Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu

José Manuel Afonso Fonseca

MuvOnSchedule
Estimativa e previsão da chegada de um meio de transporte

Outubro de 2011

MuvOnSchedule - Estimativa e previsão da chegada de um meio de transporte
IPV - ESTGV | 2011
José Afonso

Instituto Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu

José Manuel Afonso Fonseca

MuvOnSchedule
Estimativa e previsão da chegada de um meio de transporte

Tese de Mestrado
em Sistemas de Informação para as Organizações

Professor Doutor José Francisco Monteiro Morgado

Outubro de 2011

"…Ser feliz é reconhecer que vale a pena viver, apesar de todos os desafios, incompreensões e períodos de crise. Ser feliz é deixar de ser vítima dos problemas e se tornar um autor da própria história..."

Augusto Cury

Agradecimentos

Em primeiro lugar quero expressar os meus sinceros agradecimentos ao Dr. Francisco Morgado, pela sua disponibilidade e ajuda no desenvolvimento desta tese, que sem dúvida foi crucial para a conclusão deste projeto.

Ao meu colega Eng. Carlos Rodrigues que me ajudou em inúmeras situações e desafios de índole técnica na fase de desenvolvimento e sempre aliado com a sua palavra amiga em momentos mais difíceis.

Também remeto agradecimentos especiais ao Eng. Bruno Luís, que me ajudou a idealizar e estruturar inúmeras funcionalidades inerentes a aplicação móvel, sempre com a sua boa disposição e entusiasmo na discussão das ideias.

Dirijo de igual modo agradecimentos ao Eng. Filipe Manuel Gonçalves, que sempre incentivou a análise crítica e aportando sugestões construtivas, revelando-se fundamental para a evolução deste projeto.

Remeto agradecimentos ao meu irmão Fernando Manuel e a Eng. Joana Mendes, que em inúmeras ocasiões me ajudaram na revisão de alguns textos e outros aspetos importantes na redação deste documento.

Quero agradecer a minha família, pelos dias que estive menos disponível e por toda a compreensão prestada que foi sem dúvida crucial para a entrega atempada deste projeto.

Finalmente um muito obrigado a todos os que de alguma maneira contribuíram direta ou indiretamente neste projeto tornando-o possível.

Resumo

A atual conjuntura económica tem moldado a forma como as pessoas se deslocam, motivando a maior parte da população a usar os transportes públicos por forma a aumentar o conforto e reduzir custos. Contudo, existem percursos onde é necessário fazer transbordos sendo ainda mais crucial melhorar a gestão do tempo de forma a conciliar todas as ligações, para que o utente consiga chegar ao local de destino à hora pretendida.

Na Europa, em países como Holanda ou Bélgica, não é recente a adoção de planificadores intermodais de itinerários. Estes agregam informação de vários transportes coletivos disponibilizando ferramentas de apoio ao utente para incentivar, facilitar e agilizar a planificação de viagens recorrendo a esta modalidade de transporte.

Com este trabalho pretende-se estudar a exequibilidade de adoção deste tipo de plataformas em Portugal. Para tal foi realizado um inquérito por questionário de modo a verificar quais as funcionalidades pretendidas pelos utilizadores, para a planificação das suas viagens utilizando transportes públicos. Também foi efetuada uma análise a nível técnico acerca da possibilidade de integração de diversas ferramentas com um planeador intermodal de itinerários, com o objetivo de construir um protótipo de aplicação móvel de auxílio ao utente para a pesquisa e planeamento de viagens.

Abstract

The actual economic situation, caused the change in the way that many people move on the cities, due to this fact a big majority of the travelers, began to use public means of transport, both because these means are comfortable enough to make a day by day journey and because they are much more cheaper. However, there are routes where you need to use different means of public transport, on those cases improve the time management is crucial for the user, or he might arrive late on his destination.

Countries in Europe like Belgium and Holland, have adopted intermodal route planners that gather different types of information, both form public and private means of transport, and also give tools to the user, for him to plan the journey with the less effort possible.

The aim of this work is to analyze if this kind of platform could be introduced in Portugal. So a survey was made, in order to find what functionalities the user would hope to find in an application that could help him to plan a journey using public means of transport. At technical level other analysis was made in order to integrate the different tools that an intermodal route planner, could or should have, to help the user on the planification of a journey using public means of transport.

Índice

Índice de tabelas

Índice de figuras

Acrónimos

API	Application Programming Interface
CRUD	Create, Read, Update, Delete
GTFS	General Transit Feed Specification
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
OTP	Open Trip Planner
J2EE	Java 2 Enterprise Edition
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
URI	Uniform Resource Identifier
WSDL	Web Service Description Language
XML	eXtensible Markup Language

1 Introdução

Graças aos avanços tecnológicos que assistimos diariamente é possível encontrar a preços razoáveis, equipamentos móveis com capacidades interessantes, tanto ao nível de: processamento, navegação assim como de geolocalização, com margens de erro mínimas. Estas funcionalidades aliadas à crescente conectividade com a rede de dados, permite pensar mais além e idealizar projetos que aproveitem essas potencialidades para informar e facilitar deslocação de pessoas que escolhem diariamente os meios coletivos de transporte para se deslocarem.

O projeto enquadra-se na prototipagem e desenvolvimento de uma plataforma, que permita a utilizadores de transportes públicos e privados de Portugal planificar uma viagem de forma rápida e eficaz de porta a porta, fazendo uso de uma aplicação móvel integrada com um planeador intermodal de itinerários, que possibilite aos utentes o seu uso em qualquer lugar dando-lhe informações úteis sobre os meios de transporte disponíveis, entre dois pontos geograficamente distantes.

1.1 Motivação e objectivos

Atualmente existem sistemas de cálculo de itinerários como o Google Maps [11] que possibilitam a pesquisa de trajetos entre uma localização de origem e de destino, mostrando informação de alguns percursos envolvendo autocarros e metro nas principais cidades portuguesas (Lisboa, Porto), contudo estas soluções foram desenhadas e pensadas para serem utilizados através de um computador pessoal obrigando o utente a uma planificação prévia antes de seguir viagem.

Por outro lado existem soluções proprietárias como por exemplo Garmin-Asus Nuvifone A10 [5], que permitem a navegação pedestre (uma espécie de GPS para transeuntes) e possuem integração com alguns dos principais meios de transporte europeus. No entanto não permitem a filtragem de trajetos tendo como base critérios mais finos, como por exemplo se o meio de transporte faz serviço nocturno assim como outras necessidades do utilizador.

Verifica-se assim, que existe uma lacuna na integração da informação referente aos horários e rotas das diferentes empresas nacionais de transportes colectivos.
O objectivo do trabalho consiste em configurar um servidor de planeamento intermodal de itinerários e planificar/construir um protótipo de aplicação móvel *open-source* que permita a pesquisa de trajetos entre dois pontos geográficos, respeitando as restrições previamente definidas pelo utilizador, como por exemplo qual a rota mais rápida, a rota com menor distância percorrida a pé, quais as rotas que disponibilizam serviço nocturno, etc. e envolvendo um ou mais meios de transporte, permitindo assim ao utilizador a planificação da sua viagem em qualquer lugar de forma rápida e eficiente.

Com isto pretende-se também alargar e melhorar a partilha e consulta de informações sobre os horários de funcionamento dos meios de transporte entre as cidades portuguesas, facilitando e promovendo a utilização de meios públicos de transporte.

De forma a dar resposta aos objetivos propostos foram estabelecidos os seguintes passos:

- Análise de soluções/aplicações existentes no mercado, relacionadas com a pesquisa e cálculo de trajetos envolvendo a apresentação de horários entre dois pontos geográficos usando meios de transporte públicos;
- Realização de um inquérito a utilizadores regulares de meios de transporte públicos, para auferir qual o grau de penetração dos *smartphones* neste nicho de mercado assim como perceber quais seriam as funcionalidades mais interessantes numa aplicação orientada para o cálculo de trajetos;
- Desenvolver um protótipo de aplicação móvel que possibilite a integração das informações sobre horários e carreiras disponíveis numa área geográfica, por forma a permitir o cálculo de um ou vários trajetos que vão ao encontro das necessidades do utilizador.

1.2 Organização da dissertação

O presente documento encontra-se estruturado em cinco capítulos.

No capítulo 1 é a introdução, onde se enquadra o tema e apresenta-se o contexto onde o trabalho realizado se insere. São apresentadas as motivações que impulsionaram a realização do projeto e quais os objetivos a ser atingidos, incluindo também a organização da tese.

No capitulo 2 apresenta-se o estado da arte, onde se analisam quais os produtos que atualmente existem no contexto onde se enquadra a tese e onde o projeto pode ser efectivamente uma mais valia.

No capítulo 3 são identificadas as questões de investigação e indicada qual a metodologia seguida para a criação da tese, isto é a criação de um inquérito por questionário e como foi levado a cabo o levantamento e análise dos dados recolhidos.

O capítulo 4 trata sobre o desenvolvimento propriamente dito, especificando quais foram as ferramentas usadas e inclusive porque foram escolhidas. É falado sobre a arquitectura do sistema, como está constituído o protótipo a nível de classes, como foi instanciado o servidor do planificador intermodal de itinerários e finalmente são apresentadas imagens dos ecrãs do protótipo.

Por último no capítulo 5 são apresentadas as conclusões, qual o trabalho futuro a desenvolver e a bibliografia.

2 Estado da arte

Antes de apresentar o trabalho realizado, é importante fazer uma introdução sobre o que existe na área e os conceitos envolvidos para a realização do projeto apresentado.

2.1 Planificação da viagem

Em Portugal os utilizadores de transportes públicos sentem dificuldade em planificar e escolher de forma rápida e eficiente um ou vários trajetos entre cidades, envolvendo um ou mais operadores de transporte. Esta problemática afeta utilizadores esporádicos de trabalho e de lazer, nacionais e estrangeiros.

Deste modo os operadores na tentativa de prestar um melhor serviço aos seus clientes, investem no desenvolvimento de calculadores de itinerários próprios que são disponibilizados gratuitamente aos utentes. Contudo estas aplicações, como por exemplo o Itinerarium (http://www.itinerarium.net), geralmente são vocacionadas para o ambiente *web* e apresentam algumas limitações, como por exemplo, a não inclusão de informação de outros operadores, sendo pouco eficientes e/ou difíceis de utilizar e por conseguinte não geram uma mais valia real para a própria empresa e para os utentes.

Apesar de não ser um calculador de itinerários no sentido restrito da palavra o site da CP (www.cp.pt) que vemos na Figura 1, é um bom exemplo de informação útil para o planeamento de viagens onde é possível pesquisar horários, preços, avisos e outras informações importantes no âmbito dos serviços prestados por esta empresa.

Figura 1: Página inicial da empresa CP

Como é possível ver na Figura 2, é permitido ao utilizador escolher a estação de partida, chegada e a hora pretendida para planificar a sua viagem. No resultado da pesquisa é retornada uma tabela com diversas possibilidades de horas de partida, tipos de comboio/serviço e preço. Neste caso foi efetuada uma pesquisa entre a localidade de Estarreja e Coimbra, sendo apresentada uma lista detalhada com os diversos comboios disponíveis e os respetivos transbordos, assim como o preço dos

bilhetes de embarque. No caso dos comboios do tipo Alfa Pendular ou Intercidades é possível comprar por via eletrónica o bilhete de viagem.

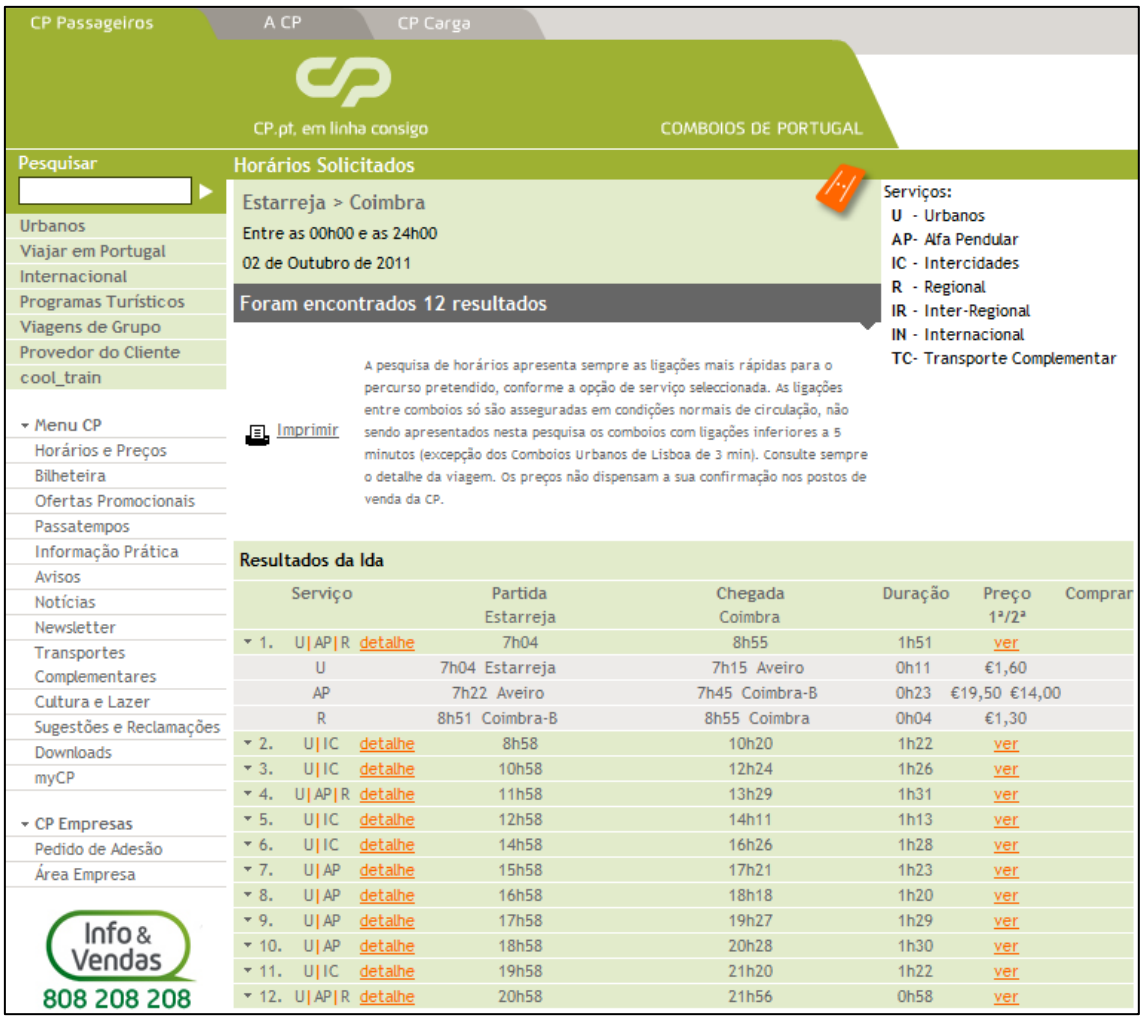


Figura 2: Resultado de uma pesquisa entre duas localidades

O TransPor (www.transpor.pt) é o único calculador de itinerários a nível nacional. Foi desenvolvido pelo Instituto de Mobilidade e Transportes Terrestres (IMTT), no âmbito do projeto SIGTII e apoiado pelo Fundo Europeu de Desenvolvimento Regional (FEDER). Como podemos observar na Figura 3, permite apenas o planeamento de viagens entre localidades e não de porta-a-porta, não inclui operadores intermunicipais tais como a Carris ou os STCP, inclui apenas uma alternativa de viagem, não são mostradas paragens intermédias e a utilização é difícil.

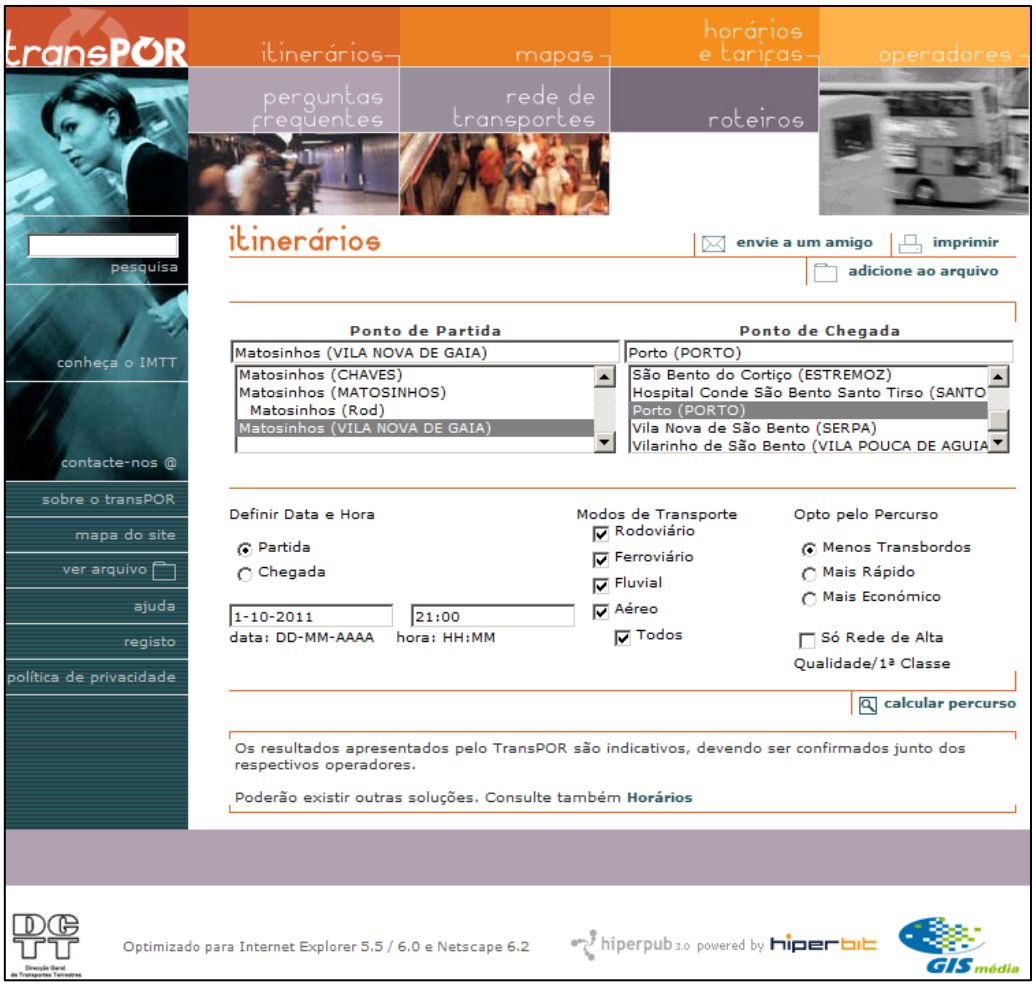


Figura 3: Formulário de pesquisa de itinerários do calculador de itinerários TransPor

Como podemos visualizar na Figura 4, foi efetuada uma pesquisa entre Vila Nova de Gaia e o centro da cidade do Porto, tendo-se obtido um único resultado onde só são envolvidos os serviços disponibilizados pela CP e não são mostradas as opções disponíveis por parte dos STCP ou do metro do Porto.

A prova da falta de popularidade da ferramenta é o número de utilizadores ser demasiado baixo para ser medido pela empresa de métricas da Internet Alexa (www.alexa.com).

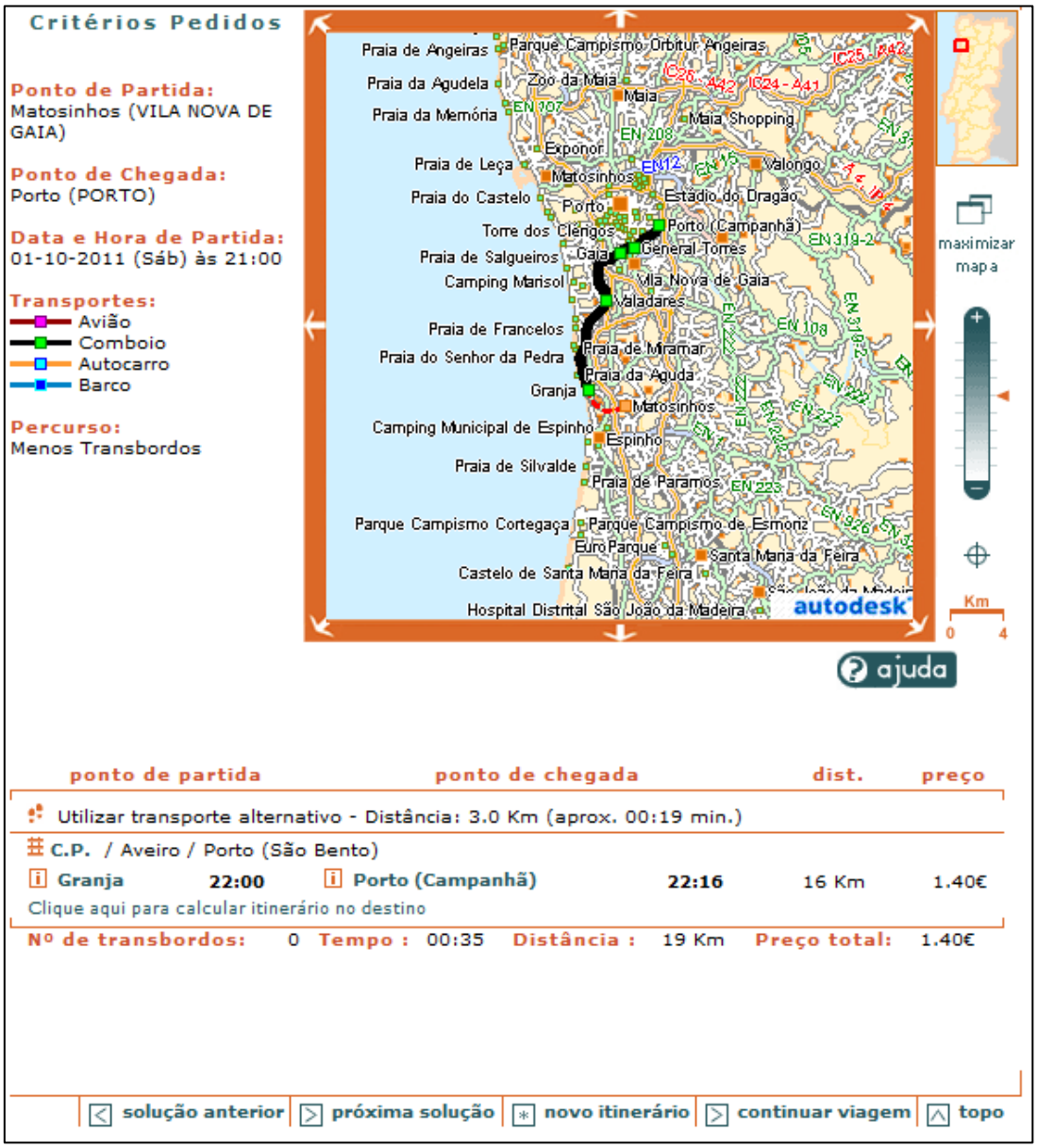


Figura 4: Resultado de uma pesquisa utilizando a ferramenta TransPor

Ao nível das Áreas Metropolitanas de Lisboa e Porto (AML e AMP) os utilizadores têm ao seu dispor ferramentas mais completas e úteis. Em Lisboa, pode ser usado o Transporlis (http://www.transporlis.sapo.pt), um sistema de informação multimodal da AML, que resultou da parceria entre os diversos operadores e Câmaras Municipais e a sua página inicial é mostrada através da Figura 5.

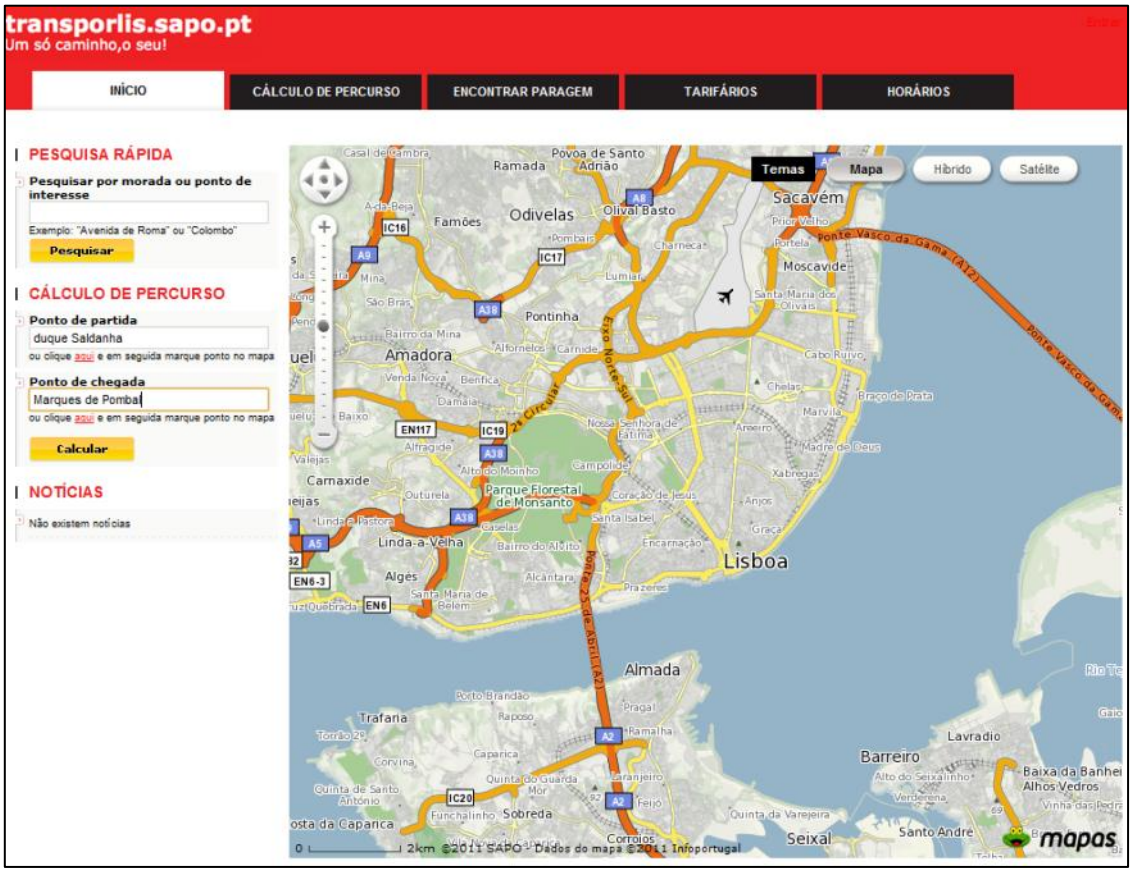


Figura 5: Página inicial da ferramenta Transporlis

Aqui é já possível aos utilizadores planear uma viagem de porta-a-porta entre moradas da AML. Como podemos ver na Figura 6, a informação fornecida em cada pesquisa é mais útil estando disponível o número de transbordos, a quantidade de CO_2, o preço, o tempo e a distância percorrida. Os resultados obtidos incluem a integração dos diversos meios de transporte disponíveis na AML como por exemplo a Carris, CP, Fertargus, Metropolitano de Lisboa**,** Rodoviária de Lisboa, Scotturb, Transportes Sul do Tejo, Transtejo e Vimeca. Carece de integração com outras redes de transporte público existentes fora de Lisboa.

A informação fornecida é útil para utilizadores esporádicos de transportes públicos mais insuficiente para um utilizador frequente. É apenas fornecido o cálculo de uma viagem e para procurar a horas diferentes é necessário refazer a pesquisa, tornando o processo moroso.

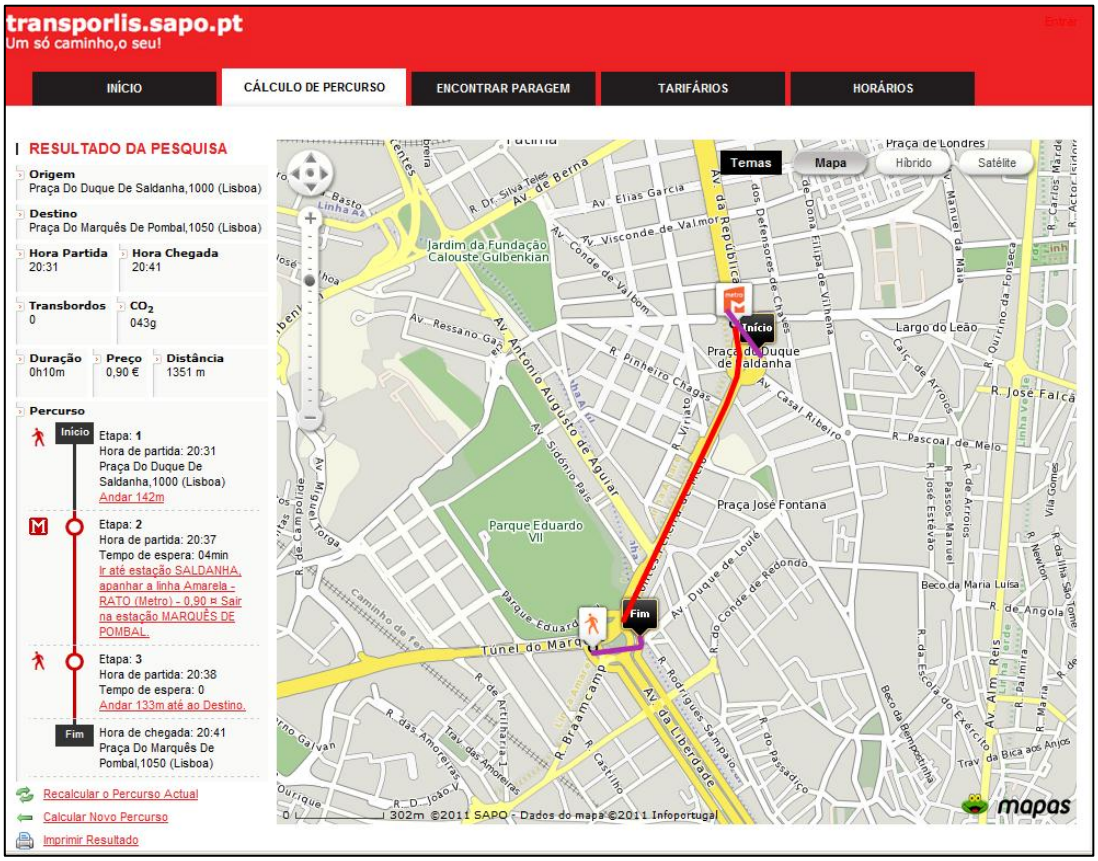


Figura 6: Resultado após pesquisa de trajetos usando a ferramenta Transporlis

Na AMP (área metropolitana do Porto) a solução encontrada foi o Itinerarium (http://www.itinerarium.net) e que podemos ver através da Figura 7.

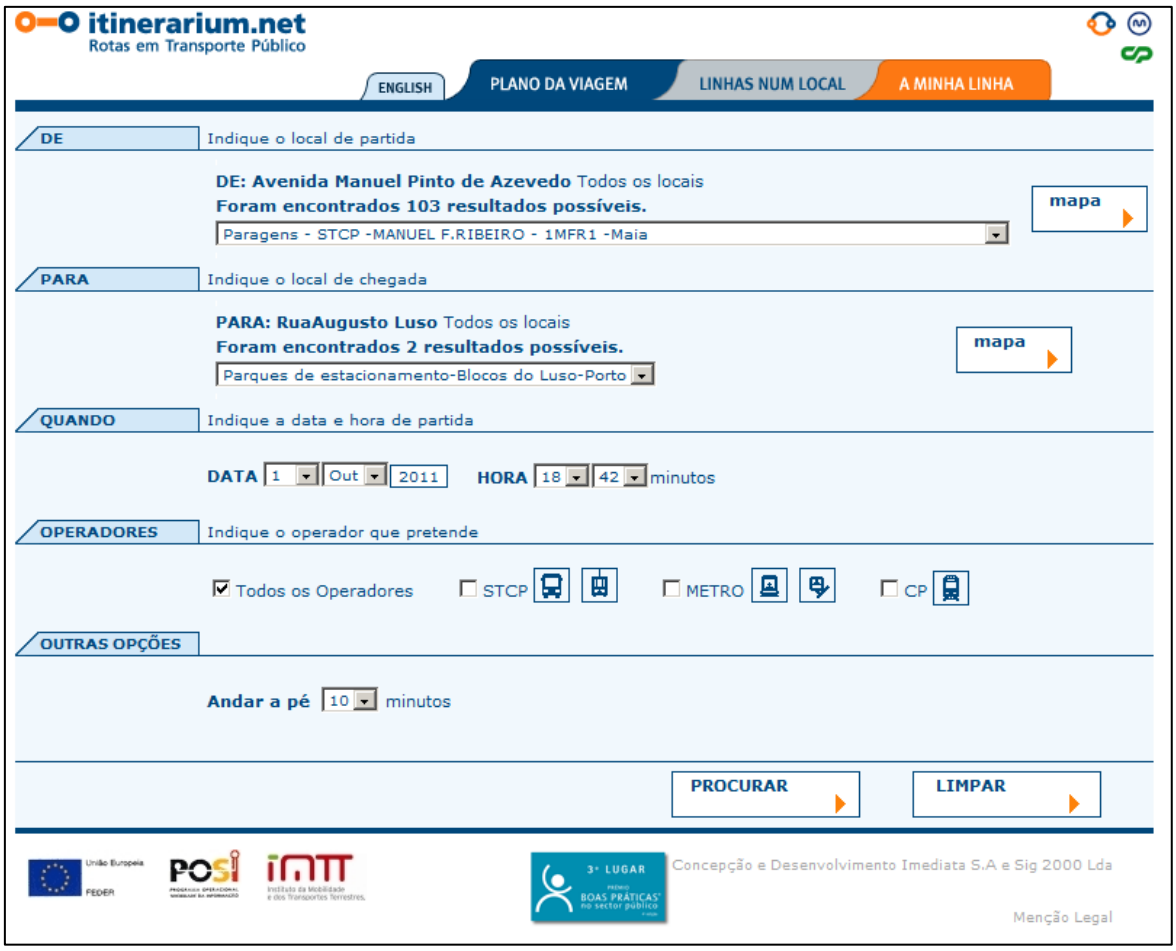


Figura 7: Página principal da ferramenta Itinerarium

Esta ferramenta agrega num único local a informação da STCP, CP Porto e Metro do Porto. Os operadores privados foram deixados de fora deste processo. A informação é mais clara no Itinerarium apesar da utilização, em particular a selecção dos locais de partida e chegada ser complicada.

Podemos verificar na Figura 8 o aspeto dos resultados de uma pesquisa de trajetos através do Itinerarium onde é possível obter informação sobre o número de zonas a percorrer, preço dos bilhetes, tempo aproximado de espera, custo total da viagem, etc. Mais uma vez é apenas fornecida uma opção e é necessário reiniciar a pesquisa caso seja necessário alterar qualquer parâmetro [3].

À semelhança do Transporlis, verificamos que carece de integração com outros meios de transporte público fora da cidade do Porto.

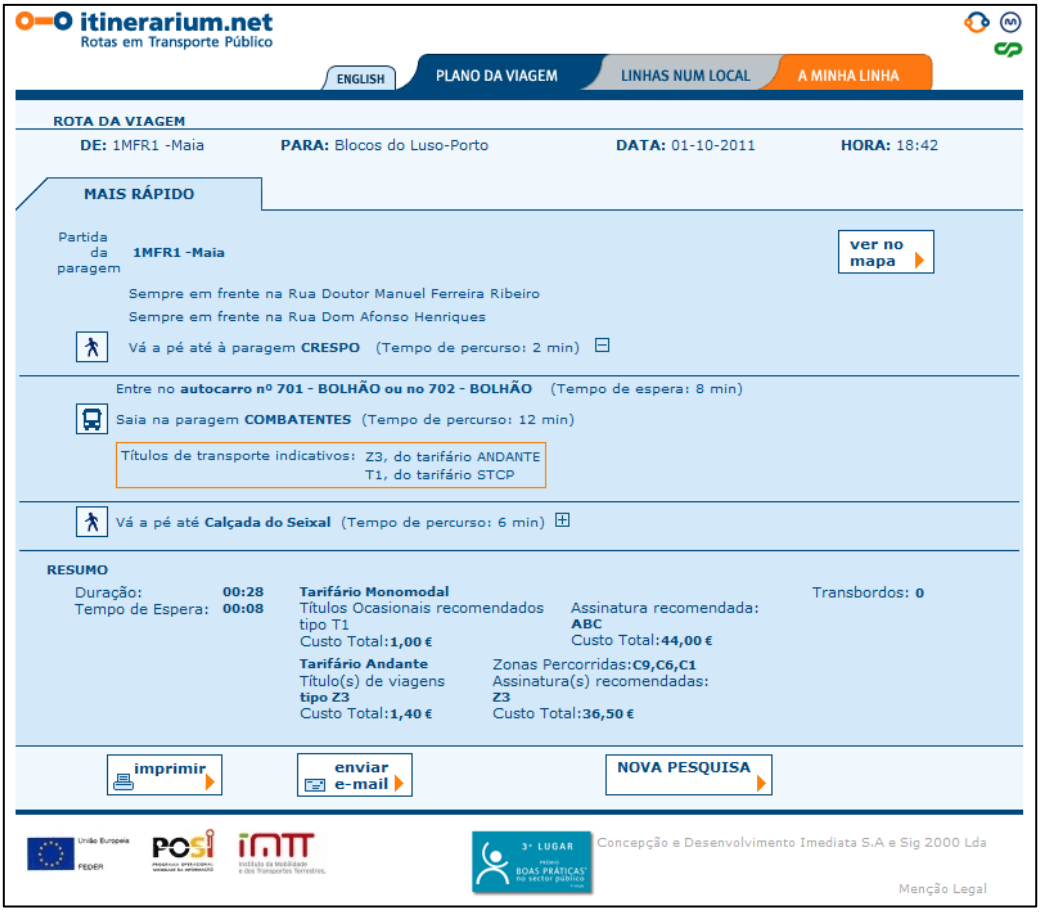


Figura 8: Resultado de pesquisa na ferramenta Itinerarium

O Google Maps Transit é um planificador de itinerários que ajuda os utilizadores a chegar ao seu destino, proporcionando informações fáceis de seguir fazendo uso dos meios públicos de transporte disponíveis. Através da Figura 9 é possível ver a página inicial de pesquisa de trajetos envolvendo transportes públicos.

Figura 9: Página inicial do Google Maps Transit

Como o transporte público é normalmente o serviço de transporte mais utilizado em qualquer cidade/estado, o Google Transit é muito útil, pois dá informações detalhadas sobre trânsito, sobre as estações, horários, duração da viagem (estimado) e até mesmo informações sobre as tarifas [27] como podemos ver na Figura 10.

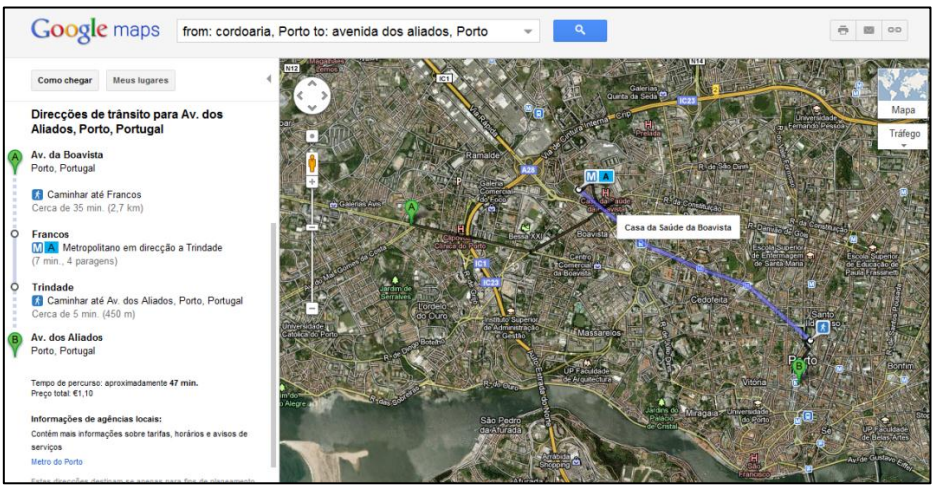


Figura 10: Resultado de uma pesquisa de trajetos usando o Google Maps Transit

Este serviço implementa diversas funcionalidades como por exemplo:

- **Obtenção de direcções:** Podemos pesquisar um trajeto a pé, de carro, bicicleta ou transportes públicos entre duas localizações (origem e destino), adaptar o caminho sugerido com ou sem portagens e fazer sinalização do mesmo (*highlight*). Permite ainda escolher qual o transporte público a considerar e o caminho com menor distância a percorrer a pé;
- **Partilha e armazenamento de mapas:** É possível guardar, partilhar e tornar público um mapa com uma direção origem e destino já previamente sinalizada (*highlight*);

Quando usada a funcionalidade **obtenção de direções**, na vertente meios de transporte, existe integração de informação com alguns transportes públicos. Por exemplo no caso da cidade do Porto encontramos informação do metro do Porto mas não sobre os autocarros STCP. Verificou-se que só em algumas das principais cidades existe informação sobre transportes públicos e os trajetos apresentados apresentam informação do custo em (tempo, dinheiro, distância). Nesta funcionalidade não estão disponíveis por exemplo filtros que permitam escolher os trajetos pelas seguintes condições:

- Rota mais rápida;
- Uso de veículo para deficientes motores;
- Tempo previsto de chegada;
- Permitir utilização de bicicleta.

Por outro lado, verificamos através da Figura 11, que se realizarmos uma pesquisa de trajetos entre duas cidades, como por exemplo Av. dos Aliados, Porto até Repeses, Viseu, não encontramos cobertura possível utilizando transportes públicos.

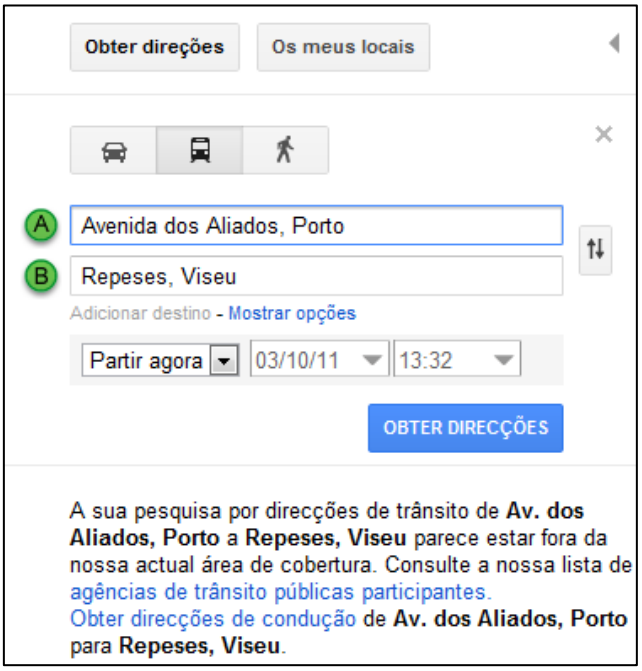


Figura 11: Pesquisa de trajetos entre a cidade do Porto e Viseu

Foi constatado que os calculadores de itinerários atualmente existentes estão direcionados à prestação de serviços num único sentido: dos operadores para os utilizadores. Normalmente é esperado que os utentes utilizem o portal na perspetiva de planificar as suas viagens, sendo estas no geral domésticas. Dos portais analisados, naqueles existentes em Portugal e mesmo quando olhando apenas para os utilizadores internos a informação fornecida fica muito aquém da que é necessária.

2.2 Aplicações de navegação pedestre

A análise de navegação terrestre recaiu sobre a aplicação Garmin Navigation CityXplorer disponível no equipamento Garmin-Asus nuvifone A10, para o sistema operativo Android [5]. Esta aplicação permite a navegação pedestre de um utilizador, pela introdução do ponto de origem e do ponto de destino à semelhança do que já existe nos GPS para automóveis convencionais, também possui integração com alguns

meios públicos de transporte nas principais cidades europeias. A vantagem deste sistema é a navegação *off-line* isto é, não é necessária a utilização da rede de dados para a obtenção de mapas nem para a indicação do caminho no mapa. Apesar de ser uma ótima solução, o número de cidades portuguesas abrangidas pela solução é limitada e só pode ser utilizado em equipamentos móveis da marca Garmin.

A plataforma OVI da Nokia [21], lançou recentemente uma versão OVI maps, que permite a navegação GPS e a navegação pedestre de forma gratuita, que está disponível para equipamentos Nokia dotados de GPS como vemos na Figura 12.

Esta solução permite ao utilizador pedestre navegar entre dois pontos, escolhendo o caminho mais curto seja utilizando pontes, parques, atalhos pedestres, etc. Ainda permite a personalização da linguagem da aplicação e definição de parâmetros como sinais sonoros ou vibração, que indicam ao utilizador onde deve virar ou simplesmente corrigir a sua rota.

Os mapas, à semelhança de outras soluções apresentadas, são de navegação *off-line* podendo ser adquiridos de forma gratuita no site da Nokia: http://maps.nokia.com, contudo carece de uma verdadeira integração com os meios de transporte públicos, como por exemplo mostrar os horários de autocarros e comboios mais próximos a um determinado local, etc.

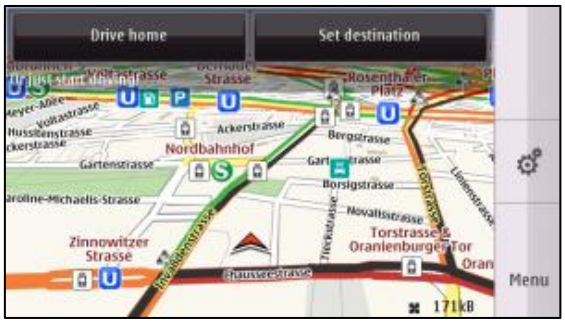


Figura 12: Nokia Ovi maps com suporte a navegação pedestre [21]

Outra aplicação que permite navegação pedestre tem como nome Pdxtrian [4]. É uma aplicação gratuita para o sistema operativo Android e está disponível para os utilizadores dos transportes públicos da cidade de Portland (Trimet). Permite consultar

um mapa das paragens de autocarro e metro e outros meios de transporte disponíveis na zona.

Também exibe uma listagem dos meios de transporte que estão a chegar a uma paragem como podemos observar na Figura 13 e utiliza o GPS do dispositivo móvel para indicar quais as paragens que estão próximas de um determinado local.

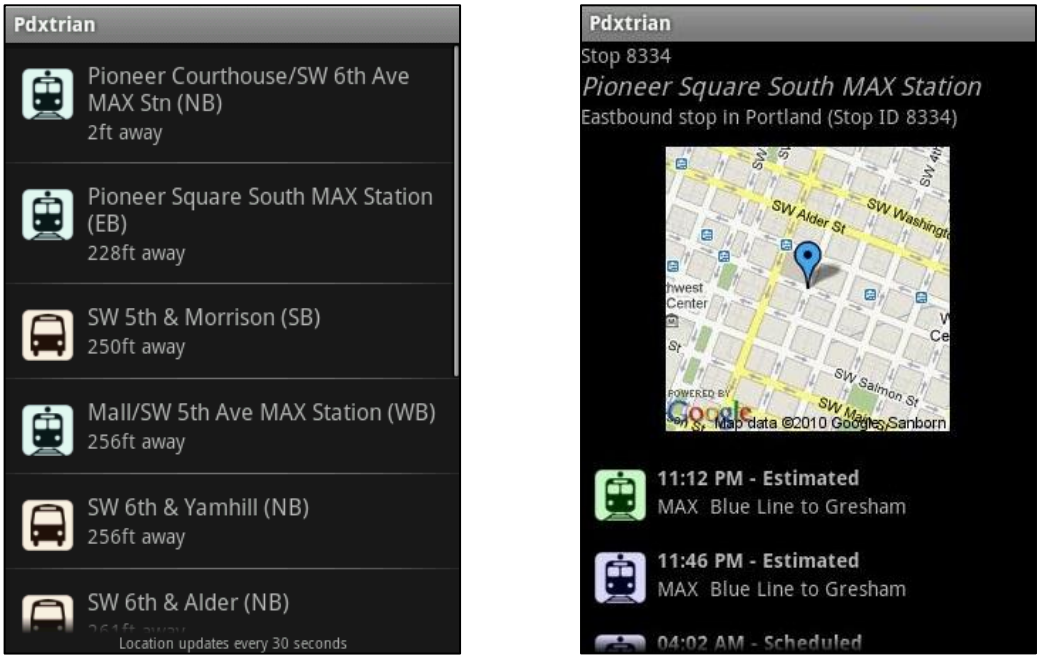


Figura 13: Aspeto da aplicação Pdxtrian [4]

3 Abordagem e processo de investigação

Este capítulo aborda os aspectos referentes ao estudo desenvolvido, mostrando as fases que o compõem, descrevendo o processo seguido e qual a sua finalidade, e a forma como foram recolhidos os dados e posteriormente analisados.

3.1 Questões de investigação

Para que o projeto se revele assertivo, torna-se crucial identificar quais as funcionalidades que vão ao encontro das necessidades dos utentes dos meios públicos de transporte e que facilitam o cálculo de trajetos e a consulta de horários, fazendo uso de uma aplicação para dispositivo móvel.

É importante identificar qual a adesão dos utilizadores dos meios de transporte à utilização de *smartphones,* no auxílio à planificação das suas deslocações, e qual o impacto do aparecimento de novas *marketplaces* com aplicações direccionadas para este tipo de público. Também é importante validar quais as funcionalidades mais apreciadas pelos utentes para que a aplicação vá ao encontro das suas necessidades.

3.2 Instrumento de investigação

O método de investigação utilizado foi o inquérito através de um questionário *on-line* e dirigido a estudantes do ensino público universitário e outros utilizadores, que usam diariamente os meios colectivos de transporte da sua respectiva cidade.

Na concepção do questionário procurou-se que este fosse objectivo e de instrução limitada, encontrando-se estruturado em vários grupos para facilitar ao inquirido a sua resposta.

No primeiro grupo de perguntas pretende-se a recolha de informação sobre a faixa etária e outros dados estatísticos de carácter pessoal através das seguintes questões:

- Qual a sua faixa etária;
- Qual o sexo;
- Quais as suas habilitações literárias;
- Possui um *smartphone*.

O segundo grupo visa avaliar informações sobre o *smartphone* e a sua utilização por parte do inquirido, fazendo uso das seguintes questões:

- Qual o sistema operativo instalado no dispositivo;
- Quantas vezes utilizou um *marketplace* de aplicações móveis;
- Que tipo de aplicações fez *download*;
- Qual o máximo que pagaria por uma aplicação móvel;
- Assumindo que estaciona o seu veículo particular, qual a distância que está disposto a andar a pé.

No último grupo são apresentadas várias funcionalidades passíveis de serem incluídas numa aplicação móvel para o cálculo de trajetos, sendo questionado qual o seu grau de importância usando a escala de Likert[1] (1 - Discordo totalmente, 2 - Discordo parcialmente, 3 - Indiferente, 4 - Concordo parcialmente, 5 - Concordo totalmente).
As funcionalidades apresentadas foram as seguintes:

- Com base na minha localização, indicar os meios de transporte disponíveis mostrando quanto tempo falta para a partida da respectiva carreira;
- Permitir a pesquisa e filtragem de trajetos possíveis entre dois pontos pelo seu custo seja em dinheiro ou tempo;
- Indicar os trajetos com menor distância percorrida a pé;
- Filtrar os transportes que permitem deslocação com bicicleta;
- Mostrar percursos optimizados para pessoas de mobilidade reduzida;
- Indicar os meios de transporte com menos transbordos entre o ponto origem e de destino;
- Receber notificações sobre possíveis atrasos;
- Receber notificação minutos antes da chegada do meio de transporte;
- Baseado na minha localização, calcular os percursos possíveis para voltar a casa;
- Mostrar nos percursos os pontos de interesse de uma cidade usando transportes públicos;

[1] Rensis Likert, criador da escala de resposta psicométrica usada comumente em questionários, usada habitualmente em pesquisas de opinião.
Likert scale. 2011 [cited 2011 September]; Available from: http://en.wikipedia.org/wiki/Likert_Scale

- Mostrar os trajetos com serviço nocturno;
- Alarme que indica ao utilizador que chegou ao destino.

Finalmente o interveniente é encorajado a partilhar as suas sugestões de outras novas funcionalidades que possam enriquecer a aplicação.

3.3 Recolha de dados

O inquérito por questionário foi difundido, através de um formulário *on-line* por correio electrónico pelas diversas universidades e institutos politécnicos do país. Decorreu entre o dia 09 de Dezembro de 2010 até dia 15 de Janeiro de 2011.

No *email* enviado aos inquiridos, foi colocada hiperligação[2] que dá acesso ao questionário on-line fazendo-se acompanhar com uma breve descrição da finalidade do inquérito e deixando claro que toda a informação recolhida é anónima e é salvaguardada a identidade das pessoas inquiridas.

3.4 Análise dos resultados obtidos

Iniciando a análise é registado que o número de respostas obtidas contabiliza um total de 114, sendo todas estas obtidas recorrendo à internet como meio de comunicação.

Os resultados recolhidos foram alvo de uma análise estatística descritiva, possibilitando a sumarização dos dados em vários gráficos de fácil leitura e compreensão.

[2] Questionário em https://docs.google.com/spreadsheet/viewform?formkey=dDRvOWlUZGR4aGtfVnhXdnMzaUtZOEE6MQ

Analisando o gráfico correspondente a Figura 14, onde é ilustrada a faixa etária alvo do estudo, verifica-se que a média de idades dos inquiridos utilizadores dos transportes públicos oscila entre os 18 e os 49 anos, mostrando uma adesão significativa a este tipo de serviço, as pessoas entre os 25 aos 34, que corresponde à faixa etária dos jovens adultos portugueses.

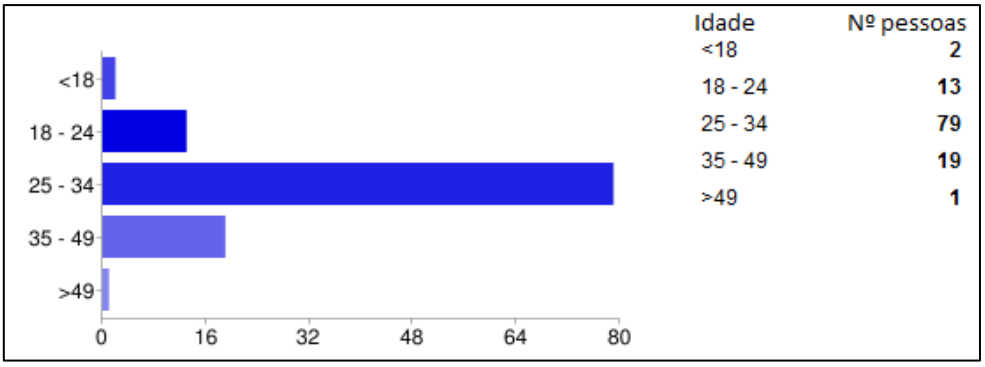


Figura 14: Faixa etária dos participantes

De seguida podemos apreciar através da Figura 15, que o género mais participativo na resposta do questionário foi o masculino, aparecendo em maior número.

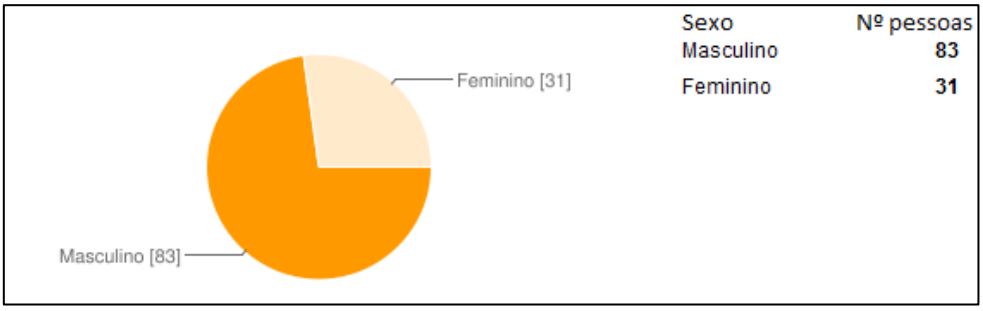


Figura 15: Relação entre os inquiridos do género feminino e masculino

No âmbito das habilitações literárias, podemos verificar que os utilizadores na sua maioria possuem pelo menos o 12º ano como se apresenta na Figura 16.

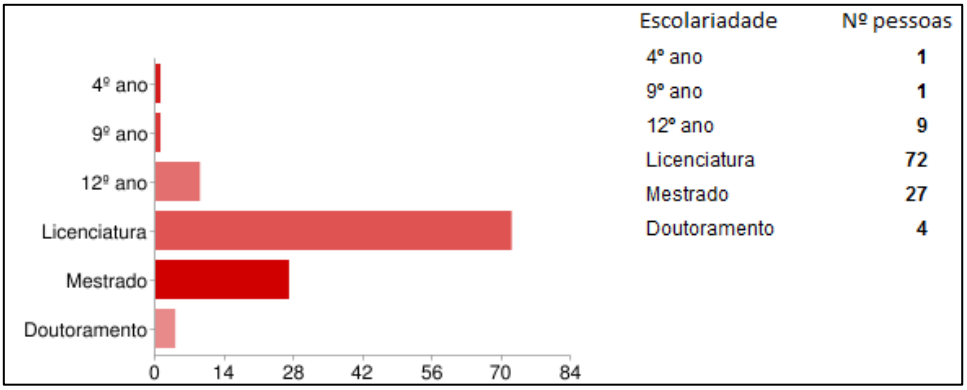


Figura 16: Escolaridade dos intervenientes

A Figura 17, regista que mais de metade das pessoas envolvidas no inquérito não possuem *smartphone*, apesar disso existe um número significativo de pessoas que têm na sua posse este tipo de equipamento.

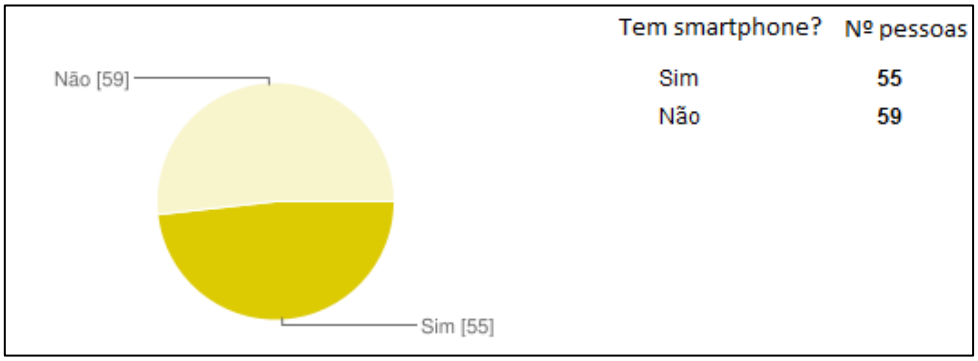


Figura 17: Utentes que possuem smartphone

Ao responder à pergunta referenciada na Figura 17, caso a resposta fosse negativa, a pessoa alvo do inquérito é convidada a responder às perguntas mostradas na Figura 18 e Figura 19, que têm como intuito perceber se o inquirido pretende em breve adquirir um *smartphone* e qual a razão de ainda não ter um equipamento destes.

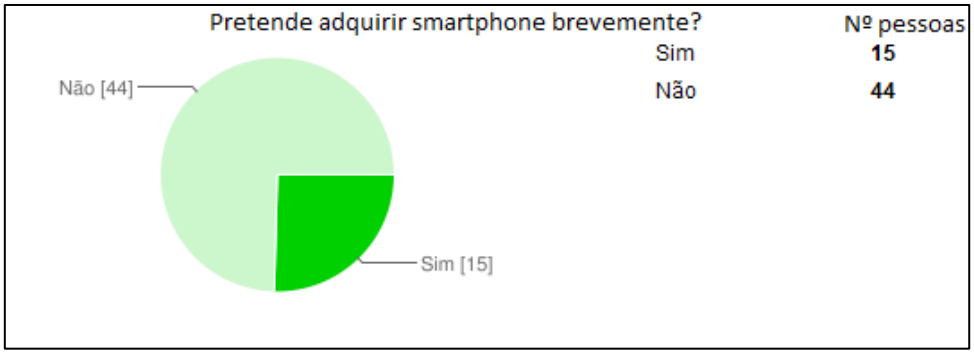


Figura 18: Número de utilizadores dispostos a comprar um *smartphone* brevemente

Como se vê na Figura 18, a maioria dos inquiridos não possui um *smartphone*. Verifica-se que, 44 pessoas não pretendem comprar este tipo de equipamento, mas 15 pessoas demonstram intenção de adquirir um.

Já na Figura 19 verificam-se quais as razões dos inquiridos mais céticos nas vantagens e potencialidades dos smartphones. Podemos destacar que 25 pessoas afirmam que não precisam deste tipo de dispositivos e que 18 pessoas acham que são muito caros.

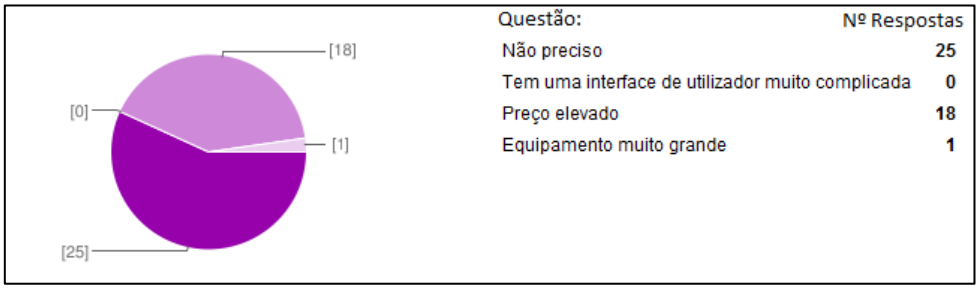


Figura 19: Razão pela qual o inquirido não tem *smartphone*

Voltando à análise das respostas dos inquiridos que possuem *smartphone*, pode-se verificar através da Figura 20, que o sistema operativo preferido é claramente o Android, não ficando muito atrás o iOS da Apple que atualmente equipa os Iphones.

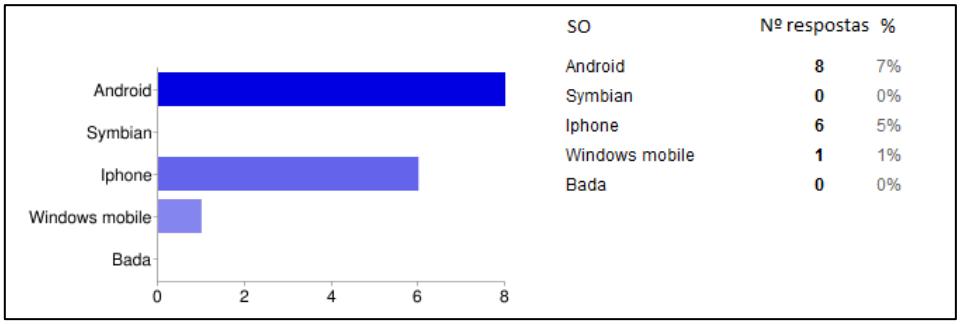


Figura 20: Sistema operativo preferido

Analisando a Figura 21, podemos observar que equipamentos adquiridos pelos intervenientes possuem na sua maioria o sistema operativo Android e Windows mobile. É importante salientar que existe uma percentagem significativa de equipamentos com o sistema operativo Symbian e iOs no grupo estudado.

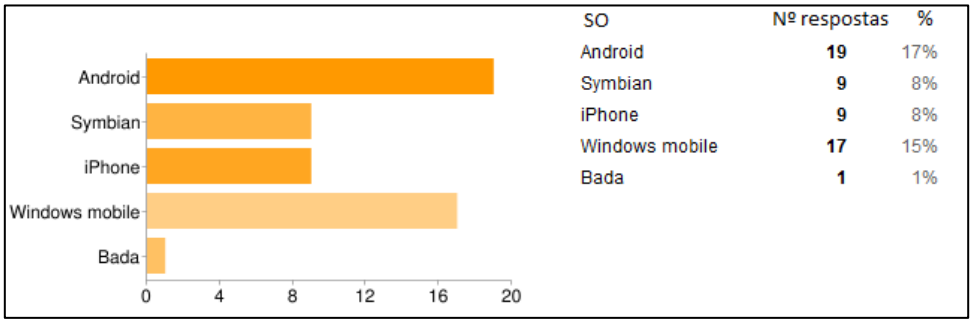
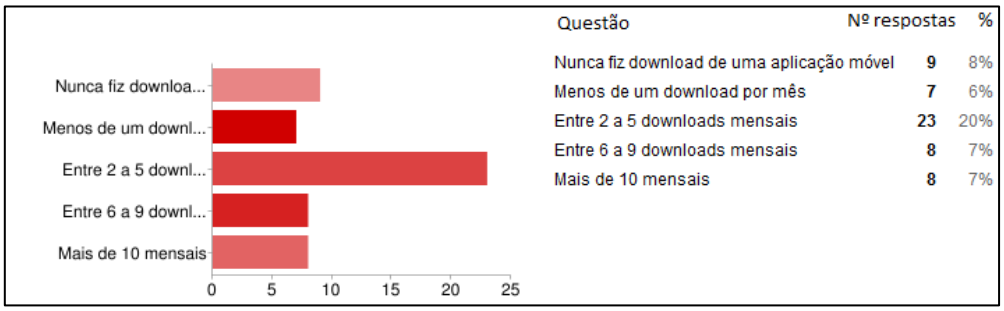


Figura 21: Sistema operativo instalado no *smartphone*

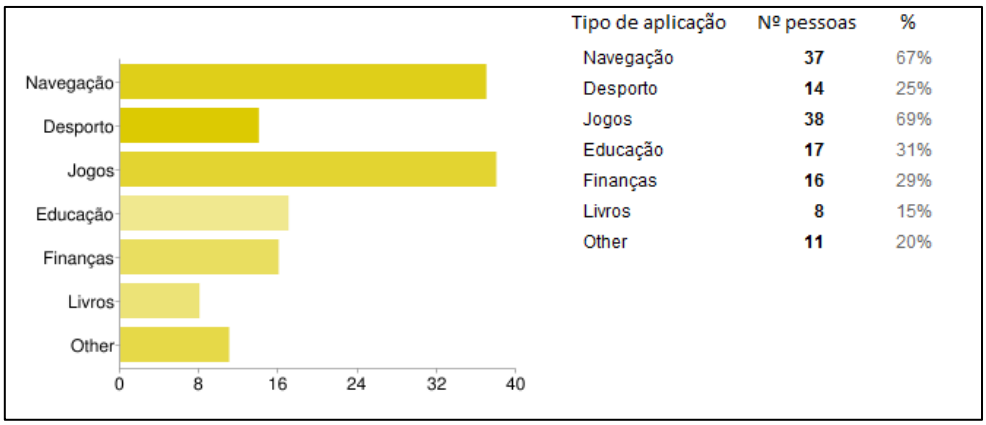
No que diz respeito a utilização de uma *AppStore* para fazer *download* de aplicações, pode-se concluir, analisando a Figura 22, que a maioria dos utilizadores optaria por efectuar entre 2 a 5 downloads mensais.



Questão	Nº respostas	%
Nunca fiz download de uma aplicação móvel	9	8%
Menos de um download por mês	7	6%
Entre 2 a 5 downloads mensais	23	20%
Entre 6 a 9 downloads mensais	8	7%
Mais de 10 mensais	8	7%

Figura 22: Frequência utilização de *marketplace* para descarregar aplicações para *smartphones*

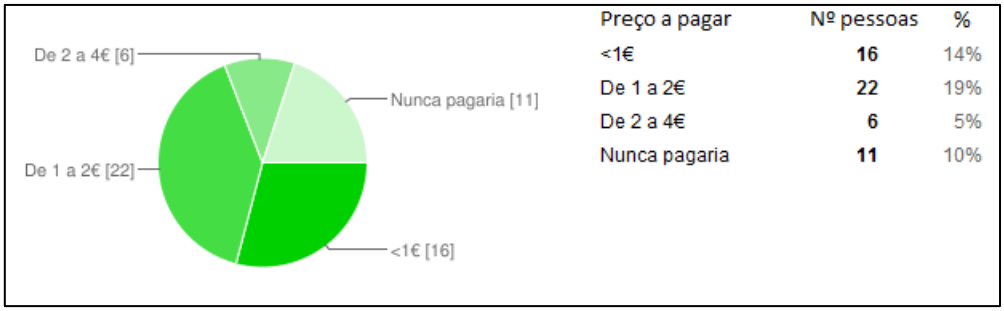
Sobre o tipo de aplicações que os utilizadores preferem, na Figura 23 verifica-se que existe um claro destaque das aplicações direccionadas para o lazer, neste caso os jogos e a navegação *on-line* com 69% e 67% respectivamente. É de salientar que a soma das percentagens pode exceder os 100% pelo simples facto que um inquirido pode escolher mais de um tipo de aplicação favorita no questionário.



Tipo de aplicação	Nº pessoas	%
Navegação	37	67%
Desporto	14	25%
Jogos	38	69%
Educação	17	31%
Finanças	16	29%
Livros	8	15%
Other	11	20%

Figura 23: Tipo de aplicações mais descarregadas pelos inquiridos

O preço médio a pagar por aplicação móvel está representado na Figura 24 e indica que 19% dos inquiridos estão dispostos a pagar em média entre 1 a 2€ por aplicação descarregada num *marketplace*.



Preço a pagar	Nº pessoas	%
<1€	**16**	14%
De 1 a 2€	**22**	19%
De 2 a 4€	**6**	5%
Nunca pagaria	**11**	10%

Figura 24: Preço médio que um utilizador pagaria por aplicação para *smartphone*

Entre os utilizadores que teoricamente estacionariam o seu veículo pessoal e deslocar-se-iam usando transportes públicos (Figura 25), verifica-se que na sua maioria estariam dispostos a andar a pé em média entre 250 e 500 metros até à respectiva paragem.

Distância	Nº pessoas	%
Menos de 100m	**9**	8%
250m	**17**	15%
500m	**23**	20%
Mais de 1000m	**6**	5%

Figura 25: Média de distância que um utilizador está disposto andar a pé

Como se referiu anteriormente, o ultimo grupo de perguntas pretendem auferir quais as funcionalidades a incluir no *software* para dispositivo móvel desenvolvido para o cálculo de trajetos.

O número de respostas dos inquiridos encontram-se condensadas na Tabela 1.

Tabela 1: Funcionalidades passíveis de inclusão no MuvOnSchedule

Funcionalidade	Discordo totalmente	Discordo parcialmente	Indiferente	Concordo parcialmente	Concordo totalmente
Com base na minha localização, indicar os meios de transporte disponíveis mostrando quanto tempo falta para a partida da respectiva carreira	0	0	4	12	39
Permitir a pesquisa e filtragem de trajetos possíveis entre dois pontos pelo seu custo seja em dinheiro ou tempo	1	0	6	24	24
Indicar os trajetos com menor distância percorrida a pé	1	1	7	22	24
Filtrar os transportes que permitem deslocação com bicicleta	2	8	20	17	8
Mostrar percursos optimizados para pessoas de mobilidade reduzida	0	2	13	15	25
Indicar os meios de transporte com menos transbordos entre o ponto origem e destino	1	0	9	25	20
Receber notificações sobre possíveis atrasos	0	0	6	19	30
Receber notificação minutos antes da chegada do meio de transporte	1	3	10	16	25
Baseado na minha localização, calcular os percursos possíveis para voltar a casa	0	1	13	20	21
Mostrar nos percursos os pontos de interesse de uma cidade usando transportes públicos	0	3	9	23	20
Mostrar os trajetos com serviço nocturno	3	0	10	21	21
Alarme que indica ao utilizador que chegou ao destino	2	6	12	16	19

Verifica-se que as funcionalidades mais apreciadas pelos inquiridos são as que estão assinaladas com a cor verde e as funcionalidades *nice to have* estão assinaladas com a cor azul.

4 Desenvolvimento

Neste capítulo são apresentadas as ferramentas, técnicas e abordagens utilizadas para resolver a problemática imposta e atingir os objectivos inicialmente propostos. É explicada a arquitectura na qual a aplicação está assente e qual a finalidade de cada um dos componentes que a integram.

4.1 Ferramentas utilizadas

Para o desenvolvimento do protótipo de sistema recorreu-se à utilização de diversas tecnologias, serviços, API's e librarias que permitiram a interligação entre subsistemas e garantem o seu funcionamento, respeitando as restrições impostas pelo tipo de cenário onde a aplicação móvel se vai encontrar inserida.

4.1.1 GTFS - General Transit Feed Specification

Este formato foi inicialmente designado por Google Transit Feed Specification e criado pela empresa Google, com a finalidade de partilhar de forma aberta e disponível as informações referentes a meios públicos de transporte, como por exemplo: horários, rotas, etc., entre a empresa prestadora do serviço e os seus utilizadores [12].

Com a criação deste formato foi criado um *standard* comum para a partilha de informações referentes aos transportes públicos. O surgimento deste formato deve-se a uma história interessante, que teve início na cidade de Portland, Oregon EUA.

Após uma viagem internacional no verão de 2005, Bibiana McHugh, responsável pelos sistemas de informação da TriMet Portland (empresa de transportes públicos da cidade de Portland), estava frustrada pois não conseguia planear a sua viagem entre dois pontos usando transportes públicos através de um programa de planeamento de itinerários como por exemplo o Mapquest.

Quando voltou aos EUA, McHugh entrou em contacto com algumas empresas como a Mapquest, Yahoo e Google, perguntando se tencionavam disponibilizar informação referente a transportes públicos nos seus mapas. Das empresas referidas apenas a Google respondeu. O engenheiro de *software* Chris Harrelson já estaria a usar 20 por cento do seu tempo ao serviço do Google para integrar este tipo de informação no Google Maps que mais tarde ser tornaria no Google Transit trip planner [26]. Desde então a TriMet e a Google trabalharam em conjunto de forma a transformar os dados de "trânsito" da TriMet num formato perceptível pelo Google Maps, surgindo deste modo o GTFS.

A TriMet foi a primeira operadora de transportes a ser integrada no Google Maps. Posteriormente mais cinco cidades americanas seguiram-lhe as pegadas: Eugene, OR; Honolulu, HI; Pittsburgh, PA; Seattle, WA e Tampa, FL. Atualmente, a Google tem acordos com mais de 100 operadoras de transporte público nos EUA e mais de 400 em todo o mundo [26]. Contudo inicialmente, a Google teve dificuldade em recolher dados, pois muitos operadores estavam relutantes em fornecer os dados relativos as suas rotas e aos seus serviços pois pensavam que a Google usaria esta informação e teria dividendos com a mesma [26].

Muitas aplicações como o Open Trip Planner são compatíveis com o formato GTFS. A forma mais simples de tornar públicas estas informações é hospedar o ficheiro no formato GTFS, num servidor *web* e publicar um anúncio através de "GTFS Data Exchange website" que o torna disponível para uso.

Entrando mais em detalhe, o GTFS possui informação sobre as rotas, horários, paragens etc., de forma versionada e distribuída em tabelas no formato CSV (*Comma Separated Values*) sendo estas posteriormente encapsuladas dentro de um ficheiro com extensão zip [17]. A especificação define os seguintes ficheiros como vemos a seguir na Tabela 2.

Tabela 2: Ficheiros que constituem o formato GTFS.

Ficheiro	Preenchimento obrigatório	Observações
Agency.txt	sim	Contem informações sobre empresa de transporte
Stops.txt	sim	Contem informações relacionadas com os pontos de paragem
Routes.txt	sim	Contem informações sobre as rotas disponibilizadas pela empresa
Trips.txt	sim	Contem informação das viagens e suas respectivas rotas. Uma viagem é uma sequência de duas ou mais paragem que ocorre num intervalo de tempo
Stop_times.txt	sim	Este ficheiro possui informação relativa aos tempos de chegada e partida de cada uma das paragens para cada viagem.
Calendar.txt	sim	Especifica em que dias o serviço está disponível.
Calendar_dates.txt	opcional	Permite definir as excepções aos dias disponíveis de serviço, definidos no ficheiro calendar.txt
Fare_attributes.txt	opcional	Especifica as tarifas a aplicar para as rotas disponíveis.
Fare_rules.txt	opcional	Permite especificar como as tarifas definidas no ficheiro Fare_attributes são aplicadas a um itinerário.
Shapes.txt	opcional	Define as coordenadas geográficas para representar as rotas de trânsito da empresa.
Frequencies.txt	opcional	Destina-se para representar horários que não tem uma lista fixa de horários na paragem.
Transfers.txt	opcional	Permite definir regras para fazer conexões nos pontos de transferência entre linhas.

Algumas das vantagens de usar o formato GTFS temos:

- Incentivar ao aumento do número de passageiros nos meios de transporte público;
- Ajudar os utentes a encontrar facilmente informações sobre horários;
- Redução de custos nas empresas de transportes mais pequenas para servir os seus clientes, complementando suas ferramentas;
- Tornar acessíveis as mudanças de informações acerca das linhas ou horários;
- Ajudar a melhorar o serviço com o feedback dos utilizadores.

Vale a pena salientar que as aplicações TransitTrips, Google Maps, Bing Maps, Tranzit, utilizam este formato.

4.1.2 Open Trip Planner

Open Trip Planner é um planeador intermodal de itinerários *open source*, que permite o planeamento de trajetos em uma ou mais viagens entre dois pontos (origem e destino). O seu núcleo foi desenvolvido em Java 2 Enterprise Edition J2EE por ser uma plataforma neutral, madura e conhecida pela maior parte dos engenheiros de *software*[22]. Implementa vários algoritmos optimizados para o cálculo de rotas como por exemplo o algoritmo de Dijkstra[3] e A*[4]. Implementa técnicas como o método de hierarquias de contracção (contraction hierarchies) para melhorar a performance do algoritmo de Dijkstra. O método de hierarquias de contracção é outra abordagem hierárquica para o cálculo do caminho mais curto. A ideia principal é organizar todos os nós de acordo com a sua importância e de seguida iterativamente contrai-los nesta ordem, adicionando atalhos para preservar as distâncias mais curtas. As consultas são efectuadas ao grafo recorrendo ao algoritmo Dijkstra bidireccional, que procura evitar os nós menos significativos usando os atalhos adicionados na etapa de pre-processamento[24].

O OTP disponibiliza as indicações para chegar a um local destino num mapa iterativo que mostra passo a passo o percurso a recorrer, os serviços que o utilizador deve usar e onde pode fazer as transferências entre os meios de transporte. Adapta-se a utilizadores com necessidades especiais, permitindo a filtragem e pesquisa de estações, meios de transporte ou autocarros que possibilitem o transporte de pessoas deficientes acomodando-se às suas necessidades. Também permite o cálculo de itinerários em vários modos de trânsito como por exemplo caminhada e ciclismo para que os seus utilizadores consigam planear uma viagem inteira de porta a porta combinando vários meios de transporte. Ao contrário de outros serviços geridos por terceiros, com o OTP podemos ter um controlo total sobre os dados do calculador de itinerários, sendo muito fácil mantê-lo atualizado face as mudanças de horários, rotas, etc.

[3] O Algoritmo de Dijkstra permite calcular o caminho de menor custo entre nós utilizando um grafo com pesos. Dijkstra's algorithm. 2011 [cited 2011 September]; Available from: http://www.algolist.com/Dijkstra's_algorithm

[4] Algoritmo A* permite a procura de um caminho entre dois pontos desde um vértice inicial até um vértice final. A* search algorithm. 2011 [cited 2011 September]; Available from: http://en.wikipedia.org/wiki/A*_search_algorithm

Quando instalamos uma instância do servidor OTP encontramos o seu projeto estruturado da seguinte forma [15]:

- **opentripplanner-routing:** Implementa os principais algoritmos de reencaminhamento e disponibiliza estruturas de dados e bibliotecas;
- **opentripplanner-api-webapp:** Aplicação *web* que disponibiliza a REST API que dá acesso ao motor de planeamento de viagens;
- **opentripplanner-webapp:** Aplicação *web* que disponibiliza uma interface de utilizador para aceder ao motor de planeamento de viagens;
- **opentripplanner-graph-buider:** Ferramenta através de linha de comandos para configurar e construir o grafo de planeamento de viagens;
- **opentripplanner-gui:** Visualizador de grafo para desenvolvimento e resolução de problemas;
- **opentripplanner-integration:** Testes de integração;
- **opentripplanner-api-extended:** Aplicação *web* que pode opcionalmente ser instanciada para mostrar o mapa de sistema. Requere do Geoserver;
- **opentripplanner-utils:** Codifica *polylines*, provavelmente para *shapefiles*;
- **opentripplanner-geocoder:** Aplicação *web* que fornece uma REST API usada pelo OTP para geocodificar lugares;

4.1.3 Geocoding

O *geocoding* é o processo que permite converter um endereço do tipo "Campus politécnico de Repeses 3504-510 Viseu" para as respectivas coordenadas geográficas: latitude 40.84269810 e longitude -7.75898320. Esta tradução é importante pois possibilita integrar os diversos sistemas e construir os pedidos de informações que são trocados entre a aplicação móvel e o planificador intermodal de itinerários, assunto que iremos abordar em pormenor mais adiante.

No âmbito deste projeto foi usado o **Google Maps API Geocoder**, que é um serviço *web* que disponibiliza um conjunto de rotinas que respondem a pedidos de *geocoding* usando pedidos HTTP num formato preestabelecido.

Além disso, o serviço permite executar a operação inversa, isto é converter coordenadas geográficas para endereço. Este processo é denominado Geocodificação reversa [13].

Também foram analisadas outros serviços *web* de *geocoding*, como o Yahoo! PlaceFinder. Anteriormente denominado Yahoo! Maps *web services*, evoluiu de forma a suportar e retornar os dados resultantes de um pedido de *geocoding* no formato JSON e que suporta geocodificação a nível mundial de endereços, pontos de interesse, aeroportos e nomes de lugares [30]. Apesar de permitir um número elevado de pedidos de geocodificação: até 50000 pedidos/dia por aplicação, face aos 2500 pedidos/dia por IP do Google geocoder API, quando comparados, a precisão do Yahoo! PlaceFinder nem sempre é melhor em algumas regiões do globo [28]. É importante salientar que apesar de não fazermos uso da API Yahoo no desenvolvimento do protótipo, é uma ferramenta a considerar em futuros desenvolvimentos para aumentar e optimizar a tarefa de *geocoding* no MuvOnSchedule.

4.1.4 GSON

O GSON é uma biblioteca Java usada para converter objetos Java na sua representação JSON. Este processo é denominado de serialização. Também é possível converter uma string no formato JSON para Java, sendo este o processo de deserialização.

GSON pode trabalhar com objetos arbitrários Java já existentes dos quais podemos não possuir o seu código fonte. Esta biblioteca nasceu como projeto interno da empresa Google e ainda é utilizada atualmente em outros projetos da referida empresa. Atualmente este *software* é de uso livre pois está disponível ao abrigo de uma licença Apache [16].

O funcionamento do GSON é explicado sucintamente, através, da classe DataObject que se encontra na Figura 26 e que possui uma variável tipo inteira, uma *string* e uma lista de *strings*.

```
import java.util.List;

public class DataObject {
    private Integer data1;
    private String data2 ;
    private List<String> list;
    public Integer getData1() {
        return data1;
    }
    public void setData1(Integer data1) {
        this.data1 = data1;
    }
    public String getData2() {
        return data2;
    }
    public void setData2(String data2) {
        this.data2 = data2;
    }
    public List<String> getList() {
        return list;
    }
    public void setList(List<String> list) {
        this.list = list;
    }
}
```

Figura 26: Implementação classe DataObject

Posteriormente é criado um ficheiro fx.json com uma *string* no formato JSON que se encontra ilustrado na Figura 27.

```
Object:{
        data1:100,
        data2:"hello",
        list:["String 1","String 2","String 3"]
}
```

Figura 27: *String* JSON contida dentro de um ficheiro

Finalmente é criada a classe GsonExample que se pode observar na Figura 28 e basicamente abre o ficheiro no formato JSON e converte todos os dados nele contidos para um objecto Java do tipo DataObject, assim deste modo é possível aceder aos dados usando os métodos getData1(), getData2() e getList().

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import com.google.gson.Gson;

public class GsonExample {
    public static void main(String[] args) {

    Gson gson = new Gson();
    try {
        BufferedReader br = new BufferedReader(
            new FileReader("c:\\fx.json"));
        //converte string no formato JSON para objecto Java
        DataObject obj = gson.fromJson(br, DataObject.class);
        System.out.println(obj);

    } catch (IOException e) {
        e.printStackTrace();
    }

    }
}
```

Figura 28:Implementação da classe GsonExample

4.1.5 RESTful + JSON

REST significa Transferência de Estado Representacional ou *REpresentational State Transfer*, e define um conjunto de princípios de arquitectura e desenho de serviços Web tendo como foco os recursos do sistema, incluindo como os estados dos recursos são endereçados e transferidos sobre HTTP por um grande numero de clientes que podem estar escritos em diferentes línguas.

Basicamente, na arquitectura REST, os dados e as funcionalidades são considerados recursos URI (*Uniform Resource Identifier*) utilizados como representação para aceder a qualquer recurso [6]. Apesar do REST ainda não ser um *standard*, funciona sob vários princípios, como cliente e servidor totalmente identificados, separados e sem estado [23]. Cada mensagem HTTP contém toda a informação necessária para fazer o pedido. Isto significa que nem o cliente nem o servidor precisam de escrever qualquer estado da comunicação entre mensagens. Na prática, muitas aplicações baseadas em HTTP utilizam

cookies e outros mecanismos para manter o estado da sessão, como por exemplo a reescrita de URLs, práticas que não são permitidas pelos princípios REST.

O REST faz uso de operações bem definidas aplicadas a esses recursos, tirando partido dos verbos bem conhecidos HTTP como GET, POST, PUT ou DELETE, que podem ser mapeados diretamente para operações CRUD (*Create, Remove, Update e Delete*). A sintaxe universal é usada para identificar recursos, expondo a sua estrutura como URI's, levando cada acção a ser identificada pela sua própria URI. Os princípios REST dizem que deve ser possível obter dados usando XML, JSON ou ambos por forma a ter mais poder e flexibilidade [10].

A implementação de um serviço RESTful oferece um conjunto de vantagens como por exemplo a escalabilidade entre componentes, uma abordagem *stateless* (sem estado), baixo *overhead*[5] (sobrecarga), apresentação dos dados num formato facilmente perceptível pelas pessoas, múltiplas interfaces, fácil construção, independência da linguagem de programação e sistema operativo e permite encorajar ao uso de XML, JSON ou ambos [20].

Um *web service* SOAP [1] tem *standards* bem definidos que lhe permite ser suportados por diferentes tipos de ferramentas e sistemas, mas seu uso significa que o programador tem que se preocupar com a criação da especificação WSDL que encapsule a *string* dentro de uma mensagem SOAP. Por outro lado, um serviço REST é simples de desenvolver, fácil de usar e introduz menos atraso entre o cliente e o prestador do serviço e depende inteiramente do protocolo HTTP. O principal problema é a falta de *toolkits* (Conjunto de ferramentas/aplicações necessárias para desenvolver uma tarefa) de desenvolvimento.

Testes realizados [18], comprovam que os serviços RESTful são mais adequados para o desenvolvimento de aplicações para dispositivos móveis. Eles são mais rápidos, proporcionam respostas mais pequenas, induzem menos sobrecarga e as respostas são mais rápidas quando comprado com um *webservice* SOAP convencional [19].

[5] Consumo em excesso de um determinado recurso seja ele processamento, armazenamento ou memória para realizar uma tarefa.

Embora o XML seja um bom formato de intercâmbio de informação, quando o ficheiro toma maiores dimensões apresenta uma sobrecarga significativa devido a descrição e identificação dos dados neles contidos. As aplicações móveis muitas vezes são executadas em dispositivos com conectividade 3G e *wireless*, e na maioria das vezes com plafond e largura de banda limitados. Deste modo torna-se imperativo encontrar um formato adequado que não comprometa o desempenho e melhore a eficiência das aplicações móveis.

Por outro lado temos o JSON (*JavaScript Object Notation*) que é um formato leve e livre de intercâmbio de dados [8]. É fácil de interpretar e gerar, é facilmente lido pelos humanos por ter natureza autodescritiva [19]. Um objecto JSON é delimitado por chavetas e podem conter vários campos separados por vírgula.

Cada campo tem um identificador de texto, seguido por dois pontos e depois pelo seu valor. Um *array* é delimitado por parêntesis retos. Na Figura 29 podemos ver um objecto JSON.

```
1   Object { field1: "value1" , field2: "value2"}
2
3   Array:[
4       {
5           field1: "value1item1",
6           field2: "value2item1"
7       },
8       {
9           field1: "value1item2",
10          field2: "value2item2"
11      },
12  ]
```

Figura 29: Representação de um objecto JSON e um array

Em JSON um valor entre aspas pode simplesmente ser uma *string*, número, boleano, *array* ou até outro objecto JSON.

JSON também aceita valores nulos. A cadeia de caracteres JSON é uma sequência de zero ou mais caracteres Unicode dentro de aspas e aceita caracteres de escape, como a maioria das linguagens de programação. JSON não permite números hexadecimais ou

octais, mas os números decimais são declarados como qualquer outra linguagem de programação.

JSON oferece várias vantagens na sua implementação, como o ser leve, ser fácil de gerar e interpretar pelos seres humanos e facilidade de integração com páginas *web* uma vez que usa sintaxe Javascript. Existem inúmeras API's que suportam e retornam resultados neste formato como por exemplo a Yahoo API. Como qualquer tecnologia o JSON tem alguns inconvenientes. Pode não ser tão legível como o XML ou HTML, o que implica perceber a sintaxe claramente. A escolha entre XML e JSON depende do trabalho que está a ser feito. Em situações onde temos necessidade de retornar dados a partir de resposta de um *webservice* e interpretar-los, o JSON e XML ambos servem perfeitamente. Se houver necessidade de atualizar partes especificas de um documento, JSON pode não ser a melhor opção, pois torna-se mais difícil a sua alteração. Neste caso será mais eficaz utilizar o formato XML e utilizar a linguagem XPath de forma a conseguir alterar facilmente as partes desejadas do documento.

Como referido anteriormente, as aplicações móveis devem ser supostamente leves e rápidas e um dos pontos a tomar em consideração é o tempo de processamento dos dados de resposta retornados no formato JSON e XML. Um processamento rápido significa que o processador do dispositivo vai estar menos tempo alocado a interpretar os dados e por consequência vai consumir menos bateria e mostrar rapidamente os resultados.

Analisando em detalhe a Figura 30, onde o eixo vertical representa o tamanho do tempo de resposta em bytes e o eixo horizontal representa o número de registos interpretados, verificamos que o formato XML introduz mais *overhead* que o JSON não importando a quantidade de registos a ser retornados. Assim sendo o uso do JSON em vez do XML como formato de troca de dados reduz consideravelmente não só largura de banda necessária para a troca, mas reduz também, e por uma grande margem, o tempo de processamento necessário para analisar o de dados utilizando um dispositivo móvel.

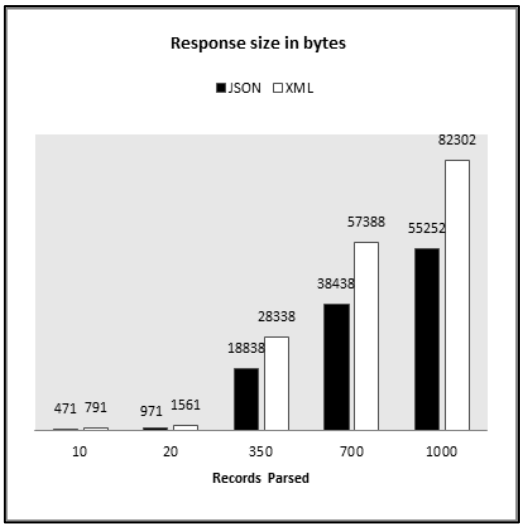


Figura 30: Tamanho da resposta em bytes, para diferentes quantidades de registos [19]

Na Figura 31, o eixo horizontal representa a contagem dos registos analisados para cada teste. Já o eixo vertical representa o tempo gasto por cada pedido. Um tempo de interpretação pequeno significa que o processo é rápido, consome menos energia e retorna os resultados ao utilizador rapidamente. As colunas de cor branca representam o tempo de interpretação gasto pelo XML e as de cor preta é o tempo de interpretação do JSON. Assim é possível concluir que se for usado um interpretador padrão para JSON e XML a partir do SDK do Android, JSON é interpretado muito mais depressa que o XML.

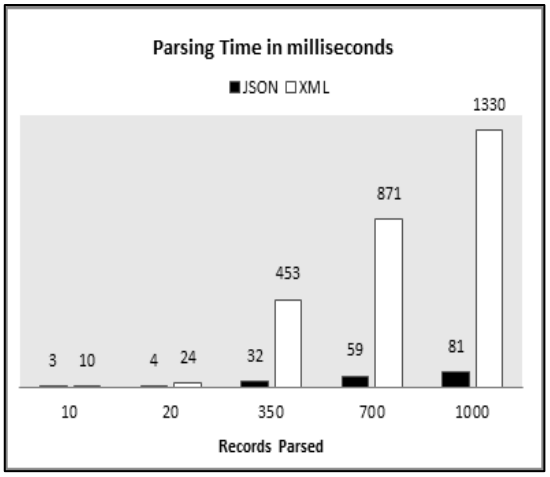


Figura 31: Tempo de processamento em milissegundos [19]

4.1.6 Sistema Operativo Android

Em Novembro de 2007 a empresa Google tornou pública a primeira plataforma *Open Source* para desenvolvimento de aplicações para dispositivos móveis, baseada na plataforma Java e tendo como base o sistema operativo Linux, sendo denominada Android. Esta plataforma é mantida pela Open Handset Alliance, grupo constituído por mais de 30 empresas cujo intuito é inovar e acelerar o desenvolvimento de aplicações e serviços, permitindo aos consumidores uma experiência mais rica e, menos dispendiosa em termos financeiros para o mercado móvel [25]. Podemos afirmar que a plataforma Android é um conjunto de *software* para uso em dispositivos móveis que inclui um sistema operativo, *midleware* (camada intermédia) e aplicações [2], constituindo a primeira plataforma de desenvolvimento móvel aberta e livre.
Após a primeira *release* da plataforma, em poucos meses, mais de um milhão de pessoas tinham descarregado o SDK do site da Google. Nos EUA a operadora T-Mobile anunciou o primeiro *smartphone* Android o G1 em Outubro de 2008 e foram vendidos até antes do final desse ano se tenham vendido centenas de milhares de estes equipamentos [25].

O sistema operativo Android definiu um novo padrão para o desenvolvimento de aplicações orientadas para dispositivos móveis e reuniu todas as condições para a criação do *Android Market*, espaço destinado a disponibilização de aplicações livres ou pagas por parte dos utilizadores.

Então porque até agora não tem sido rentável criar aplicações móveis para *smartphones*? Quais são os problemas que o Android da resposta?

- **Capacidade de execução de múltiplas aplicações em simultâneo:** O utilizador pode realizar múltiplas tarefas com o equipamento em simultâneo desde ouvir música, receber e enviar notificações ou até gravar dados GPS;
- **Permite a escolha do *hardware* a usar:** Como o Android é uma plataforma aberta, os fabricantes têm a liberdade de incluir o *hardware* que quiserem;
- **Instalação de ROMs personalizadas:** Como Android é um sistema *open source*, tem uma grande comunidade de colaboradores que constrói e disponibiliza ROMs

personalizadas para equipamentos Android. Isto permite a sua personalização gráfica e do funcionamento das seus programas;

- **Integração com redes sociais:** Os *smartphones* têm conectividade constante, permitindo aceder aos nossos *emails*, aceder ao Flickr para colocar as nossas fotos acabadas de criar com o nosso equipamento, criar comentários na nossa conta de Facebook ou até enviar uma mensagem através do Twitter, são algumas das possibilidades que nos oferece o Android de forma nativa;
- **Android market place:** O Android tem um mercado de aplicações disponíveis para os seus utilizadores comprarem *software* ou simplesmente descarregarem *software open source* das mais variadas áreas: Jogos, finanças, desporto, lazer etc.

O sistema operativo Android está subdividido em quatro camadas principais como podemos observar na Figura 32: o *kernel*, bibliotecas (*libraries*), estrutura aplicacional (*applications framework*) e as aplicações.

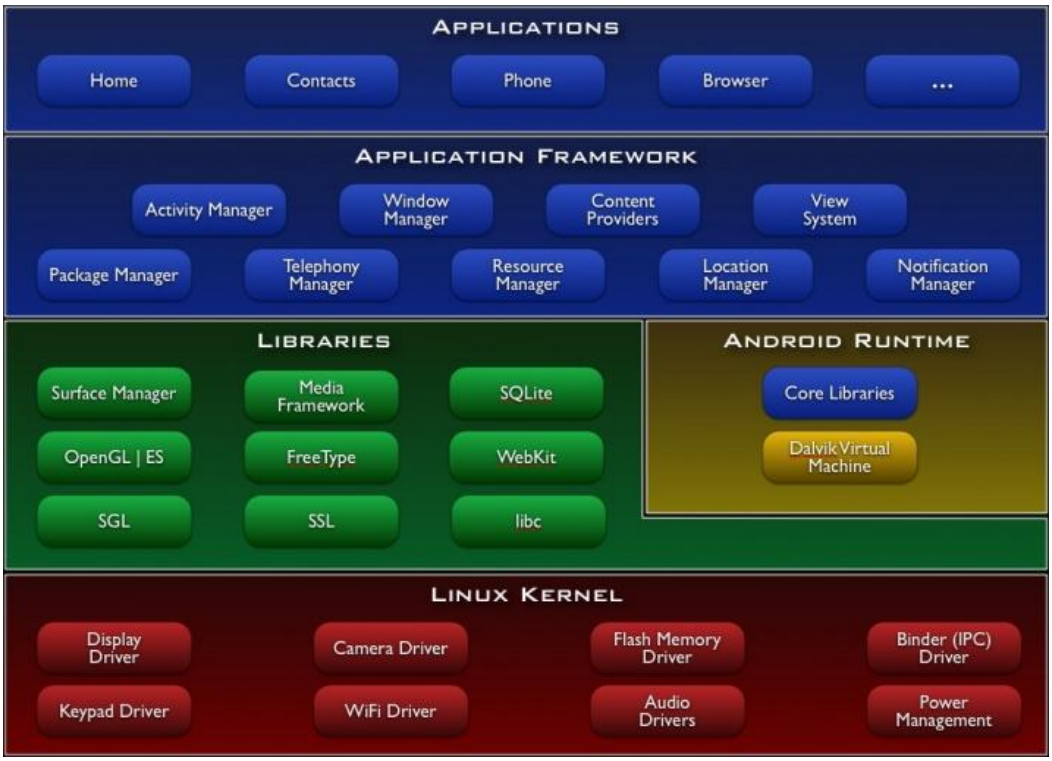


Figura 32: Arquitectura Android [14]

A camada kernel é proveniente do Linux. As bibliotecas que vêm com o Android fornecem a muitas das capacidades gráficas, multimédia e armazenamento. Embutido na camada *libraries* temos o Android *runtime* que contem a máquina virtual Dalvik que alimenta e gere as aplicações. A camada *applications framework* é a API que todas as aplicações devem usar para aceder ao nível mais baixo da arquitectura Android.

4.1.6.1 Kernel

O Linux foi escolhido como kernel do Android, uma vez que provou estabilidade e funcionalidade em sistemas desktop além disso na maioria dos casos não é necessário reescrever novos *drivers* para hardware móvel. O kernel fornece coisas como memória virtual, rede e gestão de energia. Ao examinar o kernel fornecido com o código fonte Android poucas são as mudanças significativas face às principais funcionalidades do kernel tradicional [7].

4.1.6.2 Sistema de bibliotecas

As bibliotecas do sistema Android são escritas em C/C++. São estas que fornecem as principais funcionalidades e são chamadas através de interfaces Java. Estas incluem um gestor de superfície (*Surface Manager*): para composição de janelas e gráficos 3D e 2D, *codecs* como MPEG4 e MP3, bases de dados SQL Lite e o motor de um navegador *web Webkit*. A *Dalvik Virtual Machine* é de particular interesse nesta camada. É um interpretador bytecode altamente optimizado para correr numa plataforma móvel [7].

4.1.6.3 Applications framework

Esta camada e a camada acima são totalmente escritas em Java. Aqui, o Android fornece um subconjunto substancial dos 5 pacotes *Java Standard Edition*, incluindo colecções, I/O. Também fornece todas as Android API's especificas que as aplicações devem usar incluindo coisas como a partilha de dados, acesso a funcionalidades do telefone e recepção de notificações.

Um ponto importante a ressaltar sobre o Android OS é que todas as aplicações utilizam a mesma *framework*. Isto permite uniformizar o código e reduzir os problemas relacionados com a atualização de camadas inferiores [14].

4.1.6.4 Camada aplicacional

Todas as aplicações Android são escritas em Java, e interpretadas pela *Dalvik virtual machine.* A maior parte das funcionalidades principais do telefone e das aplicações de gestão de contactos residem nesta camada. Contém *software* escrito pela equipa Android mas também *software* de terceiros que pode ser instalado no dispositivo. Isto permite que os programadores consigam manipular algumas características do *software* original e torna-lo parametrizável.

4.1.7 Java + Android SDK

Nas aplicações Android, como a maioria das aplicações para dispositivos móveis, são desenvolvidas num computador onde os recursos são mais abundantes e posteriormente é feito *download* para o dispositivo móvel para testar e usar. Por tanto as aplicações podem ser testadas e depuradas de erros num dispositivo Android ou simplesmente utilizando um emulador. Assim sendo torna-se imperativo identificar e instalar todas as ferramentas necessárias para iniciar o desenvolvimento do nosso protótipo de aplicação móvel.

O Android SDK (*Software Development Kit*) suporta diferentes IDE's (*Integrated Development Environment*) que possibilitam a escrita e a depuração de código Java que posteriormente será corrido no dispositivo móvel, sem importar qual o sistema operativo que é usado na máquina destinada ao desenvolvimento.

Basicamente são necessárias as seguintes ferramentas:

- **IDE Eclipse:** Editor que permite o desenvolvimento de código Java;
- **JDK (*Java Development Kit*):** Conjunto de utilitários que possibilitam construir peças de *software* passíveis de correr na plataforma Java. É constituído pelo compilador e bibliotecas;

- **Android SDK:** fornece as ferramentas e bibliotecas necessárias para iniciar o desenvolvimento de aplicações que podem correr em dispositivos Android, como por exemplo emulador, documentação, bibliotecas, exemplos e depurador entre outros [9];
- **ADT (*Android Developer Tool*):** Extende as capacidades do IDE neste caso o Eclipse para permitir criar rapidamente novos projetos Android, criar aplicações UI, adicionar componentes tendo como base o Android Framework API, depurar as nossas aplicações usando ferramentas do SDK e criar pacotes .apk para distribuir a aplicação [29].

4.2 Arquitetura do sistema

A arquitetura onde está assente o protótipo MuvOnSchedule, foi concebido de forma a dar resposta aos desafios propostos inicialmente e pode ser ilustrada através da Figura 33.

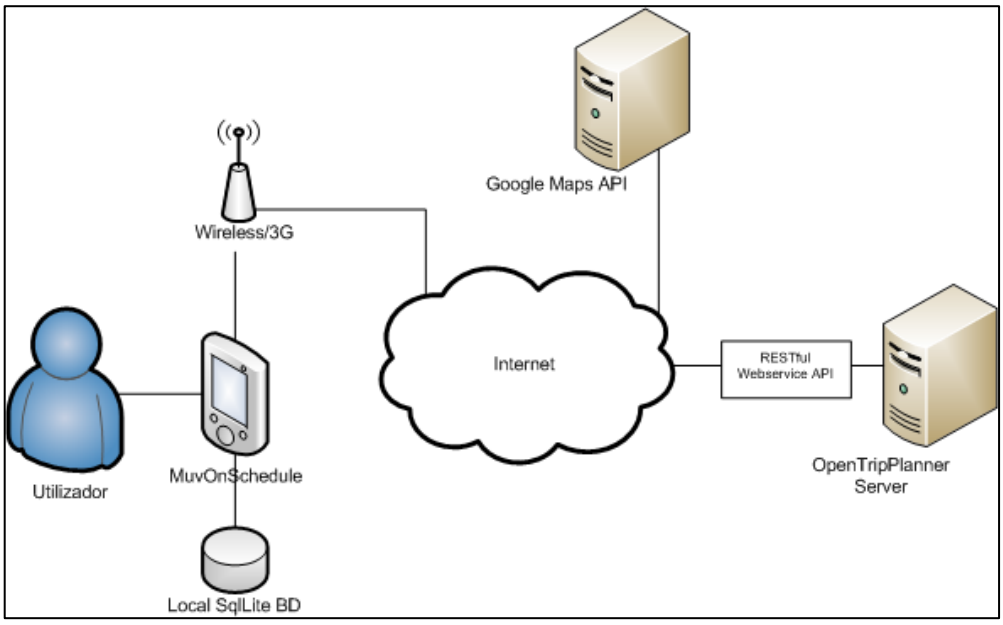


Figura 33: Arquitetura base do MuvOnSchedule

O sistema está constituído por uma aplicação servidor que neste caso é o Open Trip Planner que é o planeador intermodal de viagens, instalado num container Java Tomcat que responde a pedidos enviados efetuados a partir da aplicação cliente que está instalada no dispositivo do utilizador.

Descrevendo sucintamente o funcionamento do sistema, basicamente o utilizador instala e inicializa no seu dispositivo móvel a aplicação MuvOnSchedule, onde se indica o ponto de partida e de chegada envolvidos na viagem que quer planificar e percorrer. Com essa informação a aplicação realiza o *geocoding* desses pontos utilizando a Google Maps API Geocoder, convertendo moradas fornecidas em coordenadas geográficas (latitude e longitude). Posteriormente, com ajuda da aplicação o utilizador escolhe os meios de transporte e outros filtros disponíveis, que se encontram previamente configurados e armazenados na base de dados local do dispositivo (Sql Lite) e quando o utilizador escolhe a opção procurar é construído um pedido HTTP que é enviado ao servidor Open Trip Planner e respondido através da sua RESTful webservice API, que retorna o resultado da pesquisa no formato JSON, o que vai permitir mostrar ao utilizador os trajetos encontrados.

A aplicação cliente está estruturada seguindo o padrão *Model-View-Controller* exemplificado na Figura 34, permitindo que exista uma diferenciação clara entre os componentes que constituem a aplicação e permite a camada de controlo (*Controller*) manipular facilmente os *inputs* colocados pelo utilizador; enquanto a camada de apresentação (*View*) mostra a informação no ecrã e a camada (*model*) é responsável por guardar os dados.

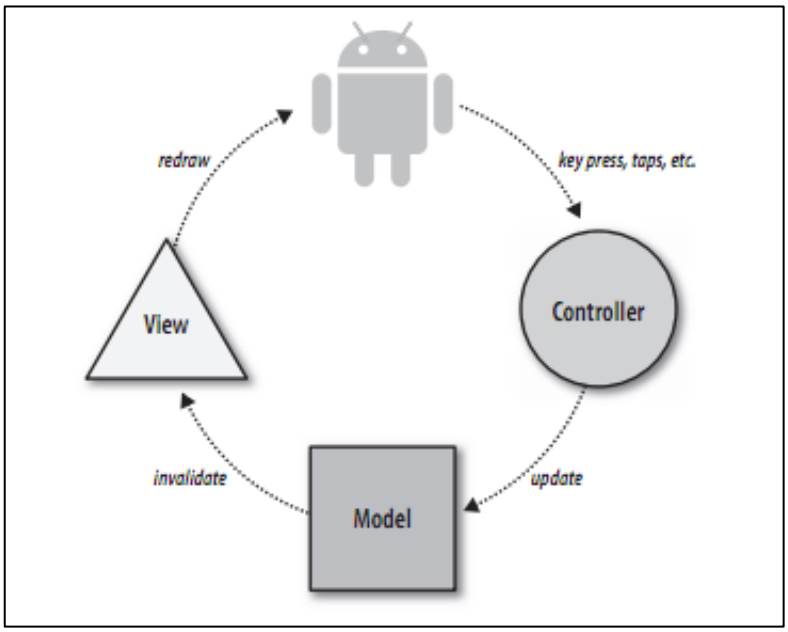


Figura 34: Conceito *Model-View-Controller*

Explicando a função de cada uma das camadas temos:

- ***Model***: Representa a camada de persistência dos dados. É onde a camada *View* e *controller* reflectem as alterações [25];
- ***View*****:** É responsável pela renderização e amostragem dos dados da aplicação ao utilizador. A *framework* UI Android percorre os componentes armazenados na *View tree* e pede a cada componente que se desenhe a si próprio e peça aos seus filhos para fazer o mesmo [25]
- ***Controller*****:** Esta parte da aplicação responde a acções externas como por exemplo toques no ecrã ou carregar em algum botão da tela, etc. É implementada como numa fila de eventos. Cada acção externa é representada por um único evento na fila. A *framework* remove cada um destes eventos da fila a medida que são tratados. Assim uma vez que o evento é tratado pelo respectivo componente em uso, este pode tomar a acção correta para mudar o estado interno da aplicação [25].

4.3 Diagrama de classes

Na construção do protótipo foram concebidos três grupos de diagramas de classes, um para suportar a informação proveniente do processo de geocodificação, outro para suportar a informação resultante das pesquisas efectuadas ao Open Trip Planner e o último possui as classes que implementam as funcionalidades disponíveis na aplicação mobile.

Previamente antes de efectuar efectivamente a pesquisa dos trajetos disponíveis no Open Trip Planner e como foi referido anteriormente, é necessário obter as coordenadas geodésicas (latitude e logitude) dos locais de partida e de chegada.

Assim sendo, é necessária uma estrutura que suporte os dados resultantes dos pedidos efectuados à Google Maps API Geocoder e que vêm no formato JSON. Utilizando a biblioteca GSON é possível a deserialização dos dados para as respetivas classes java que mostramos na Figura 35:

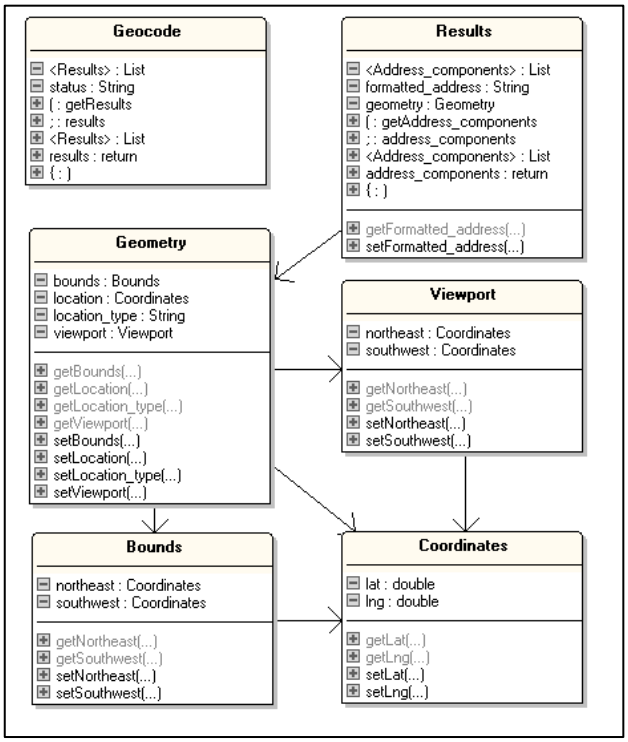


Figura 35: Diagrama de classes para o processo de *geocoding*

Na Tabela 3, explica-se resumidamente qual a funcionalidade das classes referidas na figura anterior.

Tabela 3: Função das classes de suporte ao *geocoding*

Classe	Função
Geocode	Contém uma lista de resultados e retorna o estado do processo geocoding
Results	Lista de address_components que contem nome longo e curto do endereço fornecido. Um formatted_address que contem o endereço completo formatado e contem um objeto Geometry.
Geometry	Contém objetos Bound e Coordinates que representam os limites e as coordenadas do ponto. Tem um location_type que representa o tipo de localização do ponto e um viewport.
Bounds	Armazena as coordenadas dos limites do endereço pesquisado
Coordinates	Armazena as coordenadas do endereço pesquisado
Viewport	Indica os pontos lat e lon da janela de visualização

Posteriormente após a respectiva submissão do pedido de itinerários através da API RESTful do servidor OTP, são utilizadas as classes assinaladas na Figura 36.

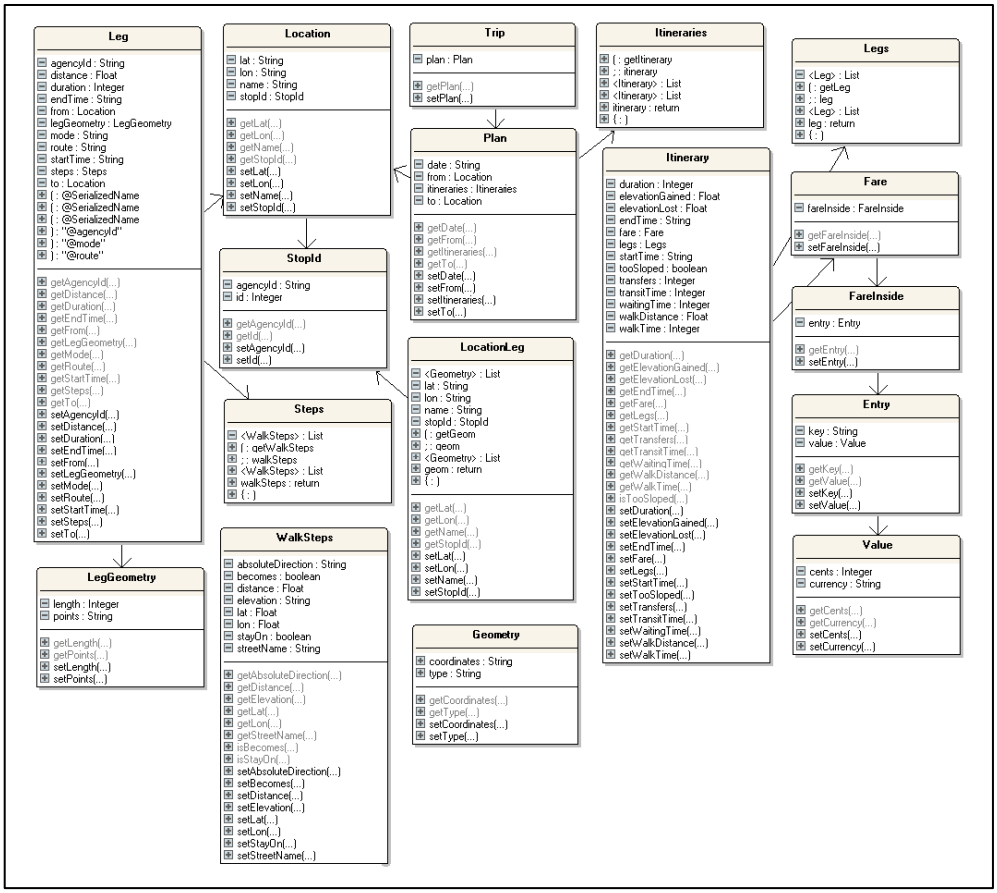


Figura 36: Diagrama de classes para mapeamento de resultados pesquisa OTP

Para compreender a utilidade e funcionalidade de cada uma destas classes, a informação foi condensada na Tabela 4.

Tabela 4: Classes que permitem acesso a informação pesquisada no servidor OTP

Classe	Função
Trip	Contém um objeto Plan
Plan	Armazena informação sobre a data da pesquisa, ponto origem, destino, objetos location e Itineraries.
Itineraries	Armazena uma lista de itinerary.
Itinerary	Armazena informação geral sobre um itinerario. Dá acesso a duração do itinerario, hora inicio, hora fim, tempo a caminhar a pé, tempo de espera, distância a percorrer a pé, declive positivo, declive negativo, número de transferências. Também possui um objecto do tipo Fare e Legs.
Fare	Armazena um objecto do tipo FareInside.
FareInside	Armazena um objeto do tipo Entry
Entry	Armazena uma key e um objeto do tipo value, normalmente usado para especificar o tipo de tarifa aplicar e preço da viagem.
Value	Guarda o preço da viagem e moeda.
Geometry	Disponibiliza o tipo de geometria e as suas coordenadas.
Legs	Armazena uma lista de Leg
Leg	Objeto que armazena informação sobre os meios de transporte e as paragens que deve aceder. Contém informação sobre a transportadora prestadora do serviço, número da carreira e tipo de transporte. Ainda informa sobre a hora de partida, hora de chegada, distancia a percorrer e duração.
LegGeometry	Contém o tamanho do Leg e os seus pontos.
Steps	Armazena uma lista de WalkSteps
WalkSteps	Informação para os percursos que são efectuados a pé. Indica distância, nome da rua, latitude, longitude e elevação do terreno.
Location	Contém o nome, identificador da paragem, latitude e longitude da localização.
LocationLeg	Contém o nome, identificador da paragem, latitude longitude da localização e uma lista de objectos do tipo Geometry.
StopId	Indica o número da paragem e a transportadora

Finalmente, aborda-se o diagrama de classes referente às classes que suportam o funcionamento do protótipo e que estão assinaladas na Figura 37.

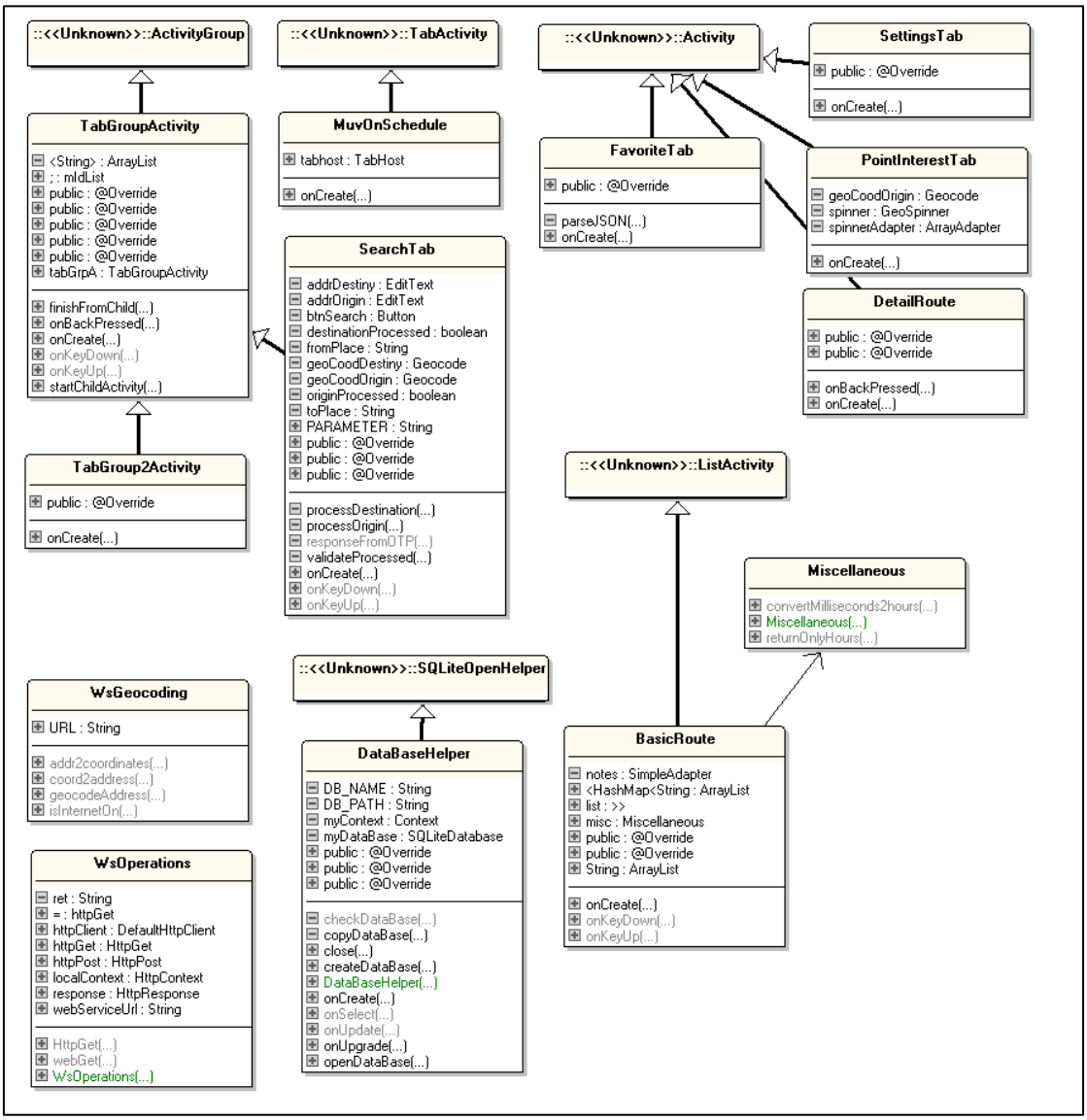


Figura 37: Diagrama representativo das classes que implementam funcionalidades

Analisando a figura anterior, verifica-se que as classes ActivityGroup, TabActivity, Activity, ListActivity e SQLiteOpenHelper são classes disponibilizadas pelo Android SDK e o desenvolvimento é realizado estendendo (extent) estas classes para implementar as funcionalidades.

Encontram-se na Tabela 5, as respectivas classes e o seu contributo na aplicação:

Tabela 5: Classes que implementam funcionalidades do protótipo

Classe	Função
MuvOnSchedule	Implementa ecrã principal da aplicação, cria a TabActivity que alberga as restantes tabs: SearchTab, FavoriteTab, PointInterestTab e SettingsTab.
SearchTab	Implementa o formulário de pesquisa de trajetos da aplicação
FavoriteTab	Implementa o formulário que mostra os locais favoritos.
PointInterestTab	Implementa o formulário que mostra os pontos de interesse.
DataBaseHelper	Implementa métodos que permitem a persistência na BD local, de configurações e informação na aplicação.
BasicRoute	Mostra os diversos trajetos resultantes da procura no servidor OTP.
DetailRoute	Mostra detalhes do trajeto escolhido pelo utilizador.
WsOperations	Permite invocar pedidos ao servidor OTP e verificar o estado da conexão com o servidor.
WsGeocoding	Implementa métodos que criam pedidos para o servidor de geocoding e efectuam a conversão de morada para coordenadas e viceversa.
Miscellaneous	Possui métodos que realizam tarefas atómicas necessárias na aplicação como por exemplo converter milissegundos em horas, etc.
TabGroupActivity/TabGroup2Activity	São métodos que permitem a iteração entre janelas, isto é passar de uma janela para outra.

4.4 Configuração Open Trip Planner

Para o protótipo funcionar foi necessário configurar uma instância do servidor Open Trip Planner. Inicialmente foi desenvolvida uma configuração no formato GTFS de uma rota fictícia de uma conhecida empresa de camionagem de Aveiro usando as ferramentas disponíveis em [12], a fim de criar o respectivo grafo com as rotas no servidor OTP.

O procedimento consistiu em instalar o OTP, fazendo *checkout* dos repositórios da empresa que desenvolveu o *software*. É importante salientar que o OTP pode utilizar varias fontes de dados como *shapefiles*, *street data* provenientes da OpenStreetMap ou simplesmente ficheiros GTFS. Optou-se por utilizar o GTFS por ser um *standard* muito usado para partilha de informação de transportes públicos.

Foi configurado o ficheiro **grap-config.xml** que se pode visualizar através da Figura 47 apresentada em anexo, onde são definidas as fontes de dados e a *bounding box*, isto é, os limites da localização geográfica onde o servidor vai atuar. Neste trabalho usou-se o ficheiro **GTFS_moveaveiro.zip** previamente gerado e validado no formato GTFS. Depois, na pasta **opentripplanner-graph-builder/Target** onde está a instalação OTP, procede-se a geração do grafo que vai permitir efectuar consultas usando o servidor usando o seguinte comando:

```
java -Xmx1024M -jar graph-builder.jar /home/demo/otp/opentripplanner-graph-builder/samples/graph-config.xml
```

Posteriormente, na diretoria raiz da instalação, recompila-se e geram-se os respectivos ficheiros opentripplanner-api-webapp.war e opentripplanner-webapp.war usando o comando:

```
mvn -e package –DskipTests.
```

Os ficheiros anteriormente gerados devem ser instalados no servidor Tomcat que irá instanciar o OTP. O ficheiro opentripplanner-webapp.war quando instanciado, permite ao utilizador fazer pesquisas de trajetos através de uma tela que pode ser parametrizável como se visualiza na Figura 38.

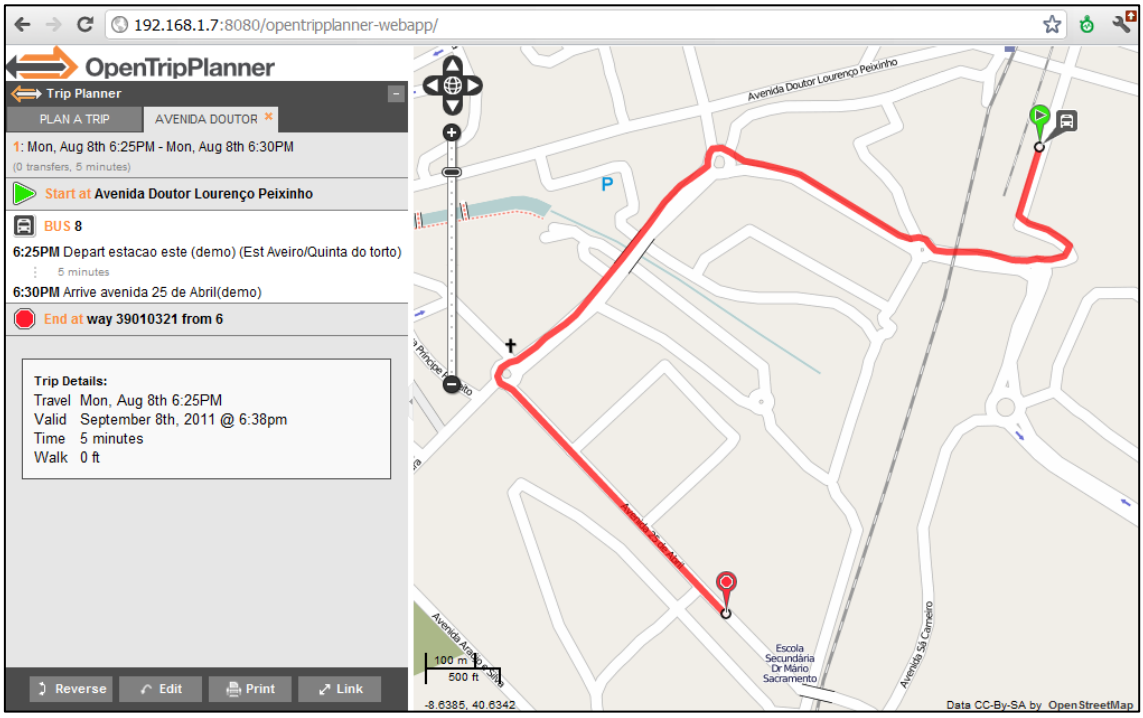


Figura 38: Aspecto de uma pesquisa de trajetos entre pontos utilizando a interface web OTP

Por outro lado, o ficheiro opentripplanner-api-webapp.war implementa a interface RESTful do OTP, que permite que sejam efectuadas consultas fazendo uso da aplicação MuvOnSchedule. Por exemplo, um utilizador faz uma pesquisa de itinerários entre dois pontos utilizando a aplicação MuvOnSchedule e é construído um pedido e enviado via http ao servidor API RESTful com o seguinte aspecto:

http://192.168.1.7:8080/opentripplanner-api-webapp/ws/plan?fromPlace=40.643901929364,-8.6417646878053&toPlace=40.635711837356,-8.6441894047547&arr=Depart&min=QUICK&maxWalkDistance=840&mode=BUSISH,TRAINISH,WALK&itinID=1&submit&date=08/08/2011&time=4:22%20pm

O servidor retorna uma resposta no formato JSON que descreve os passos que o utilizador deverá seguir para chegar ao seu destino. O aspeto da resposta devolvida pelo servidor pode ser visualizada nos anexos deste, documento especificamente na Figura 48.

4.5 MuvOnSchedule

MuvOnSchedule é o nome dado ao protótipo que será instalado em equipamentos Android e que vai permitir efetuar pesquisas de percursos entre dois pontos, utilizando a informação existente no servidor Open Trip Planner. Possui um pequeno modelo de dados que armazena, em base de dados local do dispositivo, informações sobre a aplicação, como por exemplo os filtros a disponibilizar na pesquisa, como o "caminho mais curto", o "caminho mais rápido", etc e uma tabela de muv_object que armazena o tipo de objecto que queremos persistir como exemplifica a Figura 39.

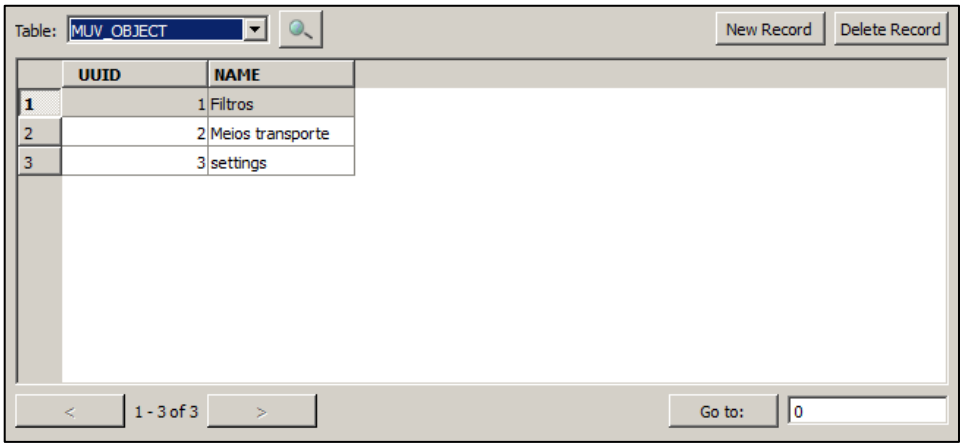


Figura 39: Aspecto da tabela MUV_OBJECT

Também existe uma tabela muv_object_values que armazena os valores respectivos para esse objecto (Figura 40):

Table: MUV_OBJECT_VALUES

New Record | Delete Record

	UUID	NAME	PARAMETER	VALUE	OBJECT_UUID	LANGUAGE
1	1	Mais rápido	MIN	QUICK	1	1
2	2	Mais curto	maxWalkDistance	160	1	1
3	3	Menos transbordos	MIN	TRANSFERS	1	1
4	4	Serviço noturno	TIME	10:00 pm	1	1
5	5	Deficientes	WHEELCHAIR	TRUE	1	1
6	6	A pé	MODE	WALK	2	1
7	7	Autocarro	MODE	BUSISH	2	1
8	8	Electrico	MODE	TRAM	2	1
9	9	Metro	MODE	SUBWAY	2	1
10	10	server_url	url	http://192.168.1.7:	3	1

< | 1 - 10 of 10 | > | Go to: | 0

Figura 40: Tipo de informação armazenada na tabela MUV_OBJECT_VALUES

A aplicação está dividida em quatro partes: procura, favoritos, pontos de interesse e definições. A primeira parte está orientada para fazer pesquisas de itinerários envolvendo um ponto de origem e de destino mostrando uma lista dos resultados onde o utilizador terá a liberdade de escolher qual o trajeto mais adequado para si. Consoante a escolha feita serão mostradas as indicações dos passos a seguir para chegar ao destino. Ainda se encontra em desenvolvimento a funcionalidade que permite escolher um trajeto e mostrá-lo no mapa, fazendo *highlight* do caminho a percorrer, como também adicionar o mesmo aos favoritos. Já no separador dos favoritos será mostrada uma lista dos trajetos assinalados como favoritos em pesquisas anteriores. Por outro lado, existirá também um separador onde serão apresentados os pontos de interesse referentes à zona envolvente relacionada com a pesquisa efetuada. O último separador é o das definições, onde será possível configurar definições como o endereço do servidor Open Trip Planner, o raio de procura de pontos de interesse, entre outras definições.

Como se pode observar na Figura 41, o utilizador ao iniciar a aplicação acede diretamente ao tabulador de pesquisa, que permite indicar o local origem, destino e parametrizar outros parâmetros inerentes a procura que quer efectuar, como por exemplo se deseja obter o caminho mais curto ou qual o tipo de transporte.

Figura 41: Ecrã inicial aplicação MuvOnSchedule

Ao colocar uma morada nas caixas “origem” ou “destino”, estas são automaticamente pesquisadas através do Google geocode API em que o utilizador recebe uma lista de moradas, para que se proceda à escolha da morada correta, como se visualiza na Figura 42:

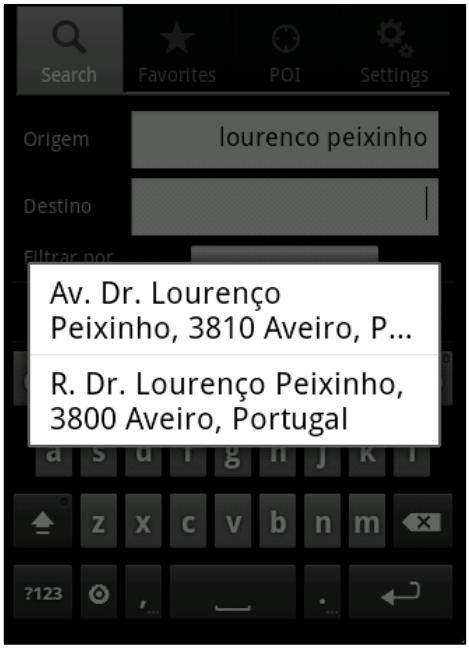


Figura 42: Exemplo processo de *geocoding*

Após a definição das moradas de origem e destino, procede-se à fazer a filtragem das carreiras cujo percurso é mais rápido ou possui menos transbordos, entre outras opções, como se pode verificar na Figura 43.

Figura 43: Filtros disponíveis no ecrã de pesquisa

Também é possível filtrar que tipo de meio de transporte se deseja envolver na pesquisa, de forma a considerar ou não todas as opções disponíveis entre duas localizações. Ao seleccionar o botão tipos de transporte aparece a caixa de selecção assinalada na Figura 44.

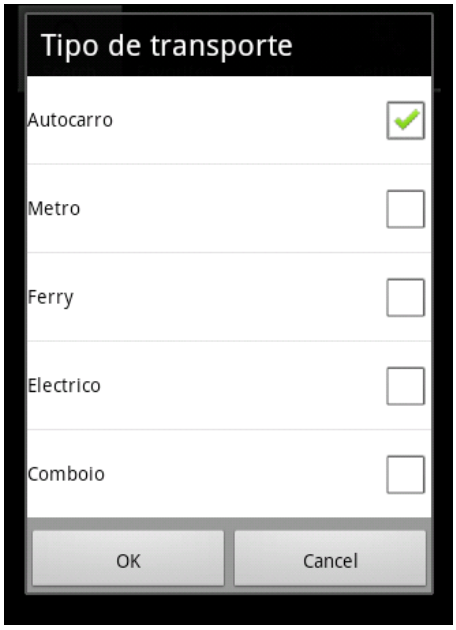


Figura 44: Filtros referentes aos tipos de transporte

É de salientar que a pesquisa só é possível indicando as moradas de origem e destino. Após indicar os filtros e os meios de transporte desejados, clica-se no botão “Procurar” onde são listados três percursos para a linha 8 entre a Avenida Doutor Lourenço Peixinho e a Avenida 25 de Abril como se pode verificar na Figura 45.

Figura 45: Listagem de itinerários resultantes da pesquisa

Por outro lado, pode-se observar na Figura 46 que o protótipo implementa filtros que não se encontram disponíveis na ferramenta Google Maps mobile (por exemplo), como a filtragem por tipo de percurso seja ele o mais rápido, com menos transbordos ou ainda o tipo de meio de transporte.

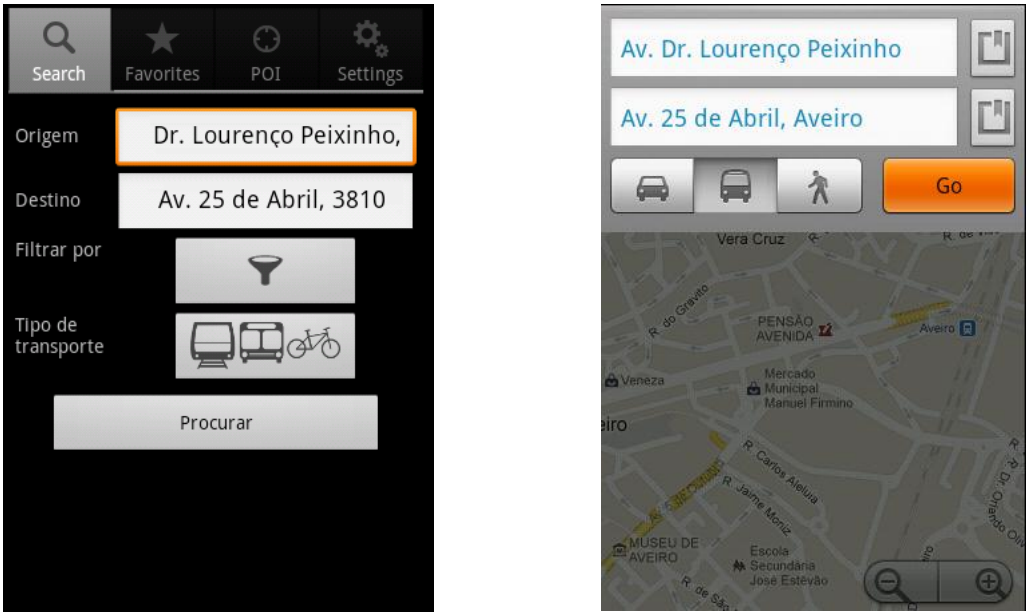


Figura 46: Comparação entre MuvOnSch e Google Maps

5 Conclusões

Após a realização e apresentação do trabalho desenvolvido é altura de realizar um balanço do trabalho realizado.

Este projeto foi iniciado com a finalidade de implementar um protótipo que permita o cálculo de itinerários fazendo uso de uma plataforma de pesquisa e partilha de informações sobre meios massivos de transporte. Este objectivo foi atingido com sucesso, permitindo que outras pessoas explorem e aprofundem as diversas funcionalidades que um sistema deste tipo pode oferecer aos utentes de meios de transporte públicos.

Foram encontradas inúmeras dificuldades no desenvolvimento do trabalho a nível de integração das diversas ferramentas, assim como ao nível do planeador intermodal de itinerários, contudo foi uma experiência enriquecedora, visto que permitiu explorar novos métodos de criação de código como o *Model-View-Controller*, experimentar o desenvolvimento usando a plataforma Android, assim como estar ciente dos desafios diários que milhares de utilizadores têm de enfrentar no momento de empreender uma viagem e sobre tudo compreender e satisfazer as suas necessidades.

Em paralelo com este trabalho foi desenvolvido um artigo[19], apresentado na conferência Centeris 2011 em Vilamoura, que permitiu avaliar e corroborar que o JSON é efectivamente o melhor formato para partilha de informação utilizando *smartphones*, por apresentar tempos de resposta mais baixos e largura de banda mais reduzida na troca de informação, face ao formato XML. Este contributo irá ajudar outros programadores que queiram desenvolver aplicações para dispositivos móveis.

A utilização do padrão *Model-View-Controller* permitiu construir a base do protótipo de forma modular, permitindo que qualquer pessoa possa programar e perceber o trabalho até agora desenvolvido e rapidamente desenvolver novas funcionalidades.

5.1 Pontos a explorar

Um dos aspectos que pode ser melhorado é a apresentação e usabilidade da aplicação móvel. Deveriam ser feitos testes de *stress* para perceber quais as limitações do protótipo e em que pontos se pode melhorar o desempenho do mesmo.

Um ponto importante seria parametrizar e desenvolver a aplicação *web* disponível por defeito no OTP de forma a permitir aos utilizadores efectuar pesquisas de trajetos via *web*. Outro ponto que catalisaria o crescimento da plataforma e facilitaria a inserção de informação na plataforma, seria a criação de aplicação mobile que permitisse aos motoristas registar as rotas fazendo uso do formato GTFS. Isto permitiria o registo de todas as rotas e trajetos de forma mais rápida e eficiente.

5.2 Trabalho futuro

Certamente que antes deste projeto entrar em produção existe muito trabalho pela frente. Um projeto desta magnitude precisa de mais recursos humanos e tempo para o desenvolvimento, maturação e refinamento das suas funcionalidades.

Deve ser evoluído o modelo de dados por forma a permitir criar e acomodar novas funcionalidades avançadas, como a pesquisa de itinerários *offline*, partilha de itinerários favoritos com outros utilizadores, etc.

Existem ainda inúmeras funcionalidades que podem ser adicionadas, como:

- Criação de módulo de notificações e alarmes que permita mostrar ao utilizador informações importantes com relação ao serviço de transporte público que pretende;
- Desenvolver modulo que permita a organização e gestão de viagens em grupo e excursões;
- Desenvolver módulo de realidade aumentada que permita aos turistas identificar e saber mais sobre os monumentos referentes a uma cidade;
- Desenvolver interface que permita utilizar várias API's de *geocoding*;

- Suporte para i18n internacionalização em diversas línguas;
- Pesquisa de trajetos com reconhecimento de voz para pessoas invisuais;
- Criação de módulo de publicidade consoante a localização do utilizador;
- Possibilidade de comentar e recomendar um itinerário a outros utilizadores.

6 Bibliografia

[1]. Allamaraju, S., ed. *RESTful Web Services Cookbook: Solutions for Improving Scalability and Simplicity*. ed. O'Reilly. 2010.

[2]. Alliance, O.H. *Android*. 2011 [cited 2011 September]; Available from: http://www.openhandsetalliance.com/android_overview.html.

[3]. Ana Rita, R.J., *TecoPlanner Bussiness Plan*. 2011. p. 1,2.

[4]. AppBrain. *Pdxtrian*. 2008 [cited 2011 September]; Available from: http://www.appbrain.com/app/pdxtrian/com.pdxtrian.alpha.

[5]. Asus, G. *nüvifone A10*. 2011 [cited 2011 April]; Available from: http://www.garminasus.com/en_GB/phones/nuvifone-a10/.

[6]. Carneiro, L., ed. *Sistemas distribuídos com RMI, CORBA e SOA*. 2008.

[7]. Coelho, P.L.d.F.e., *Open Location Based Services - Application Platform*. 2010.

[8]. Crockford, D., ed. *JavaScript: The Good Parts*. 2008, O'Reilly.

[9]. Developers, A. *ADT Plugin for Eclipse*. 2011 [cited 2011 September]; Available from: http://developer.android.com/sdk/eclipse-adt.html.

[10]. Freitag, P. *REST vs SOAP Web services*. 2005 [cited 2011 September]; Available from: http://www.petefreitag.com/item/431.cfm.

[11]. Google. *Plan a trip using public transportation*. 2007 [cited 2011 October]; Available from: http://www.google.com/intl/en/landing/transit/#mdy.

[12]. Google. *General Transit Feed Specification*. 2010 [cited 2011 April]; Available from: http://code.google.com/transit/spec/transit_feed_specification.html.

[13]. Google. *Google Maps API Web services*. 2011 [cited 2011 June]; Available from: http://code.google.com/intl/pt-PT/apis/maps/documentation/geocoding/.

[14]. Google. *What is Android?* 2011 [cited 2011 September]; Available from: http://developer.android.com/guide/basics/what-is-android.html.

[15]. Gordon, M. *OpenTripPlanner code overview*. 2011 [cited July 2011]; Available from: http://www.locationaware.usf.edu/wp-content/uploads/2011/03/OpenTripPlaner-Code-Overview.pdf.

[16]. GSON. *Google GSON*. 2008 [cited 2011 January]; Available from: http://code.google.com/p/google-gson/.

[17]. GTFS. *General Transit Feed Specification*. 2010 [cited 2011 April]; Available from: http://code.google.com/transit/spec/transit_feed_specification.html.

[18]. Hatem Hamad, M.S.e.R.A., *Performance Evaluation of RESTful Web Services for Mobile Devices*. 2009.

[19]. José Afonso, C.R.e.P.T., *Mobile Application Webservice Performance Analysis: Restful services with JSON and XML*. 2011.

[20]. Leonard Richardson, S.R.e.D.H.H., ed. *RESTful Web Services*. 2008, O'Reilly.

[21]. Nokia. *Ovi maps*. 2011 [cited 2011 April]; Available from: http://www.ovi.com/services/.

[22]. OpenTripPlanner. 2011 [cited 2011 July]; Available from: http://opentripplanner.com/learn.

[23]. Pereira, B. *Webservices REST*. 2009 [cited 2011 September]; Available from: http://brunopereira.org/tag/restful-web-services/.

[24]. Protsenko, M., *Distance problems in networks - Theory and praxis*. 2010.

[25]. Rick Rogers, J.L., Zigurd Mednieks e Blake Meike, ed. *Android application development*. 2009, O'Reilly

[26]. ROTH, M. *How Google and Portland's TriMet set the standard for Open Transit Data*. 2010 [cited 2011 July]; Available from: http://sf.streetsblog.org/2010/01/05/how-google-and-portlands-trimet-set-the-standard-for-open-transit-data/.

[27]. SHARMA, P. *Google transit: A great asset to Google Maps*. 2009 [cited 2011 June]; Available from: http://www.techpluto.com/google-transit-benefits/.

[28]. Sturgeon, P. *Geocoding API's compared*. 2011 [cited 2011 August]; Available from: http://philsturgeon.co.uk/blog/2011/02/geocoding-apis-compared.

[29]. Wikipedia. *Android software development*. 2011 [cited 2011 September]; Available from: http://en.wikipedia.org/wiki/Android_software_development.

[30]. Yahoo! *Yahoo! PlaceFinder Guide*. 2011 [cited 2011 September]; Available from: http://developer.yahoo.com/geo/placefinder/guide/.

Anexos

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd">

    <bean id="graphBundle" class="org.opentripplanner.model.GraphBundle">
        <property name="path" value="/home/demo/dirAuxOTP/graph-bundle" />
    </bean>
    <bean id="gtfsBuilder" class="org.opentripplanner.graph_builder.impl.GtfsGraphBuilderImpl">
        <property name="gtfsBundles">
            <bean id="gtfsBundles" class="org.opentripplanner.graph_builder.model.GtfsBundles">
                <property name="bundles">
                    <list>
                        <bean class="org.opentripplanner.graph_builder.model.GtfsBundle">
                            <property name="path" value="/home/demo/dirAuxOTP/GTFS_moveaveiro.zip" />
                        </bean>
                    </list>
                </property>
            </bean>
        </property>
    </bean>
    <bean id="osmBuilder" class="org.opentripplanner.graph_builder.impl.osm.OpenStreetMapGraphBuilderImpl">
    <property name="provider">
        <!-- Use an OSM provider that will automatically download OSM data in a specified region -->
        <bean class="org.opentripplanner.graph_builder.impl.osm.RegionBasedOpenStreetMapProviderImpl">
            <property name="regionsSource">
            <!-- The region to collect OSM data -->
            <bean class="org.opentripplanner.graph_builder.impl.FixedRegionSourceImpl">
                <property name="latFrom" value="40.670731" />
                <property name="lonFrom" value="-8.647540" />
                <property name="latTo" value="40.636150" />
                <property name="lonTo" value="-8.674970" />
            </bean>
            </property>
            <!-- Specify where downloaded OSM data will be cached -->
            <property name="cacheDirectory" value="/home/demo/otp/osm" />
        </bean>
    </property>
    </bean>
```

Figura 47: Ficheiro de configurações graph-config.xml

```
{
 - plan: {
      date: "2011-08-08T16:22:00+01:00",
    - from: {
         name: "Avenida Doutor LourenÃ§o Peixinho",
         lon: "-8.641764621553254",
         lat: "40.64390156281642",
         geometry: "{"type": "Point", "coordinates": [-8.641764621553254,40.64390156281642]}"
      },
    - to: {
         name: "way 39010321 from 6",
         lon: "-8.644196393132962",
         lat: "40.63571459739824",
         geometry: "{"type": "Point", "coordinates": [-8.644196393132962,40.63571459739824]}"
      },
    - itineraries: {
       - itinerary: {
            duration: "300000",
            startTime: "2011-08-08T18:25:00+01:00",
            endTime: "2011-08-08T18:30:00+01:00",
            walkTime: "0",
            transitTime: "300000",
            waitingTime: "0",
            walkDistance: "0.0",
            elevationLost: "0.0",
            elevationGained: "0.0",
            transfers: "0",
          - legs: {
             - leg: {
                  @agencyId: "MAVR",
                  @headsign: "Est Aveiro/Quinta do torto",
                  @route: "8",
                  @mode: "BUS",
                  startTime: "2011-08-08T18:25:00+01:00",
                  endTime: "2011-08-08T18:30:00+01:00",
                  distance: "909.5754807856498",
                - from: {
                     name: "estacao este (demo)",
                   - stopId: {
                        agencyId: "MAVR",
                        id: "st1"
                     },
                     lon: "-8.640243",
                     lat: "40.642427",
                     geometry: "{"type": "Point", "coordinates": [-8.640243,40.642427]}"
                  },
                - to: {
                     name: "avenida 25 de Abril(demo)",
                   - stopId: {
                        agencyId: "MAVR",
                        id: "st2"
                     },
                     lon: "-8.646272",
                     lat: "40.635646",
                     geometry: "{"type": "Point", "coordinates": [-8.646272,40.635646]}"
                  },
                + legGeometry: { … },
                  duration: "300000"
               }
            },
            tooSloped: "false"
         }
      }
   },
 + requestParameters: { … }
}
```


Figura 48: Resposta JSON do servidor OTP para pesquisa realizada pela aplicação MuvOnSchedule