Sérgio Miguel Almeida Gomes

TECO Planner

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Setembro de 2012

Sérgio Miguel Almeida Gomes

TECO Planner

Relatório de estágio submetido à Faculdade de Ciências da Universidade do Porto como parte dos requisitos para a obtenção do grau de Mestre em Engenharia de Redes e Sistemas Informáticos

Orientador: Ricardo Reis da Silva
Co-orientador: Prof. Fernando Silva

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Setembro de 2012

Resumo

Cada vez mais tem-se assistido ao aumento de ajudas ao planeamento de viagens em transporte particular, levando a que as pessoas não utilizem os transportes público. Factores como a falta de informação (operadores de transporte público, horários, tarifários, etc), calculadores de itinerários que não agregam toda a informação e a impossibilidade atual de planear viagens por transportes públicos em Portugal que envolvam mais de uma cidade e/ou operador surgiu a necessidade de criar um local único e acessível na internet. Este é o objetivo final do projeto de empreendedorismo social TECO Planner. Para apoiar o seu desenvolvimento, foi lançado o desafio para a adaptação da informação de transportes e do *software* em código aberto com vista à obtenção de um protótipo.

O processo foi iniciado com a transformação da informação na base de dados do Sistema de Informação Geográfica de Gestão de Carreiras (SIGGESC) - a fonte de informação integrada sobre transportes rodoviários em Portugal - para o formato standard a nível internacional, o formato General Transit Feed Specification (GTFS). Uma vez neste formato a informação pode ser utilizada para alimentar o OpenTripPlanner, uma das mais completas plataformas de *software* em código aberto para o cálculo de itinerários. Para atingir este objetivo foram analisados os modelos de bases de dados SIGGESC e GTFS e foi desenvolvido *software* para processamento e conversão semi-automático da informação alfanumérica e geográfica. Foram ainda propostas soluções para garantir a correta identificação dos registos no novo formato.

Por fim, a interface do OpenTripPlanner foi adaptado e integrado num portlet Liferay que fará parte de um portal com outra informação sobre transportes públicos. Concluído todo este processo, obteve-se um *software* que será utilizado futuramente para conversão dos dados da base de dados do SIGGESC e um portal *web* baseado no Liferay que permite calcular itinerários em transportes públicos.

Abstract

Increasingly has seen an increase in aid for trip planning for taking private transportation. This has the effect of reducing the number of the people that use public transportation. Factors such as the lack of information (public transport operators, timetables, tariffs, etc), journey planners that do not add all the information and the current inability to plan journeys by public transport in Portugal involving more than one city and/or operator, it became necessary to create a single and accessible on website where this would be possible. This is the ultimate goal of the social entrepreneurship project TECO Planner. To support its development, the challenge was launched to adapt the transport information and open source software in order to obtain a prototype.

The process began with the transformation of information in the database of Geographic Information System Career Management (SIGGESC in portuguese) - the source of information on integrated road transport in Portugal - into the standard international format, the General Transit Feed Specification (GTFS) format. Once in this format information can be used to feed the OpenTripPlanner, one of the most complete open source software platforms for journey planning. To achieve this goal we analyzed the databases models SIGGESC and GTFS and developed software for processing and semi-automatic conversion of alphanumeric and geographical information. Also proposed solutions to ensure the correct identification of the records in the new format.

Lastly, OpenTripPlanner's interface was adapted and integrated into a Liferay portlet that will be part of a portal with information on other public transport. Completed this process, we obtained a software that will be used for future conversion of data from the database SIGGESC and a portal web based Liferay allows calculating routes for public transport.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

A mobilidade da sociedade é extremamente importante para uma boa qualidade de vida. Num mundo cada vez mais global e com o aumento população, as necessidades de deslocação são também cada vez maiores. No entanto, o automóvel próprio é cada vez menos a solução para essas deslocações pelos seus elevados custos económicos, sociais (congestão, acidentes, etc.) e ambientais (poluição do ar, poluição sonora, etc.).

A introdução de veículos elétricos apenas resolve parcialmente esses problemas não resolvendo por exemplo o problema da congestão. Os transportes públicos são por isso um elemento essencial na transição para uma mobilidade mais sustentável.

Em Portugal e nos últimos anos tem-se assistido ao aumento dos serviços de transportes públicos [34–36], principalmente nos grandes centros urbanos do Porto e de Lisboa. Entre os diversos exemplos destacam-se o desenvolvimento do Metro do Porto, o reforço de linhas no Metro de Lisboa e novos e mais eficientes autocarros e comboios.

Para potenciar estes tipos de novos serviços é, no entanto, necessário que os utilizadores tenham conhecimento da sua existência. O planeamento de viagens porta a porta por transportes públicos em Portugal é atualmente impossível. Esta dificuldade ganha relevância quando o planeamento da mesma viagem por automóvel próprio é hoje efetuado recorrendo a um simples aparelho GPS (Sistema de Posição Global).

Um calculador de itinerários nacional é, por isso, um meio fundamental para remover uma barreira à utilização dos transportes públicos. Por outro lado, com base nas pesquisas efetuadas pelos utilizadores, a mesma ferramenta permitira obter informações importantes sobres reais necessidades de deslocação e assim permitir aos operadores

melhorar os seus serviços, transformando desta forma os transportes públicos como autocarro, comboio e metro em alternativas viáveis, rápidas, económicas, confortáveis e eficientes ao automóvel particular.

1.1 Motivação

Não existe em Portugal um calculador de itinerários que agregue todos os operadores públicos e privados de transportes, permitindo o cálculo de rotas porta a porta por transportes públicos entre os diversos pontos do país.

Atualmente para calcular um itinerário entre dois pontos de Portugal, em especial entre duas cidades, nem sempre existe informação facilmente disponível sobre os transportes possíveis, os interfaces de interligação, os horários, os tarifários e por fim o cálculo de uma rota entre cidades. Por outro lado, o planeamento de viagens por automóvel particular é cada vez mais simples podendo atualmente ser feito com recurso a um simples *smartphone*. Isto leva as pessoas a não optar por transportes públicos.

A resolução deste problema terá, por isso, impacto na mobilidade nacional ao colocar os transportes públicos numa situação mais favorável para competirem, com o automóvel particular. Um calculador de itinerários nacionais para transportes públicos fornecerá aos utilizadores informação útil (itinerários, preços, emissões CO_2, etc.) e num único local permitindo-lhes tomar decisões conscientes sobre as suas deslocações.

1.2 Objetivos

Com este projeto de estágio, pretende-se demonstrar a viabilidade de uma solução de cálculo de itinerários multimodal com software open source e fontes de dados actualmente utilizadas pelos operadores. Para esse efeito foram definidos os seguintes sub-objetivos:

- Análise e documentação do estado da arte dos calculadores de itinerários nacionais e internacionais para o conhecimento das funcionalidades, algoritmos, *software* e linguagens de programação já utilizados e que possam vir a ser implementados;
- Conhecer a base dados utilizado pelos operadores de transporte público em Portugal e desenvolvimento de *software* de conversão para o modelo standard a

nível internacional;

- Integração da interface do OpenTripPlanner para utilização num portal *web*;
- Testes de qualidade e eficácia.

A concretização destes objetivos foram plenamente sucedidos e os próximos capítulos dão conta das etapas desse percurso.

1.3 Estrutura

Para melhor compreensão do trabalho realizado, o relatório encontra-se dividido nos seguintes capítulos:

- **Estado da arte** - análise de calculadores de itinerários, ferramentas e tecnologias usadas para a concretização deste projeto;
- **De SIGGESC para GTFS** - descrição do processo de transformação da base de dados no formato SIGGESC para o formato GTFS.
- **Integração da interface do OpenTripPlanner para utilização num portal *web*** - explicação do processo de integração da interface do OpenTripPlanner para utilização num portal *web*;
- **Conclusão e trabalho futuro** - conclusão sobre o trabalho realizado, objetivos alcançados e uma descrição dos próximos passos que o projeto deve tomar.

Capítulo 2

Estado da Arte

Para a realização deste projeto foi necessário conhecer em mais detalhe, alguns calculadores de itinerários nacionais e internacionais para percebermos quais as tecnologias e *software* já utilizados e também passiveis de ser utilizados neste projeto. Foi também necessário compreender o modelo de base dados utilizado pelos operadores de transportes públicos em Portugal e conhecer os modelos utilizados a nível internacional.

2.1 Calculadores de itinerários

Um calculador de itinerários de transportes públicos permite efectuar o cálculo de um percurso entre dois pontos geográficos à escolha do utilizador, dando informação sobre quais os transportes que deve utilizar e o trajeto efetuado por eles. Além desta informação, calculador de itinerários pode fornecer mais informação como horários, tarifários, etc. O calculador de itinerários pode ser monomodal ou multimodal. Monomodal só permite efetuar o cálculo com base num modo de transporte, multimodal permite efectuar o cálculo com vários modos de transporte diferentes. Já existem diversos calculadores de itinerários de transportes públicos espalhados pelo mundo, mas para este estágio demos mais relevancia aos portugueses ou a outros que usassem software ou tecnologia importante para o projeto. Assim, esta secção analisa três calculadores de itinerários para transportes públicos: o Itinerarium [62, 66], o Transporlis [69] e o Portland Regional Trip Planner [1, 72]. Os primeiros dois porque são portugueses e são utilizados respetivamente nas áreas metropolitanas do Porto e Lisboa O terceiro é americano e está em funcionamento na cidade de Portland, estado de Oregon. O Portland Regional Trip Planner foi escolhido por ser baseado no

OpenTripPlanner [70, 71] que servirá de base ao projeto descrito neste relatório. O OpenTripPlanner é um software que permite o calculo de itinerários e as razões que levaram à escolha deste foi a inexistência de um outro que fosse de código aberto, a grande comunidade que desenvolve e suporta software e a utilização em vários países.

2.1.1 Itinerarium

O Itinerarium é um sistema de informação multimodal que engloba a Sociedade de Transportes Coletivos do Porto (STCP), os Comboios de Portugal (CP) do grande Porto e o Metro do Porto.

As funcionalidades do portal são o cálculo de planos de viagens, a localização de linhas que passam num determinado local e a consulta de informação sobre as linhas dos operadores.

O calculador calcula/planeia uma viagem entre dois pontos geográficos dando como opções ao utilizador as escolhas do início e fim da viagem, a data e hora da partida, o operador e o tempo que pretende andar a pé.

O resultado do cálculo contém informação detalhada sobre os percursos a pé a efetuar, os meios de transporte a utilizar, a duração do percurso, os tarifários aplicados nos meios de transportes, o número de transbordos e um mapa com o percurso.

O Itinerarium foi desenvolvido pelas empresas SIG2000 [65] e Imediata S.A [31] e o *software* utilizado foi o Arc Internet Map Server (ArcIMS) [2, 9] e Networking Engine Server [64].

O ArcIMS é um Sistema de Informação Geográfico (SIG) da empresa ESRI [8]. Este *software* consiste num servidor capaz de representar mapas num navegador *web* e responder a pedidos de dados efetuados pelo navegador *web* do lado do cliente.

O Networking Engine Server é uma extensão feita pela SIG2000. De acordo com a empresa, pretende complementar o *software* ArcGIS [9, 63] ao nível técnico na área de análise de redes, com as funcionalidades na determinação de percurso ótimos pedonais e rodoviários. Também determina pontos de interesse ao longo de um percurso ou percursos que obrigatoriamente passem por um conjunto de pontos de interesse. Possui ainda a capacidade de utilizar até sessenta tipos de temas de pontos de interesse, bem como a determinação de áreas de influência pedonal(zonas onde há mais vias para circulação a pé), rodoviária (zonas onde há mais movimento de transportes públicos)

e de pontos de interesse na área de influência.

2.1.2 Transporlis

O Transporlis é um sistema de informação multimodal que engloba a Carris, CP, Fertagus, Metro de Lisboa, Metro Transportes do Sul, Rodoviária de Lisboa, SCOTTURB, Soflusa, Transportes Coletivos do Barreiro, Transtejo, Transportes Sul do Tejo (TST) e Vimeca.

As funcionalidades do portal são a pesquisa por moradas ou pontos de interesse, o cálculo de planos de viagens, pesquisa de paragens, horários e tarifários dos operadores.

O calculador calcula/planeia uma viagem entre dois pontos geográficos permitindo aos utilizadores definir o início e fim da viagem, os meios de transporte ou os operadores, selecionar data e hora da partida ou chegada e ainda optar pelo percurso com menos transbordos ou menos pedonal.

O resultado desse cálculo contém informação detalhada sobre a hora da partida e chegada, o número de transbordos, as emissões de CO_2, a duração do percurso, o total de custo da viagem, o total da distância a percorrer, percurso a efetuar a pé, os meios de transporte a utilizar, os tarifários aplicados nos meios de transportes, o número de transbordos e um mapa com o percurso. A linha que representa o percurso de várias cores conforme o meio de transporte a utilizar.

O portal possui ainda informação sobre o horário das partidas e chegadas dos voos nacionais continentais e tarifários de estacionamento aplicados nos aeroportos de Lisboa, Porto e Faro.

O Transporlis foi desenvolvido pela GISMÉDIA, S.A. [15] e usa como base o Microsoft SQL Server 2008 [37], DotNetNuke Community Edition [3] e API Mapas do SAPO. O Microsoft SQL Server 2008 é um servidor de base de dados. O DotNetNuke Community Edition é sistema de gestão de conteúdos web baseado em Microsoft .Net e para o desenvolvimento de aplicações para ASP.NET.

2.1.3 Portland Regional Trip Planner

O Portland Regional Trip Planner [72] pertence à empresa TriMet [73]. O portal para além de um calculador de itinerários, permite procurar locais no mapa, dar informações

sobre as rotas dos meios de transporte, indicar os centros de trânsito (locais onde se encontram informações sobre os transportes públicos), parques de estacionamento e locais de venda de bilhetes.

O calculador é multimodal permitindo efetuar o cálculo/planeamento de uma viagem entre dois pontos geográficos a pé, carro, metro, autocarro e bicicleta. Para fazer o cálculo basta inserir o local de partida e chegada ou indicar os respetivos pontos no mapa, selecionar a hora de partida ou chegada, o dia, que meio(s) de transporte(s) a utilizar, a distância máxima a que se propõe a caminhar e a opção de itinerário com o caminho mais rápido ou menos transbordos.

O resultado desse cálculo contém informação detalhada sobre os percursos a pé a efetuar, os meios de transportes utilizar, as distâncias, a duração do percurso, o número de transbordos, o tarifário aplicado pelos operadores e um mapa com o percurso, sendo a linha que representa o percurso de varias cores conforme o meio de transporte a utilizar. O calculador oferece três opções de percurso para o utilizador escolher o que mais lhe convém. O Portland Regional Trip Planner foi desenvolvido recorrendo aos mapas do OpenStreetMaps [12], base de dados no formato General Transit Feed Specification (GTFS) [23] e *software* OpenTripPlanner [70, 71].

Tabela 2.1: Tabela comparativa dos calculadores de itinerários

	Itineratium	Transporlis	P.R. Trip Planner
escolha data e hora de partida	X	X	X
escolha data e hora de chegada	-	X	X
escolha do tempo a pé	X	-	-
escolha da distância a pé	-	-	X
escolha do caminho menos pedonal	-	X	-
escolha do caminho mais rapido	-	X	X
visualização da altitude	-	-	X
meio de transporte bicicleta	-	-	X
visualizaçao mapa satelite	-	X	X

2.2 OpenTripPlanner

O OpenTripPlanner é um calculador de itinerários multimodal que surgiu de um trabalho colaborativo entre a TriMet, operador de transportes públicos que atua na região de Portland, Oregon, EUA e um conjunto de empresas de *software* em código aberto nas áreas de transportes e informação geográfica entre as quais a OpenPlans [42].

O OpenTripPlanner, atualmente na versão 0.7.12, encontra-se em utilização final em Portland nos EUA, Valência em Espanha, Poznan e Lublin na Polónia. A versão atual, permite o cálculo de itinerários a pé, bicicleta, metro, comboio ou combinação entre eles, além de incluir *software* para estudos de acessibilidade e interligação com o Google Street View.

O cálculo de itinerários baseia-se na combinação do algoritmo A* [27, 28] com a técnica Contraction Hierarquies [14]. O algoritmo A* permite efectuar a pesquisa do percursos e a travessia em grafos. A técnica Contraction Hierarquies reduz o número de vértices em grafos com muitos vertices, permitindo assim reduzir o tempo de pesquisa dos percursos. Uma grande parte dos componentes do OpenTripPlanner está desenvolvida em Java sendo a interface do utilizador desenvolvida, em JavaScript, a principal exceção.

A arquitetura do OpenTripPlanner divide-se em três componentes, a interface *web* (Web Browser), o servidor *web* (Web Server) e o sistema de suporte (Backend System) como demonstra a figura 2.1. A interface *web* é o componente que o utilizador tem acesso através de navegador *web* e permite a visualização de mapas, obter informação e efetuar pedidos por parte do utilizador. Este componente utiliza *software* OpenLayers [10,45] e ExtJS [61] escritos em Javascript e Ajax. O Openlayers é um conjunto de bibliotecas de visualização e interação com mapas no navegador *web* do cliente. O ExtJS é uma framework para construção de aplicações web.

O servidor *web* subdivide-se nos componentes Apache e Tomcat. O Apache utiliza *software* Tilecache [26,33] escrito em Python, para a aceder aos mapas, possibilitando assim a sua renderização por parte do Openlayers. O Tomcat é uma implementação de servidor servlets e JavaPages(JSP). A estrutura é composta pelo *software* Geoserver [41, 44], PostGIS [11] e o serviço TripPlanner Web Service (TripPlanner WS). O Geoserver é um *software* escrito em Java que permite a edição de dados geográficos para serem utilizados pelo OpenLayers. Devido à sua interoperabilidade é compatível com qualquer fonte de dados desde que sejam standard (exemplos: KML, GML, OpenGIS ou GeoJSON). De salientar que o Geoserver é a implementação de referência

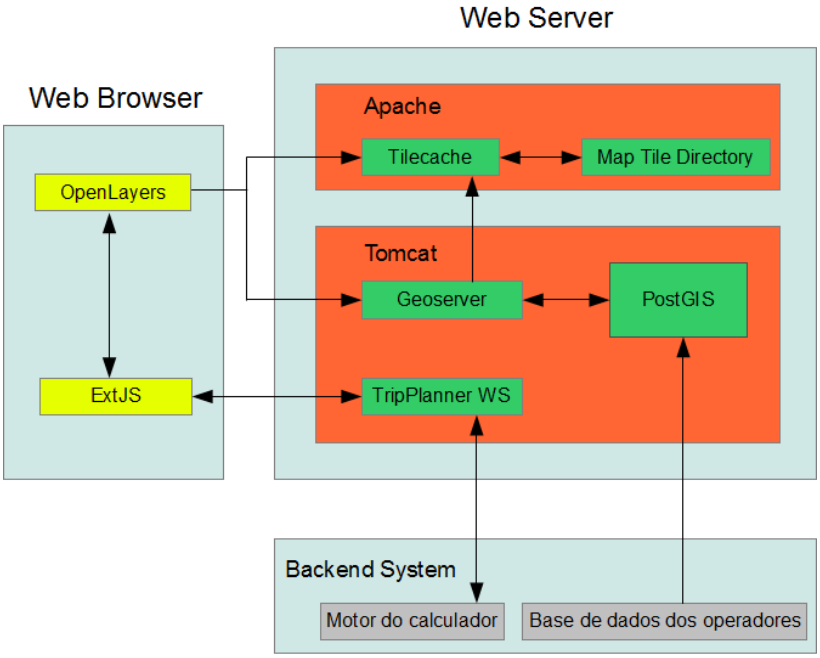

Figura 2.1: Estrutura do OpenTripPlanner

do consórcio OpenGeospatial para os serviços Web Feature Service (WFS), web Map Server (WMS) e Web Coverage Service (WCS). O PostGIS [11] é uma extensão espacial ao PostgreSQL [24] que permite a gestão de dados geográficos. O TripPlanner Web Service é o componente do OpenTripPlanner que comunica entre a interface e o motor do cálculo de itinerários. Quando se selecionam os pontos de partida e de chegada e as várias opções de viagem na interface, o TripPlanner Web Service recebe e faz o tratamentos dos dados para serem utilizados pelo motor do cálculo de itinerários.

O sistema de suporte (Backend System) é composto pelo motor do cálculo de itinerários escrito em Java e pela base de dados dos operadores de transportes públicos no formato GTFS.

2.3 Modelos de dados de transportes públicos

2.3.1 Sistema de Informação Geográfica e de Gestão de Carreiras

O Sistema de Informação Geográfica e de Gestão de Carreiras (SIGGESC) [4,54,56,60] surgiu em 2009, através de uma cooperação entre o Instituto Superior de Estatística e

de Gestão de Informação da Universidade Nova de Lisboa e o Instituto de Mobilidade e dos Transportes Terrestres (IMTT) [6]. O sistema consiste numa aplicação *desktop* denominada de Sistema de Informação de Carreiras dos Operadores(SICO) e um portal *web* SIGGESC. A aplicação SICO permite a introdução de dados por parte de cada um dos operadores e a criação de uma base de dados individual. O portal SIGGESC é uma aplicação *web*/*webgis* composto por uma interface *web* e uma aplicação ArcGis Server, a interface *web* SIGGESC permite a gestão de pedidos e transferências de base de dados entre os operadores e o SIGGESC. A aplicação ArcGis Server permite a visualização de outputs gráficos e geográficos sobre as bases de dados dos operadores facilitando a gestão dos transportes coletivos de passageiros por parte do IMTT.

2.3.2 Base de Dados do SIGGESC

A base de dados do SIGGESC contém informação sobre os transportes coletivos. A informação é constituída por dados alfanuméricos e geográficos, sendo cedida pelos operadores de transportes ao IMTT via *web*, através do portal SIGGESC. Não existe informação pública disponível sobre a constituição da base de dados. O acesso a uma base de dados no formato SIGGESC foi feita através dos dados cedidos por um operador de transporte coletivo.

A base de dados é constituída por trinta tabelas referentes aos dados alfanuméricos e duas referentes aos dados geográficos. Para clareza de exposição é exposto a seguir apenas as principais tabelas e campos essenciais para a compreensão do trabalho descrito no capitulo 3. Para mais detalhes é sugerido a consulta do anexo C1.

PARAGEM - identificação de cada paragem. O código de paragem único do SIGGESC como chave primária, a localização indicando a rua ou lugar onde se situa a paragem, o nome comum por que é conhecida (designacao) e o código de identificação da paragem para o respetivo operador (figura 2.3).

TROCOS - identificação de cada troço. O código de troço único do SIGGESC como chave primária, as duas paragens que definem a localização de inicio e o fim do troço, a extensão do troço em metros e os tempos mínimos, médios e máximos para percorrer o troço (figura 2.4).

OPERADORES - identificação de cada operador. O código de operador único do SIGGESC como chave primária, o nome do operador (designoperador) e o url para o site do operador (figura 2.5).

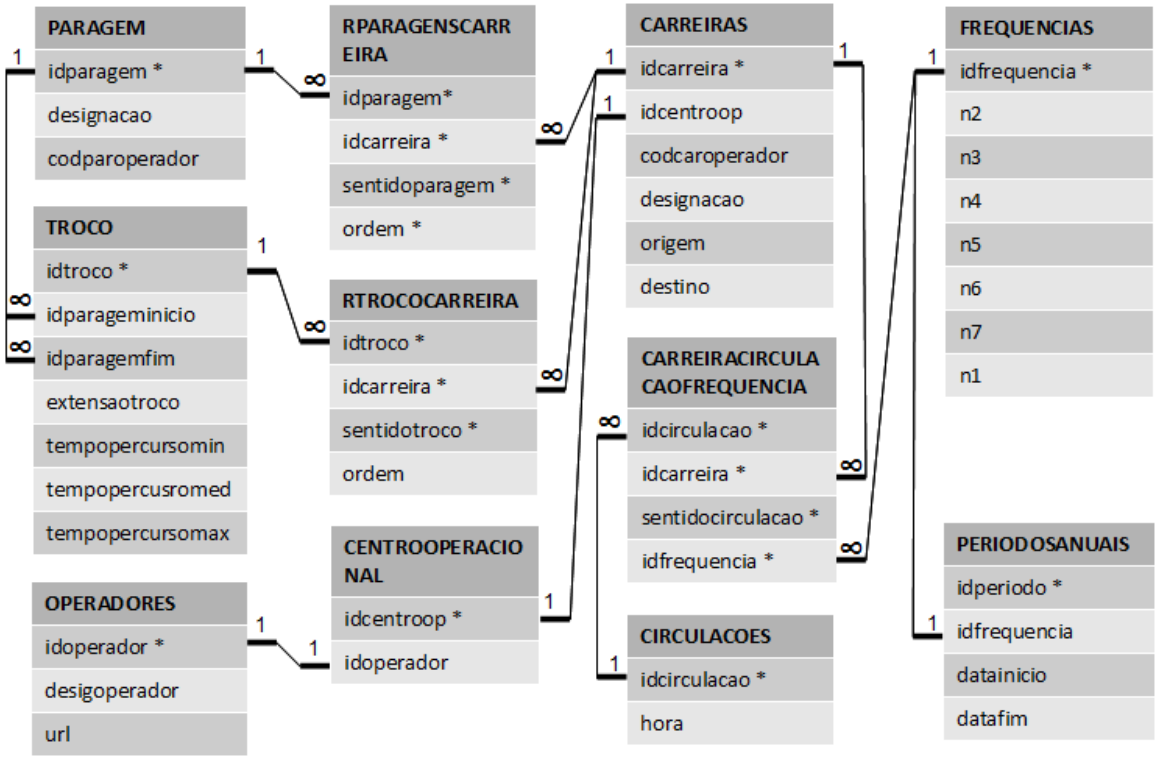

Figura 2.2: Tabelas e campos essenciais da base de dados alfanumérica do SIGGESC(Modelo Relacional)

idparagem*	designacao	codcaroperador
7511	Sampaio	219

Figura 2.3: Exemplo de dados da tabela PARAGEM

idtroco*	idparagemi nicio	idparage mfim	extensaot roco	tempopercur somin	tempopercurs omed	tempopercurs omax
123	7502	7503	136	1	1	1

Figura 2.4: Exemplo de dados da tabela TROCOS

idoperador*	designoperador	url
187	António da Silva Cruz & Filhos, Ldª	http://www.maiatransportes.pt

Figura 2.5: Exemplo de dados da tabela OPERADORES

CARREIRAS - identificação de cada carreira. O código de carreira único do SIGGESC como chave primária, código do centro operacional onde é operado a carreira, código da carreira nos sistemas do operador, nome comum por que é conhecida (designacao) e a localização do início e fim da carreira em origem e destino (figura 2.6).

idcarreiras*	idcentroop	codcaroperador	designacao	origem	destino
2820	1	13	Ermesinde(estação cp)-Maia(centro	Ermesinde	Maia

Figura 2.6: Exemplo de dados da tabela CARREIRAS

CIRCULACOES - identificação de todas as horas possíveis. O código de circulação que identifica cada hora e o campo hora com o valor da hora (figura 2.7).

idcirculacao*	hora
1	00:00:00
2	00:01:00
3	00:02:00
...	...
1440	23:59:00

Figura 2.7: Exemplo de dados da tabela CIRCULACOES

FREQUENCIAS - identificação de cada frequência. O código de frequência único do SIGGESC como chave primária e os campos seguintes identificam os dias da semana em que há serviço (figura 2.8).

idfrequencia*	n2	n3	n4	n5	n6	n7	n1
34	True	True	True	True	True	False	False

Figura 2.8: Exemplo de dados da tabela FREQUENCIAS

PERIODOSANUAIS - identificação de períodos anuais. O código do período é único do SIGGESC como chave primária. Ao período está associado uma frequência e uma data de início e fim do período respetivamente (figura 2.9).

RPARAGENSCARREIRAS - identificação das paragens que cada carreira possui. Os campos idparagem, idcarreira, sentidoparagem e ordem compõem a chave primária. Assim consegue-se saber que paragens contém determinada carreira, em que sentido e ordem está a paragem na carreira (figura 2.10).

idperiodo*	idfrequencia	datainicio	datafim
1	34	01-01-2012	31-12-2012

Figura 2.9: Exemplo de dados da tabela PERIODOSANUAIS

idparagem*	idcarreira*	sentidoparagem*	ordem*
7502	2820	2	28

Figura 2.10: Exemplo de dados da tabela RPARAGENSCARREIRAS

RTROCOCARREIRAS - identificação dos troços que cada carreira possui.Os campos idtroco, idcarreira, sentidotroco e ordem compõem a chave primária. Assim conseguem saber que troços contém determinada carreira, o sentido e a ordem do troço na carreira (figura 2.11).

idtroco*	idcarreira*	sentidotroco*	ordem*
123456	2820	1	20

Figura 2.11: Exemplo de dados da tabela RTROCOCARREIRAS

CENTROOPERACIONAL - identificação dos centros onde os operadores operam em caso de grupos. O código do centro operador do SIGGESC é único e chave primária e o idperador indica o operador (figura 2.12).

CARREIRACIRCULACAOFREQUENCIA - identificação da frequência, o sentido de circulação e uma hora de partida de uma determinada carreira (figura 2.13).

2.3.3 General Transit Feed Specification

General Transit Feed Specification (GTFS) [23] é um formato aberto de representação de dados para gestão de transportes públicos e informação geográfica, desenvolvido pela Google. A base de dados permite que os operadores de transporte publiquem os seus dados e programadores desenvolvam aplicações na área dos transportes como calculadores de itinerários. Para facilitar a portabilidade dos dados, as diferentes tabelas são individualmente exportadas como ficheiros Comma Separated Value (CSV) e agregadas num único ficheiro zip, possuindo informação sobre os operadores, rotas, paragens, viagens, horários, etc.

De acordo com Matthew Roth [59], o General Transit Feed Specification surgiu em

idcentroop*	idoperador
1	187

Figura 2.12: Exemplo de dados da tabela CENTROOPERACIONAL

idcirculacao*	idcarreira*	sentidocirculacao*	idfrequencia*
11	2820	1	34

Figura 2.13: Exemplo de dados da tabela CARREIRACIRCULACAOFREQUENCIA

PARAGENS
carreira*
parcelar*
variante*
sentido*
ordempar
codparagem*
the_geom

TROCOS
carreira*
parcelar*
variante*
sentido*
ordemtro
codparin*
codparfi*
the_geom

Figura 2.14: Tabelas e campos essenciais da base de dados geográfica do SIGGESC

carreira*	parcelar*	variante*	sentido*	ordempar	codparagem*	the_geom
10	1	0	0	1	7802	010100002065

Figura 2.15: Exemplo de dados da tabela PARAGENS

carreira*	parcelar*	variante*	sentido*	ordemtro	codparin*	codparfi*	the_geom
10	0	1	1	10	7809	78010	0105000020E61

Figura 2.16: Exemplo de dados da tabela TROCOS

2005 depois de Bibiana McHugh, gestora de tecnologias de informação do operador de transporte TriMet da área de Portland, ter viajado para fora dos EUA e não ter conseguido planear a sua viagem. No regresso aos EUA, Bibiana McHugh contatou as empresas Mapquest, Yahoo e Google no sentido de saber quais os planos que as empresas tinham para incorporar dados de trânsito nos seus serviços de mapas, e se, a TriMet poderia ser parceiro no projeto. A única a responder foi a Google através do engenheiro de *software* Chris Harrelson, que usava parte do seu tempo numa interface de dados de trânsito com o Google Maps ao qual se tornou Google Transit Trip Planner. A partir daí a TriMet e a Google trabalharam em conjunto na preparação dos dados de trânsito da própria TriMet e definiram um formato para trabalhar com o Google Maps.

2.3.4 GTFS *feed*

GTFS *feed* [18, 22] consiste num conjunto de treze ficheiros, seis obrigatórios e sete opcionais que constituem a base dados GTFS. Os ficheiros contêm campos que podem ser obrigatórios ou opcionais, devem estar no formato CSV, a primeira linha deve conter os nomes dos campos e as linhas seguintes o conteúdo correspondente a cada campo. É este grupo de ficheiros agregados num único ficheiro .zip que alimenta (*feed*) calculadores de itinerários e outras aplicações na área dos transportes.

Para ajudar os operadores na construção da GTFS, a Google disponibiliza um exemplo de GTFS [17], um guia de referência e ferramentas de validação [21]. Disponibiliza também um site [20] onde permite que os operadores incluam os seus dados em formato GTFS na base de dados do Google Maps.

2.4 Ambientes de desenvolvimento

Para o projeto, o ambiente de desenvolvimento a ser usado tinha que ter como principais requisitos:

- o suporte para JEE devido a necessidade uma plataforma completa de ferramentas necessárias ao desenvolvimento de aplicações *web*;
- ser grátis e código aberto para diminuir os custos de início do projeto.

Encontrámos vários ambientes de desenvolvimento [74]: Eclipse, Netbeans, IntelliJ_IDEA, Jbuilder, JDeveloper, MyEclipse e IBM Rational Application Developer.

Os ambientes mais conhecidos e que respondiam aos requisitos são o Eclipse [13] e o NetBeans [49]. Verifiquei que não há consenso sobre o qual é melhor. Mas existem opiniões que vão no mesmo sentido: o Netbeans possui uma melhor interface gráfica, utiliza mais memória RAM (Ramdom Acess Memory) e de origem vem mais completo mas o Eclipse devido aos *plugins* é mais personalizável, mais completo e mais portátil pois não é preciso instalar no computador.

Ambos os ambientes são multiplataforma (correm em Windows, Linux e Mac OS) e possuem licenças de código aberto. O Eclipse é desenvolvido por uma comunidade que pertence à fundação Eclipse. O Netbeans é desenvolvido pela Oracle.

O ambiente escolhido foi o Eclipse porque para além de corresponder aos requisitos, tanto eu como a empresa temos experiência na utilização deste ambiente.

2.5 Sistemas de Gestão de Base de Dados

Para que pudessemos executar a tranformação da base de dados no formato SIGGESC para o formato GTFS (3) era necessário um *software* de gestão de bases de dados com suporte para dados geográficos [46]. Ao fazer a pesquisa aos calculadores de itinerários, encontrámos o Microsoft SQL Server, então pesquisámos por outros gestores de base de dados geográficos e descobrimos PostgreSQL [24] com PostGIS [46], Oracle Spatial or Locator [50], IBM DB2 Spatial Extender [29], IBM DB2 Informix with Spatial Blade [30], MySQL Spatial [48] . Todos eles seguem o modelo definido pelo Open Geospatial Consortium chamado SFSQL("Simple Feature for SQL"). SFSQL [40, 47] "especifica regras particulares para construção de geometria válidas, representação de geometrias no formato ASCII e binário, e um conjunto de funções básicas para construção, inspeção, medição e manipulação de geometrias".

O IBM DB2 Spatial Extender e IBM DB2 Informix with Spatial Blade são dois gestores pouco conhecidos e atualmente desenvolvidos pela empresa IBM. As licenças são proprietárias.

O MySQL Spatial é um gestor atualmente desenvolvido pela Oracle. O número de funções MySQL suportado é pequeno o que dificulta o uso das operações sobre base de dados e o motor do MySQL, o MyISAM (Método de Acesso Sequencial de Indexado)

não possui suporte para transações sobre objetos geográficos. A licença é licença publica geral(GPL) [16].

O Oracle Spatial or Locator são dois gestores desenvolvidos pela empresa Oracle. O Oracle Locator possui menos funcionalidades que o Spatial. Para além da Oracle Spatial ser mais completo, possui um conjunto de funcionalidades mais complexas. Isto permite que o Oracle Spatial seja o gestor de base dados mais completo que todos os referênciados. A licença é proprietária.

O Microsoft SQL Server é desenvolvido pela Microsoft. Em relação a outros gestores possui algumas diferenças ao nível da sintaxe e o índex espacial usa o esquema de rede multinível em vez do R-tree usado no PostgreSQL e Oracle. Também possui algumas limitações que são: correr só em servidores Windows e não possuir suporte para transformação entre os vários tipos de coordenadas. A licença é proprietária.

O PostgreSQL é desenvolvido pelo Grupo de Desenvolvimento Global PostgreSQL. O gestor é desenvolvido para ser possível a utilização da extensão PostGIS, assim permitindo uma muito boa integração por parte da extensão e permitindo o tratamento de dados geográficos no PostgreSQL. A principal limitação é o não suporte direto a coordenadas geográficas sendo necessárias funções para o tratamento de coordenadas geográficas. O PostgreSQL tem licença com código aberto e livre. O PostGIS tem licença GPL.

Para o projeto foi escolhido o PostgreSQL. As razões que levaram à escolha deste sistema de gestão de base de dados foram a eficiência quando aplicado num servidor, que se mostra como um dos melhores [38], a utilização do PostGIS, a extensão geográfica do PostgreSQL, também no OpenTripPlanner e finalmente pelo facto de ser o único que é grátis e de código aberto.

Capítulo 3

De SIGGESC para GTFS

Para ser possível utilizar os dados SIGGESC no OpenTripPlanner, é necessário fazer uma transformação para uma base de dados no formato GTFS. Embora esteja previsto, no futuro, o acesso através de um serviço *web* à base de dados SIGGESC, nesta fase optou-se por aceder aos dados de um operador de forma desconexa. A componente alfanumérica foi-nos fornecida na forma de uma base de dados enquanto que a componente geográfica foi-me fornecida na forma de duas *shapefiles*[1] uma contendo informação sobre paragens (Paragens.shp) e a outra sobre percursos (Troços.shp).

A conversão de shapefiles para uma base de dados PostgreSQL, bem como toda a manipulação de dados geográficos no PostgreSQL foi feita utilizando o PostGIS, a extensão espacial do PostgreSQL.

3.1 Uma nota sobre coordenadas geográficas

A representação de um corpo curvo e a três dimensões (a Terra) num plano a duas dimensões (mapa) preservando os ângulos retos entre os meridianos e os paralelos é feita com recurso a modelos matemáticos. Dependendo da utilização prevista do mapa e a área do planeta que se pretende colocar em evidência, são utilizados diferentes modelos e pontos de referência (chamados datum) [32, 67].

Em Portugal continental são utilizados fundamentalmente os datum Lisboa [5,58] com o ponto de fixação situado no Castelo de São Jorge e o datum 73 [5, 57] com o ponto

[1] *Shapefile* é um formato de dados vetoriais geoespaciais que guarda informação de geometrias(pontos,linhas,poligonos).

de fixação no vértice geodésico de 1ª ordem na serra de Melriça, distrito de Castelo Branco. O datum 73 encontra-se obsoleto, embora uma boa parte da cartografia actual ainda use este sistema de coordenadas. O sistema de referência standard (SRS) actual é PT-TH06-ETS89 (european terrestrial ref system) de 1989. O posicionamento global, por outro lado, é feito utilizando o Sistema Geodésico Mundial (World Geodetic System) definido em 1984 e conhecido por WGS84. Este sistema utiliza o centro de massa da Terra como ponto de referência [68].

É essencial que a projeção no mapa de determinada informação geográfica seja feita utilizando o mesmo sistema sob pena de as localizações mapeadas não corresponderem às localizações pretendidas. No entanto, uma vez conhecido o sistema em que se encontra a informação é sempre possível converte-la para qualquer outro sistema.

No PostgreSQL com extensão PostGIS esta conversão é feita pela função:

```
ST_Transform (geometry, SRID sistema de coordenadas final)
```

onde o SRID ou Spatial Reference System Identifier é um número inteiro que define univocamente cada sistema de coordenadas e é padronizado internacionalmente pelo Oil & Gas Producers Surveying & Positioning Committee [39].

No caso particular tratado neste projeto, a informação geográfica do SIGGESC está definida com base no datum 73 enquanto que nas tabelas GTFS são esperadas coordenadas em WGS84.

3.2 Tabela SHAPES

Toda a informação essencial à tabela SHAPES do GTFS está presente na tabela TROCOS da componente geográfica do SIGGESC embora nem toda esteja diretamente acessível. Esta tabela está organizada em termos de percursos entre paragens (troços) e não possui, por isso, um identificador único para cada trajeto (shape).

É, portanto, necessário construir um identificador que seja único e ao mesmo tempo visualmente informativo para o preenchimento do campo `shape_id`.

Uma vez que cada `shape_id` deverá identificar de forma unívoca um determinado trajeto de uma determinada carreira de um determinado operador num determinado sentido, a solução foi exatamente a inclusão de toda esta informação como mostra a figura 3.1.

idoperador-carreira-parcelar-variante-sentido
187-10-1-1-1

Figura 3.1: Exempo de um valor do campo `shape_id`

Como já referido na secção 2.3.2, cada registo da tabela TROCO define um trajeto entre duas paragens. Esse trajeto é constituído por geometrias [46] do tipo LINESTRING ou MULTILINESTRING. Uma `linestring` representa um conjunto de pontos e os segmentos de reta que os unem. Uma `multinestring` representa um conjunto de várias linestrings como mostram a figura 3.2 e os exemplos abaixo.

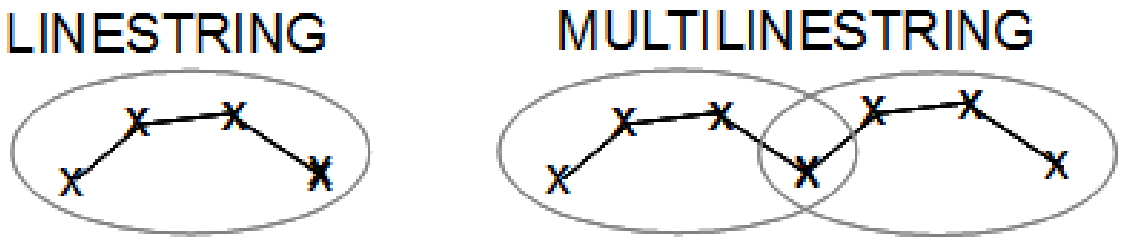

Figura 3.2: Exempo do que é uma LINESTRING e MUTILINESTRING

```
LINESTRING
((-8.5931332227459 41.2317127457025, -8.59402749239055 41.2322850754101))

MULTILINESTRING
((-8.5931332227459 41.2317127457025, -8.59402749239055 41.2322850754101),
(-8.59305322512081 41.2316627475326, -8.5931332227459 41.2317127457025),
(-8.59278769301871 41.231481563882, -8.59305322512081 41.2316627475326
```

Ao contrário do SIGGESC onde cada registo da tabela TROCOS representa um segmento, na tabela SHAPES do GTFS cada registo representa um ponto. É portanto necessário extrair as coordenadas dos pontos constituintes de cada `(multi)linestring`.

Relembro que cada `(multi)linestring` possui vários pontos pelo que o número de registos da tabela shapes é bastante superior ao número de registos da tabela TROCOS da qual provêm.

A obtenção dos valores dos campos da tabela SHAPES resulta da relação esquematizada na figura 3.3.

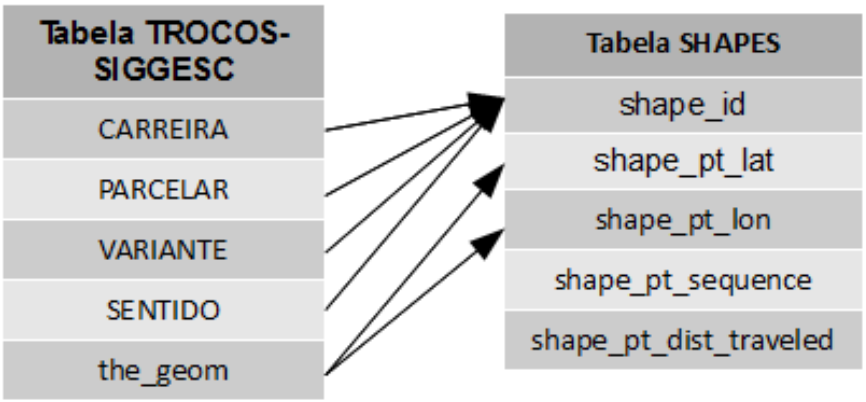

Figura 3.3: Representação esquemática da transformação para a tabela SHAPES

Para preenchermos a tabela SHAPES, inicialmente criámos uma função que para cada registo da tabela TROCOS extraía os pontos do campo `the_geom` e por cada ponto inseria um registo na tabela SHAPES. A função foi feita com sucesso, mas os testes revelaram um problema que ilustramos na figura 3.4.

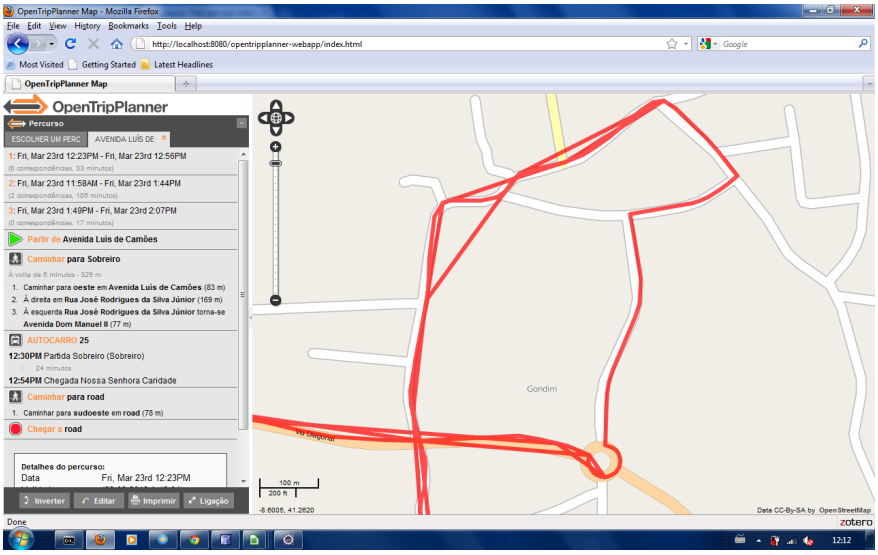

Figura 3.4: Problema encontrado na tabela SHAPES

Analisámos os dados da tabela SHAPES e descobrimos problemas com os dados. A ordem na qual se encontravam os pontos que constituiam as `multilinestrings`, nem sempre era a ordem em que esses pontos estavam quando projetados num mapa. As figuras 3.5, 3.6 e 3.7 pretendem ilustrar as situações encontradas. Recordo que cada `multilinestring` representa um trajeto entre duas paragens e é composta por mais do que uma `linestring` cada uma representando um sub-trajeto. Para que o trajeto seja

representado de forma contínua existem, por isso, pontos comuns entre as diferentes `linestrings`.

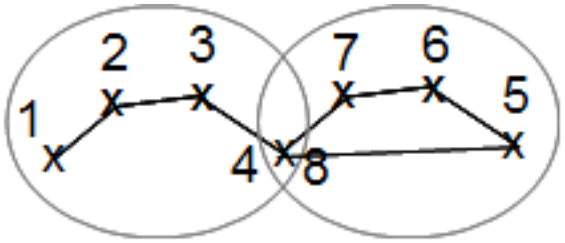

Figura 3.5: Problema um encontrado na tabela SHAPES

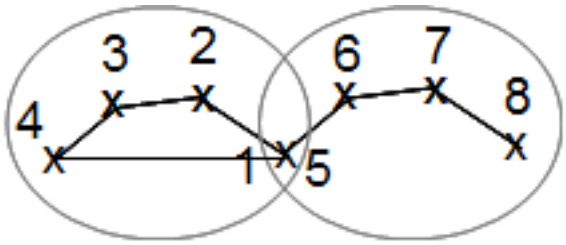

Figura 3.6: Problema dois encontrado na tabela SHAPES

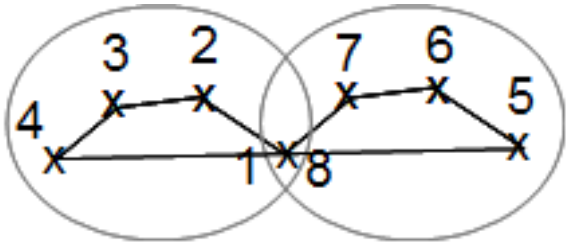

Figura 3.7: Problema três encontrado na tabela SHAPES

Os números nas figuras 3.5, 3.6 e 3.7 indicam a ordem encontrada para os pontos de duas `linestrings` nos dados originais enquanto que a sua posição relativa pretende representar a posição relativa num mapa. Por outras palavras, os pontos 4 e 8 na figura 3.5 possuem as mesmas coordenadas geográficas embora não apareçam consecutivamente na tabela.

O OpenTripPlanner traça no mapa uma linha que liga os pontos seguindo a ordem que está na tabela SHAPES e não a sua posição geográfica. Explicitamente, isto significa que na figura 3.5, o OpenTripPlanner traça uma linha do ponto 4 ao ponto 5, quando o deveria fazer na ordem 4, 8, 7, 6 e 5. O resultado deste problema no mapeamento

de trajetos no mapa com dados reais é ilustrado na figura 3.4. As figuras 3.5, 3.6 e 3.7 representam os três casos possíveis, o primeiro quando duas `linestrings` consecutivas estão desordenadas (figura 3.7) e os outros dois quando o problema se restringe apenas a uma de duas `linestrings` consecutivas (figuras 3.5 e 3.6)

Para resolver estes problemas criámos uma função(ver código B.1 passo 2) que vai percorrer todos os registos/troços da tabela. Quando se verifica que o valor do campo `ordemtro` é igual a um, é atribuído à variável **valor**, o valor de zero. A variável é importante pois vais guardar a posição/sequência do ponto na shape.

O PostGIS inclui um conjunto de funções que simplificam a manipulação de dados geográficos. A função `ST_NumGeometries`, quando aplicada a uma `multilinestring` fornece o número de `linestrings` que a constituem. Cada `linestring` é depois percorrida para obtenção dos pontos nelas contidos. As funções `ST_StartPoint`, `ST_EndPoint` e `ST_NumPoints` aplicadas a `linestrings` permitem obter o ponto inicial, final e o número de pontos total respetivamente.

Para cada `multilinestring`, o primeiro ponto da primeira `linestring` (ver código B.1 passo 2.4) é, desta forma, comparado com o primeiro e o último ponto da segunda `linestring`. Uma vez que existem pontos comuns entre `linestrings` consecutivas, desta comparação sairá a ordem correta para os pontos quer mantendo a ordem original, quer invertendo uma das `linestrings`.

Este procedimento garante que a primeira e segundas `linestrings` se encontram agora ordenadas corretamente, pelo que os pontos da terceira `linestring` são comparados com os da segunda e ordenados se necessário. O procedimento repete-se para as restantes `linestrings`.

Para o caso em que o primeiro registo/troço de uma shape seja `linestring`, os pontos são gravados na tabela pela ordem da `linestring`(ver código B.1 passo 2.7).

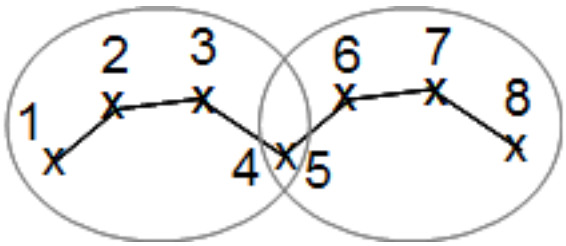

Figura 3.8: Shapes com pontos na ordem correta

A função preencheu todos os campos da tabela SHAPE exceto o campo `distance`.

Para preencher o campo `distance` criámos uma função (ver código B.1 passo 3) que percorre todos os registos da tabela e verifica(ver código B.1 passo 3.1) se o valor do campo `sequence` é igual um. Isto indica que iniciou uma shape e então coloca no campo `distance`, o valor zero. Se o valor do campo `sequence` for superior a um (ver código B.1 passo 3.2), é calculado o valor da distância do ponto atual somando a distância já anteriormente calculado do ponto anterior com resultado do valor a função `ST_distance_sphere` que calcula a distância entre o ponto anterior e o ponto atual.

3.3 Tabela AGENCY

Para criar a tabela AGENCY do GTFS precisámos da tabela OPERADORES da base de dados SIGGESC. Criámos uma consulta de seleção com os campos da tabela OPERADORES que preenche os campos da tabela AGENCY como mostra a figura 3.9.

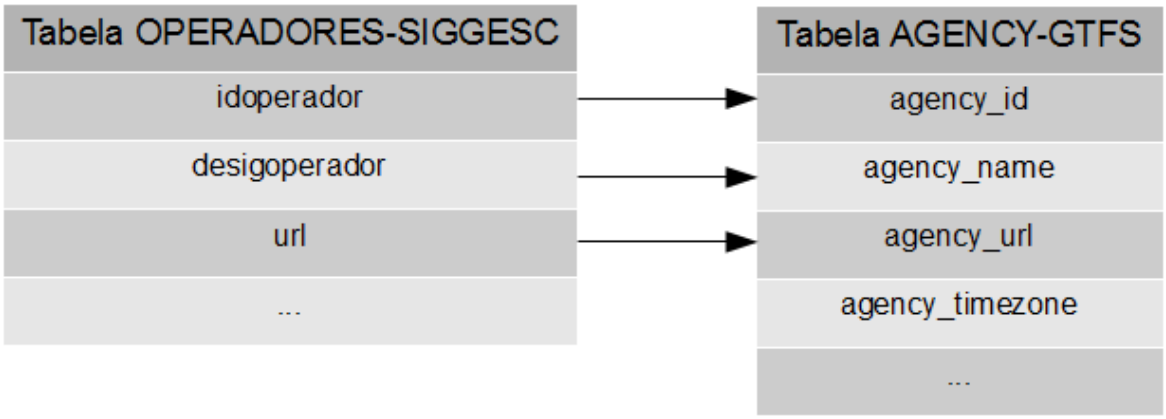

Figura 3.9: Representação esquemática da transformação para a tabela AGENCY

A base de dados SIGGESC, por ser focada apenas em operadores portugueses, não inclui um campo equivalente ao `agency_timezone` da tabela AGENCY. Este campo foi preenchido com o código Europe/Lisbon que define o fuso horário de Portugal continental. Também os campos `agency_lang`, `agency_phone` e `agency_fare_url` não existem atualmente na base de dados SIGGESC e para este caso, foram preenchidos manualmente.

Para futuro, propusémos que a informação em falta conste de uma base de dados de operadores com informação complementar ao SIGGESC.

3.3.1 Carateres proibidos

As tabelas criadas no formato GTFS serão posteriormente exportadas como ficheiros CSV para alimentarem o calculador de itinerários. Para manter a consistência da informação, é essencial garantir que todas as virgulas presentes nos campos de texto são substituídas por outro carater. Assim, optámos por substituir ","(virgula) por " " (espaço) utilizando a função `regexp_replace( texto, texto_a_substituir, novo_texto)`. Por exemplo, no caso da tabela AGENCY o nome "António da Silva Cruz & Filhos, Ldª."foi transformado em "António da Silva Cruz & Filhos Ldª."

3.4 Tabela STOPS

A tabela PARAGENS que resulta da *shapefile* com o mesmo nome contém toda a informação essencial para a construção da tabela STOPS do GTFS. É apenas necessário obter a latitude e longitude da informação geográfica codificada no campo `the_geom` através das funções `ST_Y()` e `ST_X()` respetivamente (figura 3.10).

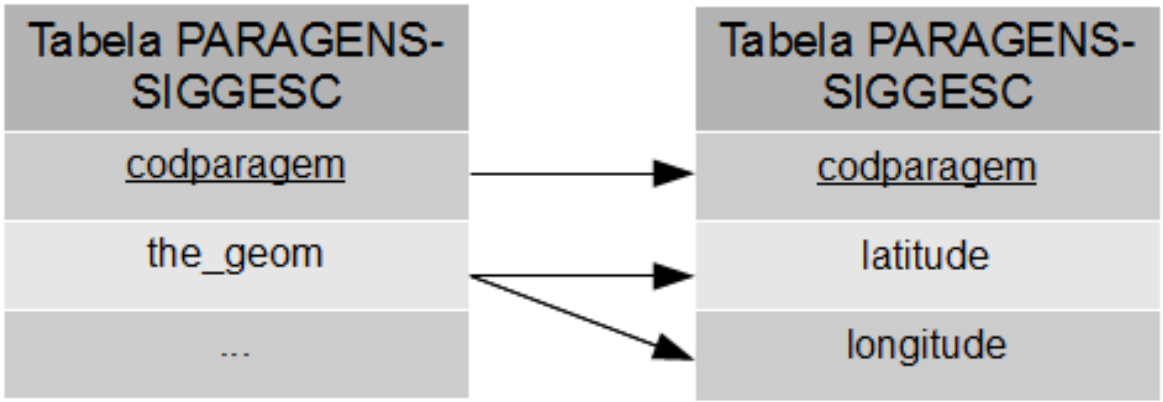

Figura 3.10: Representação esquemática da alteração efetuada na tabela PARAGENS

As tabelas necessárias para criar a tabela STOPS foram as tabelas PARAGENS e PARAGEM estando relacionadas pelos campos `codparagem` e `idparagem` pois contêm o mesmo tipo de código que identifica a paragem. Assim, criámos uma consulta de seleção (ver código B.1 passo 4) com os campos das duas tabelas e preenchemos os campos da tabela STOPS como mostra a figura 3.11.

O campo `stop_code` é preenchido com um código que identifica a paragem de forma única para os passageiros. Embora este campo seja opcional, optámos por incluí-lo,

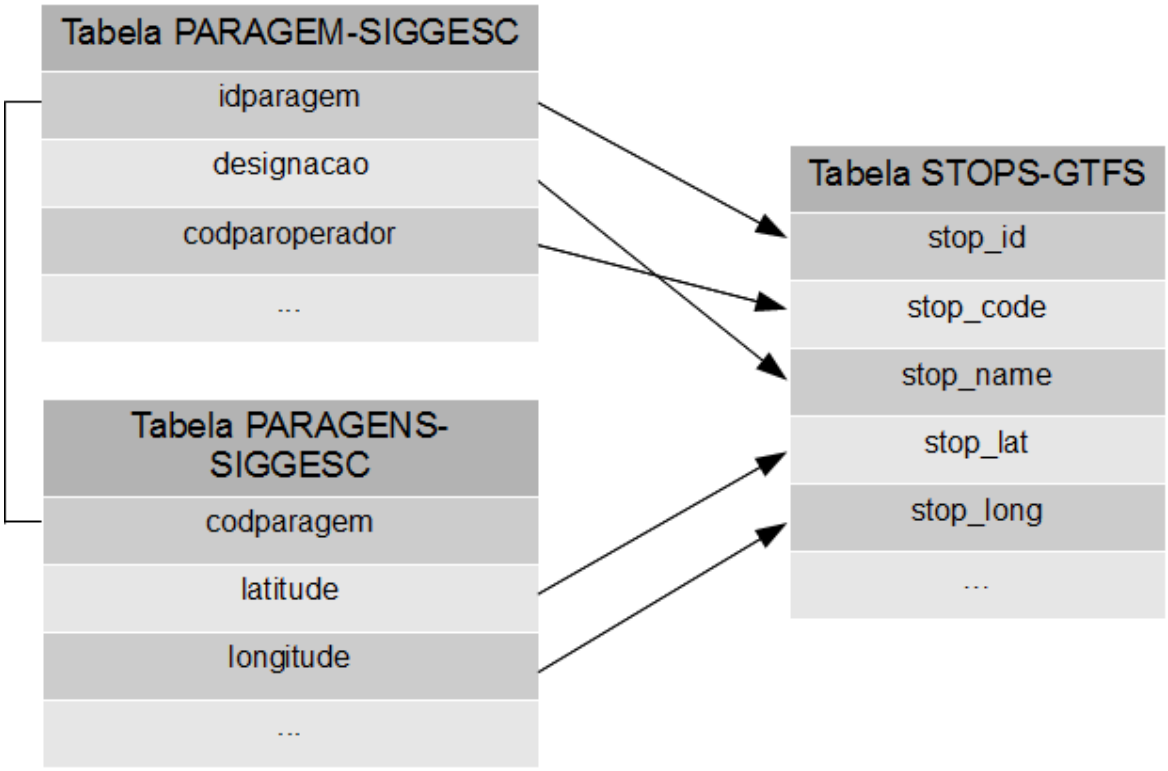

Figura 3.11: Representação esquemática da transformação para a tabela STOPS

desde já, na tabela STOPS pela relevância que tem para os utilizadores. Este código é utilizado por operadores como os STCP, a Carris e outros, para disponibilizarem os tempos de espera dos veículo através de SMS. Na base de dados SIGGESC este corresponde ao campo `codparoperador` da tabela PARAGEM.

3.5 Tabela ROUTES

A tabela ROUTES resulta da combinação das tabelas CENTROOPERACIONAL e CARREIRAS da base de dados SIGGESC relacionadas pelos campos `idcentroop` existentes nas duas tabelas. Para preencher os campos das tabelas ROUTES como mostra a figura 3.12, fizémos uma consulta de seleção com os campos das tabelas CENTROOPERACIONAL e CARREIRAS.

Todos os operadores na base de dados do SIGGESC são operadores rodoviários pelo que o campo `route_type` que define o tipo de transporte utilizado foi preenchido por defeito com o valor "3"que corresponde a autocarros.

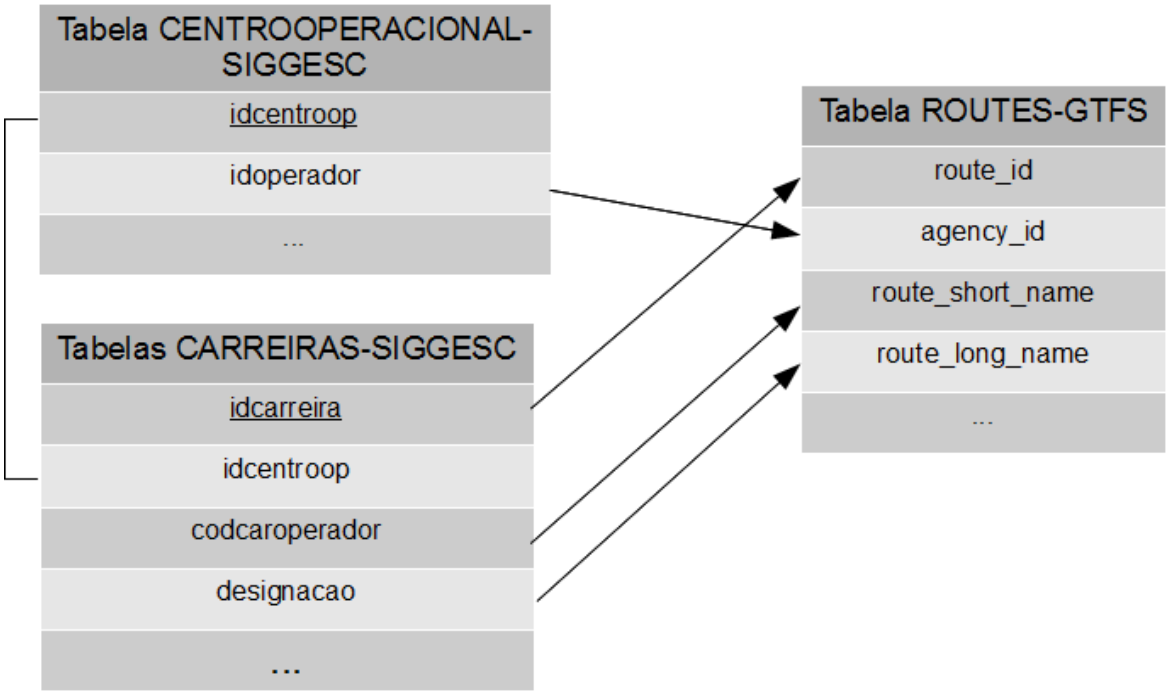

Figura 3.12: Representação esquemática da transformação para a tabela ROUTES

3.6 Tabela CALENDAR

Para criar a tabela CALENDAR do GTFS precisámos das tabelas FREQUENCIAS e PERIODOSANUAIS da base de dados do SIGGESC. As duas tabelas estão relacionadas pelo campo `idfrequencia`. Não existe, no entanto nenhum campo diretamente equivalente ao campo `service_id` da tabela CALENDAR.

O campo `service_id` é a chave primária da tabela CALENDAR e deveria identificar inequivocamente o período e a frequência do serviço, que atualmente se encontram em tabelas diferentes. Para preservar a identificação existente no SIGGESC, ao mesmo tempo que assegura a unicidade do registo, propus que o campo `service_id` resulte da união dos campos `idperíodo` e `idfrequencia` da forma exemplificada na tabela 3.1. Uma vez definida a construção do `service_id`, criámos uma consulta de seleção com os campos das tabelas PERIODOSANUAIS e FREQUENCIAS e preenchemos os campos da tabela CALENDAR como mostra a figura 3.13.

Tabela 3.1: Tabela demonstrativa do formato do campo `service_id` na tabela CALENDAR

idperiodo	idfrequencia	service_id
1	34	134
2	34	234
3	35	235

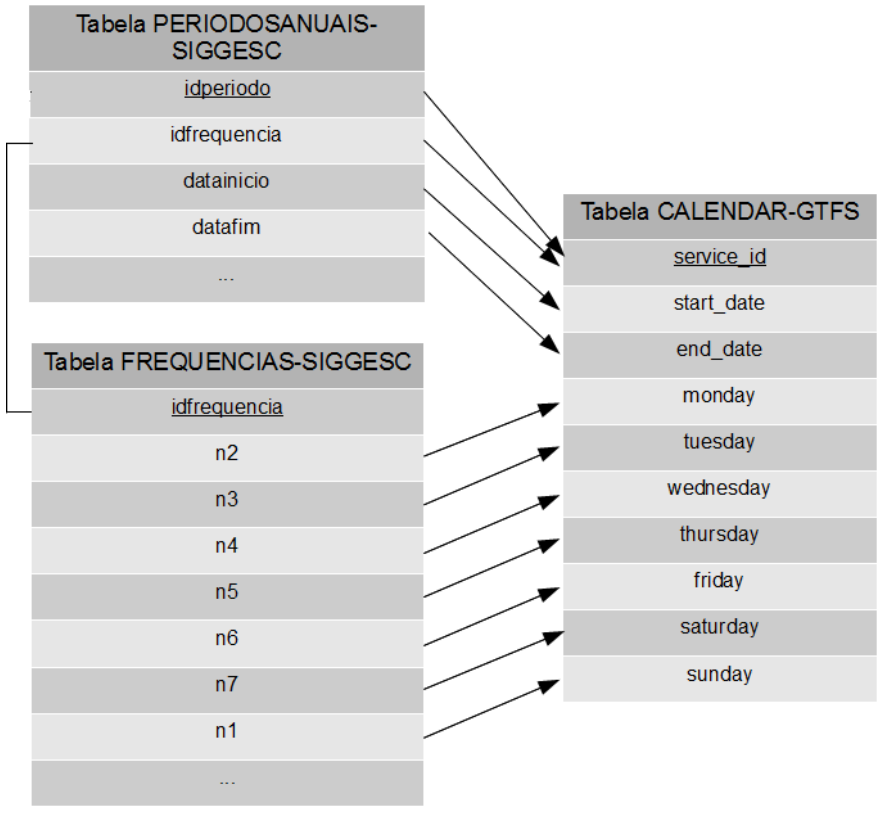

Figura 3.13: Representação esquemática da transformação para a tabela CALENDAR

3.7 Tabela TRIPS

A tabela TRIPS no GTFS identifica todas as viagens realizadas por todas as carreiras do operador. Uma viagem é definida como uma sequência de paragens percorridas num determinado sentido e num horário pré-determinado. Assim, por exemplo:

- Uma viagem é o percurso do autocarro da carreira "X"desde o início na Avenida dos Aliados às 16:00h até ao término no Mercado de Matosinhos às 16:45h;
- Uma viagem diferente da anterior é o percurso do autocarro da mesma carreira "X"desde o início na Avenida dos Aliados às 17:00h até ao término no Mercado de Matosinhos às 17:45h;
- Uma outra viagem diferente das duas anteriores é o percurso do autocarro da carreira "X"desde o início no Mercado de Matosinhos às 16:50h até ao término na Avenida dos Aliados às 17:35h.

A tabela TRIPS resulta da combinação das tabelas CENTROOPERACIONAL, CARREIRAS e CARREIRACIRCULACAOFREQUENCIA da base de dados SIGGESC e tabela CALENDAR do GTFS gerada anteriormente. As tabelas CARREIRACIRCULACAOFREQUENCIA e CARREIRAS estão relacionadas pelos campos `idcarreira`, as tabelas CARREIRACIRCULACAOFREQUENCIA e CALENDAR estão relacionadas pelos campos `idfrequencia` e as tabelas CARREIRAS e CENTROOPERACIONAL estão relacionadas pelos campos `idcentroop`.

Novamente surgiu aqui o problema de construção do campo `trip_id` (a chave primária da tabela TRIPS) procurando preservar os códigos de identificação das tabelas originais do SIGGESC. Este campo deveria permitir identificar:

- o operador a que pertence a carreira - `idoperador`;
- a carreira a que pertence a viagem - `idcarreira`;
- o sentido da viagem - `sentidocirculacao`;
- a hora a que se inicia a viagem - `idcirculação`;
- o período do ano e os dias da semana em que a viagem está disponível - `service_id`.

idoperador-idcarreira-sentidocirculacao-idfrequencia-idcirculacao-service_id
187-2820-1-35-0900-235

Figura 3.14: Exemplo de um valor de `trip_id`

A solução encontrada foi combinar todos os códigos de identificação anteriores num único código como mostra a figura 3.14.

Como o tamanho de alguns dos códigos constituintes do campo `trip_id` varia (por exemplo por serem números sequenciais), não é possível extrair um componente do código final utilizando apenas a sua posição e o número de carateres constituintes. Por essa razão, optei por separar cada elemento pelo carater "-" facilitando, no futuro, a utilização das funções de reconhecimento de padrões do PostgreSQL [25].

Encontrada a solução para o campo `trip_id` o próximo passo foi preencher a tabela. Foi necessário criar duas consultas de seleção, uma para cada sentido da viagem. A razão para esta divisão está no facto de o campo `trip_headsign` ser preenchido com o destino de cada viagem que depende do sentido em que é feita essa viagem. Assim, quando a viagem é efetuada no sentido origem-destino, o campo `trip_headsign` é preenchido com o conteúdo do campo `destino`, quando a viagem é efetuado no sentido contrário o campo `trip_headsign` é preenchido com o conteúdo do campo `origem`. Os restantes campos da tabela TRIPS foram obtidos através da relação esquematizada na figura 3.15.

A tabela TRIPS tem o campo `shape_id` como opcional. No entanto, e dado que a tabela SHAPES, (opcional na base de dados) já existe, o campo `shape_id` pode ser incluído na tabela TRIPS. Recordo que a tabela SHAPES cuja chave primária é o campo `shape_id` teve origem na shapefile Trocos.shp pertencendo à componente geográfica do SIGGESC. A introdução do campo `shape_id` na tabela TRIPS requer por isso um relacionamento entre esta componente e a componente alfanumérica e permite, em termos práticos, relacionar cada viagem com o percurso efetuado pelo veículo no mapa. Verifiquei, contudo, que com a informação que tenho disponível, não existe qualquer relacionamento entre essas duas componentes. Para resolver este problema foi preciso analisar os dados das tabelas das bases de dados e a solução passou por criar uma tabela e preencher os seus campos manualmente. A tabela a que eu dei o nome de AUXILIAR5, cria um relacionamento entre a componente alfanumérica e a componente geográfica da base de dados e consequentemente entre a tabela TRIPS e SHAPES e possui a estrutura como mostra a figura 3.16.

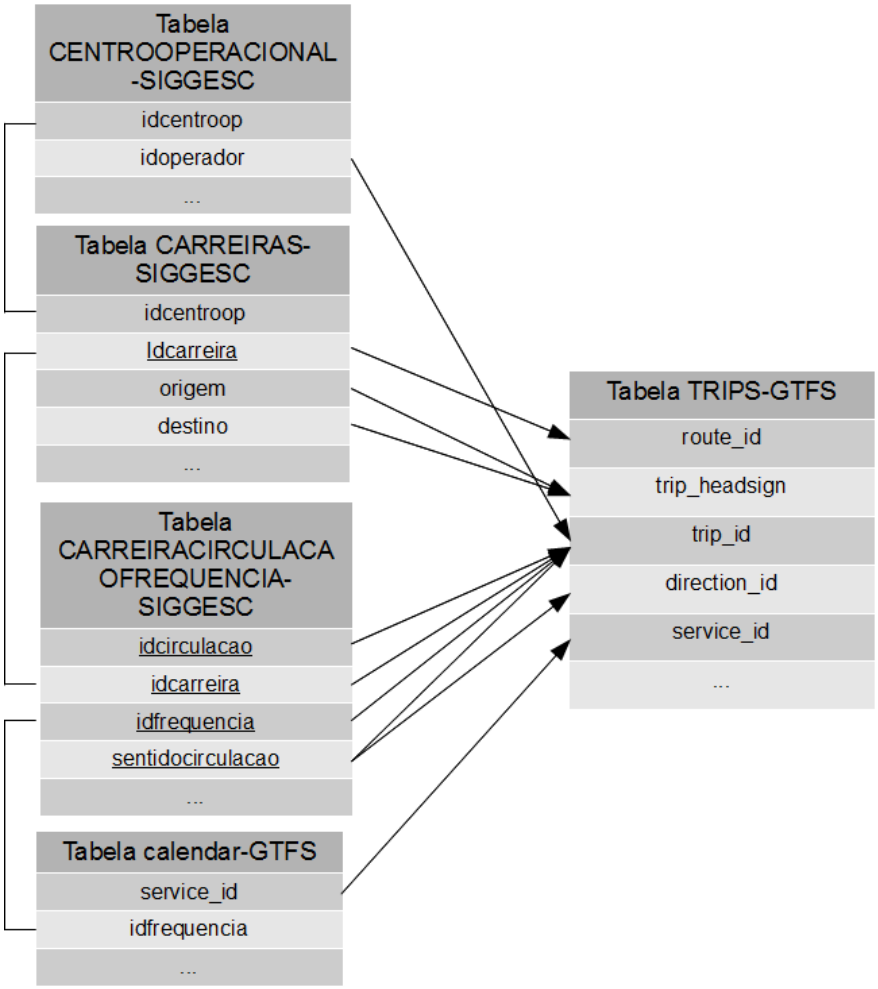

Figura 3.15: Representação esquemática da transformação para a tabela TRIPS

AUXILIAR5
idoperador*
idcarreira*
carreiranome
parcelar
variante
sentido*

Figura 3.16: Tabela AUXILIAR5

A construção da tabela AUXILIAR5 é apenas possível para um operador de pequenas dimensões como aquele cujos dados são utilizados neste relatório e que conta com cerca de 15 carreiras. Para grandes operadores de transportes públicos esta abordagem é inviável.

Os dados obtidos têm como origem um operador e não diretamente o IMTT. Por essa razão, é expetável que o relacionamento entre a parte alfanumérica e a parte geográfica exista internamente no SIGGESC mas que os operadores apenas tenham acesso aos dois componentes de forma desconexa. Se for este o caso, a formação da tabela AUXILIAR5 não será necessária no futuro. Se não existir, de facto qualquer relacionamento entre a componente alfanumérica e a componente geográfica a abordagem que proponho é de, juntamente com cada operador, ser desenvolvida uma tabela global com o formato da tabela AUXILIAR5.

3.8 Tabela STOP_TIMES

A tabela STOP_TIMES possui informação do tempo de chegada e partida das paragens de todas as viagens que se encontram na tabela TRIPS. É por isso a tabela do GTFS

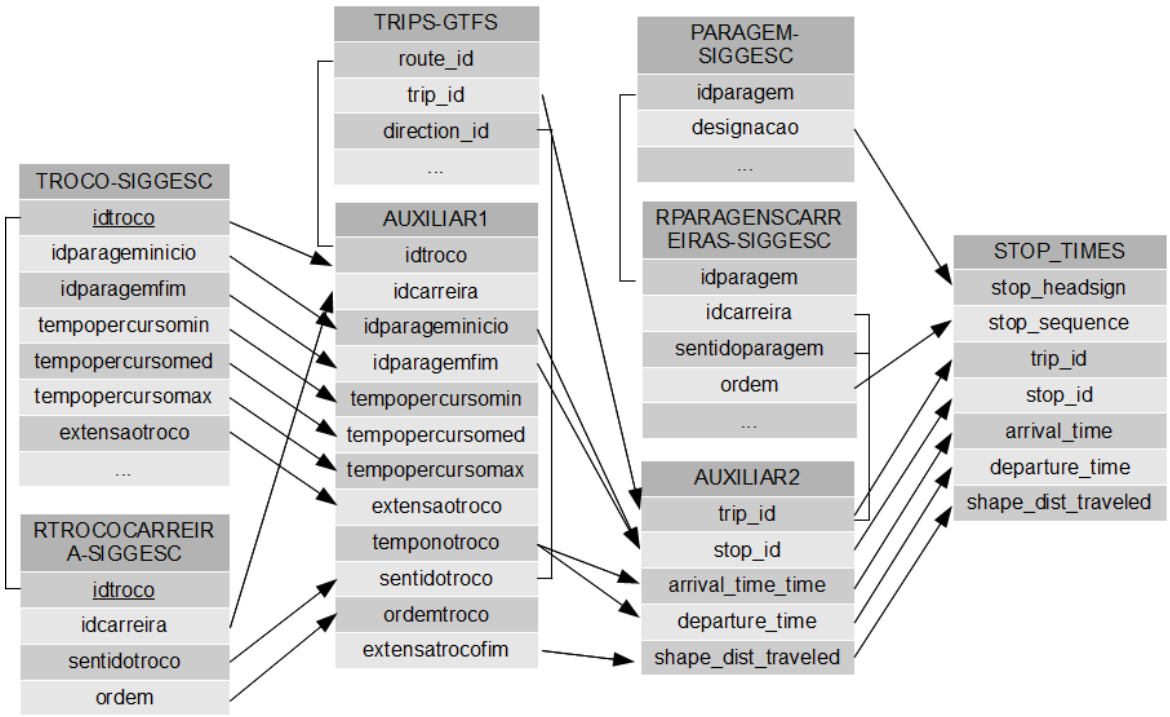

Figura 3.17: Representação esquemática da transformação para a tabela STOP_TIMES

com o maior número de registos.

Uma observação das tabelas do SIGGESC mostra que não existe nenhuma que, de forma direta (ou quase direta) dê origem à tabela STOP_TIMES. A construção desta tabela é ainda dificultada por dois outros motivos:

- A contagem do tempo no GTFS tem regras particulares para as viagens que ultrapassam as 24 horas de um dia;
- Ao contrário do que acontece no GTFS, a base de dados SIGGESC armazena apenas os horários de cada viagem na primeira paragem e os tempos (mínimo, médio e máximo) necessários para percorrer a distância entre duas paragens.

Nas próximas secções é analisado cada uma destas situações individualmente.

3.8.1 Contagem do tempo no GTFS

Em toda a base de dados GTFS os campos que se referem a tempo devem ter cinco ou seis digitos, agrupados dois a dois e separados pelo carater ":", isto é, HH:mm:ss (H:mm:ss é aceite no caso de a hora começar por 0). As horas devem ser fornecidas no formato 24h.

No caso de viagens que ultrapassem o dia em que se iniciaram, o horário deve ser inserido com valores superiores a 24:00:00. Exemplo: Supondo que para a viagem V o veículo chega à paragem 1234 às 23:45:00 do dia D e à paragem 5678 à 01:05:00 do dia D+1. Para a paragem 5678, o horário a colocar na tabela STOP_TIMES é 25:05:00. A tabela 3.2 resume esta situação.

Tabela 3.2: Tabela explicativa do formato da hora utilizado na tabela STOP_TIMES

Horário	No GTFS
8 da manhã	08:00:00
8 da tarde	20:00:00
01:05:00	01:05:00 25:05:00 - se corresponder a uma viagem do dia D que se prolonga para o dia D+1

Combinar estas regras, o facto de o PostgreSQL não permitir (em formato hora) valores superiores a 23:59:59 e a necessidade de fazer cálculos com valores de tempo para determinar o horário em cada paragem(ver secção 3.8.2) justifica a não utilização do formato hora. Optámos, por isso, por converter os tempos e efetuar todos os cálculos auxiliares em segundos, armazenando o resultado (convertido para horas, minutos e segundos) como `varchar`.

3.8.2 Cálculo do horário de chegada/partida em cada paragem

A base de dados SIGGESC armazena apenas os horários de cada viagem na primeira paragem e os tempos (mínimo, médio e máximo) necessários para percorrer a distância entre duas paragens. O GTFS, por outro lado, requer que em cada paragem os horários de chegada e de partida do veículo sejam preenchidos explicitamente (campos `arrival_time` e `departure_time` da tabela STOP_TIMES) e de acordo com as regras definidas para os campos de tempo (ver secção 3.8.1). O procedimento para obtenção desses valores é analisado nesta secção.

Em geral, o tempo de permanência de um veículo numa paragem é demasiado curto para ser significativo e é por isso incluído no tempo total do percurso entre paragens. Este é o caso também no SIGGESC pelo que ao preencher a tabela STOP_TIMES do GTFS, os campos `arrival_time` e `departure_time` são iguais para a mesma paragem.

Como não existe informação sobre as alturas do dia em que para o tempo de percurso entre paragens deve ser utilizado o tempo mínimo ou o tempo máximo, o tempo médio é a estimativa mais fiável para ser utilizada ao longo de todo o dia. Alguns registos no SIGGESC, no entanto, não possuíam tempos médios atribuídos, de forma que, para garantir a consistência dos resultados, atualizei na tabela TROCOS, estes casos com a média aritmética entre os tempos mínimos e máximos.

A hora de partida de cada viagem é dada pelo campo `idcirculação` da tabela CARREIRACIRCULACAOFREQUENCIA que é também um dos constituintes do campo `trip_id` da tabela TRIPS (ver secção 3.7). Assim, e de uma forma sumária, para todas as viagens na tabela TRIPS, a função (ver código B.1 passo 13) vai somar o valor da hora de início (`idcirculação` de `trip_id`) ao tempo médio necessário para percorrer cada troço dessa viagem (`temponotroço` da tabela AUXILIAR1) para obter a hora de chegada. O resultado será posteriormente utilizado para preencher os campos `arrival_time` e `departure_time`.

A mesma função calcula ainda a distância percorrida desde o início da viagem até cada paragem uma vez que tal como com os tempos as distâncias percorridas para cada troço encontram-se armazenadas na tabela TROCO do SIGGESC.

Todos estes resultados são armazenado na tabela temporária AUXILIAR2 que resulta da seleção de campos das tabelas TRIPS e AUXILIAR1. Esta última é uma tabela temporária com informação das paragens que cada carreira possui, bem como o tempo e a distância a que cada paragem se encontra da paragem inicial(ver código B.1, passo 9). Para exemplo, alguns registos das tabelas AUXILIAR1 e AUXILIAR2 estão representados na figura 3.18, como também o problema existente com o qual solucionei criando uma função(ver código B.1 passo 14).

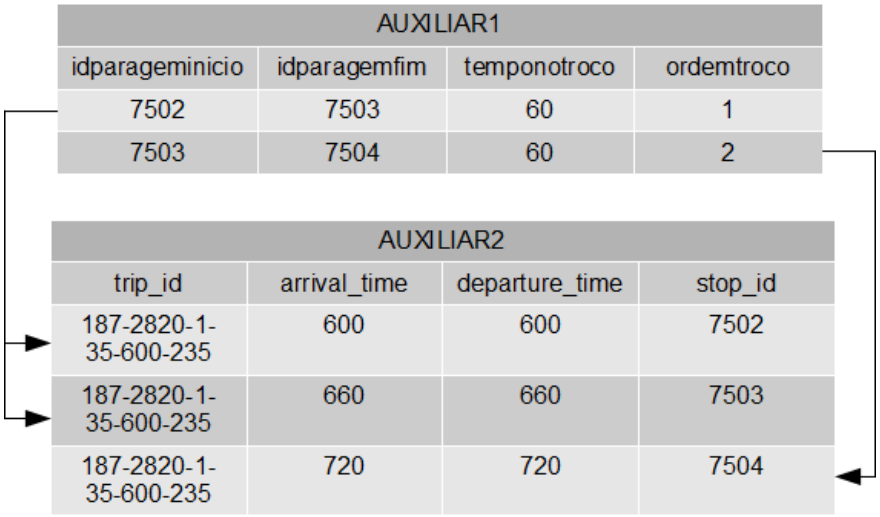

Figura 3.18: Imagem representativa dos passos 14.1 e 14.2 do código B.1

Depois de preenchida a tabela STOP_TIMES verifiquei que havia repetição em registos de algumas paragens. Isto acontece por erro na digitação dos dados ou possivelmente em determinadas carreiras onde os autocarros param numa paragem por um tempo prolongado. Quando isso acontece, a tabela contém para a mesma paragem dois valores para `idparagens` iguais.

Enquanto não existe confirmação por parte dos operadores da necessidade de manter esta situação, criámos uma função (ver código B.1 passo 16) que percorre os registos da tabela e quando encontra dois registos que possuem tempos de chegada e tempos de partida iguais e elimina um registo. Para os objetivos deste estágio, esta solução resolve o problema, no entanto propomos que no futuro, e uma vez que o GTFS suporta horas de chegada e partida para cada paragem, seja analisada com os operadores uma solução a longo prazo.

3.9 Testes

Como já referimos na secção Tabela SHAPES 3.2 foram efetuados testes para ver se a base de dados no formato GTFS estava correta. Os critérios usados foram:

- verificação do percurso calculado traçado no mapa
- a descrição detalhada do percurso calculado corresponde ao percurso traçado no mapa

Depois de efetuado o cálculo do itinerário, o primeiro teste efetuado, é um teste visual 3.19 para verificar se o percurso traçado no mapa passava por locais que não eram estradas, ruas ou por cima de edifícios como acontece na figura 3.4.

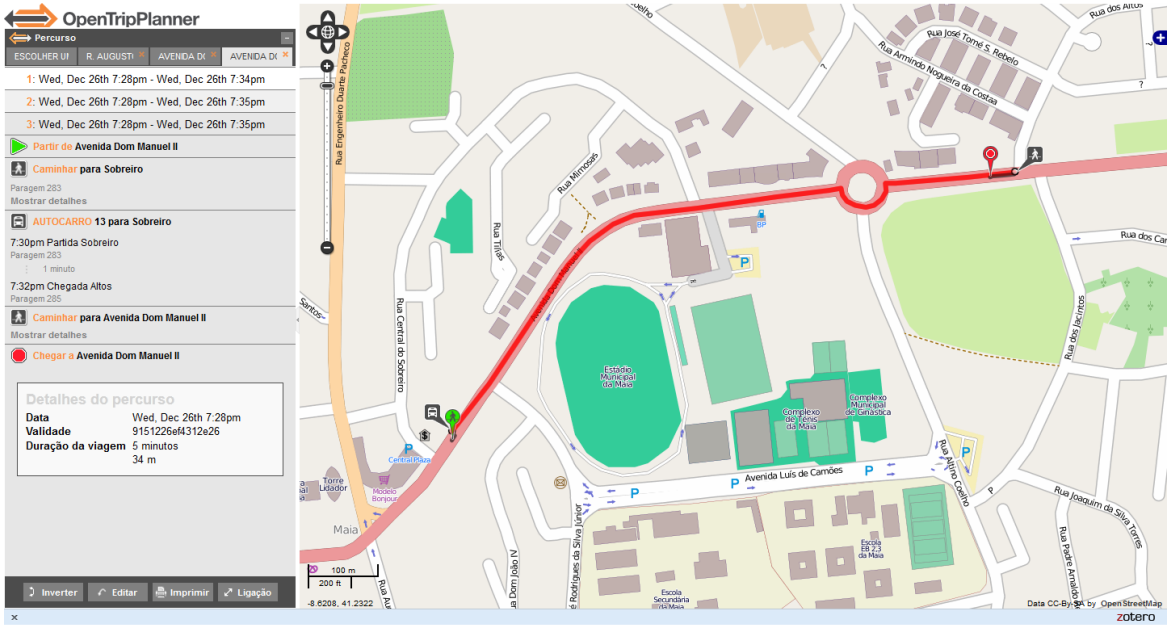

Figura 3.19: Imagem de percurso calculado no OpenTripPlanner

O segundo teste incide sobre os dados retirados da descrição detalhada de um percurso. Assim com base nos números dos autocarros (rotas) e os códigos das paragens de inicio e fim do percurso como mostra a figura 3.19, retirámos os pontos de coordenadas que definem o percurso entre duas as duas paragens e marcámos no Google Maps como mostra a figura 3.20.

Este teste permitiu-nos comparar o percurso traçado no mapa com o percurso traçado no Google Maps. Assim verificámos se a descrição detalhada do percurso corresponde

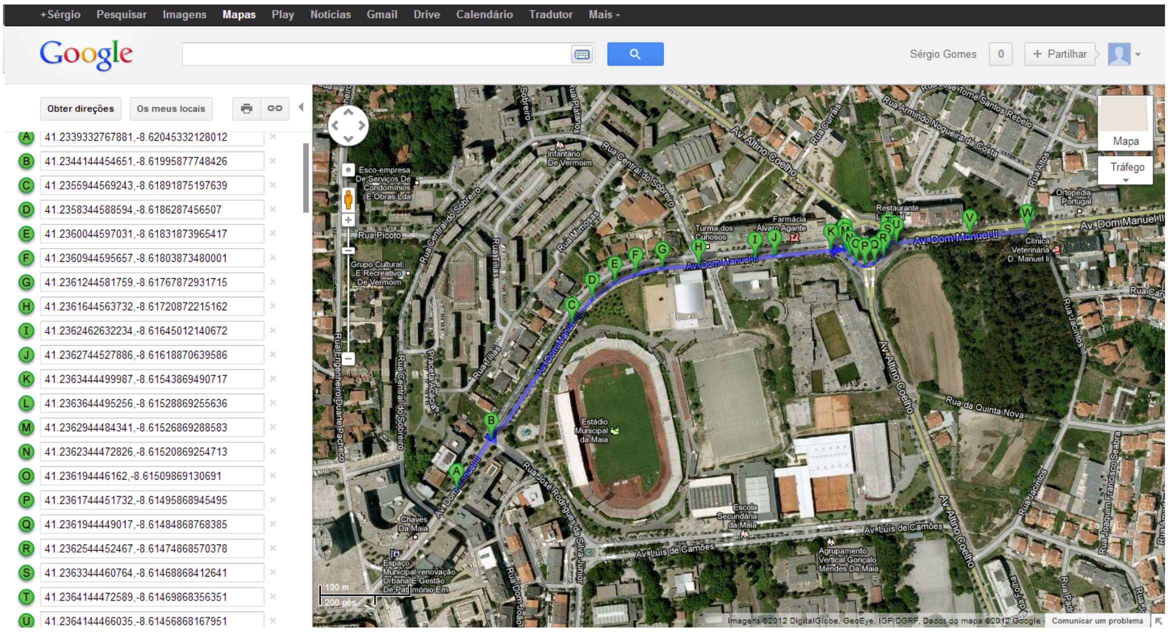

Figura 3.20: Imagem do percurso traçado no Google Maps usando os dados da base de dados GTFS

ao percurso traçado no mapa. Também com base neste teste pudemos encontrar e analisar os problemas descritos na secção Tabela SHAPES 3.2, caso estes existam. Para garantir que o maior número de dados estão corretos, foram efetuados testes a um grande número de cálculos de itinerários, de maneira que todos os número de autocarros (rotas) em diferentes horas do dia fossem testados.

Capítulo 4

Integração da interface do OpenTripPlanner para utilização num portal *web*

O OpenTripPlanner foi desenvolvido para funcionar como uma aplicação independente para cálculo de itinerários por transportes públicos e modos suaves daí que o interface do utilizador na versão standard (figura 4.1) esteja desenhado para ocupar toda a janela do navegador *web*. Para os objetivos do Teco Planner é importante que

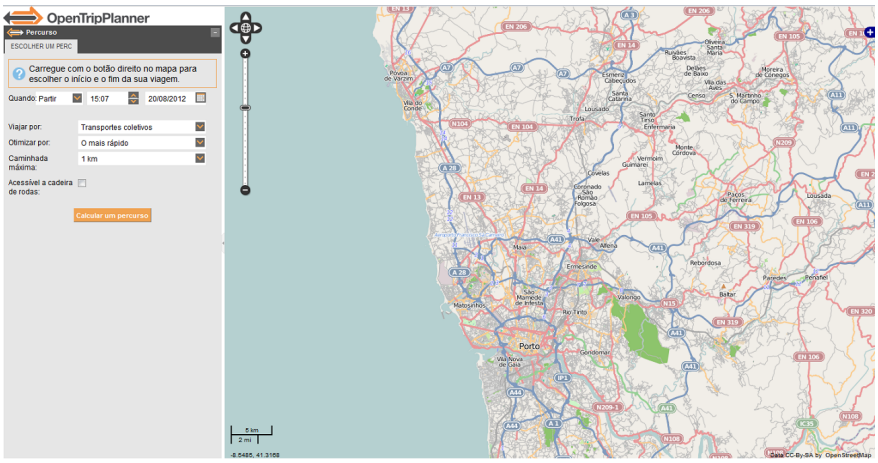

Figura 4.1: Interface do OpenTripPlanner

outra informação (por exemplo, sobre passes, sobre alterações de horários etc.) seja disponibilizada aos utilizadores. É também importante que os utilizadores possam eles próprios introduzir informação relevante (sugestões de viagem ou de funcionalidades,

por exemplo) e interagir entre utilizadores e com o Teco Planner numa filosofia Web2.0 daí que a interface do utilizador no OpenTripPlanner necessite de ser modificado. Existem três soluções genéricas de como isso pode ser feito:

1. Construir um página *web* de raiz;
2. Utilizar um sistema de gestão de conteúdos *web* (Web Content Management System);
3. Utilizar um portal corporativo (Enterprise Information Portal);

A solução a encontrar deveria permitir a incorporação do OpenTripPlanner, a inclusão de informação extra e a interação dos utilizadores com essa informação. Deverá ainda ser suficientemente flexível para permitir atualizações rápidas mesmo por gestores de conteúdos com conhecimentos de informática básicos.

O portal corporativo combina muitas das caraterísticas da criação de páginas de raiz e dos sistemas de gestão de conteúdos com algumas vantagens adicionais como a maior rapidez de criação, uma vez que as funcionalidades mais utilizadas (redes sociais, blogs, etc) já estão disponíveis.

4.1 Liferay

A empresa Gisgeo propôs a utilização do Liferay, considerado o líder nesta área [7]. O Liferay é um portal corporativo código aberto compatível com a maior parte dos sistemas operativos incluindo Windows, Linux e Mac Os e com vários servidores de aplicação e containers de servlets incluindo Weblogic, JBoss e Tomcat.

O Liferay é completamente personalizável e mesmo na versão standard é disponibilizado com um conjunto de portlets para Wikis, blogs e redes sociais. Ao Liferay está ainda associada uma grande comunidade de utilizadores e programadores que continuamente desenvolve novos portlets, corrige erros e contribui para a dinamização de fóruns de suporte.

Atualmente um portal *web* permite com uma alguma facilidade e rapidez a construção de uma página *web* e também criar uma página *web* mais agradável ao utilizador.

A integração do OpenTripPlanner no portal Liferay foi conseguida através da criação de um portlet no Liferay. O portlet [55] é um componente *web* que funciona dentro

de um contentor portlet. O contentor funciona como um quadro onde é gerado dinamicamente conteúdo *web*, independente da estrutura da página *web*.

O portal agregando vários portlets, permite facilmente criar, modificar a estrutura e conteúdo de uma página *web* pois os portlets facilmente podem ser adicionados, modificados ou removidos. Outra vantagem é que uma página *web* não precisa de estar completamente indisponível quando se pretende modificar alguma coisa, pois só é necessário remover os portlets que se pretende alterar, deixando a página *web* disponível com os outros portlets que prestam outros serviços.

Antes de explicar como foi feita a integração do interface do OpenTripPlanner num portlet Liferay, vale a pena analisarmos a estrutura do código de ambos.

4.2 Estrutura do OpenTripPlanner

Os ficheiros e pastas do OpenTripPlanner estão estruturados como mostra a figura 4.2. Na pasta `bin` (executáveis) e na pasta `cache` (dados) está localizada a informação para

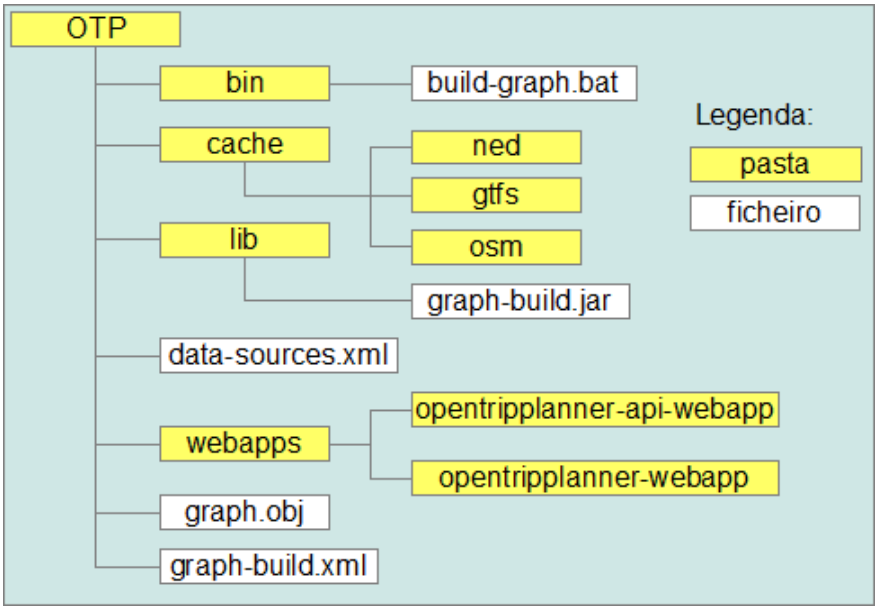

Figura 4.2: Estruturação dos ficheiros e pastas do OpenTripPlanner

construção do grafo de rotas. No interior de cada uma das sub-pastas da pasta chache encontram-se um ou mais ficheiros `.zip` contendo dados de rotas e horários no formato GTFS (pasta `GTFS`), um ficheiro OSM no formato XML com informação geográfica obtida do OpenStreetMap (pasta `OSM`) e se disponível um ficheiro com informação de

altitude (no formato do US National Elevation Dataset - `NED`). Quando disponível, a informação sobre altitude, permite ao OpenTripPlanner sugerir rotas mais suaves, nomeadamente no planeamento de viagens por bicicleta.

O ficheiro `build-graph.bat` da pasta `bin` ativa a geração do grafo de rotas (pelo ficheiro `graph-build.jar`) e a criação do ficheiro `graph.obj`. As instruções (quais os ficheiros GTFS, OSM e NED a utilizar, qual localização do ficheiro `graph.obj`, etc.) para a criação do grafo de rotas são definidas no ficheiro `graph-build.xml`.

O ficheiro `graph.obj` é um ficheiro do grafo construído com os dados da pasta `cache` é gerado na execução do ficheiro `build-graph.bat` que se encontra na pasta `bin`. O `build-graph.bat` executa o ficheiros `graph-builder.jar` e `graph-builder.xml` gerando o grafo.

Os dois outros componentes essenciais ao funcionamento do OpenTripPlanner são as aplicações *web* localizadas na pasta `opentripplanner-api-webapp` e na pasta `opentripplanner-webapp`. A primeira inclui os ficheiros da componente servidor *web* que permite o cálculo dos itinerários e outros serviços disponibilizados no OpenTripPlanner. A segunda possui os ficheiros da componente interface *web* permitindo a visualização e a interação com o navegador *web* do cliente.

Como a pasta `opentripplanner-webapp` possui os ficheiros que controlam a interface do utilizador é necessário conhecermos mais pormenorizadamente o seu conteúdo e a forma como esses ficheiros se interligam. A figura 4.3 ilustra a estrutura desta pasta. A pasta `opentripplanner-webapp` está organizada em três pastas principais: `images` (contendo imagens como por exemplo os ícones dos diferentes modos de transportes e logótipos do OpenTripPlanner), `WEB-INF` (onde se encontram os ficheiros de configuração da aplicação *web*) e `js`. Esta última contém os ficheiros de configuração do OpenTripPlanner e será analisada em detalhe abaixo.

O ficheiro `index.html` é a página *web* padrão que permite carregar os ficheiros javascript, css e iniciar o processo que dá origem à interface do OpenTripPlanner. O ficheiro `planner.css` permite estruturar a interface do OpenTripPlanner e o ficheiro `print.html` cria uma nova página html com as informações de um itinerário para impressão.

A pasta `js` encontra-se dividida em bibliotecas de *software* externo (pasta `lib`) e ficheiros internos(pasta `otp`). A pasta `lib` é constituído pelos ficheiros do *software* ExtJS, Openlayers referidos anteriormente (secção 2.2) e GeoExt [43] que juntamente com o OpenLayers permite a interação com o mapa.

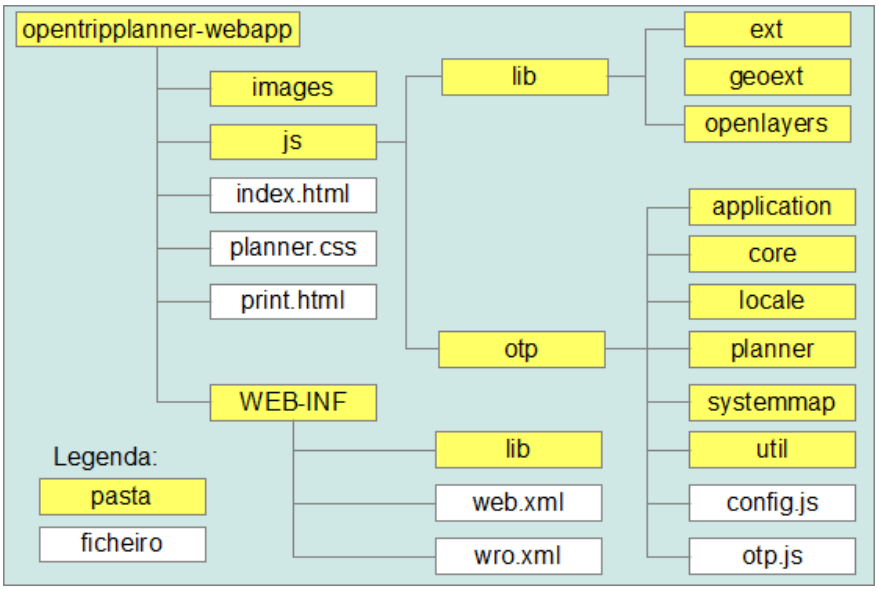

Figura 4.3: Estruturação da pasta `opentripplanner-web-app`

A pasta `otp` contém os ficheiros que formam a interface do OpenTripPlanner, permitindo a sua construção. Para melhor estruturação, os ficheiros encontram-se divididos em pastas consoante as suas funções. O ficheiro `otp.js` juntamente com os ficheiros da pasta `application` permitem iniciar o OpenTripPlanner desencadeando processos necessários para tal, a pasta `core` constrói a interface inicial e a `planner` efetua as alterações para mostrar o cálculo de itinerário no mapa e uma descrição verbal do mesmo, a pasta `locale` possui os ficheiros de tradução do OpenTripPlanner para várias linguas. A pasta `utils` permite definir valores de constantes, formatos de valores como a data e hora, os valores opcionais que aparecem nas caixas da interface do OpenTripPlanner apontar o caminho de imagens. O ficheiro `config.js` permite definir configurações por defeito quando o OpenTripPlanner inicia: a língua, o tipo de mapa, o zoom inicial bem como ativar e desativar funcionalidades.

4.3 Estrutura do Portlet Liferay

O portlet [55] utilizado pelo Liferay, segue o padrão Java Specification Request 286 [51–53]. Este padrão garante a portabilidade e interoperabilidade dos portlets em diferentes plataformas de portais *web*. A estrutura do portlet encontra-se representada na figura 4.4. A estrutura do portlet encontra-se dividida em três componentes: os ficheiros do lado do cliente (ficheiros `css`, `js`, `jsp` e imagens), os ficheiros de

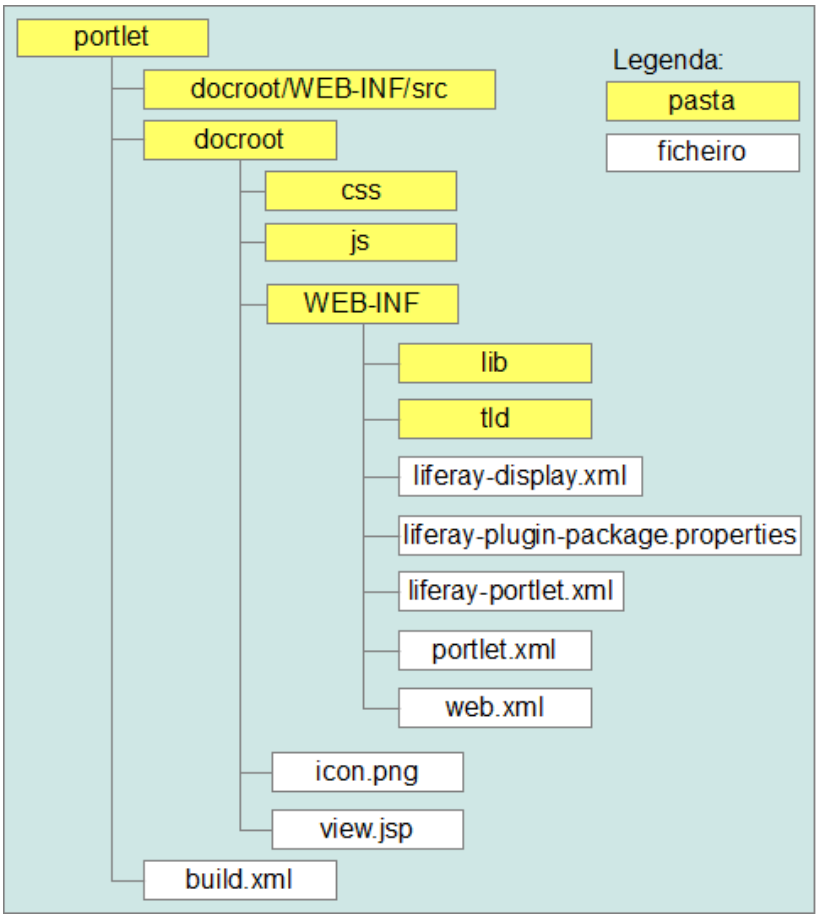

Figura 4.4: Estruturação dos ficheiros e pastas do portlet

configuração do portlet que se encontram dentro da pasta `WEB-INF` e os ficheiros em java, são guardados na pasta `docroot/WEB/src`.

A pasta `docroot` funciona como um diretório mãe para os ficheiros do portlet. Dentro do portlet as pastas `css`, `js` permitem colocar respetivamente os ficheiros do tipo css e javascript. A pasta `WEB-INF` contém os ficheiros de configuração do portlet, permitindo a identificação da categoria do portlet no liferay(ficheiro `liferay-display.xml`), definir propriedades do plugin do portlet(ficheiro `liferay-plugin-packege.properties`)e carregar ficheiros que normalmente são carregados nas página html do tipo css e javascript(ficheiro `liferay-portlet-xml`). As pastas `lib` e `tld` que contém ficheiros inerentes ao portlet.

4.4 Configuração do OpenTripPlanner no Portlet Liferay

4.4.1 Componente interface *web*

O primeiro passo para a integração do OpenTripPlanner no portal Liferay foi a configuração dos ficheiros do OpenTripPlanner no portlet do liferay. Os ficheiros `planner.css` e `wro.xml` e as pastas `images`, `js`, `lib` foram copiados para as respetivas pastas indicadas pelas setas (figura 4.5). O ficheiro `web.xml` do portlet recebeu o

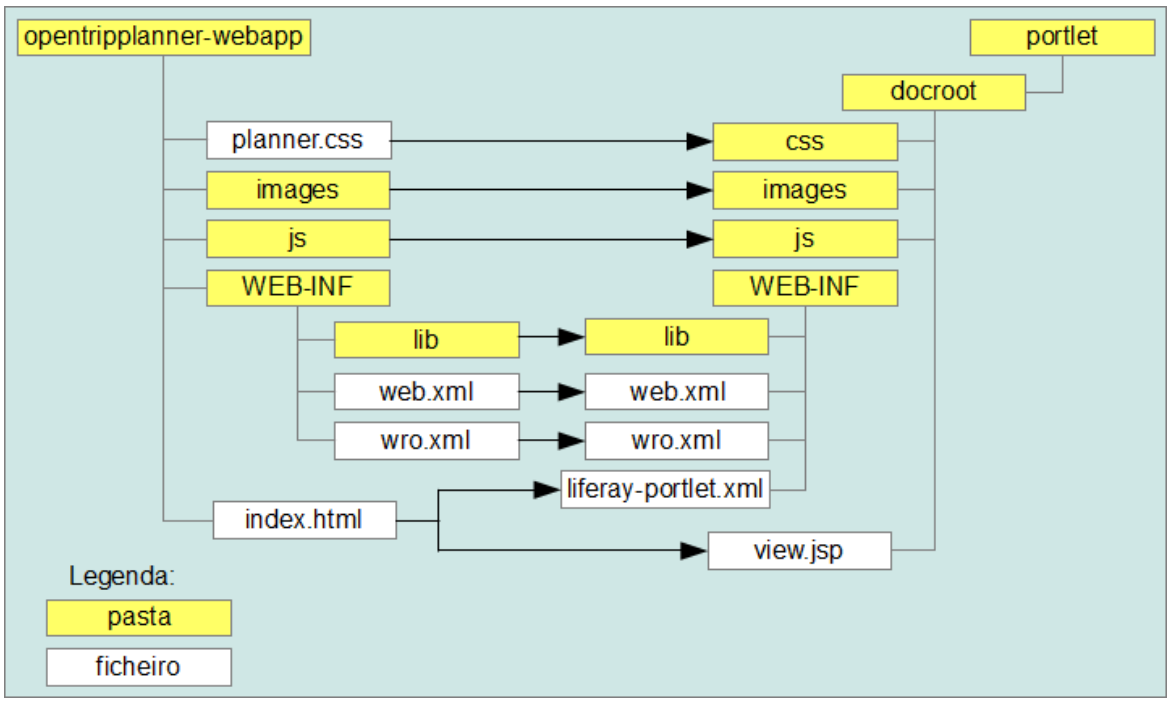

Figura 4.5: Configuração do OpenTripPlanner no portlet

código (figura 4.6) do ficheiro `web.xml` do OpenTripPlanner: Isto permite definir o tipo de aplicação *web* criado.

É no ficheiro `index.html` que se encontram definidos os *scripts* para iniciação da interface do utilizador do OpenTripPlanner. Embora a estrutura do portlet seja semelhante à estrutura da aplicação *web*, no portlet não existe um equivalente direto ao ficheiro `index.html` da aplicação *web*. O código deste ficheiro será portanto distribuído pelos ficheiros `view.jsp` e `liferay-portlet.xml`.

Ao ficheiro `view.jsp` foi adicionado o código do ficheiro `index.hml` como mostra a

```
<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <mime-mapping>
    <extension>js</extension>
    <mime-type>application/x-javascript</mime-type>
  </mime-mapping>
</web-app>
```

Figura 4.6: Código do ficheiro web.xml

figura 4.7:

```
<script>otp.util.AnalyticsUtils.importGoogleAnalytics();</script>
<script type="text/javascript">
Ext.onReady(function()
{
    otp.util.AnalyticsUtils.initGoogleAnalytics();
    new otp.application.Controller(otp.config);
});
</script>
```

Figura 4.7: Linhas de código para iniciar o OpenTripPlanner

O Google Analytics [19] é um serviço gratuito oferecido pela Google servindo para monotorizar páginas *web* como o número de visitas, trafego, etc. Apesar do código iniciar quando o OpenTripPlanner inicia, o serviço disponibilizado pelo GoogleAnalytics só funcionará quando for ativado pelo administrador da página. A linha de código `new otp.application.Controller(otp.config)` inicia o OpenTripPlanner.

Os scripts de carregamento dos ficheiros `css` e `js` que fazem parte do `index.html`, por outro lado, são específicos para o portlet e terão, por isso de ser incorporados no ficheiro `liferay-portlet.xml`. Isto é feito utilizando as *tags*

`<header-portlet-javascript>endereço_ficheiro</header-portlet-javascript>`

para ficheiros `js` e

`<header-portlet-css>endereço do ficheiro<header-portlet-css>`

para ficheiros `css`. Aqui, o endereço do ficheiro necessário para as *tags*, é definido relativamente ao caminho do portlet onde a pasta `docroot` é a pasta raíz. No contexto dos portlets Liferay, as tags

`<header-portlet-css>` e `</header-portlet-javascript>`

definem os endereços dos ficheiros `css` e `js` respetivamente que irão ser referenciados no cabeçalho da página.

As linhas de código (figuras 4.8 4.9) e exemplificam a situação descrita acima para um ficheiro `css` e um ficheiro `js`.

`index.html`

```
<link rel="stylesheet" type="text/css" href="js/lib/ext/resources/css/ext-all-notheme.css" />
<script src="js/lib/openlayers/OpenLayers.js"></script>
```

Figura 4.8: Linhas de código para carregar ficheiros na página `index.html`

`liferay-portlet.xml`

```
<header-portlet-css>/js/lib/ext/resources/css/ext-all-notheme.css</header-portlet-css>
<header-portlet-javascript>/js/lib/openlayers/OpenLayers.js</header-portlet-javascript>
```

Figura 4.9: Linhas de código para carregar ficheiros no `liferay-portlet.xml`

A figura 4.10 mostra o primeiro desenvolvimento do portlet. A visualização do portal é feita na versão 12 do navegador *web* Firefox com a extensão Firebug. O Firebug é um conjunto de ferramentas de desenvolvimento *web* que permitem editar, depurar e monitorar CSS, HTML e JavaScript qualquer página da *web*.

Embora o desenvolvimento do portlet tenha sido feito corretamente, observam-se problemas ao nível da interface, nomeadamente:

1. Diversas imagens não estão ser encontradas;
2. O campo de visualização continua a ser gerado para ocupar toda a janela de visualização aparecendo como fundo e não no interior do portlet como pretendido;
3. O componente servidor *web* não está a ser encontrada;
4. Traduções para Português do interface e alterações no formato da data e hora;

Os erros relacionados com as imagens foram provocados pela deslocalização da pasta `images` e de todos os ficheiros aí contidos. Foi por isso necessário percorrer e atualizar todas as instâncias do código que referenciavam a pasta `images` com o novo endereço.

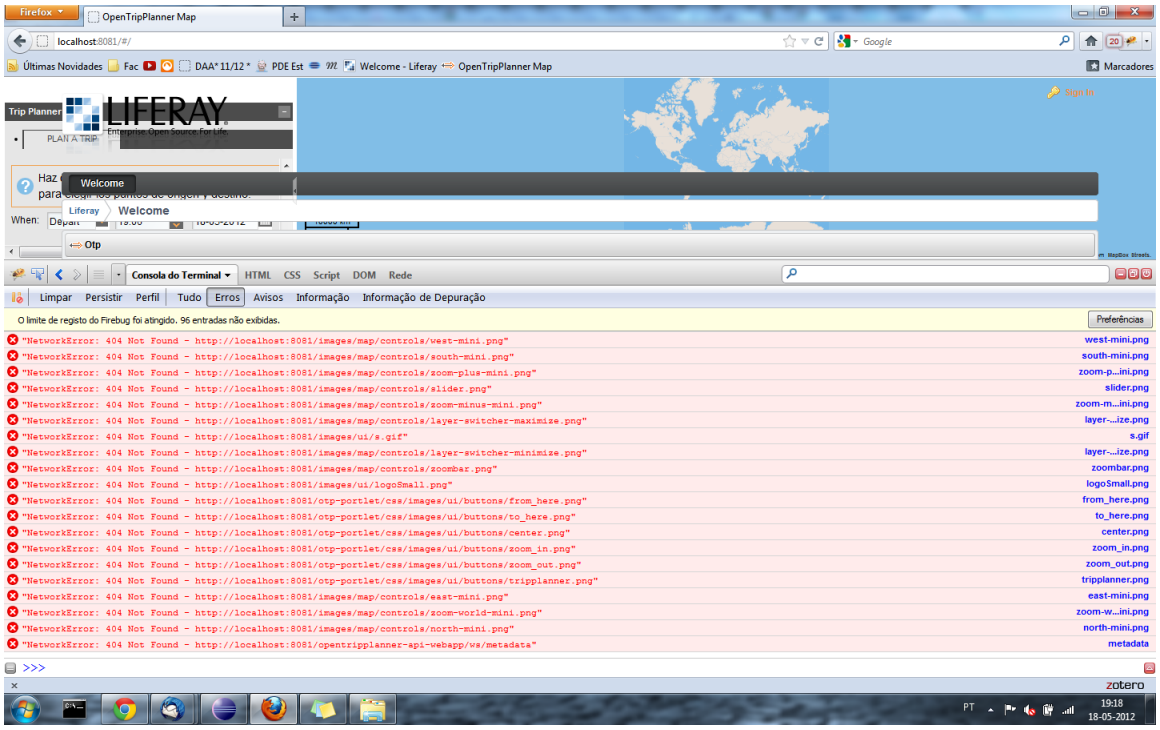

Figura 4.10: Erros

Resolvido o problema das imagens o passo seguinte foi colocar a interface do OpenTripPlanner dentro do portlet. Como já referimos, o OpenTripPlanner utiliza *software* ExtJS para fazer a renderização da página *web*. O ExtJS permite ao programador a criação de uma página *web* utilizando painéis. Cada painel contém uma parte da página *web*. A grande vantagem é que se pode criar, modificar, remover e juntar painéis sem mexer nos restantes componentes. O interface OpenTripPlanner encontra-se dividido em três painéis como mostram as figuras4.11 e 4.12. O painel oeste mostra as opções para o cálculo do itinerário e depois de efetuado o cálculo, descreve verbalmente o itinerário calculado. Os painéis sul e mapa formam o painel central(painel cor cinza) que descreve graficamente o itinerário. O painel mapa mostra o mapa com o itinerário calculado traçado. O painel sul mostra a elevação do solo durante o itinerário. Como não possuimos dados sobre a elevação do solo de Portugal, desativámos o painel sul.

Apesar do OpenTripPlanner estar a correr no portlet, a sua interface está configurada para mostrar numa página completa, situação que se encontra na figura 4.13. Para resolver o problema era necessário mostrar os painéis dentro do portlet. No ficheiro `UI.js` encontra-se o código de cada painel que compõe a interface. A cada painel adicionámos o código `renderTo: id`, obrigando o painel a mostrar na secção da página *web* com o nome `id` criada para o efeito no ficheiro `view.jsp`.

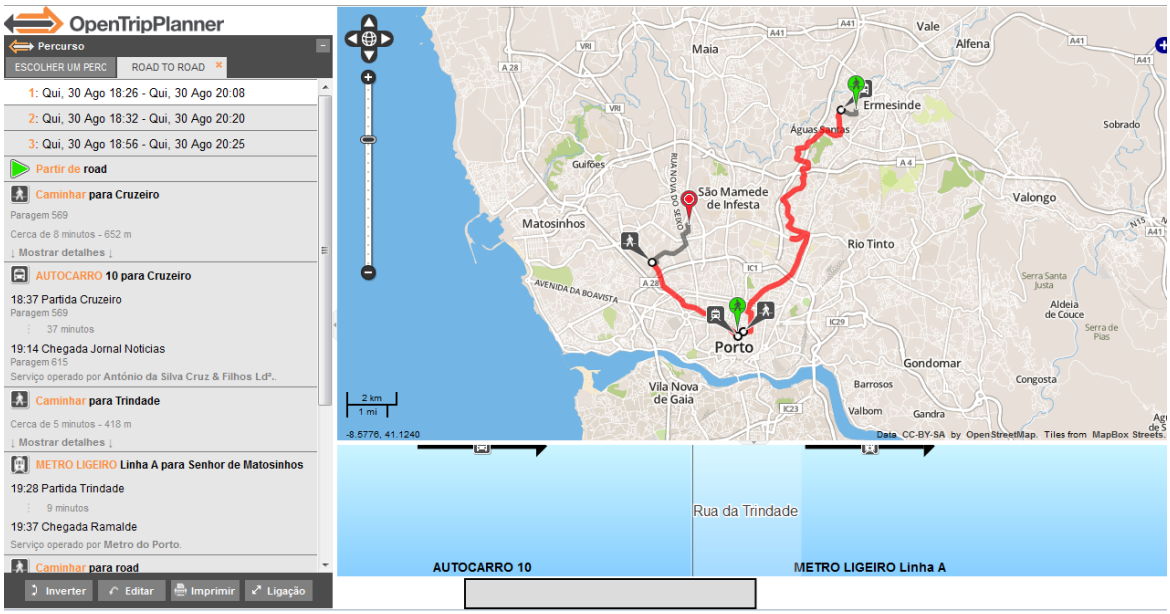

Figura 4.11: Interface OpenTripPlanner composta por paineis

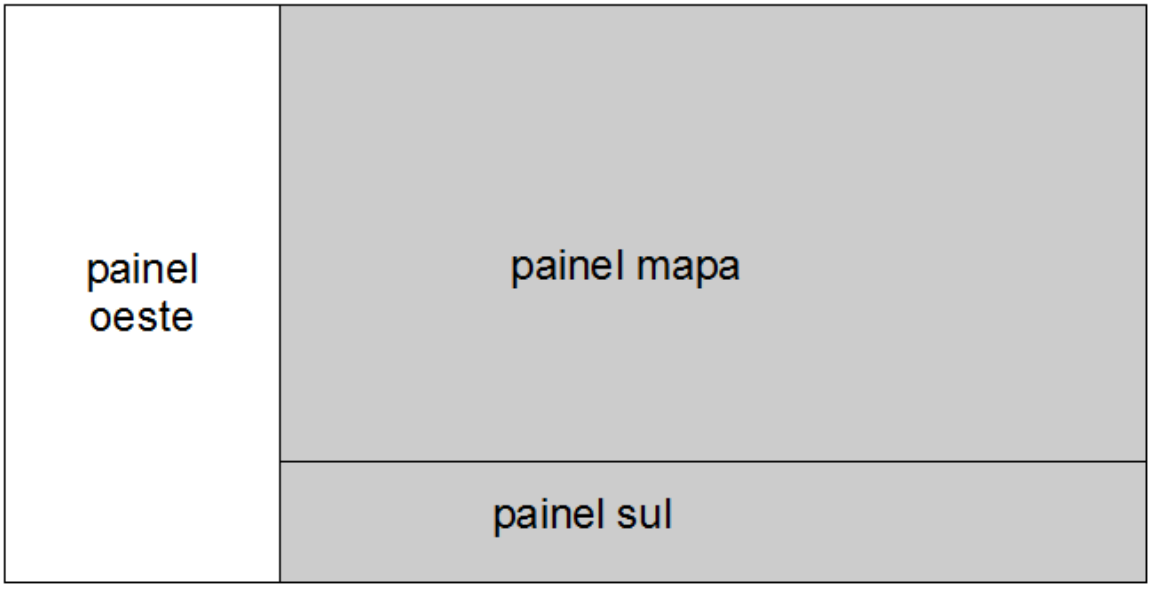

Figura 4.12: Paineis do OpenTripPlanner

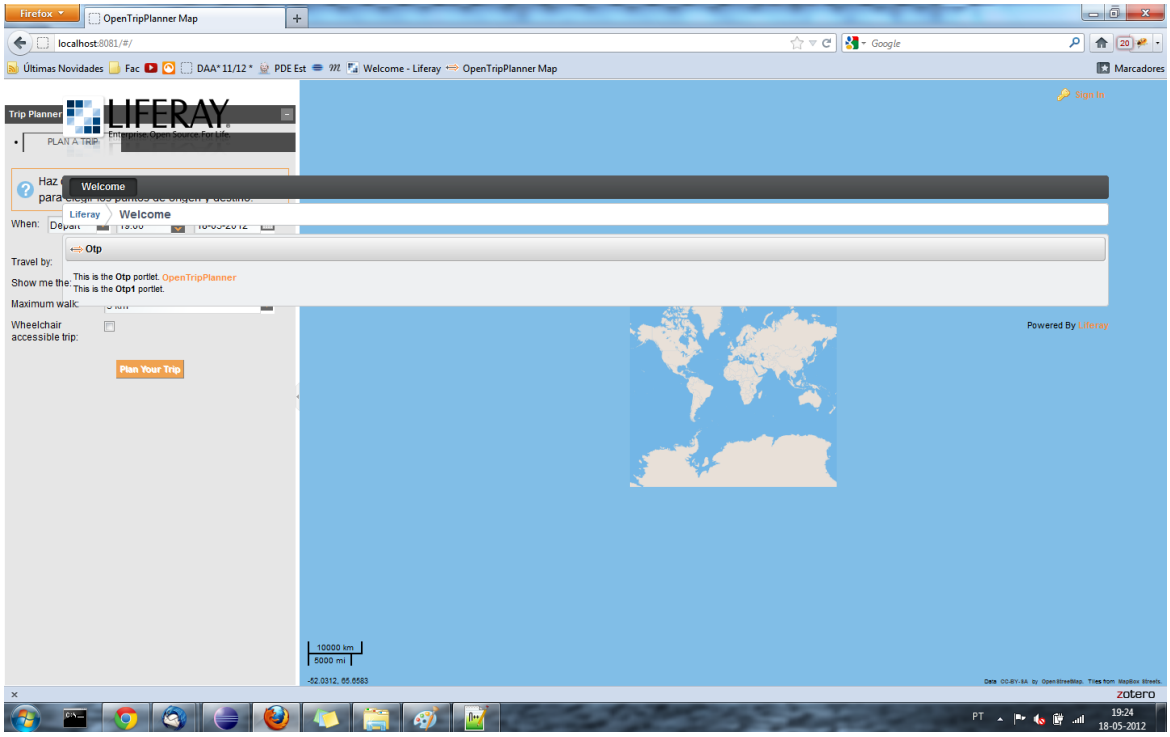

Figura 4.13: Interface OpenTripPlanner por trás do Liferay

No ficheiro `view.jsp` também criámos o código hml (figura 4.14). Isto permite

```
<table>
<tr>
<td><div id="demoMapLeft" align="left"  style="width=:300px"></div></td>
<td><div id="demoMapRight" align="right" style="width=:565px"></div></td>
</tr>
</table>
```

Figura 4.14: Código para criar uma tabela

criar uma tabela invisível com duas colunas, cada coluna é uma secção com o `id` dos painéis. Assim os dois painéis do OpenTripPlanner renderizam dentro de cada coluna permitindo ficar lado a lado mas dentro do portlet como demonstra a figura 4.15.

O OpenTripPlanner tem a possibilidade de alterar algumas configurações como a língua e o sistema métrico para determinados países. A língua e o sistema métrico que vem por defeito é o inglês. Para ser possível efetuar a alteração destas configurações, o OpenTripPlanner possui vários ficheiros com traduções para algumas línguas na pasta `locale`, o português não é exceção. Para configurar para Portugal, alterámos a linha de código `otp.locale.English` para `otp.locale.Portuguese` em todos os ficheiros

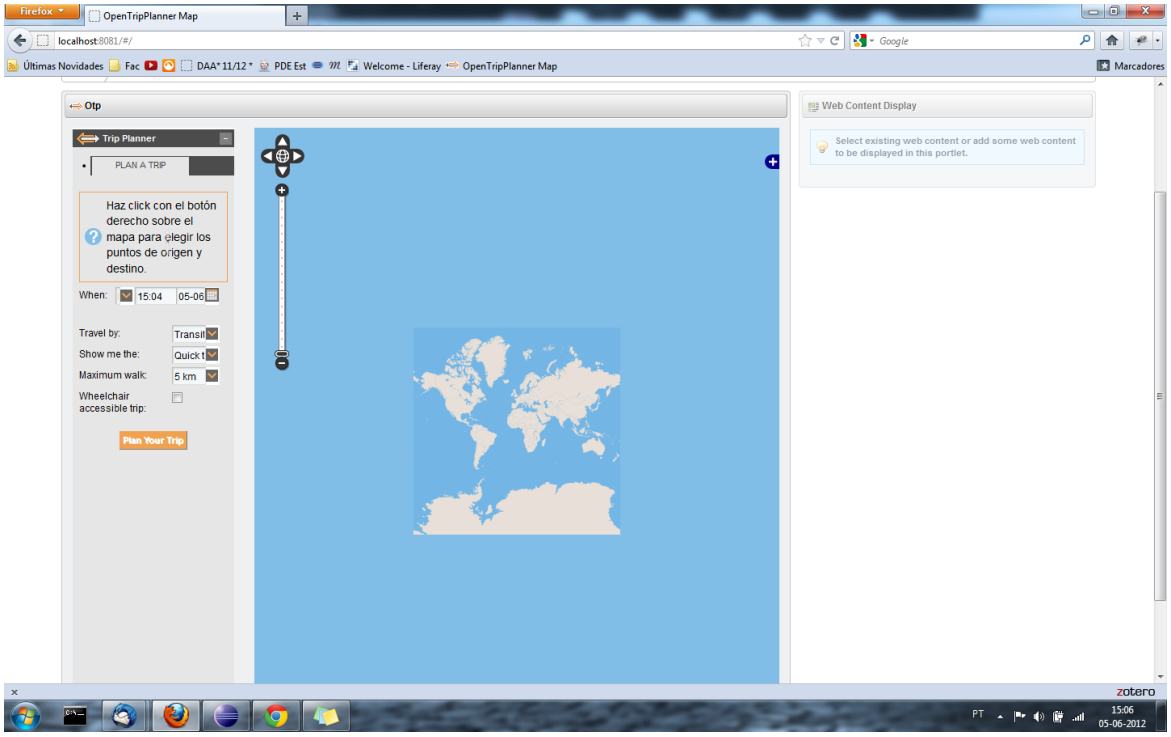

Figura 4.15: Interface OpenTripPlanner correta

que estão dentro da pasta `otp` e suas subpastas.

4.4.2 Componente servidor *web*

Foi também necessário configurar a componente servidor *web*. Os ficheiros da componente servidor *web* para além de se encontrarem na pasta `opentripplanner-api-webapp`, também se encontram compactados no ficheiro `opentripplanner-api-webapp.war`.

A componente servidor *web* não necessita de qualquer modificação, necessita apenas de ser instalada e reconhecida pela componente interface *web*. Os ficheiros do tipo `.war` (Web Application Archive) permitem uma instalação rápida e fácil das aplicações *web* pelo que vantajosa a sua utilização neste caso.

Foi por isso apenas necessário copiar o ficheiro .war para a pasta `deploy` do servidor Tomcat no Liferay. Todos os ficheiros nesta pasta são automaticamente instalados no servidor assim que este é iniciado, ficando disponiveis para utilização.

Depois de integrar a componente servidor web, foi realizado um teste a todas as funcionalidades do portal *web*. Verificámos no entanto que a funcionalidade de impressão

não funcionava. O próximo passo foi a configuração da opção de impressão.

4.4.3 Configuração da opção de impressão

Uma das opções que o OpenTripPlanner disponibiliza, é a impressão do mapa com o trajeto marcado e uma explicação sobre todo o trajeto a ser efetuado. Ao contrário das opções editar e inverter, quando o utilizador carrega no botão para imprimir, é gerada uma nova página *web* numa nova janela do navegador *web* e assim disponibilizada informação numa estrutura mais adequada para impressão. A nova página é gerada pelo ficheiro `print.html` que se encontra dentro da pasta `opentripplanner-webapp` como demonstrado na figura 4.3.

A solução encontrada para resolver o problema foi copiar os ficheiros `print.html`, `planner.css`, as pastas `otp`, `lib` e `images` para dentro da pasta html que se encontra no Tomcat do Liferay. A pasta `html` permite gerar páginas que não sejam integradas na página do Liferay mas que estejam ligadas por algum URL ao Liferay.

Ao realizar este passo voltámos a realizar o teste a todas as funcionalidades, às quais demonstraram funcionar corretamente.

Capítulo 5

Conclusão e trabalho futuro

5.1 Conclusão

O projeto da criação do protótipo do calculador de itinerários Teco Planner teve fases distintas. A pesquisa e análise de calculadores de itinerários a nível nacional e internacional permitiu-nos conhecer sistemas utilizados na construção de calculadores de itinerários bem como as suas funcionalidades mais comuns. Permitiu-nos também obter mais conhecimentos sobre o funcionamento dos transportes públicos. Com o estudo do *software* OpenTriPlanner e da base de dados do SIGGESC a próxima fase foi a criação de um *software* que permitisse converter, transformar os dados da base de dados no formato SIGGESC para uma base de dados no formato GTFS utilizado pelo OpenTripPlanner. O *software* mostrou-se eficaz e eficiente pois transforma os dados sem necessitar de muito espaço permanente para armazenar dados, é de simples execução uma vez que o responsável por esta tarefa não necessitará de conhecimentos sobre o *software*, só terá de o executar e também foi rápido para tratamentos dos dados da base de dados que tivemos durante o projeto.

A integração do OpenTripPlanner no portal corporativo Liferay revelou-se uma boa opção. Pese embora o tempo dispendido a conhecer a interface do OpenTripPlanner e a estrutura e construção de portlets no Liferay, a integração entre ambos revelou-se bastante robusta. Economizou-se também tempo em relação a construir uma página *web* de raiz e permite futuramente uma melhor interação dos utilizadores com o prório portal *web* utilizando as aplicações sociais existentes no portal corporativo Liferay. O cálculo dos itinerários efetuados nos testes não revelaram qualquer problema. Com isto, o objetivo principal do projeto de ter um protótipo do futuro calculador de

itinerários Teco Planner foi atingido.

5.2 Trabalho futuro

Apesar do protótipo Teco Planner já efetuar o cálculo de itinerários, ainda há muito trabalho pela frente. Pode-se dizer que para o Teco Planner crescer há dois rumos a traçar: a colaboração com os operadores de transportes e a evolução do portal *web*. Dentro destes rumos há casos específicos que devem ser tratados:

- **Colaboração com os operadores para melhorar os dados**- em Portugal a base de dados do SIGGESC é recente e por isso os operadores muitas vezes têm dificuldades em inserir os dados corretamente no sistema. A base de dados que recebi para este projeto é um exemplo disso pois continha vários erros, alguns dos quais corrigi através da programação de scripts e outros casos manualmente. Por isso é importante a colaboração com os operadores;
- **Criação um *software* e de uma base de dados que permita guardar e tratar as pequisas efetuadas pelos utilizadores**- as pesquisas efetuadas pelos utilizadores podem revelar informações úteis. Se houver muitas pesquisas para duas cidades entre as quais não existam serviços de transportes, é possivel informar os operadores que a ligação entre aquelas duas cidades é muito pesquisada de forma a que eles possam analisar a disponibilização de serviços;
- **Melhorar a interface do portal *web***- a interface atual do protótipo, é a interface padrão do portal corporativo Liferay com a integração da interface do OpenTripPlanner no mesmo. Assim é necessário melhorar a interface do portal para permitir uma agradável utilização do Teco Planner bem como a disponibilização de outra informação útil aos utilizadores;
- **Adaptação das futuras versões do *software* OpenTripPlanner ao portal *web***- o *software* OpenTripPlaner está em constante evolução permitindo maior rapidez no cálculo de itinerários bem como a adição de novas funcionalidades. A evolução contínua implica também uma contínua adaptação do portal para suportar estas novas funcionalidades.

Apêndice A

Glossário e Lista de Acrónimos

ASCII	- American Standard Code for Information Interchange
AJAX	- Asynchronous Javascript and XML
API	- Application Programming Interface
ArcIMS	- Arc Internet Map Server
ArcGIS	- Arc Geografic Information Sistem
ArcSDE	- Arc Spatial Database Engine
CP	- Comboios de Portugal
CSS	- Cascading Style Sheets
CSV	- Comma Separated Value
GPL	- General Public License
GPS	- Global Positional System
GTFS	- General Transit Feed Specification
HTML	- HyperText Markup Language
IBM	- International Business Machines
IMTT	- Instituto de Mobilidade e dos Transportes Terrestres
JEE	- Java Enterptrise Edition
JS	- JavaScript
JSP	- Java Server Page
NED	- National Elevation Dataset
OS	- Operating System
OSM	- Open Street Map
RAM	- Random Access Memory
SFSQL	- Simple Feature for Structured Query Language
SICO	- Sistema de Informação de Carreiras dos Operadores

SIG	- Sistema de Informação Geográfico
SIGGESC	- Sistema de Informação Geográfica e Gestão de Carreiras
SMS	- Short Message System
STCP	- Sociedade de Transportes Colectivos do Porto
SQL	- Structured Query Language
TST	- Transportes Sul do Tejo
XML	- Extensible Markup Language
WGS	- World Geodetic System
URL	- Uniform Resource Locator

Apêndice B

Código de conversão de SIGGESC para GTFS

```
1  --Parte Geografica
2  --Passo1-------------------------------------------------------------
3  --Converter as coordenadas em Trocos de datum 73 para WGS84
4  ALTER TABLE "Trocos" DROP CONSTRAINT enforce_srid_the_geom;
5  UPDATE "Trocos"    SET the_geom = ST_Transform(the_geom, 4326);
6  ---------------------------------------------------------------------
7
8  --criar a tabela nao no formato gtfs
9  --CREATE temporary TABLE shapes
10 CREATE TABLE shapes
11 (
12   shape_id CHARACTER VARYING NOT NULL,
13   shape_pt_lat DOUBLE PRECISION,
14   shape_pt_long DOUBLE PRECISION,
15   shape_pt_sequence INTEGER NOT NULL,
16   shape_pt_traveled INTEGER,
17   ordemtroco INTEGER,
18   codparini INTEGER,
19   codparfim INTEGER,
20   gid serial NOT NULL,
21   CONSTRAINT shapes_pkey PRIMARY KEY (shape_id , shape_pt_sequence )
22 )
23 WITH (
24   OIDS=FALSE
25 );
26 ALTER TABLE shapes
27   OWNER TO postgres;
```

```
28
29 --Passo2-----------------------------------------------------------
30 --ordenacao dos pontos em Trocos e insercao nas shapes
31 CREATE OR REPLACE FUNCTION ordenacaoPontosShapes() RETURNS void AS $$
32 DECLARE
33  tbrow     RECORD;
34  subtotal VARCHAR;
35  m INTEGER;
36  g GEOMETRY;
37  t GEOMETRY;
38  j GEOMETRY;
39  n INTEGER;
40  p GEOMETRY;
41  valor INTEGER;
42 BEGIN
43   subtotal:=0;
44   --Passo2.1-----
45   FOR tbrow IN SELECT CAST(CAST("carreira" AS VARCHAR) || '-' || CAST("
         parcelar" AS VARCHAR)
46     || '-' || CAST("variante" AS VARCHAR) || '-' ||  CAST("sentido" AS
          VARCHAR) AS VARCHAR) AS shape_id, "codparin","codparfi",the_geom,
           "ordemtro"
47    FROM "Trocos" ORDER BY gid
48   LOOP
49     --Passo2.2-- caso o troco = 1 ----------------------------
50     IF tbrow."ordemtro" = 1 THEN
51       ---- caso seja multilinestring ---------------------
52       valor:=0;
53       --Passo2.3--
54       IF GeometryType(tbrow.the_geom) LIKE 'MULTI%' THEN
55         FOR m IN SELECT generate_series(1, ST_NumGeometries(tbrow.
               the_geom)) LOOP
56         ---- comparacao para verificar se a 1ª linestring ta correcta ou
                inversa
57         -- m=1
58           --Passo 2.4
59           IF m = 1 THEN
60             g :=ST_GeometryN(tbrow.the_geom, m);
61             t :=ST_GeometryN(tbrow.the_geom, m+1);
62             --Passo 2.5
63             IF ST_StartPoint(g)= ST_StartPoint(t) OR ST_StartPoint(g)=
                    ST_EndPoint(t) THEN
64               n=ST_NumPoints(g);
65               while n > 0 LOOP
66                 p := ST_PointN(g, n);
```

```
67                valor:=valor +1;
68                n=n-1;
69                INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                      shape_pt_sequence,ordemtroco,codparini,codparfim)
70                  VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                        ordemtro",tbrow."codparin",tbrow."codparfi");
71              END LOOP;
72            ELSE
73              FOR n IN SELECT generate_series(1, ST_NumPoints(g)) LOOP
74                p := ST_PointN(g, n);
75                valor:=valor +1;
76                INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                      shape_pt_sequence,ordemtroco,codparini,codparfim)
77                  VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                        ordemtro",tbrow."codparin",tbrow."codparfi");
78              END LOOP;
79            END IF;
80          ---- fim da comparacao para verificar se a 1ª linestring ta
                correcta ou inversa
81          ELSE
82            -- m seguintes
83            ---- comparacao para verificar se as linestrings seguintes
                  tao correctas ou inversas
84            --Passo 2.6
85            g :=ST_GeometryN(tbrow.the_geom, m);
86            IF p = ST_StartPoint(g) THEN
87              FOR n IN SELECT generate_series(1, ST_NumPoints(g)) LOOP
88                p := ST_PointN(g, n);
89                valor:=valor +1;
90                INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                      shape_pt_sequence,ordemtroco,codparini,codparfim)
91                  VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                        ordemtro",tbrow."codparin",tbrow."codparfi");
92              END LOOP;
93            ELSIF p = ST_EndPoint(g) THEN
94                n=ST_NumPoints(g);
95                while n > 0 LOOP
96                  p := ST_PointN(g, n);
97                  valor:=valor +1;
98                  n=n-1;
99                  INSERT INTO shapes(shape_id, shape_pt_lat,
                        shape_pt_long,shape_pt_sequence,ordemtroco,
                        codparini,codparfim)
100                   VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                          ordemtro",tbrow."codparin",tbrow."codparfi");
```

```
101                 END LOOP;
102               END IF;
103               ---- fim da comparacao para verificar se as linestrings
                      seguintes tao correctas ou inversas
104             END IF;
105           END LOOP;
106         ---- fim do caso seja multilinestring para ordemtro =1
107         ELSE
108         ---- como não tem nenhuma linestring no inicio nao entra nesta
              condicao
109           ---- caso o primeiro seja linestring----------
110           --Passo2.7
111           FOR n IN SELECT generate_series(1, ST_NumPoints(tbrow.the_geom))
                LOOP
112             p := ST_PointN(g, n);
113             valor:=valor + 1;
114             INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                  shape_pt_sequence,ordemtroco,codparini,codparfim)
115               VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."ordemtro"
                    ,tbrow."codparin",tbrow."codparfi");
116           END LOOP;
117         END IF;
118       ---- fim do caso o troco = 1 ----------------------------------
119       ELSE
120       ---- caso os trocos > 1 ----------------------------------------
121       ---- caso seja multilinestring ----------aqui compara sempre com o
            ponto anterior
122         --Passo 2.8
123         IF GeometryType(tbrow.the_geom) LIKE 'MULTI%' THEN
124           FOR m IN SELECT generate_series(1, ST_NumGeometries(tbrow.
                the_geom)) LOOP
125             g :=ST_GeometryN(tbrow.the_geom, m);
126             --Passo2.9
127             IF ST_StartPoint(j)= ST_StartPoint(g) OR ST_EndPoint(j)=
                  ST_StartPoint(g) THEN
128               FOR n IN SELECT generate_series(1, ST_NumPoints(g)) LOOP
129                 p := ST_PointN(g, n);
130                 valor:=valor +1;
131                 INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                      shape_pt_sequence,ordemtroco,codparini,codparfim)
132                   VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                        ordemtro",tbrow."codparin",tbrow."codparfi");
133               END LOOP;
134             ELSE
135               n=ST_NumPoints(g);
```

```
136             WHILE n > 0 LOOP
137               p := ST_PointN(g, n);
138               valor:=valor +1;
139               n=n-1;
140               INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                      shape_pt_sequence,ordemtroco,codparini,codparfim)
141                 VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                        ordemtro",tbrow."codparin",tbrow."codparfi");
142             END LOOP;
143           END IF;
144           j :=ST_GeometryN(tbrow.the_geom, m);
145         END LOOP;
146         ---- fim da  comparacao para verificar se a 1ª linestring ta
              correcta ou inversa
147       ELSE
148         ---- caso seja linestring
149         g :=tbrow.the_geom;
150         --Passo2.10
151         IF ST_StartPoint(j)= ST_StartPoint(g) OR ST_EndPoint(j)=
              ST_StartPoint(g) THEN
152           FOR n IN SELECT generate_series(1, ST_NumPoints(g)) LOOP
153             p := ST_PointN(g, n);
154             valor:=valor + 1;
155             INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                    shape_pt_sequence,ordemtroco,codparini,codparfim)
156               VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                      ordemtro",tbrow."codparin",tbrow."codparfi");
157           END LOOP;
158         ELSE
159           n=ST_NumPoints(g);
160           while n > 0 LOOP
161             p := ST_PointN(g, n);
162             valor:=valor +1;
163             n=n-1;
164             INSERT INTO shapes(shape_id, shape_pt_lat,shape_pt_long,
                    shape_pt_sequence,ordemtroco,codparini,codparfim)
165               VALUES(tbrow.shape_id,ST_Y(p),ST_X(p),valor,tbrow."
                      ordemtro",tbrow."codparin",tbrow."codparfi");
166           END LOOP;
167         END IF;
168         j :=tbrow.the_geom;
169         ---- fim da comparacao para verificar se as linestrings
              seguintes tao correctas ou inversas
170       END IF;
171     END IF;
```

```
172      ---- fim do caso seja multilinestring ----------------------
173    END LOOP;
174    RETURN;
175 END;
176 $$ LANGUAGE plpgsql;
177
178 SELECT * FROM ordenacaoPontosShapes();
179 ------------------------------------------------------------------
180
181 --Passo3-----------------------------------------------------------
182 --calculo da distancia de shapes
183 CREATE OR REPLACE FUNCTION calculoDistanciaShapes() RETURNS void AS $$
184 DECLARE
185 tbrow     RECORD;
186 p GEOMETRY;
187 distance DOUBLE PRECISION;
188 BEGIN
189    FOR tbrow IN SELECT * FROM shapes ORDER BY gid
190    LOOP
191      --Passo 3.1
192      IF tbrow.shape_pt_sequence = 1 THEN
193         distance=0;
194         UPDATE shapes
195         SET shape_pt_traveled =distance
196         WHERE gid = tbrow.gid;
197
198         p:=ST_MakePoint(tbrow.shape_pt_lat,tbrow.shape_pt_long);
199      ELSE
200         --Passo3.2
201         distance:=distance + ST_distance_sphere(p,ST_MakePoint(tbrow.
               shape_pt_lat,tbrow.shape_pt_long));
202         p:=ST_MakePoint(tbrow.shape_pt_lat,tbrow.shape_pt_long);
203         UPDATE shapes
204         SET shape_pt_traveled=distance
205         WHERE gid = tbrow.gid;
206
207      END IF;
208    END LOOP;
209
210 END;
211 $$ LANGUAGE plpgsql;
212
213 SELECT * FROM calculoDistanciaShapes();
214 ------------------------------------------------------------------
215
```

```
216 --Passo4-----------------------------------------------------------
217 -- colocacao do identificador do operador no valor da shape_id
218 UPDATE shapes
219 SET shape_id=CAST(idoperador AS varchar) ||'-'|| a.shape_id
220 FROM operadores,shapes AS a
221 WHERE shapes.gid=a.gid;
222
223 -- eliminacao de campos para ficar igual a tabela shapes do gtfs
224 ALTER TABLE shapes
225 DROP ordemtroco,
226 DROP codparini,
227 DROP codparfim,
228 DROP gid;
229 ----------------------------------------------------------------------
230
231 --Passo5-----------------------------------------------------------
232 --Converter as coordenadas em Paragens de datum 73 para WGS84
233 ALTER TABLE "Paragens" DROP CONSTRAINT enforce_srid_the_geom;
234 UPDATE "Paragens" SET the_geom = ST_Transform(the_geom, 4326);
235
236 --Adicionar latitude e longitude à tabela Paragens
237 ALTER TABLE "Paragens" ADD COLUMN latitude DOUBLE PRECISION;
238 ALTER TABLE "Paragens" ADD COLUMN longitude DOUBLE PRECISION;
239
240 UPDATE "Paragens" SET latitude  = ST_Y(the_geom);
241 UPDATE "Paragens" SET longitude = ST_X(the_geom);
242 ----------------------------------------------------------------------
243
244 --Parte alfanumérica
245 --criar as tabelas do gtfs(agency,calendar,routes,stop_times,stops,trips
    )
246 --CREATE temporary TABLE agency
247 CREATE  TABLE agency
248 (
249   agency_id INTEGER NOT NULL,
250   agency_name CHARACTER VARYING(56),
251   agency_url CHARACTER VARYING(256),
252   agency_timezone CHARACTER VARYING(56),
253   agency_lang CHARACTER VARYING(8),
254   agency_phone CHARACTER VARYING(13),
255   agency_fare_url CHARACTER VARYING(256),
256   CONSTRAINT agency_pkey PRIMARY KEY (agency_id )
257 )
258 WITH (
259   OIDS=FALSE
```

```
260 );
261 ALTER TABLE agency
262   OWNER TO postgres;
263
264
265 --CREATE temporary TABLE calendar
266 CREATE TABLE calendar
267 (
268   service_id CHARACTER VARYING(32) NOT NULL,
269   monday BIT(1),
270   tuesday BIT(1),
271   wednesday BIT(1),
272   thursday BIT(1),
273   friday BIT(1),
274   saturday BIT(1),
275   sunday BIT(1),
276   start_date CHARACTER(8),
277   end_date CHARACTER(8),
278   CONSTRAINT calendar_pkey PRIMARY KEY (service_id )
279 )
280 WITH (
281   OIDS=FALSE
282 );
283 ALTER TABLE calendar
284   OWNER TO postgres;
285
286
287 --CREATE temporary TABLE routes
288 CREATE TABLE routes
289 (
290   route_id INTEGER NOT NULL,
291   agency_id INTEGER,
292   route_short_name CHARACTER VARYING(32),
293   route_long_name CHARACTER VARYING(64),
294   route_desc CHARACTER VARYING(256),
295   route_type SMALLINT,
296   route_url CHARACTER VARYING(256),
297   route_color CHARACTER(6),
298   route_text_color CHARACTER(6),
299   CONSTRAINT routes_pkey PRIMARY KEY (route_id )
300 )
301 WITH (
302   OIDS=FALSE
303 );
304 ALTER TABLE routes
```

```
305   OWNER TO postgres;
306
307
308 --CREATE temporary TABLE stops
309 CREATE  TABLE stops
310 (
311   stop_id INTEGER NOT NULL,
312   stop_code INTEGER,
313   stop_name CHARACTER VARYING(75),
314   stop_desc CHARACTER VARYING(50),
315   stop_lat DOUBLE PRECISION,
316   stop_lon DOUBLE PRECISION,
317   zone_id CHARACTER VARYING(32),
318   stop_url CHARACTER VARYING(256),
319   location_type "char",
320   parent_station "char",
321   CONSTRAINT stops_pkey PRIMARY KEY (stop_id )
322 )
323 WITH (
324   OIDS=FALSE
325 );
326 ALTER TABLE stops
327   OWNER TO postgres;
328
329
330 --CREATE temporary TABLE trips
331 CREATE TABLE trips
332 (
333   route_id INTEGER NOT NULL,
334   service_id CHARACTER VARYING(32),
335   trip_id CHARACTER VARYING(32) NOT NULL,
336   trip_headsign CHARACTER VARYING(256),
337   trip_short_name CHARACTER VARYING(75),
338   direction_id INTEGER,
339   block_id CHARACTER VARYING(32),
340   shape_id CHARACTER VARYING,
341   CONSTRAINT trips_pkey PRIMARY KEY (trip_id )
342 )
343 WITH (
344   OIDS=FALSE
345 );
346 ALTER TABLE trips
347   OWNER TO postgres;
348
349 --Passo6-----------------------------------------------------
```

```
350 --inserir valores na tabela agency
351 INSERT INTO agency(agency_id, agency_name, agency_url, agency_timezone,
        agency_lang,agency_phone, agency_fare_url)
352 SELECT idoperador,regexp_replace(desigoperador, ',', ' '),
        regexp_replace(url, ',', ' '),'Europe/Lisbon', 'PT', '+351221465897',
         'http://www.maiatranpostes.pt/tickect'
353 FROM operadores;
354 ------------------------------------------------------------
355
356 --Passo7------------------------------------------------------
357 --inserir valores na tabela stops
358 INSERT INTO stops(stop_id,stop_code,stop_name,stop_lat,stop_lon)
359 SELECT distinct idparagem, codparoperador, regexp_replace("Paragens".
        designacao, ',', ' '), latitude, longitude
360 FROM paragem,"Paragens"
361 WHERE idparagem=codparagem;
362 ------------------------------------------------------------
363
364 --Passo8------------------------------------------------------
365 --inserir valores na tabela routes
366 INSERT INTO routes(route_id, agency_id, route_short_name,
        route_long_name, route_type)
367 SELECT idcarreira,a.idoperador,CAST(codcaroperador AS VARCHAR),
        regexp_replace(designacao, ',', ' '),3
368 FROM centrooperacional AS a,carreiras AS b
369 WHERE a.idcentroop = b.idcentroop
370 ORDER BY idcarreira;
371 ------------------------------------------------------------
372
373 --Passo9------------------------------------------------------
374 --inserir valores na tabela calendar
375 INSERT INTO calendar(service_id, monday, tuesday, wednesday, thursday,
        friday, saturday,sunday, start_date, end_date)
376 SELECT CAST(b.idperiodo AS VARCHAR)||CAST(a.idfrequencia AS VARCHAR),
377         CAST(CAST(a.n2 AS INTEGER) AS BIT),CAST(CAST(a.n3 AS INTEGER) AS
              BIT),CAST(CAST(a.n4 AS INTEGER) AS BIT),
378         CAST(CAST(a.n5 AS INTEGER) AS BIT),CAST(CAST(a.n6 AS INTEGER) AS
              BIT),CAST(CAST(a.n7 AS INTEGER) AS BIT),
379         CAST(CAST(a.n1 AS INTEGER) AS BIT),
380         substr(CAST(b.datainicio AS VARCHAR),1,4) || substr(CAST(b.
              datainicio AS VARCHAR),6,2)||substr(CAST(b.datainicio AS
              VARCHAR),9,2),
381         substr(CAST(b.datafim AS VARCHAR),1,4)||substr(CAST(b.datafim AS
              VARCHAR),6,2)||substr(CAST(b.datafim AS VARCHAR),9,2)
382 FROM frequencias AS a,periodosanuais AS B
```

```
383 WHERE a.idfrequencia=b.idfrequencia;
384 ------------------------------------------------------------
385
386 --Passo10------------------------------------------------------
387 --inserir valores na tabela trips no sentido 0
388 INSERT INTO trips(route_id, service_id, trip_id, trip_headsign,
        direction_id)
389 SELECT c.idcarreira,d.service_id AS t, CAST(a.idoperador AS VARCHAR) ||'
        -'|| CAST(c.idcarreira AS VARCHAR)||'-'|| CAST((c.sentidocirculacao
        -1) AS VARCHAR)
390         ||'-'|| CAST(c.idfrequencia AS VARCHAR)||'-'|| SUBSTR(CAST(((c.
                idcirculacao-1)/10000.00) AS VARCHAR),3,4) ||'-'|| d.
                service_id, CAST(b.destino AS VARCHAR),(c.sentidocirculacao)-1
391 FROM centrooperacional AS a,carreiras AS b,carreiracirculacaof AS c,
        calendar AS d
392 WHERE b.idcarreira=c.idcarreira AND a.idcentroop= b.idcentroop AND c.
        sentidocirculacao=1 AND CAST(c.idfrequencia AS VARCHAR)= SUBSTR(d.
        service_id,2,2);
393
394 --inserir valores na tabela trips no sentido 1
395 INSERT INTO trips(route_id, service_id, trip_id, trip_headsign,
        direction_id)
396 SELECT c.idcarreira,d.service_id AS t, CAST(a.idoperador AS VARCHAR) ||'
        -'|| CAST(c.idcarreira AS VARCHAR)||'-'|| CAST((c.sentidocirculacao
        -1) AS VARCHAR)
397         ||'-'|| CAST(c.idfrequencia AS VARCHAR)||'-'|| SUBSTR(CAST(((c.
                idcirculacao-1)/10000.00) AS VARCHAR),3,4)||'-'|| d.service_id
                , CAST(b.origem AS VARCHAR),(c.sentidocirculacao)-1
398 FROM centrooperacional AS a,carreiras AS b,carreiracirculacaof AS c,
        calendar AS d
399 WHERE b.idcarreira=c.idcarreira AND a.idcentroop= b.idcentroop AND c.
        sentidocirculacao=2 AND CAST(c.idfrequencia AS VARCHAR)= SUBSTR(d.
        service_id,2,2);
400 ------------------------------------------------------------
401
402 --Passo11------------------------------------------------------
403 --inserir o valor de shape_id
404 UPDATE trips
405 SET shape_id =(CAST(idoperador AS varchar)
406      ||'-'||CAST(carreiranome AS varchar)
407      ||'-'||CAST(parcelar AS varchar)
408      ||'-'||CAST(variante AS varchar)
409      ||'-'||CAST(sentido AS varchar))
410 FROM auxiliar5
411 where route_id= idcarreira AND direction_id= sentido-1;
```

```
412 ------------------------------------------------------------
413
414 --criar a uma tabela auxiliar1
415 --CREATE temporary TABLE auxiliar1
416 CREATE TABLE auxiliar1
417 ( gid serial NOT NULL,
418   idtroco BIGINT NOT NULL,
419   idcarreira BIGINT NOT NULL,
420   idparageminicio BIGINT,
421   idparagemfim BIGINT,
422   tempopercursomin REAL,
423   tempopercursomed REAL,
424   tempopercursomax REAL,
425   temponotroco REAL,
426   sentidotroco BIGINT NOT NULL,
427   ordemtroco BIGINT NOT NULL,
428   extensaotroco BIGINT,
429   extensaotrocofim BIGINT,
430   CONSTRAINT auxiliar1_pkey PRIMARY KEY (idcarreira , sentidotroco ,
          ordemtroco )
431 )
432 WITH (
433   OIDS=FALSE
434 );
435 ALTER TABLE auxiliar1
436   OWNER TO postgres;
437
438 --Passo12------------------------------------------------
439 --inserir valores na tabela auxiliar1
440 INSERT INTO auxiliar1(idtroco, idcarreira, idparageminicio, idparagemfim
      , tempopercursomin, tempopercursomed, tempopercursomax, sentidotroco,
       ordemtroco,extensaotroco)
441 SELECT b.idtroco,b.idcarreira,a.idparageminicio,a.idparagemfim,a.
     tempopercursomin,a.tempopercursomed,a.tempopercursomax,b.sentidotroco
     ,b.ordem AS ordemtroco, a.extensaotroco
442 FROM troco AS a,rtrococarreira AS b
443 WHERE a.idtroco=b.idtroco
444 ORDER BY idcarreira,sentidotroco,ordemtroco;
445
446 ---corrigir o tempopercursomed usado no calculo do tempo por troco
447 UPDATE auxiliar1
448    SET tempopercursomed= ((CAST((tempopercursomin+tempopercursomax) AS
          FLOAT)) )/ 2.0
449  WHERE tempopercursomin> tempopercursomed OR tempopercursomin >
      tempopercursomax OR tempopercursomed > tempopercursomax ;
```

```
450 ------------------------------------------------------------
451
452 --Passo13------------------------------------------------------
453 -- calcular o tempo e a distancia de cada troco na tabela Auxiliar1
454 CREATE OR REPLACE FUNCTION calculoDistanciaAuxiliar1() RETURNS VOID AS
        $$
455 DECLARE
456   tbrow RECORD;
457   valor1 DOUBLE PRECISION;
458   valor2 INTEGER;
459 BEGIN
460   valor1 := 0;
461   valor2 := 0;
462  FOR tbrow IN
463    SELECT gid,ordemtroco,tempopercursomed,extensaotroco FROM auxiliar1
         ORDER BY gid
464  LOOP
465    IF tbrow.ordemtroco = 1 THEN
466     valor1 := tbrow.tempopercursomed;
467     valor2 := tbrow.extensaotroco;
468     UPDATE auxiliar1 SET temponotroco = valor1, extensaotrocofim=valor2
          WHERE auxiliar1.gid=tbrow.gid;
469  ELSE
470    valor1:=valor1+tbrow.tempopercursomed;
471    valor2:=valor2+tbrow.extensaotroco;
472    UPDATE auxiliar1 SET temponotroco =valor1,extensaotrocofim=valor2
         WHERE auxiliar1.gid=tbrow.gid;
473    END IF;
474  END LOOP;
475  RETURN;
476 END;
477 $$ LANGUAGE plpgsql;
478
479 SELECT * FROM calculoDistanciaAuxiliar1();
480 ------------------------------------------------------------
481
482 --criar uma tabela auxiliar2
483  --CREATE temporary TABLE auxiliar2
484  CREATE TABLE auxiliar2
485 (
486   trip_id CHARACTER VARYING(32),
487   arrival_time INTEGER,
488   departure_time INTEGER,
489   stop_id INTEGER,
490   stop_sequence INTEGER,
```

```
491    stop_headsign CHARACTER VARYING(32),
492    pickup_type INTEGER,
493    drop_off_type INTEGER,
494    shape_dist_traveled DOUBLE PRECISION,
495    gid serial NOT NULL,
496    CONSTRAINT auxiliar2_pkey PRIMARY KEY (gid )
497 )
498 WITH (
499    OIDS=FALSE
500 );
501 ALTER TABLE auxiliar2
502    OWNER TO postgres;
503
504 --Passo14-----------------------------------------------
505 --inserir valores na tabela auxiliar2
506 CREATE OR REPLACE FUNCTION calculoValoresAuxiliar2() RETURNS VOID AS $$
507 DECLARE
508    tbrow1      RECORD;
509    tbrow2      RECORD;
510    valor INTEGER;
511    valor1 INTEGER;
512    --sbrow      sub;
513 BEGIN
514    valor := 0;
515    valor1:= 0 ;
516    FOR tbrow1 IN  SELECT * FROM trips
517    LOOP
518      FOR tbrow2 IN SELECT * FROM auxiliar1
519      LOOP
520    --Passo14.1
521    IF   tbrow1.route_id = tbrow2.idcarreira AND tbrow1.direction_id =
           tbrow2.sentidotroco-1 THEN -- relacionamento das tabelas.
522      IF tbrow2.ordemtroco <> 1 THEN
523        valor:= CAST(substr(tbrow1.trip_id,15,4) AS INTEGER) *60 + tbrow2.
             temponotroco *60;
524        valor1:=valor;
525        INSERT INTO auxiliar2 (trip_id,arrival_time,departure_time,
             stop_id,shape_dist_traveled)
526              VALUES (tbrow1.trip_id,valor,valor,tbrow2.idparagemfim,
                   tbrow2.extensaotrocofim);
527
528      ELSE
529        --Passo14.2
530        INSERT INTO auxiliar2 (trip_id,arrival_time,departure_time,stop_id
             , shape_dist_traveled)
```

```
531                 VALUES (tbrow1.trip_id,CAST(substr(tbrow1.trip_id,15,4) AS
                        INTEGER)*60 ,CAST(substr(tbrow1.trip_id,15,4) AS INTEGER
                        ) *60 ,tbrow2.idparageminicio,0);
532         valor:= CAST(substr(tbrow1.trip_id,15,4) AS INTEGER)*60  + (tbrow2
                .temponotroco * 60);
533         INSERT INTO auxiliar2 (trip_id,arrival_time,departure_time,
                stop_id,shape_dist_traveled)
534                 VALUES (tbrow1.trip_id,valor,valor,tbrow2.idparagemfim,
                        tbrow2.extensaotrocofim);
535       END IF;
536     END IF;
537       END LOOP;
538     END LOOP;
539     RETURN;
540 END;
541 $$ LANGUAGE plpgsql;
542
543 SELECT * FROM calculoValoresAuxiliar2();
544 ------------------------------------------------------
545
546 --criar a tabela stop_times-----------------------
547 --CREATE temporary TABLE stop_times
548 CREATE TABLE stop_times
549 (
550   trip_id CHARACTER VARYING(256),
551   tempo_chegada INTEGER,
552   tempo_partida INTEGER,
553   arrival_time CHARACTER VARYING(8),
554   departure_time CHARACTER VARYING(8),
555   stop_id INTEGER,
556   stop_sequence INTEGER,
557   stop_headsign CHARACTER VARYING(256),
558   pickup_type INTEGER,
559   drop_off_type INTEGER,
560   shape_dist_traveled DOUBLE PRECISION,
561   gid serial NOT NULL,
562   CONSTRAINT stop_times_pkey PRIMARY KEY (gid,trip_id,  stop_sequence )
563 )
564 WITH (
565   OIDS=FALSE
566 );
567 ALTER TABLE stop_times
568   OWNER TO postgres;
569 ------------------------------------------------------
570
```

```
571 --Passo15-----------------------------------------
572 --inserir valores na tabela stop_times
573 INSERT INTO stop_times (trip_id,tempo_chegada,tempo_partida,stop_id,
        stop_sequence,stop_headsign,shape_dist_traveled)
574 SELECT trip_id, arrival_time, departure_time, stop_id, b.ordem,
        regexp_replace(c.designacao, ',', ' '), shape_dist_traveled
575 FROM auxiliar2 AS a, rparagenscarreiras AS b,paragem AS c
576 WHERE CAST(substr(a.trip_id,5,4) AS INTEGER) = b.idcarreira AND a.
        stop_id=b.idparagem AND CAST(substr(a.trip_id,10,1) AS INTEGER)=(b.
        sentidoparagem)-1
577 AND b.idparagem = c.idparagem
578 --and arrival_time<>departure_time
579 ORDER BY trip_id,arrival_time,stop_sequence;
580 ----------------------------------------------------
581
582 --Passo16-----------------------------------------
583 --apagar valores tepetidos em stop_times
584 CREATE OR REPLACE FUNCTION apagarRepetidosStopTimes() RETURNS VOID AS $$
585 DECLARE
586   tbrow1     RECORD;
587   valor1 INTEGER;
588   valor2 INTEGER;
589   valor3 INTEGER;
590   valor4 INTEGER;
591 BEGIN
592 valor1:=-999;
593 valor2:=-999;
594 valor3:=-999;
595 valor4:=-999;
596 FOR tbrow1 IN  SELECT * FROM stop_times
597 LOOP
598   --Passo16.1
599   IF (tbrow1.tempo_chegada = valor1 AND tbrow1.tempo_partida = valor2
        AND tbrow1.stop_id=valor3) OR tbrow1.stop_sequence= valor4  THEN
600     DELETE FROM stop_times WHERE gid=(tbrow1.gid);
601   ELSE
602     --Passo 16.2
603     valor1:=tbrow1.tempo_chegada;
604     valor2:=tbrow1.tempo_partida;
605     valor3:=tbrow1.stop_id;
606     valor4:=tbrow1.stop_sequence;
607   END IF;
608 END LOOP;
609 END;
610 $$ LANGUAGE plpgsql;
```

```
611 SELECT * FROM apagarRepetidosStopTimes();
612 ----------------------------------------------------
613
614 --Passo17----------------------------------------------
615 --conversao do tempo de chegada e partida para o formato hora na tabela
        stop_times
616 Update stop_times
617 SET (arrival_time, departure_time)=(substr(CAST(tempo_chegada /360000.00
        AS varchar),3,2)||':'|| substr(CAST((tempo_chegada %3600)/6000.00 AS
        varchar),3,2) ||':' || substr(CAST(((tempo_chegada %3600)%60)/100.00
        AS varchar),3,2),
618  substr(CAST(tempo_chegada /360000.00 AS varchar),3,2)||':'|| substr(
        CAST((tempo_partida %3600)/6000.00 AS varchar),3,2) ||':' || substr(
        CAST(((tempo_partida %3600)%60)/100.00 AS varchar),3,2));
619
620 --eliminacao dos campos nao necessarios
621 ALTER TABLE stop_times
622 DROP tempo_chegada,
623 DROP tempo_partida,
624 DROP gid;
625 ----------------------------------------------------
```

Listing B.1: Código SQL para transformação da base de dados no formato SIGGESC em GTFS

Apêndice C

Anexos

C.1 Base de dados GTFS

As tabelas e os campos sublinhados representam tabelas e campos obrigatórios.

AGENCY- Este ficheiro possui informação sobre os operadores.

- **agency_id**- código identificador do operador. Este código é único para cada operador.
- **agency_name**- nome completo do operador.
- **agency_url**- URL do site do operador. O URL deve começar por http:// ou https://.
- **agency_timezone**- fuso horário do operador [75].
- **agency_lang**- língua oficial utilizada pela operador.
- **agency_phone**- o número de telefone do operador para prestação de assistência.
- **agency_fare_url**- URL do site do operador para compra de bilhetes.

STOPS- Este ficheiro possui informação sobre as paragens.

- **stop_id**- código identificador da paragem. Este código deve ser único para cada paragem.

- **stop_code**- código que é mostrado ao passageiro para identificar a paragem.
- **stop_name**- nome da paragem.
- **stop_desc**- descrição da paragem.
- **stop_lat**- valor da latitude do ponto de coordenada da paragem no formato WGS 84.
- **stop_lon**- valor da longitude do ponto de coordenada da paragem no formato WGS 84.
- **zone_id**- código identificador da zona onde está inserida a paragem.
- **stop_url**- URL sobre a paragem. O URL deve começar por http:// ou https://.
- **location_type**- indica se a paragem possui estrutura física ou não.
- **parent_station**- identifica a estação de autocarros em que a paragem está inserida.
- **stop_timezone**- fuso horário da localização da paragem.

ROUTES- Este ficheiro possui informação sobre rotas.

- **route_id**- código identificador da rota. Este código deve ser único para cada rota.
- **agency_id**- código identificador da operador. Este código é único para cada operador de transporte público.
- **route_short_name**- pequeno nome identificador da rota. Este nome não indica lugares ou destinos apenas identifica a rota aos passageiros. Ex: 13
- **route_long_name**- nome completo da rota. Ex: Ermesinde (Estação CP) - Maia (centro).
- **route_desc**- descrição da rota.
- **route_type**- identifica o tipo de transporte utilizado na rota. Ex:3-autocarro, 0-metro ligeiro, etc.
- **route_url**- URL do site contendo informação sobre a rota.

- **route_color**- indica a cor da linha usada para traçar a rota num mapa.
- **route_text_color**- indica a cor do texto usado para traçar a rota num mapa.

TRIPS- Este ficheiro possui informação sobres as viagens.

- **route_id**- código identificador da rota. Este código deve ser único para cada rota.
- **service_id**- código identificador da disponibilidade do serviço.
- **trip_id**- código identificador de uma viagem. Este código é único para cada viagem.
- **trip_headsign**- o valor deste campo deve identificar o destino da viagem aos passageiros.
- **trip_short_name**- pequeno nome identificador da viagem.
- **direction_id**- código identificador do sentido da viagem numa determinada rota.
- **block_id**- identifica um bloco de duas ou mais viagens em sequência feitas pelo mesmo transporte.
- **shape_id**- código identificador de um determinado percurso. Este código é único para percurso.

STOP_TIMES- Este ficheiro contém informações sobre as paragens de cada viagem.

- **trip_id**- código identificador da viagem. Este código deve ser único para cada viagem.
- **arrival_time**- hora de chegada à paragem.
- **departure_time**- hora de partida da paragem.
- **stop_id**- código identificador da paragem.
- **stop_sequence**- a ordem de sequência de paragem na viagem.
- **stop_headsign**- permite alterar a informação de destino de uma viagem caso esta mude entre paragens.

- **pickup_type**- código identificador do tipo de recolha dos passageiros numa paragem.
- **drop_off_type**- código identificador do tipo de deixados dos passageiros numa paragem.
- **shape_dist_traveled**- a distancia percorrida desde o inicio da viagem até determinada paragem.

CALENDAR- Este ficheiro contém informações sobre a disponibilidade dos serviços efectudos pelos operadores.

- **service_id**- código identificador de um conjunto de datas em que o serviço está disponível.
- **monday**- valor binário, 0 caso o serviço não esteja disponivel nesse dia e 1 caso esteja.
- **twesday**- valor binário, 0 caso o serviço não esteja disponivel nesse dia e 1 caso esteja.
- **wednesday**- valor binário, 0 caso o serviço não esteja disponivel nesse dia e 1 caso esteja.
- **thursday**- valor binário, 0 caso o serviço não esteja disponivel nesse dia e 1 caso esteja.
- **friday**- valor binário, 0 caso o serviço não esteja disponivel nesse dia e 1 caso esteja.
- **saturday**- valor binário, 0 caso o serviço não esteja disponivel nesse dia e 1 caso esteja.
- **sunday**- valor binário, 0 caso o serviço não esteja disponivel nesse dia e 1 caso esteja.
- **start_date**- contém a data de inicio do serviço.
- **end_date**- contém a data de fim do serviço.

CALENDAR_DATES- Este ficheiro contém informações sobre dias em que são adicionados ou retirados serviços.

- **service_id**- código identificador do serviço.
- **date**- data em que o serviço é diferente do normal.
- **exception_type**- indica se o serviço foi adicionado ou removido.

FARE_ATTRIBUTES- Este ficheiro contém informação sobre as preços das tarifas aplicados nos transportes.

- **fare_id**-código identificador da tarifa. Este código deve ser único para cada tarifa.
- **price**- preço.
- **current_type**- define a moeda usada.
- **payment_method**- código identificador do método de pagamento da tarifa. Pago antes do embarque ou a bordo.
- **transfers**- código identificador do numero de transbordos que a tarifa permite.
- **transfers_duration**- indica o periodo de tempo permitido antes de um transbordo expirar.

FARE_RULES- Este ficheiro contém informações sobre as regras a aplicar sobra as tarifas.

- **fare_id**- código identificador da tarifa. Este código deve ser único para cada tarifa.
- **route_id**- código identificador de uma determinada rota. Este código deve ser único para cada rota.
- **origin_id**- código da zona da paragem de origem da viagem do passageiro.
- **destination_id**- código da zona da paragem de fim da viagem do passageiro.
- **contains_id**- código da zona que o passageiro pode viajar com determinada tarifa.

SHAPES- Este ficheiro contém informação sobre um determinado percurso. Permite traçar linhas em mapas para representar as viagens.

- **shape_id**- código identificador de um determinado percurso. Este código é único para o percurso.
- **shape_pt_lat**- valor da latitude de um ponto de coordenada de um determinado percurso no formato WGS 84.
- **shape_pt_lon**- valor da longitude de um ponto de coordenada de um determinado percurso no formato WGS 84.
- **shape_pt_sequence**- ordem de sequência do ponto de coordenada no percurso.
- **shape_dist_traveled**- distância percorrida no percurso desde o primeiro ponto. Este valor não é o valor em linha recta entre dois pontos mas sim o trajecto efectivo do transporte público.

FREQUENCIES- Este ficheiro contém informação sobre as frequências das viagens.

- **trip_id**- código identificador da viagem. Este código deve ser único para cada viagem.
- **start_time**- hora de inicio das viagens.
- **end_time**- hora de fim das viagens.
- **headway_secs**- indica o tempo entre partidas numa paragem.
- **exact_times**- indica se a frequência das viagens devem-se só basear no tempo do campo headway_secs.

TRANFERS- Este ficheiro contém informação sobre as transferências entre paragens.

- **from_stop_id**- código identificador da paragem entre duas rotas. Paragem onde passageiro sai de um transporte.
- **to_stop_id**- código identificador da paragem entre duas rotas. Paragem onde o passageiro apanha um novo transporte.
- **transfer_type**- contém um código que indica a disponibilidade da transferência entre duas rotas naquela localidade.
- **min_transfer_time**- tempo disponível para ligação entre duas rotas.

FEED_INFO- Este ficheiro contém informação sobre quem publica a informação GTFS.

- **feed_publisher_name**- nome da organização que publica a base de dados GTFS.
- **feed_publisher_url**- URL da organização que publica a base de dados GTFS. O URL deve começar por http:// ou https://.
- **feed_lang**- código que indica que língua é utilizada a base de dados GTFS.
- **feed_start_date feed_end_date**- data da publicação ou data de fim da publicação da base de dados GTFS.
- **feed_version**- indica a versão dos dados.

C.2 Base de dados do SIGGESC

Figura C.1: Base de dados alfanumérica do SIGGESC

Referências

[1] TriMet Bibiana McHugh. The OpenTripPlanner project. In *The OpenTripPlanner Project*, pages 12–16. Agosto 2011. Acedido em Agosto de 2012.

[2] Brasgeo. ArcIMS - Soluções em Geoinformação. `http://www.brasgeo.com.br/novo/servidores-gis/arcims`. Acedido em Agosto de 2012.

[3] DotNetNuke Corp. DotNetNuke community edition CMS. `http://www.dotnetnuke.com/Products/Community-Edition.aspx`. Acedido em Agosto de 2012.

[4] Fernando Costa. SIGGESC - Uma experiência útil. `http://www.imtt.pt/sites/IMTT/Portugues/Noticias/Documents/SIGGESC%2011%20Maio%202011%20-%20PDFs/SIGGESC2011%20-%20Fernando_Costa_VIMECA.pdf`. Acedido em Janeiro de 2012.

[5] Instituto Superior de Agronomia. O Datum Lisboa. `http://www.isa.utl.pt/dm/sig/sig20002001/TemaSGR/datum_lisboa.htm`. Acedido em Agosto de 2012.

[6] Instituto de Mobilidade Terrestre. `http://www.imtt.pt`.

[7] Liferay Enterprise. Liferay. `http://www.liferay.com/`. Acedido em Agosto de 2012.

[8] ESRI. Esri portugal. `http://www.esriportugal.pt/`. Acedido em Agosto de 2012.

[9] ESRI. Arcgis 9 what is ArcGis? In *Arcgis 9 What is ArcGis?*, pages 19–24. 2001. Acedido em Agosto de 2012.

[10] Open Source Geospatial Foundation. OpenLayers: home. `http://openlayers.org/`. Acedido em agosto de 2012.

[11] Open Source Geospatial Foundation. PostGIS. `http://postgis.refractions.net/`. Acedido em Agosto de 2012.

[12] OpenStreetMap Foundation. OpenStreetMap. `http://www.openstreetmap.org/`. Acedido em Agosto 2012.

[13] The Eclipse Foundation. Eclipse. `http://www.eclipse.org/`. Acedido em Agosto de 2012.

[14] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In Catherine McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 319—333. Springer Berlin / Heidelberg, 2008.

[15] GISMÉDIA. GISMÉDIA 2012. `http://www.gismedia.pt/`. Acedido em Agosto de 2012.

[16] GNU. The GNU general public license v3.0 - GNU project - free software foundation FSF. `http://www.gnu.org/copyleft/gpl.html`. Acedido em Agosto de 2012.

[17] Google. Example GTFS feed. `https://developers.google.com/transit/gtfs/examples/gtfs-feed`. Acedido em Fevereiro de 2012.

[18] Google. General transit feed specification reference - transit - google developers. `https://developers.google.com/transit/gtfs/reference`. Acedido em Fevereiro de 2012.

[19] Google. Google analytics. `http://www.google.com/analytics/`. Acedido em Agosto de 2012.

[20] Google. GTFS data exchange. `http://www.gtfs-data-exchange.com/`. Acedido em Fevereiro de 2012.

[21] Google. Tools - transit - google developers. `https://developers.google.com/transit/tools`. Acedido em Janeiro de 2012.

[22] Google. Understanding GTFS concepts - transit partners help. `https://support.google.com/transitpartners/bin/answer.py?hl=en&answer=1106431`. Acedido em Fevereiro de 2012.

[23] Google. What is GTFS? - transit - google developers. `https://developers.google.com/transit/gtfs/`. Acedido em Agosto 2012.

[24] PostgreSQL Global Development Group. PostgreSQL:. `http://www.postgresql.org/`. Acedido em Agosto de 2012.

[25] PostgreSQL Global Development Group. PostgreSQL: documentation: 7.4: Pattern matching. `http://www.postgresql.org/docs/7.4/static/functions-matching.html`. Acedido em Agosto de 2012.

[26] Geoweb Guru. WMS tile caches. `http://www.geowebguru.com/articles/152-wms-tile-caches`. Acedido em Agosto de 2012.

[27] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael. Correction to a formal basis for the heuristic determination of minimum cost paths. *ACM SIGART Bulletin*, (37):28–29, 1972.

[29] IBM. IBM - DB2 spatial extender. `http://www-01.ibm.com/software/data/spatial/db2spatial/features.html`. Acedido em Agosto de 2012.

[30] IBM. IBM - informix spatial DataBlade - software. `http://www-01.ibm.com/software/data/informix/blades/spatial/`. Acedido em Agosto de 2012.

[31] Imediata. Imediata - Sistemas Multimédia, SA. `http://www.imediata.pt`. Acedido em Agosto de 2012.

[32] Instituto Geográfico Português. Perguntas mais frequentes. `http://www.igeo.pt/perguntas.htm#1`. Acedido em Agosto de 2012.

[33] MetaCarta Labs. TileCache, from MetaCarta labs. `http://tilecache.org/`. Acedido em Agosto de 2012.

[34] Metro de Lisboa. Metro de lisboa. obras em curso. `http://www.metrolisboa.pt/obras/obras-em-curso/`. Acedido em Agosto de 2012.

[35] Metro de Lisboa. Metro de lisboa. um pouco de história. `http://www.metrolisboa.pt/empresa/um-pouco-de-historia/`. Acedido em Agosto de 2012.

[36] Metro do Porto, SA. Metro do porto. a vida em movimento. `http://www.metrodoporto.pt/PageGen.aspx?WMCM_PaginaId=23939`. Acedido em Agosto de 2012.

[37] Microsoft. Microsoft SQL server. `http://www.microsoft.com/sqlserver`. Acedido em Janeiro 2012.

[38] Gábor Fazekas Nóra Sterbinszky. Comparison of the efficiency of combination of database servers, application servers and operating systems with the TPC-W benchmark. *Proceedings of the 8th International Conference on Applied Informatics Eger, Hungary*, 1:361–370, jan 2010. Acedido em Setembro de 2012.

[39] Oil and Gas Producers. Geomatics committee. `http://www.epsg.org/`. Acedido em Agosto de 2012.

[40] Open Geospatial Consortium. Simple feature access. `http://www.opengeospatial.org/standards/sfs`. Acedido em Agosto de 2012.

[41] OpenGeo. OpenGeo : GeoServer. `http://opengeo.org/technology/geoserver/`. Acedido em Agosto de 2012.

[42] OpenLans. OpenPlans | helping cities work better. `http://openplans.org/`. Acedido em Agosto de 2012.

[43] OpenPlans. GeoExt. `http://geoext.org/`. Acedido em Agosto de 2012.

[44] OpenPlans. GeoServer. `http://geoserver.org`. Acedido em Agosto de 2012.

[45] OpenPlans. OpenGeo : OpenLayers. `http://opengeo.org/technology/openlayers/`. Acedido em agosto de 2012.

[46] OpenPlans. OpenGeo : Spatial database tips and tricks : Introduction. `http://workshops.opengeo.org/postgis-spatialdbtips/introduction.html`. Acedido em Janeiro de 2012.

[47] OpenPlans. Spatial database tips and tricks. `http://workshops.opengeo.org/postgis-spatialdbtips/introduction.html`. Acedido em Agosto de 2012.

[48] Oracle. MySQL-Spatial. `http://dev.mysql.com/doc/refman/5.1/en/spatial-extensions.html`. Acedido em Agosto de 2012.

[49] Oracle. NetBeans. `http://netbeans.org/`. Acedido em Agosto de 2012.

[50] Oracle. Oracle spatial and graph - spatial features. `http://www.oracle.com/technetwork/database/options/spatial/index.html`. Acedido em Agosto 2012.

[51] Oracle. Portlet. http://developers.sun.com/portalserver/reference/techart/jsr168/. Acedido em Agosto de 2012.

[52] Oracle. Portlet. http://today.java.net/pub/a/today/2009/01/20/jsr-286-portlet-irrelevance.html. Acedido em Agosto de 2012.

[53] Oracle. Portlet. http://developers.sun.com/portalserver/reference/techart/jsr286/jsr286.html. Acedido em Agosto de 2012.

[54] Marco Painho. Sistema de Informacao Geográfica de Gestão de Carreiras - SIGGESC: a transiçao para web. http://www.imtt.pt/sites/IMTT/Portugues/Noticias/Documents/SIGGES%2011%20Maio%202011%20-%20PDFs/SIGGESC2011%20-%20Marco_Painho_ISEGI_UNL.pdf. Acedido em Janeiro de 2012.

[55] Patternizando. Portlet-introdução à tecnologia portlet. http://www.patternizando.com.br/2011/03/introducao-a-tecnologia-portlet/. Acedido em Agosto de 2012.

[56] Esri Portugal. Esri portugal :: Instituto da mobilidade e dos transportes terrestres. http://www.esriportugal.pt/eventos/9-encontro-de-utilizadores-esri-portugal/eue2011/resumo/imtt-isegi/.

[57] Instituto Geografico Português. Datum 73. http://www.igeo.pt/produtos/Geodesia/inf_tecnica/sistemas_referencia/Datum_73.htm. Agosto de 2012.

[58] Instituto Geografico Português. Datum lisboa. http://www.igeo.pt/produtos/Geodesia/inf_tecnica/sistemas_referencia/Datum_Lx.htm. Acedido em Agosto de 2012.

[59] Matthew Roth. How google and portland's TriMet set the standard for open transit data | streetsblog san francisco. http://sf.streetsblog.org/2010/01/05/how-google-and-portlands-trimet-set-the-standard-for-open-transit-data/. Acedido em Janeiro de 2012.

[60] Isabel Seabra. SIGGESC- Sistemas de Informação Geografica de Gestão de carreiras. http://www.imtt.pt/sites/IMTT/Portugues/Noticias/Documents/SIGGESC%2011%20Maio%202011%20-%20PDFs/SIGGESC2011%20-%20Isabel_Seabra_IMTT.pdf. Acedido em Janeiro de 2012.

[61] Sencha. Web application development with sencha ext JS framework | sencha ext JS | products | sencha. http://www.sencha.com/products/extjs. Acedido em Agosto de 2012.

[62] SIG2000. Itinerarium - STCP. http://www.sig2000.net/uploads/itinerarium-sistemasinformacaogeografica.pdf. Acedido em Agosto de 2012.

[63] SIG2000. Produtos SIG2000 ArcGIS. http://www.sig2000.net/productdetails.asp?id=5#ArcGIS. Acedido em Agosto de 2012.

[64] SIG2000. Produtos SIG2000 Networking Engine. http://www.sig2000.net/productdetails.asp?id=4. Acedido em Agosto de 2012.

[65] SIG2000. Sistemas de Informação Geográfica SIG2000. http://www.sig2000.net/. Acedido em Agosto de 2012.

[66] STCP. Itinerarium. http://www.itinerarium.net. Acedido em Agosto 2012.

[67] National Geodetic Survey. datum1. http://www.ngs.noaa.gov/faq.shtml#WhatDatum. Acedido em Agosto de 2012.

[68] National Geodetic Survey. National geodetic survey - frequently asked questions. http://www.ngs.noaa.gov/faq.shtml#WGS84. Acedido em Agosto de 2012.

[69] Transporlis. http://www.transporlis.sapo.pt/Default.aspx?tabid=36. Acedido em Agosto de 2012.

[70] TriMet. Home openplans/OpenTripPlanner wiki · GitHub. https://github.com/openplans/OpenTripPlanner/wiki/. Acedido em Agosto de 2012.

[71] TriMet. OpenTripPlanner. http://opentripplanner.com/. Acedido em Agosto de 2012.

[72] TriMet. TriMet interactive map. http://rtp.trimet.org/#/. Acedido em Agosto de 2012.

[73] TriMet. TriMet: public transit in the portland area. http://trimet.org/. Acedido em Agosto de 2012.

[74] Wikipedia. Comparison of integrated development environments - wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments. Acedido em Agosto 2012.

[75] Wikipedia. List of tz database time zones - wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/List_of_tz_zones`. Acedido em Agosto de 2012.