	public void setQuantity(double quantity) {
		this.quantity = quantity;
	}

	public ClientLoadTypeEntity getClientLoad() {
		return clientLoad;
	}

	public void setClientLoad(ClientLoadTypeEntity clientLoad) {
		this.clientLoad = clientLoad;
	}
}
```

Anexo E – Método CompareTo

```
@Override
public int compareTo(ClientsToEachRoute another) {
	if (orderBySize)
	{
		if (this.listCostRoutes.size() >
another.listCostRoutes.size())
			return 1;
		else if (this.listCostRoutes.size() ==
another.listCostRoutes.size())
			return 0;
		else
			return -1;
	}
	else
	{
		if (this.getLowerCostDiff() > another.getLowerCostDiff())
			return 1;
		else if (this.getLowerCostDiff() ==
another.getLowerCostDiff())
			return 0;
		else
			return -1;
	}
}

public double getLowerCostDiff() {
	double lowerCost = Integer.MAX_VALUE;
	double scndLowerCost = Integer.MAX_VALUE;
	for (CostRoute cRoute : listCostRoutes)
	{
		if (cRoute.cost < lowerCost)
		{
			scndLowerCost = lowerCost;
			lowerCost = cRoute.cost;
		}
		else if (cRoute.cost < scndLowerCost)
		{
			scndLowerCost = cRoute.cost;
		}
	}
	return scndLowerCost - lowerCost;
}
```

Anexo F – Classe ClientData

```
public class ClientData extends SQLiteOpenHelper {

	private static final int DATABASE_VERSION =
MySQLiteHelper.DATABASE_VERSION;
	private static final String DATABASE_NAME =
MySQLiteHelper.DATABASE_NAME;
	public static final String C_ID_CLIENT =
MySQLiteHelper.C_ID_CLIENT;
	public static final String C_NAME = MySQLiteHelper.C_NAME;
	public static final String C_DESCRIPTION =
MySQLiteHelper.C_DESCRIPTION;
	public static final String C_ID_TYPE_LOAD =
MySQLiteHelper.C_ID_TYPE_LOAD;
	public static final String C_QUANTITY =
MySQLiteHelper.C_QUANTITY;
	private static final String TABLE_CLIENT =
MySQLiteHelper.TABLE_CLIENT;
	public ClientLoadTypeData clientLoadType;

	public ClientData(Context context) {
		super(context, DATABASE_NAME, null, DATABASE_VERSION);
		clientLoadType = new ClientLoadTypeData(context);
	}

	private static final String[] COLUMNS = { C_ID_CLIENT,
		C_NAME, C_DESCRIPTION, C_ID_TYPE_LOAD, C_QUANTITY };

	public void addClient(ClientEntity client) {
		SQLiteDatabase db = this.getWritableDatabase();
		ContentValues values = new ContentValues();
		values.put(C_NAME, client.getName());
		values.put(C_DESCRIPTION, client.getDescription());
		values.put(C_ID_TYPE_LOAD, client.getIdTypeLoad());
		values.put(C_QUANTITY, client.getQuantity());
		db.insert(TABLE_CLIENT,
				null,
				values);
		db.close();
	}

	public int addClientWithId(ClientEntity client) {
		SQLiteDatabase db = this.getWritableDatabase();
		ContentValues values = new ContentValues();
		values.put(C_NAME, client.getName());
		values.put(C_DESCRIPTION, client.getDescription());
		values.put(C_ID_TYPE_LOAD, client.getIdTypeLoad());
		values.put(C_QUANTITY, client.getQuantity());

		long idClient = db.insert(TABLE_CLIENT,
					null,
					values);

		db.close();
		return (int)idClient;
	}
```

```
	public ClientEntity getClient(int idClient) {
		SQLiteDatabase db = this.getReadableDatabase();
		Cursor cursor = db.query(TABLE_CLIENT,
				new String[] { C_ID_CLIENT,
				C_NAME, C_DESCRIPTION, C_ID_TYPE_LOAD,
C_QUANTITY },
				C_ID_CLIENT + " = ?",
				new String[] { String.valueOf(idClient) },

				null
				null,
				null,
				null);
		if (cursor != null)
			cursor.moveToFirst();
		ClientEntity client = new ClientEntity();

		try {

	client.setIdClient(Integer.parseInt(cursor.getString(0)));
			client.setName(cursor.getString(1));
			client.setDescription(cursor.getString(2));

	client.setIdTypeLoad(Integer.parseInt(cursor.getString(3)));

	client.setQuantity(Integer.parseInt(cursor.getString(4)));

	client.setClientLoad(clientLoadType.getClientLoadType(client.get
IdTypeLoad()));
		} catch (Exception e) {
		}

		cursor.close();

		return client;
	}

	public List<ClientEntity> getAllClients() {
		List<ClientEntity> clients = new
LinkedList<ClientEntity>();

		String query = "SELECT * FROM" + TABLE_CLIENT;

		SQLiteDatabase db = this.getWritableDatabase();
		Cursor cursor = db.rawQuery(query, null);

		ClientEntity client = null;
		if (cursor.moveToFirst()) {
			do {
				client = new ClientEntity();

	client.setIdClient(Integer.parseInt(cursor.getString(0)));
				client.setName(cursor.getString(1));
				client.setDescription(cursor.getString(2));

	client.setIdTypeLoad(Integer.parseInt(cursor.getString(3)));
```

```
        client.setQuantity(Integer.parseInt(cursor.getString(4)));

        client.setClientLoad(clientLoadType.getClientLoadType(client.get
IdTypeLoad()));
                        clients.add(client);
                    } while (cursor.moveToNext());
                }

                cursor.close();
                return clients;
        }

        public void deleteClient(ClientEntity client) {
                SQLiteDatabase db = this.getWritableDatabase();
                db.delete(TABLE_CLIENT,
                        C_ID_CLIENT + " = ?",
                        new String[] {
String.valueOf(client.getIdClient()) });
                db.close();
        }
}
```

Anexo G – Criação de marcadores

No caso de ser o primeiro ponto, significa que se está a inserir o armazém (imagem do armazém), caso contrário será um novo cliente (respetiva imagem de cliente).

```
boolean isFirst = listMarkers.size() == 0;
newMarker = isFirst ? map.addMarker(new MarkerOptions()
            .position(pointMap)
            .title(clientName.getText().toString())
            .icon(BitmapDescriptorFactory

      .defaultMarker(BitmapDescriptorFactory.HUE_BLUE))
            .draggable(true))
            : map.addMarker(new MarkerOptions()
                        .position(pointMap)
                        .title(clientName.getText().toString())
                        .draggable(true));
```

Após inserir, é calculado e editado o tamanho de cada um dos clientes, tendo em conta a quantidade de carga a transportar para os mesmos.

```
public void changeImageFromMarkes() {
      double totalLoad = 0;
      for (int idClient : listClients) {
            if (listClients.indexOf(idClient) != 0) {
                  totalLoad +=
db.client.getClient(idClient).getQuantity();
            }
      }

      double avgLoad = totalLoad / (listClients.size() - 1);

      for (Marker markerChange : listMarkers) {
            if (listMarkers.indexOf(markerChange) != 0) {
                  double sizeRate = db.client.getClient(

      listClients.get(listMarkers.indexOf(markerChange)))
                              .getQuantity()
                              / avgLoad;

                  sizeRate = sizeRate < 0.7 ? 0.7 : sizeRate > 1.3 ?
1.3
                              : sizeRate;

                  BitmapDrawable b = (BitmapDrawable)
getResources().getDrawable(
                              R.drawable.exmark);
                  Bitmap d = b.getBitmap();

                  int newSize = (int) ((maxSizeToImages * 0.03) *
sizeRate);
```

```
                    Bitmap bhalfsize = Bitmap.createScaledBitmap(d,
(int) newSize,
                            (int) newSize, false);

                    markerChange.setIcon(BitmapDescriptorFactory
                            .fromBitmap(bhalfsize));

                } else {
                    BitmapDrawable b = (BitmapDrawable)
getResources().getDrawable(
                            R.drawable.warehouse2);
                    Bitmap d = b.getBitmap();

                    int newSize = (int) (maxSizeToImages * 0.04);

                    Bitmap bhalfsize = Bitmap.createScaledBitmap(d,
(int) newSize,
                            (int) newSize, false);

                    markerChange.setIcon(BitmapDescriptorFactory
                            .fromBitmap(bhalfsize));
                }
            }
}
```

Anexo H – Método de aquisição de dados do Google Maps

Método assincrono chamado inicialmente para preparer a ligação ao WebService.

```
private class AsyncCallWSToString extends AsyncTask<Void, Void,
String> {
	ArrayList<String> listPointsWS = new ArrayList<String>();
	private ProgressDialog progressDialog;
	String msgToDisplay = "";

	AsyncCallWSToString(ArrayList<Marker> listPoints, String
msgToDisplay) {
		this.msgToDisplay = msgToDisplay;
		for (int i = 0; i < listPoints.size(); i++) {

	listPointsWS.add(listPoints.get(i).getPosition().latitude + ";"
				+
		listPoints.get(i).getPosition().longitude);
		}
	}

	@Override
	protected void onPreExecute() {
		super.onPreExecute();
		progressDialog = new ProgressDialog(context);
		progressDialog.setMessage("Getting data, please wait...");
		progressDialog.setIndeterminate(true);
		progressDialog.show();
	}

	@Override
	protected String doInBackground(Void... params) {
		String result = getAllPoints(listPointsWS).toString();
		return result;
	}

	@Override
	protected void onPostExecute(String result) {
		super.onPostExecute(result);
		progressDialog.hide();
		if (result != "NOK") {
			if (processData(result)) {
				if (isToRegenerateRoutesFromNewData
						||
isToRegenerateRoutesFromDragMarker) {
					canMarkPoints = false;
					canMarkNewPoints = false;
					canTraceRoutes = true;
					canEditRoutes = false;
					canRemoveFromRoutes = false;
					canRemovePoints = false;

					List<routeOfTheMap>
auxListRoutesOfTheMap = listRoutesOfTheMap;
```

```
                                listRoutesOfTheMap = new
ArrayList<routeOfTheMap>();

                                for (routeOfTheMap routeToDraw :
auxListRoutesOfTheMap) {
                                    List<Integer> orderClientIdList =
new ArrayList<Integer>();

                                    for (int i = 0; i <
routeToDraw.listClientsID
                                            .size(); i++) {

        orderClientIdList.add(listClients

        .indexOf(routeToDraw.listClientsID

        .get(i)));
                                    }

                                    new connectAsyncTaskToDrawNewPath(
                                            orderClientIdList,
routeToDraw).execute();
                                }

                                if (isToRegenerateRoutesFromNewData)
                                    msgToDisplay =
stringAfterGetDataFromNewPoints;
                                else if
(isToRegenerateRoutesFromDragMarker)
                                    msgToDisplay =
stringSuccessAfterDragPoint;

                                isToRegenerateRoutesFromNewData = false;
                                isToRegenerateRoutesFromDragMarker =
false;
                            }

                            if (msgToDisplay != "") {
                                Toast.makeText(context, msgToDisplay,
Toast.LENGTH_LONG)
                                        .show();
                                Toast.makeText(context,
stringBeforeSeedsRoutes,
                                        Toast.LENGTH_LONG).show();
                            }
                        } else {
                            Toast.makeText(context,
stringErrorGettingData,
                                    Toast.LENGTH_LONG).show();
                        }
                    } else {
                        Toast.makeText(context, stringErrorGettingData,
                                Toast.LENGTH_LONG).show();
                    }
                }
            }
```

Método que faz os pedidos ao WebService.

```
public String getAllPoints(ArrayList<String> listPoints) {
	try {
		SoapObject request = new SoapObject(NAMESPACE,
METHOD_NAME);

		String[] listMarkerString = new String[listPoints.size()];

		String allResults = "";
		String allRequests = "";

		for (int i = 0; i < listPoints.size(); i++) {
			listMarkerString[i] = listPoints.get(i);
			allRequests += listPoints.get(i) + ":";
		}

		PropertyInfo propList = new PropertyInfo();
		propList.setName("listAlPointss");
		propList.setValue(allRequests);
		propList.setType(String.class);
		request.addProperty(propList);

		SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(
				SoapEnvelope.VER11);
		envelope.dotNet = true;
		envelope.setOutputSoapObject(request);
		AndroidHttpTransport androidHttpTransport = new
AndroidHttpTransport(
				URL);

		try {
			androidHttpTransport.call(SOAP_ACTION, envelope);
			SoapPrimitive response = (SoapPrimitive)
envelope.getResponse();
			allResults = response.toString();
		} catch (Exception e) {
			e.printStackTrace();
		}

		return allResults;
	} catch (Exception e) {
		return "NOK";
	}
}
```

ANEXO I – Artigo Científico apresentado na MIC2015

An Application for Handheld Devices to Solve Capacitated Vehicle Routing Problems

Carlos Carreto[1], Ricardo Antunes[1]

[1] Unidade de Investigação para o Desenvolvimento do Interior
Instituto Politécnico da Guarda
Av. Dr. Francisco Sá Carneiro 50, 6300-559 Guarda, Portugal
ccarreto@ipg.pt, antenas963@hotmail.com

Abstract

This paper presents an application for handheld devices (smartphones and tablets), to solve capacitated Vehicle Routing Problems. The application implements a visual interactive solution method that integrates the insight and experience of the scheduler, and the power and precision of the heuristics, in an interactive environment. The solution method was implemented to make use of the interesting features and services of the current handheld devices. Results show that despite the restrictions on the processing power of these devices, their functionalities are very interesting to implement visual interactive methods, allowing the common user to easily solve vehicle routing problems.

1 INTRODUCTION

This paper presents the work in progress to develop an application for Android handheld devices (smartphones and tablets) to solve real-life capacitated Vehicle Routing Problems (VRP) [4], based on a visual interactive solution method implemented through a Greedy Randomized Adaptive Search Procedure (GRASP) [6]. The aim of the application is to take advantage of the features and resources of these devices, such as geographical map and location services and the visual interaction capabilities to allow the development of an advanced application that can be used by common users, which is capable of solving real life problems.

The motivation to use a visual interactive method is to integrate the insight and experience of the user, and the power and precision of the heuristics, in an interactive environment. Through interaction the user is able to control the solution process by selecting initial parameters, selecting algorithms and adjusting solutions. The user can also include special knowledge of the real life problem guiding the heuristics towards promising areas of the solution space. An interactive solution method also facilitates the inclusion of constraints and makes the solutions more acceptable because the user participates actively in the solution process.

Previous work [1] has shown that excellent results for the VRP can be obtained by a visual interactive method and GRASP. The method was later extended to tackle different variations of the VRP [2, 7], showing that the method is flexible and adaptable to different types of problems. The application developed uses an adapted version of this solution method, which takes into account the interesting features mentioned before, but also the limitations and restrictions of the handheld devices, especially in what concerns to processing power and memory capacity. A careful implementation of the visual interactive solution method is necessary in order to overcome those limitations.

The main contribution of this work is the implementation of a visual interactive solution method on an application for handheld devices that can solve basic vehicle routing problems.

Although there are some examples of applications for this type of devices that solve routing problems, the authors could only find applications that can handle a unique route.

2 THE VISUAL INTERACTIVE SOLUTION METHOD

The visual interactive solution method has three different phases: *Seeds Selection*, *Routes Construction and Improvement*, and *Solution Refinement*. Figure 1 illustrates the three phases that are described in more detail below.

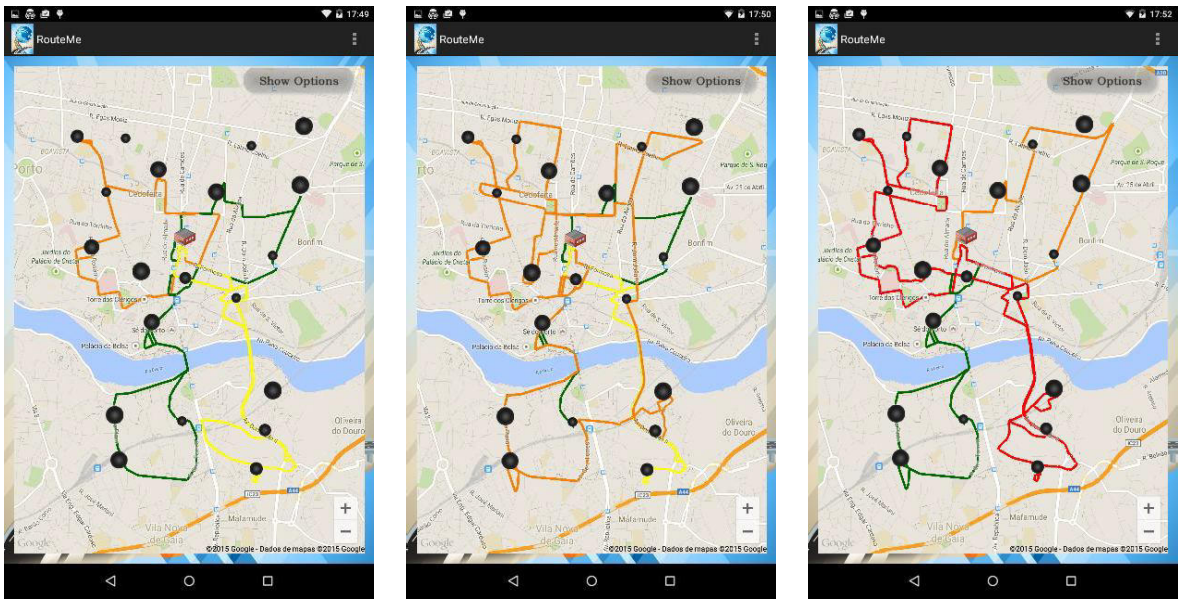


Figure 58 – Illustration of the three phases of the solution method; a) outline of routes after seeds selection; b) a solution obtained after route construction and improvement; c) a solution obtained after solution refinement

The solution method starts with a problem definition where the user creates a new problem instance and selects the locations of the customers and the depot directly in a real map, by using a simple touch interface, or by saving his/her actual location while traveling from one location to another. For each customer the user also gives the corresponding demand. The customers are represented on the map as circles with diameters proportional to their demands. After selecting the customers, the application uses the Google Maps services to get the distances and routes between the customers and the depot, used by the solution method.

In the *Seeds Selection* phase, with the customer locations displayed on a real map, the user selects one or more seed customers in the order that the user would expect them to be visited, to form an outline route. The seed customers are the customers that for some reason the user forces to be serviced by specific vehicles. The user selects the seed customers for each vehicle in accordance with his/her local knowledge and experience of a real life problem, or by identifying certain patterns such as clusters of customers, customers with very high demands or isolated customers. During the selection process the user is helped by the application, which checks for constraint violations related to the capacity of the vehicles. Figure 1 a) shows a typical outline of routes after selection seed costumers.

The *Routes Construction and Improvement* phase, constructs the routes by clustering the remaining customers according to the vehicles defined by the seeds. This clustering is done

without user intervention and is implemented in the construction phase of the GRASP algorithm. The heuristic may find a feasible solution where all the customers have been allocated to the vehicles; or the clustering may result in an infeasible solution, because there was a violation of constraints.

The clustering heuristic allocates each customer to the position in a route where the insertion cost is minimized, giving allocation priority to customers with a more obvious insertion route (given by the difference between the second smallest and the smallest insertion cost), and to customers for which the number of routes they can go on is smallest, and while applying a 2-optimal heuristic [5] in order to reduce the total distance travelled by each vehicle. The clustering heuristic is followed by a local search phase that tries to improve the solution obtained by the clustering, if feasible. The GRASP local search phase implements a one-node interchange procedure with a first-improve policy, which immediately accepts the first interchange that gives a positive saving. In order to speed-up the local search phase, the algorithm implements a *p-Neighbourhood* strategy, similar to the one presented by Gendreau et al [3], that reduces significantly the number of interchanges to consider by selecting as destination routes of the interchanges, only those that are "close" to the customer that is being repositioned. The "closeness" is defined by a metric which corresponds to the distance on the map between customers. Figure 2 b) shows an example of a feasible solution obtained from the initial set of seeds. The solution obtained in the construction phase can be either feasible or infeasible. The solution method saves the best feasible solution at each iteration, but if the heuristic fails to find a feasible solution, it saves the best infeasible solution, which the user can try to improve in the solution refinement phase. The best infeasible solution is defined as the infeasible solution with the minimum total weight unallocated, as this is considered to provide the best potential for finding a feasible solution during the refinement phase.

In the *Solution Refinement* phase, the user can try to refine the solution in order to improve it (especially if it is infeasible), by using interactive tools. With no formal rules to follow, the interactive system we developed is sufficiently flexible to allow creativity and imagination from the user, and different refinement techniques are possible for using these tools. The interactive tools are implemented based on a simple touch interface. The simplest example consists of the removal of some customers from their current routes. This is followed by a new call to the construction and improvement heuristics, allowing these customers to be inserted into different routes. This is a very simple but effective procedure when there is some overlap between routes, and can lead to several different feasible solutions in a very short period of time. Figure 1 c) shows a possible final solution after applying the interactive tools.

The user has several interactive tools he can choose: remove selected customers from a route, delete a selected route, insert a customer in a selected route and return to the seeds configuration. The interactive tools are implemented base on the natural touch screen interface of the handheld devices.

3 CONCLUSIONS

The paper presented a visual interactive method to solve real-life capacitated vehicle routing problems. Interactive methods for vehicle routing are appealing because they integrate the insight and experience of the user, and the power and precision of heuristics in an interactive environment. The method was implemented in an application for handheld devices that is likely to appeal to schedulers who wish to retain some control over the routes that are adopted, especially if there are local constraints on route structure that need to be taken into consideration. Preliminary results show that despite the restrictions on the processing power of the current handheld devices, their functionalities are very interesting to implement visual interactive solution methods, allowing the common user to easily solve vehicle routing problems.

References

[1] Barrie Baker and Carlos Carreto. A visual interactive approach to vehicle routing. *Comput. Operations Research*, 30.3, pp. 321–337, 2003.

[2] Carlos Carreto and Barrie Baker. A GRASP interactive approach to the vehicle routing problem with Backhauls. In Celso Ribeiro and Pierre Hansen, editors, *Essays and Surveys on Metaheuristics*, pp. 185–200, Kluwer Academic Publishers, 2002.

[3] Michel Gendreau, Alain Hertz and Gilbert Laporte. New insertion and post optimization procedures for the traveling salesman problem. *Operations Research* 40.6, pp. 1086–1094, 1992.

[4] Paolo Toth and Daniele Vigo. The vehicle routing problem. Society for Industrial and Applied Mathematics, 2001.

[5] Shen Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44, pp. 2245–2269, 1965.

[6] Thomas Feo and Mauricio Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6.2, pp. 109–133, 1995.

[7] Yazgi Tutuncu, Carlos Carreto and Barrie Baker. A visual interactive approach to classical and mixed vehicle routing problems with backhauls. *Omega*, 37.1, pp. 138–154, 2009.