SIAT - Sistema de Informações e Alertas de Trânsito

Diogo Filipe Dias Silva

Relatório do projecto realizado no âmbito do curso de Mestrado em Engenharia Informática e de Computadores sob orientação do Eng. Fernando Miguel Santos Lopes de Carvalho.

Setembro, 2011

Esta página foi intencionalmente deixada em branco.

SIAT - Sistema de Informações e Alertas de Trânsito

Setembro de 2011

Relatório do projecto realizado no âmbito do curso de Mestrado em Engenharia Informática e de Computadores

AUTOR:

Diogo Filipe Dias Silva, nr. 30700
diogofdsilva@gmail.com

ORIENTADOR:

Fernando Miguel Santos Lopes de Carvalho
mcarvalho@cc.isel.ipl.pt

Esta página foi intencionalmente deixada em branco.

Resumo

A evolução tecnológica e das sociedades permitiu que, hoje em dia, uma boa parte da população tenha acesso a dispositivos móveis com funcionalidades avançadas. Com este tipo de dispositivos, temos acesso a inúmeras fontes de informação em tempo-real, mas esta característica ainda não é, hoje em dia, aproveitada na sua totalidade.

Este projecto tenta tirar partido desta realidade para, utilizando os diversos dispositivos móveis, criar uma rede de troca de informações de trânsito. O utilizador apenas necessita de servir-se do seu dispositivo móvel para, automaticamente, obter as mais recentes informações de trânsito enquanto, paralelamente, partilha com os outros utilizadores a sua informação. Apesar de existirem outras alternativas no mercado, com soluções que permitem usufruir do mesmo tipo de funcionalidades, nenhuma utiliza este tipo de dispositivos (*GPS's* convencionais, por exemplo).

Um dos requisitos necessário na implementação deste projecto é uma solução de *geocoding*. Após terem sido testadas várias soluções, nenhuma cumpria, na totalidade, os requisitos deste projecto, o que originou o desenvolvimento de uma nova solução que cumpre esses requisitos.

A solução é, toda ela, muito modular, formada por vários componentes, cada um com responsabilidades bem identificadas. A arquitectura desta solução baseia-se nos padrões de desenvolvimento de uma *Service Oriented Architecture*. Todos os componentes disponibilizam as suas operações através de *web services*, e a sua descoberta recorre ao protocolo *WS-Discovery*. Estes vários componentes podem ser divididos em duas categorias: os do núcleo, responsáveis por criar e oferecer as funcionalidades requisitadas neste projecto e os módulos externos, nos quais se incluem as aplicações que apresentam as funcionalidades ao utilizador.

Foram criadas duas formas de consumir a informação oferecida pelo serviço SIAT: a aplicação móvel e um *website*. No âmbito dos dispositivos móveis, foi desenvolvida uma aplicação para o sistema operativo *Windows Phone 7*.

Palavras-chave Trânsito, rede de partilha, dispositivos móveis, SOA, WS-Discovery, Web services, WP7.

Esta página foi intencionalmente deixada em branco.

Abstract

Technological progress and the evolution of our societes allowed that, today, a good part of the population has access to mobile devices with advanced features. With such devices, we have access to numerous information sources in real time, however, this feature is not fully explored.

This project tries to take advantage of this fact using various mobile devices, to create a network for exchanging traffic information. The user only needs to make use of his mobile device, and will automatically get the latest traffic information while, at the same time, shares your information with other users. Although there are other alternatives on the market with solutions that allow you to enjoy the same functionality, none uses this type of devices.

One of the requirements needed in the implementation of this project is a solution of geocoding. After being tested several solutions, none met in full, the requirements of this project, which led to the development of a new solution, that meets these requirements.

The solution is very modular, composed by several components, each with clearly identified responsibilities. The architecture of this solution is based on Service Oriented Architecture design pattern. All components provide their operations through web services, and their discovery uses the protocol WS-Discovery. These components can be divided into two categories: core, responsible for creating and offering the required features in this project and the external modules, which include applications that provide the functionalities to the user.

We created two ways to consume the information offered by SIAT service: a mobile application and a website. To the mobile devices, was developed an application for the Windows Phone 7 operating system.

Keywords Traffic, sharing network, mobile devices, SOA, WS-Discovery, Web services, WP7.

Esta página foi intencionalmente deixada em branco.

Agradecimentos

Quero agradecer a:

Aos meus pais, pela paciência, tanto nas alturas em que estava presente com mau humor como quando não estava.

Joana, também pela paciência, pela minha ausência e pela revisão do relatório, mas sobretudo por todo o amor e carinho, indispensáveis nos melhores e piores momentos.

Ao Prof. Miguel Carvalho, pela preciosa orientação e acompanhamento.

Ao resto da família, em particular à minha Tia Bela, pela revisão deste relatório.

A todos os amigos, que me fizeram rir quando mais precisei.

Esta página foi intencionalmente deixada em branco.

Conteúdo

Lista de Figuras

Esta página foi intencionalmente deixada em branco.

Listagens

Esta página foi intencionalmente deixada em branco.

1 Introdução

O presente documento representa o relatório do projecto de mestrado Sistema de Informações e Alertas de Trânsito (SIAT). Este projecto tira partido da tecnologia actual dos dispositivos móveis (incluindo todos os dispositivos com antena móvel, como sejam, *PDA*, *smartphones*, *tablets*, entre outros), promovendo uma maior mobilidade aos automobilistas, que utilizando um dispositivo móvel poderão ter acesso a informações de trânsito, em tempo real. Este objectivo é conseguido através da partilha de informação entre todos os automobilistas.

Se todos os telemóveis com *GPS* contribuírem com a sua informação, é possível criar uma rede dinâmica e em tempo real de troca de informações de trânsito entre estes terminais. Além da troca de informações, estes poderão, igualmente, ser úteis na emissão de alertas de acidentes rodoviários.

1.1 Motivação

Este projecto emerge da necessidade de promover a mobilidade, informando os condutores acerca de possíveis acidentes ou afluxos anormais de trânsito. Com esse objectivo, pretende-se utilizar um dispositivo móvel para criar uma rede de troca de informações de trânsito.

O telemóvel é um equipamento amplamente utilizado pela população portuguesa. Os estudos da ANACOM indicam que existem cerca de 15,87 milhões de telemóveis [1]. Dentro destas estações, existe uma tendência, [2] de mercado, para um crescente número de estações móveis que possuem funcionalidades avançadas, os chamados *smartphones*. Dentro destas funções avançadas, inclui-se a funcionalidade de *Global Positioning System* (vulgarmente chamado de GPS) que permite saber a posição exacta do dispositivo.

Se existir uma estação móvel com GPS dentro de um carro em marcha, é possível obterem-se informações acerca da posição e movimento desse veiculo, nomeadamente a sua velocidade actual. Estes dados podem ser analisados e extrapolados, de modo a obter informações de trânsito em tempo-real. Os restantes automobilistas que circulam no mesmo itinerário, ao terem acesso a este tipo de

informações, poderiam, caso desejassem, optar por outras vias alternativas.

1.2 Estado da arte

A ideia subjacente neste projecto não é totalmente inovadora, existindo no mercado produtos com funcionalidades semelhantes. Na sua essência têm o mesmo objectivo, ou seja, a utilização de dispositivos móveis que estão, habitualmente, presentes nos veículos de hoje, para prever o trânsito. Dentro dos produtos analisados encontra-se o *TomTom HD Traffic*[3] e o *CellInt TrafficSense*[4].

1.2.1 TomTom HD Traffic

A marca *TomTom* é prestigiada no fabrico e comercialização de *GPS's* para navegação terrestre. Além da funcionalidade mais básica de navegação, a marca acrescenta funcionalidades a esses dispositivos pela via do *software* e de serviços *online*. Dentro destes serviços encontra-se o *TomTom HD Traffic*.

Estes dispositivos obtêm informação geográfica (localização e velocidade actual) e enviam-na para uma unidade central de processamento dessa informação, a qual é gerida pela empresa. Como a tecnologia usada nos GPS não permite enviar dados, quem utiliza este serviço está comprometido com uma subscrição de um serviço móvel GSM/GPRS de uma operadora.

Para além da informação recolhida por cada um destes dispositivos, este serviço procura obter outros meios de informação, como por exemplo, dados provenientes de entidades oficiais. Toda a informação recolhida é processada numa unidade central, onde é filtrada e validada.

1.2.2 CellInt TrafficSense

A *CellInt* é uma empresa especializada em fornecer informações de tráfego a partir da recolha de dados provenientes de antenas móveis. O seu produto principal é o *Traffic Sense*, um serviço capaz de monitorizar tráfego automóvel e dar informações de acidentes em tempo real.

Segundo a documentação fornecida acerca deste produto, o seu funcionamento consiste em analisar o movimento das antenas móveis de pessoas anónimas. O resultado destas medições é extrapolado para gerar previsões de tráfego, com base nos dados históricos.

Este serviço não usa a técnica de triangulação, com base na posição das antenas, por não ser suficientemente precisa. O processo de funcionamento deste serviço consiste em percorrer as estradas (onde se deseja suportar este serviço) *à priori*, registando a intensidade do sinal das diferentes antenas disponíveis e associando esses valores a coordenadas geográficas. A posterior localização de uma antena móvel é efectuada através do processo inverso, ou seja, de acordo com os níveis de intensidade das diferentes antenas encontra-se a posição exacta.

1.3 Objectivos gerais

A solução proposta por este projecto é composta por uma unidade de análise de dados centralizada, denominada ***SIAT Core***. Esta unidade é abastecida por dados de diferentes fontes, os quais são tratados de modo a construir as informações de trânsito que posteriormente são transmitidas aos consumidores interessados. A Figura 1.1 consiste numa descrição visual do processo desenvolvido no ***SIAT Core***.

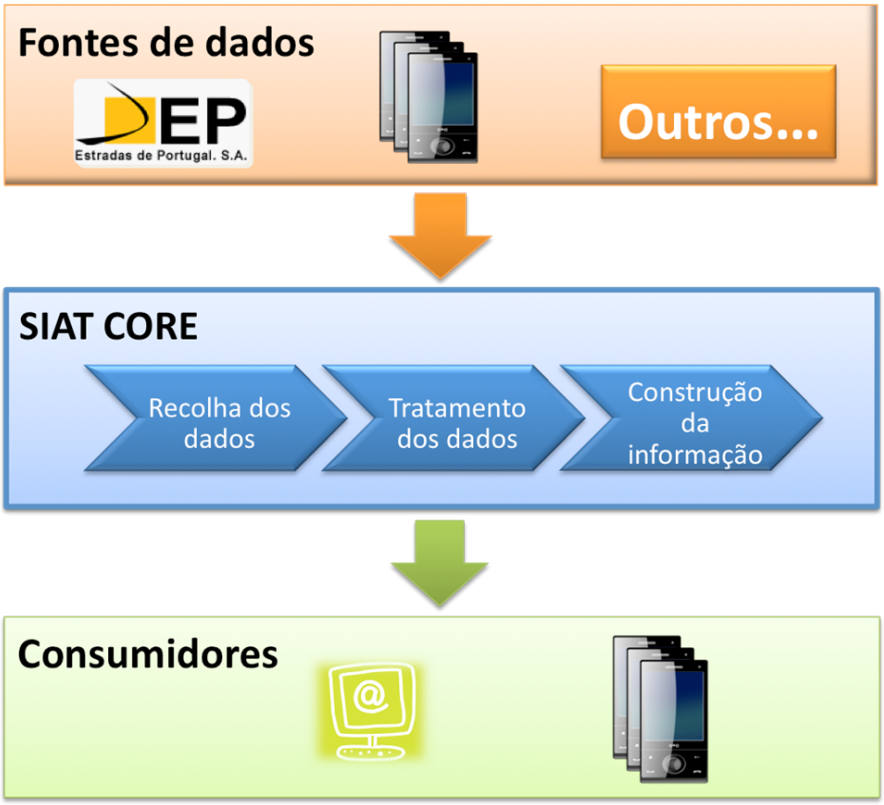

Figura 1.1: SIAT - Processos de funcionamento do Core.

A principal fonte de dados são os dispositivos móveis individuais. Esses dispositivos devem possuir tecnologia que lhes permita obter a sua localização geográfica (GPS ou AGPS, por exemplo) e conseguir estabelecer uma ligação à *internet*, de forma a enviar os dados recolhidos.

Outras fontes de dados provêm de entidades externas que já possuam ferramentas para fornecer informações de trânsito, como por exemplo, as Estradas de Portugal.

A unidade de análise de dados processa os dados provenientes das diversas fontes, validando e transformando esses dados em informação útil para o utilizador final, o que neste caso se traduz em atribuir níveis de intensidade de tráfego para as diferentes estradas.

Durante este processo, é necessário obter informações geográficas (*geocoding*) com bastante frequência, como tal, a solução utilizada para obter este tipo de informações deve de, para além de cumprir os requisitos deste projecto, oferecer o melhor desempenho possível.

Existem diversos consumidores desta informação. O automobilista poderá ser notificado no telemóvel sobre as informações de trânsito, ou então, se quiser poderá realizar uma consulta a um *Website* que apresente essa informação.

1.4 Organização do Documento

1.4.1 Estrutura

Este documento é composto por seis capítulos. De seguida é apresentada uma lista com os vários capítulos e uma breve descrição do seu conteúdo.

Capitulo 2 - Enquadramento Contém um breve enquadramento, sobre as soluções existentes em algumas das tecnologias utilizadas neste projecto.

Capitulo 3 - Solução de Geocoding Solução desenvolvida para suprir as necessidades geográficas deste projecto.

Capitulo 4 - Arquitectura global Arquitectura global da solução.

Capitulo 5 - Implementação Apresenta uma descrição da implementação desenvolvida.

Capitulo 6 - Conclusões Conclusões retiradas do desenvolvimento desta solução, bem como uma série de tópicos que apresentam algum do trabalho futuro.

1.4.2 Convenções

No decorrer deste documento, todos os nomes de classes estão a **negrito**, sendo que as classes desenvolvidas no âmbito deste projecto estão também a ***negrito e itálico***. Também nos nomes dos componentes do sistema é usado o ***negrito e itálico***.

No texto que se segue, foram utilizados os seguintes acrónimos:

Dispositivo Móvel Pode ser *Smartphones*, *Tablets*, *PDA's* ou até mesmo um simples portátil.

WP7 *Windows Phone 7* - Sistema operativo para dispositivos móveis desenvolvido pela Microsoft

WCF *Windows Communication Foundation*

GPS *Global Positioning System*

CRUD *Create, Read, Update, Delete* - as quatro operações de dados mais básicas.

SOA *Service-oriented Architecture*

WSDL *Web Services Description Language*

SOAP *Simple Object Access Protocol*

JSON *JavaScript Object Notation*

OASIS *Organization for the Advancement of Structured Information Standards*

OGC *Open Geospatial Consortium*

DAL *Data Access Layer*

UML *Unified Modeling Language*

Esta página foi intencionalmente deixada em branco.

2 Enquadramento

Este projecto é composto por várias interfaces com o utilizador, as quais utilizam o sistema central de análise de dados, tanto para fornecer *input* como para obter o seu *output*. Uma das interfaces será um dispositivo móvel capaz de obter e enviar as suas informações geográficas através da *internet*. O serviço terá de, com essas informações geográficas, identificar qual a estrada de onde estes dados provêm, atribuindo um nível de intensidade de trânsito.

A concepção deste projecto coloca dois desafios essenciais. O primeiro é escolher a melhor plataforma móvel (sistema operativo), para dar suporte às funcionalidades necessárias e o segundo é utilizar um serviço que permita obter a cartografia corrente (mapas). Tendo em conta o número de soluções disponíveis para ambos os casos, torna-se importante fazer uma análise cuidada dos produtos para escolher os mais adequados à utilização dada por este projecto.

2.1 Dispositivos móveis

O mercado dos dispositivos móveis encontra-se numa expansão extremamente competitiva, com a existência de vários sistemas operativos, provenientes de diversos fornecedores. Nesta secção, faz-se uma análise comparativa das diferentes plataformas, o que permite optar pela mais adequada para o desenvolvimento da aplicação móvel deste projecto.

2.1.1 Windows Phone 7

O *Windows Phone 7* é o mais recente sistema operativo para dispositivos móveis desenvolvido pela Microsoft. Este sistema vem substituir o anterior *Windows Mobile*, não existindo qualquer tipo de compatibilidade entre estes dois sistemas operativos.

A Microsoft definiu requisitos de hardware exigentes, mas que permitem aos programadores terem uma base estável de desenvolvimento.

Para desenvolver aplicações basta descarregar gratuitamente as ferramentas do *Windows Phone*. Estas podem ser obtidas através do site da *Microsoft* e incluem uma versão base do Visual Studio 2010, o emulador e as *frameworks* necessárias. Só existem dois modos de desenvolvimento de aplicações para o *Windows Phone 7*: *Silverlight* e *XNA Game Studio*. O primeiro é indicado para o desenvolvimento de aplicações, enquanto o segundo é especializado no desenvolvimento de jogos. Ambas as plataformas assentam sobre a *.Net framework*, como tal, o desenvolvimento pode ser feito numa das várias linguagens compatíveis com esta *framework*. Actualmente, não existem bibliotecas nativas para desenvolvimento de aplicações.

A única forma de descarregar e utilizar aplicações externas é através do *Windows Phone Marketplace*. As aplicações são empacotadas num ficheiro do tipo *XAP* e submetidas a aprovação. Esta é bastante rigorosa impedindo a aplicação de interferir com o bom funcionamento do telemóvel. Após a validação da aplicação, esta fica disponível para ser descarregada e utilizada por outros utilizadores.

2.1.2 Android

O sistema operativo *Android*, tem como base o *kernel Linux* e é desenvolvido pela empresa *Google* com o apoio da *Open Handset Alliance* [5].

O desenvolvimento de aplicações para este sistema operativo é realizado na linguagem de programação Java, sendo também possível desenvolver bibliotecas em código nativo que, posteriormente, são integradas nas aplicações desenvolvidas.

As aplicações de sistema executam sob uma plataforma orientada a objectos, que contém bibliotecas de sistema, fornecendo aos programadores acesso às diferentes funcionalidades do mesmo. O código é executado numa máquina virtual criada para este sistema operativo, denominada *Dalvik* [6]. As aplicações vêm empacotadas no formato *Dalvik Executable* (.dex), tendo sido idealizadas para sistemas com recursos limitados (como memória e velocidade de processamento). As aplicações são distribuídas num formato compactado (.apk), onde se encontram os ficheiros binários e respectivos recursos.

2.1.3 iOS

O *iOS* é o sistema operativo da *Apple* para dispositivos móveis e tem como base o *kernel* do *MAC OSX*. Alguns dos dispositivos que utilizam este sistema operativo são o iPhone, iPad e iPod Touch.

O iPhone veio revolucionar a forma como são encarados os dispositivos móveis por ser extremamente *user-friendly*. O sistema operativo é um dos grandes responsáveis desta revolução, uma vez que a sua implementação teve como objectivo proporcionar ao utilizador a experiência de ter a informação na ponta dos dedos, descartando a utilização da típica caneta que acompanhava os antigos PDA's.

Tal como as outras plataformas, esta também disponibiliza um *Marketplace* e as aplicações só podem ser descarregadas por esta via. A inserção de novas aplicações implica a sua validação por

parte da *Apple*. O desenvolvimento é feito na linguagem de programação típica nos produtos *Apple*, o *Objective-C*, mas as ferramentas de desenvolvimento são disponibilizadas exclusivamente para a plataforma *MAC OSX*.

2.2 Serviço de Mapas

Existem vários serviços que disponibilizam cartografia *online*. Neste projecto, é necessário um serviço deste tipo, como tal, irá ser feita uma análise das diferentes soluções, escolhendo a mais adequada.

Um dos requisitos que importa avaliar neste tipo de serviços é a quantidade e qualidade das funcionalidades que disponibilizam. De entre as funcionalidades habitualmente oferecidas encontram-se, por exemplo:

- Mapas - a partir de uma determinada posição, devolve uma imagem com o mapa dessa zona.
- *Geocoding* - é o processo de conversão de endereços (como "Av Brasil, Lisboa") em coordenadas geográficas (como latitude e longitude)
- *Reverse Geocoding* - processo inverso ao *Geocoding*, ou seja, a partir de uma coordenada geográfica, devolve dados geográficos (usualmente o endereço).
- Rotas - Devolve um possível trajecto entre dois pontos geográficos.

Para a resolução deste projecto a funcionalidade mais importante é a de *Reverse Geocoding*, como tal, será a que vai ser mais detalhadamente analisada.

2.2.1 Google Maps

O *Google Maps* é o serviço de visualização de mapas *online* desenvolvido pela *Google* [7]. Este serviço pode ser consultado na página *web* ou pode ser consumido remotamente com o recurso a uma API, das quais destacam-se a API *JavaScript* e a API *REST*.

Nestas API's estão disponíveis mecanismos que permitem consumir todas as funcionalidades previamente enumeradas (mapas, *Geocoding*, *Reverse Geocoding* e rotas).

2.2.2 Microsoft Bing Maps

O *Bing Maps* [8] é um serviço idêntico ao *Google Maps*, mas desenvolvido pela Microsoft. Suporta todas as funcionalidades previamente enumeradas e disponibiliza vários métodos de consumir a informação. Para além das habituais API's *REST* e *JavaScript*, este serviço acrescenta um *WebService SOAP* e um controlo *Silverlight*.

2.2.3 Sapo Mapas

O Sapo Mapas API é uma biblioteca que permite incluir os mapas do Sapo usando *Javascript*. Esta biblioteca disponibiliza um conjunto de utilidades e serviços que manipulam os mapas do Sapo [9]. O Sapo Mapas tem uma API bem documentada, oferecendo as funcionalidades de Mapas, *Geocoding* e Rotas. Segundo a documentação, não existe serviço de *Reverse Geocoding* disponível, o que inviabiliza a sua utilização neste projecto. Outra condicionante é o facto da API ser apenas *Javascript*, condicionando o método de desenvolvimento a ser utilizado.

2.2.4 Open Street Maps

O OpenStreetMap é um projecto aberto à comunidade, que tem o objectivo de disponibilizar dados geográficos livres e gratuitos[10]. A partir desta fonte de dados, foram criados vários projectos (a maioria *open source*) para disponibilizar as funcionalidades típicas neste tipo de serviços. O projecto Mapnik [11] disponibiliza o serviço de Mapas e o Nominatim [12] disponibiliza as funcionalidades de *GeoCoding* e *Reverse Geocoding*. Existem vários projectos que disponibilizam a funcionalidade de Rotas.

Como os dados são públicos e os projectos são *open source*, eles podem ser descarregados facilmente, dando a oportunidade de criar um servidor local.

Sendo este um projecto de cariz comunitário, onde os dados são recolhidos voluntariamente, existem, infelizmente, zonas do país que não se encontram bem cartografadas, no entanto, estes problemas acontecem tipicamente em zonas rurais e em estradas municipais.

Nominatim

O *Nominatim* é um projecto que pode ser dividido em duas componentes distintas. A primeira é uma base de dados com o nome *Gazeteer* alojada num servidor *Postgres* com o *plug-in Postgis* . A segunda componente é uma API *Rest-like* construída em PHP, a qual contém grande parte da lógica de negócio desta aplicação e através da qual é possível aceder aos dados.

O processo de instalação e configuração do *Nominatim* disponibilizado está totalmente vocacionado para a sua utilização em ambientes Linux. Este requisito aumenta a heterogeneidade do ambiente de desenvolvimento, uma vez que o resto da solução está a ser implementado em tecnologias *Microsoft*. Acresce ainda o menor domínio demonstrado tanto em ambientes Linux, como em bases de dados *Postgres*, o que cria um factor de ineficiência e de complexidade acrescida.

2.2.5 Comparação e escolha

Dos sistemas estudados, conclui-se que o *Google Maps* e o *Bing Maps* não permitem cumprir (todos) os requisitos deste projecto. Esta conclusão deve-se, sobretudo, à falta de coerência e precisão

demonstrada nos testes realizados. Como se tratam de sistemas comerciais não disponibilizam acesso directo aos seus dados internos, nomeadamente à informação geográfica. Assim, torna-se impraticável manipular ou transformar os dados destes sistemas para que sejam utilizáveis neste projecto.

Como meio de contornar esta dificuldade surge o *Open Street Maps* (OSM). A principal vantagem deste sistema é a possibilidade de transformar os dados geográficos, e apesar de existirem zonas do país que não se encontram bem cartografadas, este problema não têm relevância para este projecto, porque o mais importante é ter a principal rede viária cartografada.

A utilização do projecto *Nominatim* permite tirar partido dos dados geográficos reutilizando uma solução já existente. No entanto, verificou-se que este não permite cumprir o requisito de identificar o sentido actual de um condutor numa estrada, e a única solução seria estudar e alterar o seu código. Fazer alterações em código desenvolvido em tecnologias que são desconhecidas seria altamente contraproducente, pelo que optou-se por outra abordagem que será apresentada no próximo capítulo.

Esta página foi intencionalmente deixada em branco.

3 Solução de *Geocoding*

O estudo efectuado às plataformas de mapas disponíveis permitiu concluir que nenhuma cumpre, na totalidade, os requisitos do projecto. Por este motivo, foi desenvolvida uma solução que permite resolver todos os problemas encontrados até este ponto. Esta solução tem o nome de ***OSM Querys*** e é inspirada no *Nominatim*, mas focalizada no cumprimento dos requisitos necessários, nomeadamente:

- *Reverse geocoding* que permita identificar a estrada actual;
- Capacidade de identificar vários pontos do percurso recente de um condutor;
- Capacidade de identificar o sentido do automóvel.

Um dos problemas encontrado nas soluções estudadas é a diversidade de tecnologias utilizada no desenvolvimento, o que causa problemas de heterogeneidade. Para evitar esse tipo de complicações, esta solução foi desenvolvida para o sistema operativo *Windows*. Foi utilizada uma base de dados *SQL Server 2008*, pois é a primeira versão deste servidor com suporte para funções geográficas, e o resto da solução foi desenvolvida em *.Net*. Desta forma será mais fácil integrar esta solução no resto do projecto.

Um dos objectivos, tido em conta, no desenvolvimento desta solução é que esta deve ser totalmente independente do resto do projecto. Por essa razão, foram construídas ferramentas autónomas e um guia de instalação das mesmas que pode ser utilizado, posteriormente, em qualquer máquina.

A solução utiliza os dados do *OSM* com um modelo de dados próprio. Para criar esse modelo de dados são disponibilizados uma série de *scripts SQL* que devem ser executados numa determinada ordem, a qual é descrita no manual de instalação. Esta solução inclui também uma ferramenta (denominada ***OSMtoDB***) que carrega os dados do ficheiro *OSM* para a base de dados e, por fim, um serviço que permite aceder a esses dados.

Para especificar o contrato de um serviço deste tipo, foi criado um *assembly* que contém uma interface que estabelece esse contrato, bem como um conjunto de classes que representam as entidades utilizadas (ver a Figura 3.1).

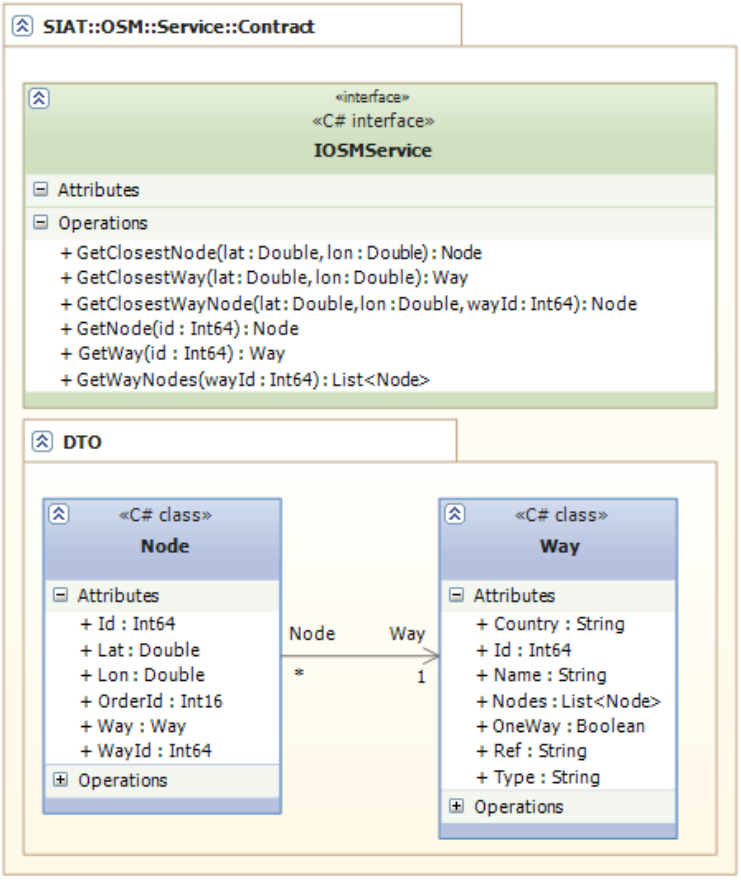

Figura 3.1: Diagrama UML dos contratos.

3.1 Modelo de dados

A análise realizada ao ficheiro *OSM* permitiu dividir o seu conteúdo em apenas três elementos: **Node**, **Way** e **Relation**. O **Node** é o elemento mais básico, usado para definir um ponto geográfico. Uma **Way** permite agregar ordenadamente um conjunto de nós, quer seja para formar uma linha ou uma área. Por fim, uma **Relation** permite agrupar objectos (*ways* e *nodes*) que estejam, de alguma forma, geograficamente relacionados. Com esta análise foram adquiridos conhecimentos suficientes para interpretar um ficheiro deste género e identificar o que é relevante neste projecto.

Com base nos dados do OSM foi extraído um modelo entidade-associação muito simples, ilustrado na Figura 3.2, e que é composto apenas por duas entidades: **Node** e **Way**.

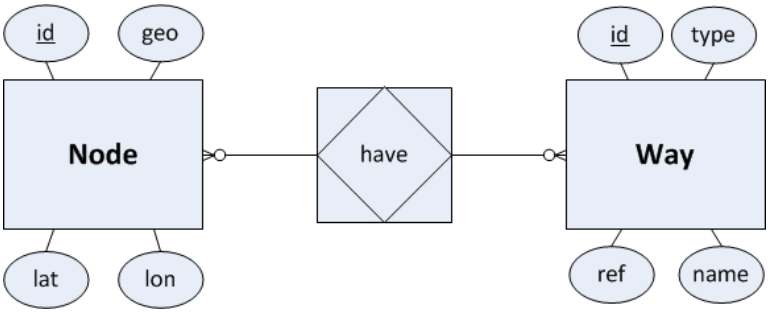

Figura 3.2: Modelo EA construído a partir dos dados do OSM.

3.1.1 Relação *Node - Way*

Como o mesmo **Node** pode pertencer a várias **Way**, e uma **Way**, por sua vez, tem vários **Node**, existe uma relação N..N entre estas duas entidades.

Seguindo as metodologias de normalização, chegámos a um modelo de dados composto por três tabelas: **Node**, **Way** e **WayNode**, sendo que esta última é uma consequência da relação N..N. Após uma análise mais pormenorizada sobre os dados, verificou-se que a tabela **WayNode** tinha apenas mais 10% de entradas comparativamente com a tabela **Node**. Isto indica que, na grande maioria das situações, esta é uma relação de 1..N e não de N..N. Por esta razão, é preferível desnormalizar esta relação, ficando a entidade ***Node*** com um identificador da ***Way*** a que pertence. Além disso, esta desnormalização torna as junções entre estas duas entidades mais eficiente e, como esta operação é frequentemente usada, a sua *performance* é critica para o sistema. Como estas tabelas não sofrem actualizações nem remoções, alguns dos efeitos nocivos da desnormalização, nomeadamente, as perdas de consistência nos dados, que acontecem frequentemente quando os dados são sujeitos a este tipo de operações, não se verificam.

3.1.2 Entidade *Node*

Tal como foi dito anteriormente, um objecto do tipo ***Node*** representa, essencialmente, um ponto geográfico. Esta entidade é composta pelos seguintes campos:

- **Id** - Identificador único;
- **WayId** - Identificador da estrada à qual pertence;
- **OrderId** - Número que identifica a posição deste elemento na estrada a que pertence (usado para determinar o sentido);
- **Latitude** - Latitude do ponto;
- **Longitude** - Longitude do ponto;

- **Geo** - Campo do tipo geográfico onde é criado um objecto do tipo **Point**, que representa a coordenada geográfica.

3.1.3 Entidade *Way*

Uma **Way** permite agregar ordenadamente um conjunto de nós. Normalmente é composto por um conjunto de ***Nodes*** e representa uma estrada. Neste sistema, a entidade **Way** é composta pelos seguintes atributos:

- **Id** - Identificador único;
- **Type** - Indica o tipo da estrada (auto-estrada, nacional, etc.);
- **Ref** - Sigla que identifica a estrada no seu país;
- **Name** - Nome da estrada por extenso;
- **OneWay** - Valor booleano para indicar se a estrada é de sentido obrigatório;
- **Country** - País à qual a estrada pertence (não está a ser usado, e deverá ser removido);
- **Geo** - Campo do tipo geográfico onde é criado um objecto do tipo **LineString**, que representa o percurso da estrada.

3.1.4 Campos espaciais (geográficos)

Ambas as entidades apresentadas são compostas por um campo que representa algo no espaço multi-dimensional, aos quais dá-se o nome de campos espaciais. Estes campos são, geralmente, divididos em duas categorias: geométricos e geográficos. Os da primeira categoria representam alguma entidade geométrica, tal como um conjunto de triângulos ou linhas rectas, numa superfície plana. A segunda categoria é indicada para representar algo no espaço geográfico, ou seja, uma posição na superfície da terra. A diferenciação entre estes dois tipos deve-se, sobretudo, ao nível das escalas usadas para cada tipo. Enquanto os objectos geométricos são representados com pares (X, Y), e a diferença entre dois pontos é facilmente calculada com o teorema de Pitágoras, nos objectos geográficos, os campos são representados com pares (lat, lon), e a distância entre dois pontos tem em conta a curvatura da terra.

A utilização dada aos campos espaciais, neste projecto, é claramente identificável como sendo do tipo geográfico. Um dos problemas deste tipo, é multiplicidade de escalas existentes para representar os pares latitude/longitude. Essas escalas são identificadas univocamente através de um *standard*, com o nome *Spatial Reference System Identifier* (*SRID*).

Neste projecto foi utilizada a escala *World Geodetic System 84* (*WSG 84*), a qual é utilizada pela maioria dos *GPS's*, e é identificada no *SRID* pelo número 4326.

3.1.5 Índices

Os índices são uma parte importante desta base de dados, devido ao elevado número de consultas a que será exposta e à necessidade de as mesmas terem uma resposta rápida. Para cumprir estes objectivos foram criados os seguintes índices:

- Node(geo)
- Node(wayId)
- Way(geo)

Os índices foram criados nos campos chave mais utilizados nas consultas: ambos os campos geográficos e o campo *wayId* da tabela **Node**. Os campos geográficos das duas tabelas são utilizados nas consultas que permitem obter a estrada ou nó mais perto de um determinado ponto e, o campo *wayId* da tabela **Node**, é utilizado nas consultas onde se pretende obter todos os nós de uma estrada. Neste último caso, o índice deve, inclusive, ser criado como *clustered*, o que obrigaria a uma ordenação em disco por essa ordem, que beneficiaria algumas consultas.

3.1.6 Funções

Por fim, foram criadas um conjunto de *stored procedures* que efectuam algumas consultas que são frequentemente utilizadas neste sistema, nomeadamente a procura pela estrada e nó mais próximo de um determinado ponto geográfico. Essas consultas tiram partido dos índices previamente criados e utilizam as funções geográficas embutidas no *SQL Server 2008*, as quais são um standard definido pela *Open Geospatial Consortium* (*OCG*).

3.2 Camada de acesso a dados

Para aceder aos dados guardados segundo o modelo de dados da Figura 3.2, foi criada uma biblioteca *.NET* com um conjunto de classes e métodos que permitem aceder aos dados, à qual se deu o nome de camada de acesso a dados (*Data Access Layer* - DAL).

Esta solução teve a preocupação de abstrair as camadas superiores da tecnologia usada para fazer o acesso a dados e, por isso, ela pode ser dividida em duas partes: a primeira é uma série de classes e interfaces genéricas, usadas para definir os contratos; a segunda é a implementação destes contratos utilizando uma tecnologia de acesso a dados, neste caso, a *ADO.Net Entity Framework*. No modelo de dados apresentado existem duas entidades: ***Way*** e ***Node***.

Esta camada de acesso a dados baseia-se sobretudo em dois padrões: *data mapper* e o *data transfer object* (*DTO*) [13].

O primeiro padrão é usado para mapear as entidades de domínio com uma granulosidade bastante fina, expondo não só as operações *CRUD* (*Create - Read - Update - Delete*) sobres estes objectos, mas também operações que permitem cumprir algumas funcionalidades mais específicas, como, por exemplo, encontrar o objecto ***Way*** mas próximo de uma determinada posição geográfica. Neste contexto existem dois *data mappers*: ***INodesDataMapper*** e ***IWaysDataMapper*** que mapeiam, respectivamente, as entidades ***Node*** e ***Way***.

O padrão *DTO* é muito simples, e consiste apenas em criar classes que representam as entidades da base de dados. Neste caso isso até já tinha sido feito, estando essas classes (***Node*** e ***Way***) no *assembly* partilhado que definiu os contratos do serviço (Figura 3.1). Estas classes são usadas como retorno nos métodos definidos pelos *data mappers*.

A classe ***OSMDataAccessLayer*** é, no fundo, quem disponibiliza a camada de acesso a dados aos seus utilizadores, sendo usada uma instância desta classe por cada transacção. Esta classe segue o padrão *Unit of Work*, abstraindo o utilizador da forma como é feito o tratamento dos dados em modo desconectado. Este deve, antes de terminar a utilização deste objecto, fazer *commit* ou *rollback* das alterações efectuadas. O acesso aos dados é feito através de duas propriedades, mais precisamente, uma para cada *data mapper* criado.

O diagrama *UML* com estas classes pode ser visualizado na Figura 3.3.

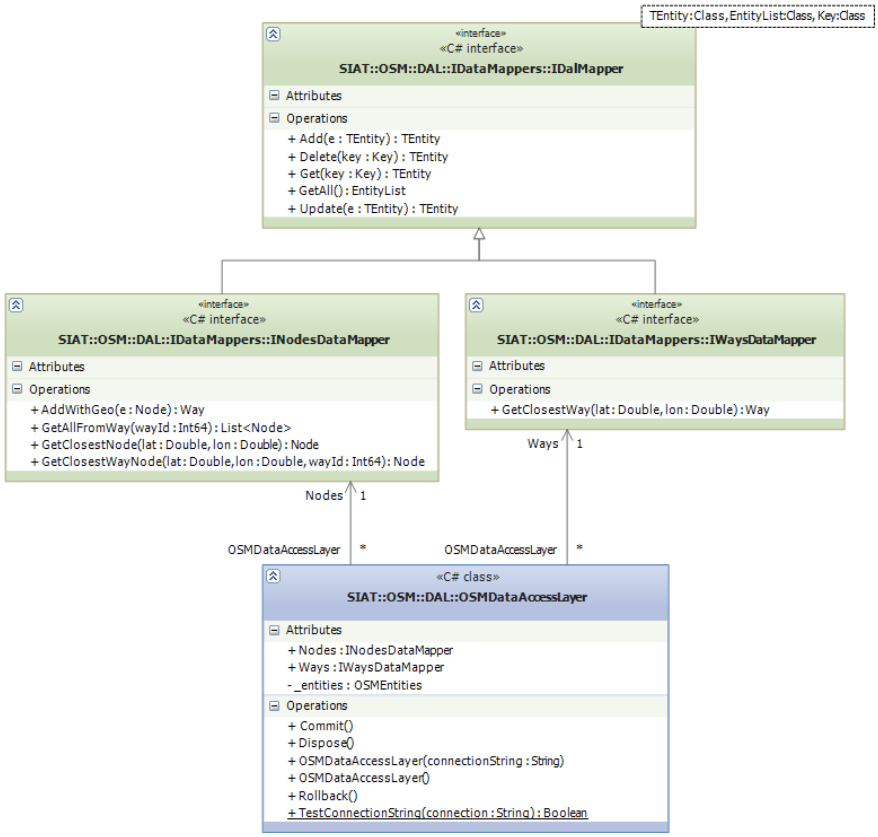

Figura 3.3: Diagrama UML da DAL.

3.2.1 Implementação utilizando a *ADO.Net Entity Framework*

A implementação desta *DAL* foi feita recorrendo à *Entity Framework* (*EF*) na sua versão 4.0. Apesar de ser utilizada esta *framework*, foram minimizadas as dependências da mesma através da utilização de alguns padrões e tecnologias.

Normalmente a utilização desta *framework* gera as classes que representam as entidades e o contexto. Este tipo de utilização cria dependências que, no caso das entidades, são resolvidas fazendo a conversão entre essas classes e os *DTO*. Para evitar o *overhead* associado a esta operação, a versão 4.0 da *EF* já permite a utilização de classes do tipo *POCO* (*Plain Old CLR Objects*) directamente como entidades, substituindo as habitualmente geradas pela *framework*. Este tipo de classes não contém qualquer tipo de lógica de persistência, apenas os campos que compõem a entidade, evitando a criação de qualquer tipo de dependência.

Para utilizar esta técnica seguem-se os mesmos procedimentos, no que toca ao desenho do modelo de dados, de forma gráfica, sendo esse desenho transformado num ficheiro do tipo **edmx**. A única diferença reside na indicação dada à *framework* (através das propriedades desse ficheiro), para que esta não gere algum código, nomeadamente, as classes que representam os objectos das entidades, e a classe que disponibiliza o acesso a essas entidades, à qual dá-se, vulgarmente, o nome de contexto.

Neste caso, essas classes até já tinham sido criadas (são os *DTO's* previamente mencionados), como tal, foram reaproveitadas para serem utilizadas por esta *framework*. A utilização destas técnicas permite que o objecto percorra a totalidade das camadas do componente sem nunca ser necessário fazer qualquer transformação de tipos nem criar dependências com tecnologias.

Nestas situações o contexto é criado manualmente, mas deve obedecer a um determinado padrão, de forma a ser utilizado da pela *EF*. Esse padrão é muito simples, basta que a classe criada extenda da classe **ObjectContext** e disponibilize um objecto do tipo **ObjectSet** para cada uma das entidades, o qual é criado através do método **CreateObjectSet<T>()**, disponibilizado pela *framework*.

Será a utilização deste padrão, aquando do desenvolvimento desta classe, que fará ligação com o código da *EF*. A implementação da classe ***OSMEntities*** é apresentada na Listagem 1.

Foram criadas duas classes que implementam os *data mappers*, nomeadamente, ***EFNodesDataMapper*** e ***EFWaysDataMapper***, as quais utilizam um objecto do tipo ***OSMEntities*** para aceder à base de dados e retornar a informação pretendida. Este ultimo é recebido por parâmetro, no construtor.

Por fim, é a classe ***OSMDataAccessLayer*** que faz a ligação entre tudo isto. Apesar das entidades que expõe serem as interfaces dos *data mappers*, no momento da sua primeira utilização são afectados com objectos do tipos previamente apresentados. Neste momento, é passado aos mesmos um objecto do tipo ***OSMEntities***, criado previamente, no construtor da classe ***OSMDataAccessLayer***. A implementação desta classe é apresentada na Listagem 2.

```
1  public class OSMEntities : ObjectContext
2  {
3          ...
4          private ObjectSet<Node> _nodes;
5          public ObjectSet<Node> Nodes
6          {
7                  // Cria o objecto que representa e entidade Node
8                  get { return _nodes ?? (_nodes = CreateObjectSet<Node>()); }
9          }
10
11         private ObjectSet<Way> _ways;
12         public ObjectSet<Way> Ways
13         {
14                 get { return _ways ?? (_ways = CreateObjectSet<Way>()); }
15         }
16
17         public ObjectResult<Node> GetClosestNode(double latitude,double longitude)
18         {
19                 // Chamada ao Stored Procedure com o nome GetClosestNode
20                 return base.ExecuteFunction<Node>("GetClosestNode",
21                                                   new ObjectParameter("lat", latitude),
22                                                   new ObjectParameter("lon", longitude));
23         }
24         ...
25 }
```

Listagem 1: Implementação do contexto usando a EF.

Para ter uma melhor independência, este problema poderia ter sido abordado de outra forma. Poderiam ter sido criados dois *assemblies*: um para as classes que são independentes da tecnologia de acesso a dados; e outro para as classes que são dependentes de uma determinada tecnologia, neste caso, o *ADO.Net Entity Framework*.

Os utilizadores desta DAL apenas referenciavam o primeiro *assembly* e, para definir qual o tipo de tecnologia a utilizar, definiam o nome do respectivo *assembly* no ficheiro de configuração. O carregamento deste último seria feito de forma dinâmica, através da implementação de uma *factory*, que ficará inserida no primeiro.

Esta implementação traz benefícios, mas também tem problemas. A principal vantagem é a clara separação entre o que são os contratos e as implementações, o que elimina totalmente as dependências. Em contrapartida, o carregamento dinâmico de um *assembly* é um processo demorado, no entanto, este só é feito uma única vez.

Na solução actualmente implementada, quem usa continua a não precisar de fazer alterações no código, mas é obrigado a compilar novamente a sua solução com o novo *assembly*.

3.3 Processo de carregamento

O processo de carregamento dos ficheiros *OSM* na base de dados está dividido em várias tarefas, cada uma pensada para minimizar o tempo que esta operação acarreta.

Inicialmente, quando as tabelas são criadas na base de dados, são desactivadas as várias restrições presentes nessas tabelas (de chave primária e de chaves estrangeiras). Outra opção tomada neste âmbito é delegar a criação dos índices apenas para o final do carregamento. Estas duas medidas têm

```
1  public class OSMDataAccessLayer : IUnitOfWork
2  {
3          private readonly OSMEntities _entities;
4
5          public OSMDataAccessLayer()
6          {
7                  _entities = new OSMEntities();
8          }
9
10         private INodesDataMapper _nodes;
11         public INodesDataMapper Nodes
12         {
13                 get { return _nodes ?? (_nodes = new EFNodesDataMapper(_entities)); }
14         }
15
16         private IWaysDataMapper _ways;
17         public IWaysDataMapper Ways
18         {
19                 get { return _ways ?? (_ways = new EFWaysDataMapper(_entities)); }
20         }
21 }
```

Listagem 2: Implementação da classe ***OSMDataAccessLayer***.

o objectivo de melhorar o desempenho das escritas na base de dados, num processo onde é feito um carregamento massivo de informação.

De seguida, é utilizada a ferramenta ***OSMtoDB*** para carregar os dados para o servidor. Neste processo, os campos geográficos de ambas as entidades não ficam preenchidos, esse processo é feito *à posteriori* por um *stored procedure* criado para o efeito. Esta opção resulta da experiência de utilização deste procedimento, que demonstrou que esta opção é mais rápida do que criar o campo imediatamente na inserção do tuplo. Este acontecimento não parece ter explicação plausível, por isso poderá ser uma falha do servidor de base de dados.

As várias etapas do processo de instalação de uma base de dados OSM são apresentadas na Figura 3.4.

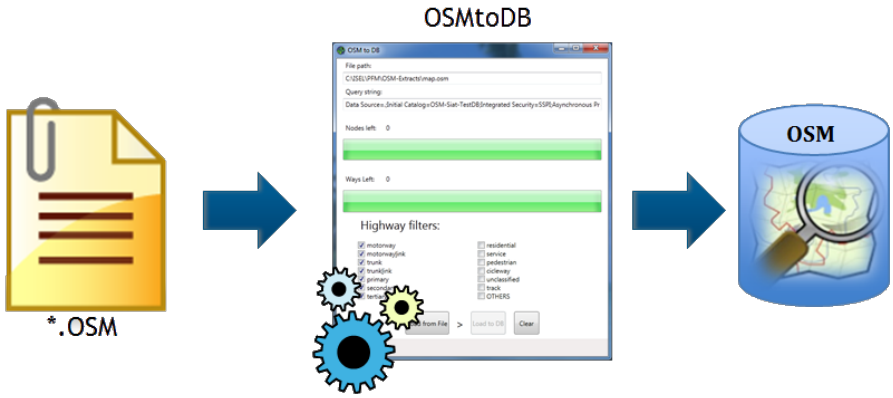

Figura 3.4: Processo de instalação da base de dados OSM.

3.3.1 OSMtoDB

A ***OSMtoDB*** é uma aplicação *desktop* desenvolvida com o intuito de auxiliar o processo de instalação deste sistema. Foi desenvolvida em *.Net* usando a *framework Windows Presentation Foundation*, e

tem como função ler os ficheiros *XML*, transformar esses dados e carregá-los na base de dados previamente criada com o modelo apresentado.

Esta aplicação permite indicar o caminho para o ficheiro *OSM* que serve de fonte de dados e a *query string* da base de dados onde estes deverão ser inseridos posteriormente (ver Figura 3.5). O seu desenvolvimento foi contextualizado nos objectivos do projecto SIAT, e por essa razão apenas é possível filtrar as entradas por tipo de estrada. No entanto, é possível indicar que se deseja carregar todas as entradas lidas e nesta situação não existe qualquer tipo de filtragem.

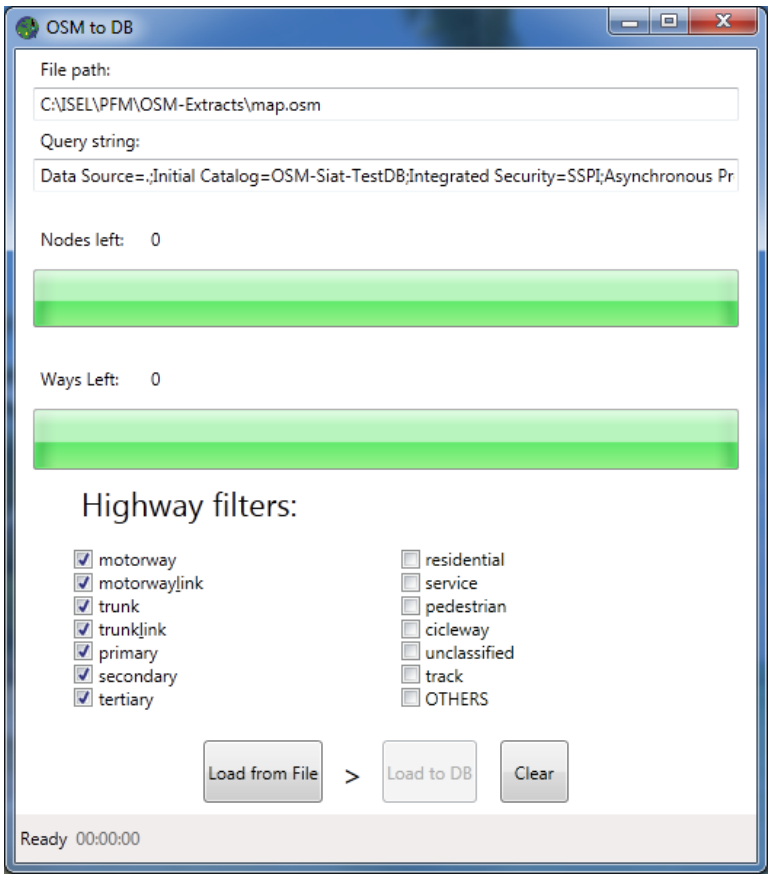

Figura 3.5: Aplicação OSM to DB.

Internamente esta aplicação funciona da seguinte forma: existe um fio de execução que lê o ficheiro *XML* e alimenta duas listas, uma com os objectos ***Node*** e outra com os objecto ***Way***. Paralelamente, existem um conjunto de fios de execução que inserem os dados dessa lista na base de dados, utilizando a camada de acesso a dados previamente apresentada. Ambas as listas são objectos da *framework .Net* 4.0, do tipo **ConcurrentBag**, preparados para acessos concorrentes sem o uso de *locks*, o que permite uma maior eficiência. No entanto, os resultados da utilização desta ferramenta demonstram que ler o ficheiro corresponde a apenas 5% do tempo necessário para carregar os dados recolhidos.

Foram realizados testes à ferramenta que permitem comprovar que a mesma funciona com os seguintes servidores de bases de dados:

- *SQL Server 2008 R2*
- *SQL Server Code Name 'Denali' CTP3*
- *SQL Azure*

3.4 Serviço

Neste ponto optou-se por uma abordagem diferente em relação ao *Nominatim*. Enquanto que este último disponibiliza um serviço *REST*, nesta solução as funcionalidades são disponibilizadas através de um *web service* do tipo *SOAP*. No futuro, deverá ser seguida uma abordagem semelhante à do *Nominatim*, mas neste ponto esse requisito não é fundamental pois não acrescenta nada de novo ao sistema.

Este *web service* foi construído em *Windows Communication Foundation* (*WCF*) e expõe o conjunto de operações especificado na interface ***IOSMService*** (previamente apresentada na Figura 3.1). Foi necessário adaptar o uso destes contratos ao *WCF* e, por essa razão, tanto a interface como as classes (*DTO's*) foram marcadas como contratos de serviço e de dados, respectivamente.

O serviço está configurado para funcionar em modo *Single*, ou seja, com uma única instância, mas com pedidos concorrentes.

A implementação da interface ***IOSMService*** é feita na classe ***OSMService***, que limita-se a servir de fachada para as funções correspondentes da camada de acesso a dados. No entanto, esta classe é responsável por lidar com a validação e escolha da conexão mais indicada para a pesquisa, mas essa funcionalidade será detalhada na Subseção 3.4.1.

Por fim, importa acrescentar que este serviço utiliza a funcionalidade *WCF Discovery* para se anunciar e ser descoberto numa rede local. Esta funcionalidade permite que esta aplicação se integre com naturalidade numa solução *Service Oriented Architechture*. Mais uma vez, esta funcionalidade resulta de um requisito do projecto SIAT, o qual será explicado no Capitulo 5.

3.4.1 Escalabilidade

Os serviços já existentes que são similares a este são sujeitos a uma grande quantidade de consultas diárias, além disso, as consultas por este efectuadas são computacionalmente exigentes, o que implica que exista um número limitado de consultas às quais este serviço consegue responder com alguma eficiência.

Torna-se assim importante incluir alguma escalabilidade na solução. Neste ponto, a solução encontrada para este problema foi muito simplista e consiste em ter uma *pool* de bases de dados com a informação replicada. De seguida cada pedido é redireccionado para uma base de dados diferente de forma circular, ou seja, caso existam três pedidos e apenas duas bases de dados o primeiro pedido

vai para a base de dados número um, o segundo pedido vai para a base de dados número dois e, por fim, o terceiro pedido volta a ser para a base de dados número um. Este algoritmo é semelhante ao *Round-robin DNS* [14].

Existem também ganhos significativos quanto à disponibilidade do sistema, porque um maior número de bases de dados permite tolerar as falhas individuais.

O comportamento desta funcionalidade começa quando o serviço é criado. Nesse ponto o ficheiro de configurações é lido e todas as *connection strings* são validadas. Em *runtime*, é possível editar ou adicionar conexões através do ficheiro de configurações. O serviço deve ser informado acerca desta modificação através do *input* da consola, desencadeando o processo de leitura deste ficheiro e consequente actualização da lista de conexões. Durante a execução do serviço as suas operações requisitam a conexão através do método (***GetNextConnection***). A implementação deste método, consiste em fazer a divisão inteira entre o número de chamadas (*ticketNumber*) e o número de conexões disponíveis. De seguida, o valor do *ticket* é incrementado atomicamente. A implementação deste algoritmo é apresentado na Listagem 3.

```
1  private int _ticketNumber;
2  private string[] _connectionStrings;
3
4  private string GetNextConnection()
5  {
6          int connectionNumber = Interlocked.Increment(ref _ticketNumber);
7          connectionNumber = connectionNumber % _connectionStrings.Length;
8          return _connectionStrings[connectionNumber];
9  }
```

Listagem 3: Algoritmo para obter balanceamento de carga nas bases de dados.

Apesar de existir a consciência que a solução implementada não é a melhor, ela oferece algum nível de escalabilidade com um custo de computação muito reduzido. No futuro deverão ser exploradas outras soluções, quer ao nível do algoritmo utilizado como na estrutura que armazena a informação. Quanto a este ultimo ponto, existe uma particularidade que nunca pode ser violada, que é a seguinte: qualquer sistema utilizado para aumentar a escalabilidade da solução tem de ter em consideração que o suporte para funções geográficas é necessário. Esta particularidade impede que se usem, pelo menos por enquanto, outros tipos de sistemas de armazenamento persistente de informação que estão mais preparados para objectivos de escalabilidade, tais como os *No-SQL*.

3.5 Resultados

As duas soluções previamente apresentadas (*Nominatim* e *Sql Server*) foram submetidas a testes (consultar o **??**) com o objectivo de determinar a sua *performance*. Os valores apresentados no teste revelam que a segunda versão é ligeiramente mais lenta, cerca de 15 %. No entanto deve ter-se em consideração que esta versão cumpre os requisitos especificados, e para o fazer tem de recorrer a uma procura computacionalmente mais exigente.

3.6 Disponibilização à comunidade *OSM*

Como o desenvolvimento deste componente é o resultado de uma lacuna, existente no conjunto de projectos desenvolvidos pela comunidade do *OSM*, um dos objectivos é disponibilizar a solução encontrada, para esse mesma comunidade possa, a partir desse momento, contribuir para o desenvolvimento dessa solução.

Com esta publicação, passará a existir uma ferramenta que permitirá a outros projectos tirarem partido da informação do *OSM*, utilizando uma base de dados *MS SQL Server*.

Foi criado um repositório público no *Github*, que disponibiliza um conjunto de ferramentas e outros documentos auxiliares, bem como todo o código fonte usado no desenvolvimento desta solução. Esse repositório pode ser encontrado em `https://github.com/diogofdsilva/MS-OSM-Querys`.

Esta página foi intencionalmente deixada em branco.

4 Arquitectura geral

De uma forma geral, os componentes implementados nesta solução podem ser divididos em dois grupos: os do núcleo (***SIAT Core***) e os módulos externos. O conjunto de componentes que formam o ***SIAT Core*** permitem disponibilizar as funcionalidades do sistema de informações. Os módulos apenas tiram partido destas funcionalidades, interagindo e alimentando o sistema. Na Figura 4.1 são apresentados os componentes da arquitectura geral da solução implementada.

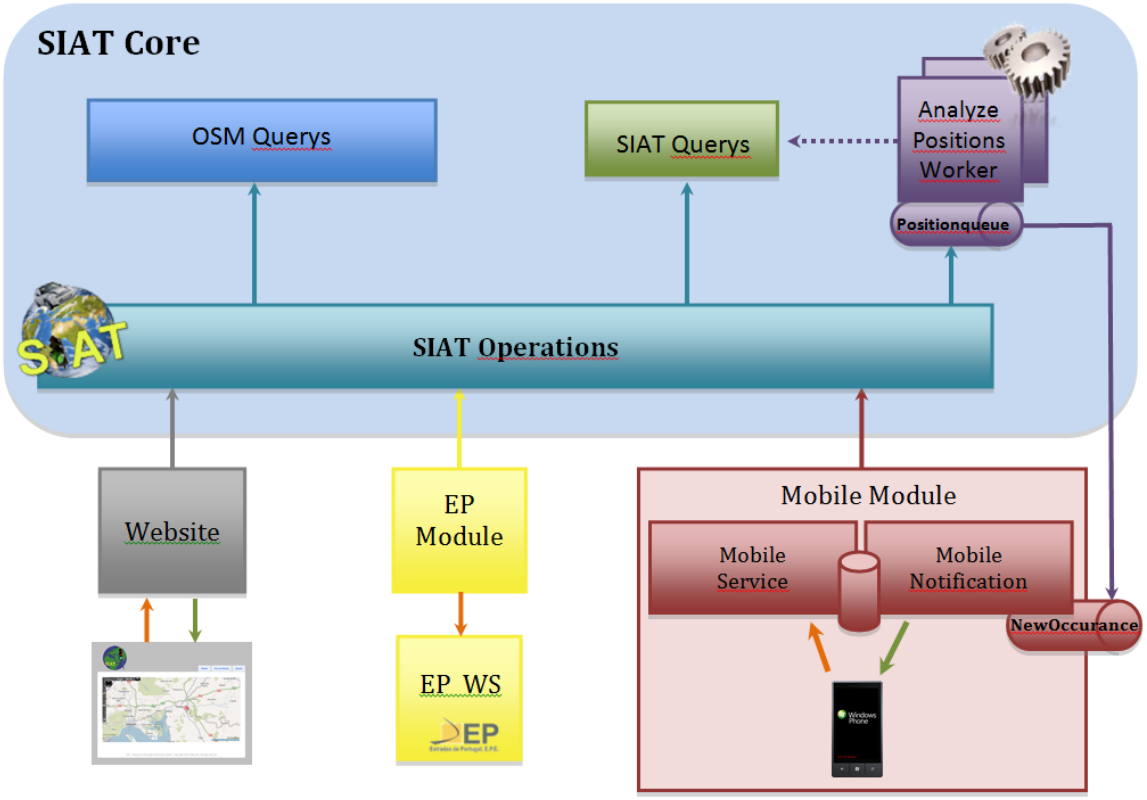

Figura 4.1: Arquitectura Geral da solução.

O SIAT Core é composto por quatro componentes: ***OSM Querys***, ***SIAT Querys***, ***Analyze Positions Worker*** e ***SIAT Operations***.

O ***OSM Querys*** é um componente que se dedica, exclusivamente, a *querys* puramente geográficas como, por exemplo, saber qual a estrada onde se encontra o veiculo, e que não interfiram com os dados do sistema de informações. Este componente será exposto na forma de um serviço.

O componente ***SIAT Querys*** serve para as restantes consultas, ou seja, as que envolvam dados do sistema de informações, tais como saber todas as ocorrências de trânsito detectadas. Neste componente existe um modelo de dados próprio criado para suprir as necessidades do projecto SIAT. Seria possível agregar estes componentes num único, mas esta separação permite criar um balanceamento de carga que evite ou, pelo menos, minimize, o estrangulamento de um único componente. Este componente é exposto da mesma forma que o anterior.

O ***Analyze Positions Worker*** é um componente que surge com o objectivo de melhorar a *performance* da solução. Como o algoritmo que analisa as informações de trânsito recebidas é computacionalmente exigente, não faz sentido concentrar a sua execução (mesmo que em paralelo) num único processo, sobretudo se esse processo estiver a ser utilizado para outros fins (como no caso do serviço que comunica com os dispositivos móveis). A comunicação com este componente é feita, exclusivamente, através de filas de mensagens, de forma assíncrona.

A arquitectura desta solução, em particular, dos componentes do ***SIAT Core***, foi desenvolvida segundo os princípios de desenho *Service-oriented architecture* (*SOA*)[15].

Os dois primeiros componentes apresentados são *web services*, expõem um contrato *WSDL* e a comunicação é feita via *SOAP*. Este conjunto de contratos de comunicação, partilhados entre as várias componentes, é precisamente um dos requisitos deste tipo de arquitecturas. Aliás, o seu desenvolvimento adoptou a grande maioria dos princípios *SOA*, com particular relevância para a solução que permite a descoberta dos serviços. A comunicação com estes serviços é sincrona e, como são *stateless* (outro dos princípios *SOA*), podem ser executados paralelamente.

O único componente que escapa a esta lógica de desenvolvimento é o ***Analyze Positions Worker***. Este componente, por utilizar os restantes, não e totalmente autónomo porque não encapsula a totalidade da sua lógica de negócio. Como já foi mencionado, a comunicação com este componente é assíncrona, baseada numa fila de mensagens, o que permite que exista um acoplamento fraco com as entidades que o usam, característica que é, também, uma dos princípios das arquitecturas *SOA*.

Para os módulos externos, o componente ***SIAT Operations*** é a *"porta de entrada"* no ***SIAT Core***. Este componente funciona como uma *business layer*, encapsulando a lógica do negócio e tudo o que se refere à interacção com os componentes anteriores. Pode-se dizer, seguindo as metodologias *SOA*, que este componente faz a orquestração dos restantes serviços.

Como já foi explicado anteriormente, todos os módulos externos interagem com o sistema central para lhes fornecer/obter informações de trânsito. Os dispositivos móveis são as fontes de informação

por excelência, no contexto deste projecto. Por essa razão, o módulo específico para estes dispositivo é composto por duas componentes, um para que os dispositivos possam enviar a informação recolhida e outro para receber novas actualizações.

A solução inclui também um *website*, no qual é possível consultar as actuais informações de trânsito. Por fim, resta o módulo das Estradas de Portugal, responsável por recolher periodicamente a informação disponível no *web service* desta entidade, com o objectivo de agregar o maior número de informação possível.

No próximo capítulo serão analisados todos estes componentes mais detalhadamente.

Esta página foi intencionalmente deixada em branco.

5 Implementação

A implementação desta solução é feita quase exclusivamente em tecnologias Microsoft, desde a base de dados utilizada (*SQL Server 2008*) até à linguagem de programação escolhida, o *C#* (*.Net*), incluindo o sistema operativo móvel escolhido, o *Windows Phone 7*. Esta escolha visa diminuir ao máximo os problemas decorrentes de uma implementação demasiado heterogénea.

Como foi referenciado no Capitulo 4 esta arquitectura foi construída segundo os princípios SOA. Um desses princípios é a descoberta de serviços, que neste projecto foi implementada tirando partido de uma nova funcionalidade do *WCF 4.0*, o *WCF Discovery*. Esta funcionalidade utiliza o protocolo *WS-Discovery* para, em *runtime*, anunciar ou descobrir serviços mediante um determinado critério. Tipicamente o critério usado é o tipo da interface implementada pelo serviço, no entanto existem outros critérios como, por exemplo, através de *scopes*, que são *URI's* utilizados para categorizar *endpoints* [16].

Uma das grandes vantagens deste protocolo em relação aos seus concorrentes é a possibilidade de explorar o *multicast* em redes locais, deixando de ser necessário um ponto central com o registo de todos os serviços. O cliente quando necessita de um determinado serviço envia uma mensagem a requisitá-lo apresentando os seus critérios, por sua vez o serviço verifica se corresponde aos critérios e anuncia-se em caso afirmativo.

Neste capitulo irá ser abordado a implementação de cada um dos componentes desenvolvidos, partindo da explicação de como estão implementados os componentes que fornecem as funcionalidades do serviço até aos componentes que o disponibilizam ao utilizador.

No componente ***OSM Querys*** foi utilizada a solução previamente apresentada no Capitulo 3, como tal a explicação exaustiva da sua implementação não irá ser novamente apresentada.

5.1 SIAT Querys

O componente ***SIAT Querys*** integra-se nesta arquitectura para colmatar uma necessidade muito específica: extender as funcionalidades geográficas que já existem no componente ***OSM Querys***, acrescentando a informação do projecto *SIAT*. Esta necessidade surge porque, sendo o ***OSM Querys*** um componente autónomo, focado em oferecer serviços de *geocoding*, não deve conter alguma informação mais específica que está intrínseca à utilização dada neste projecto.

Actualmente, esta necessidade surge apenas para marcar geograficamente as ocorrências de trânsito mas, sendo esta a principal funcionalidade do projecto, é importante não descurar a sua implementação da melhor forma. Futuramente, podem ser adicionadas novas funcionalidades ao projecto que requisitem de um suporte geográficos, tais como: permitir que o utilizador crie zonas, nas quais deseja ser notificado de novas ocorrências.

Invariavelmente, este componente será semelhante ao ***OSM Querys*** em termos de arquitectura, ou seja, é composto por uma camada de dados, uma camada de acesso a esses dados e uma camada de serviços que disponibiliza as funcionalidades aos consumidores interessados. Também em termos de tecnologias escolhidas os dois componentes são semelhantes, em particular na utilização das *frameworks Entity Framework* e *WCF*. As diferenças resumem-se praticamente aos tipos de dados manipulados.

Entre as competências deste componente inclui-se a manipulação e consulta das ocorrências do sistema. Para especificar o contrato deste serviço foi novamente criado um *assembly* partilhado que contém a interface que estabelece esse contrato e a classe que representa uma ocorrência (ver a Figura 5.1).

Este componente faz parte do conjunto que formam a rede *SOA* criada nesta solução.

5.1.1 Modelo de dados

Mais uma vez recorreu-se a uma base de dados com o servidor *SQL Server 2008* para armazenar a informação deste componente. Esta escolha deve-se à utilização de campos do tipo *geography* nas entidades deste modelo.

Nesta fase de desenvolvimento do sistema este componente apenas manipula ocorrências, e esta será a única entidade deste modelo de dados. O diagrama entidade-associação deste modelo de dados é apresentado na Figura 5.2.

Occurrence

A entidade ***Occurrence*** armazena a informação das ocorrências presentes no sistema. Para este sistema, uma ocorrência deve conter a seguinte informação:

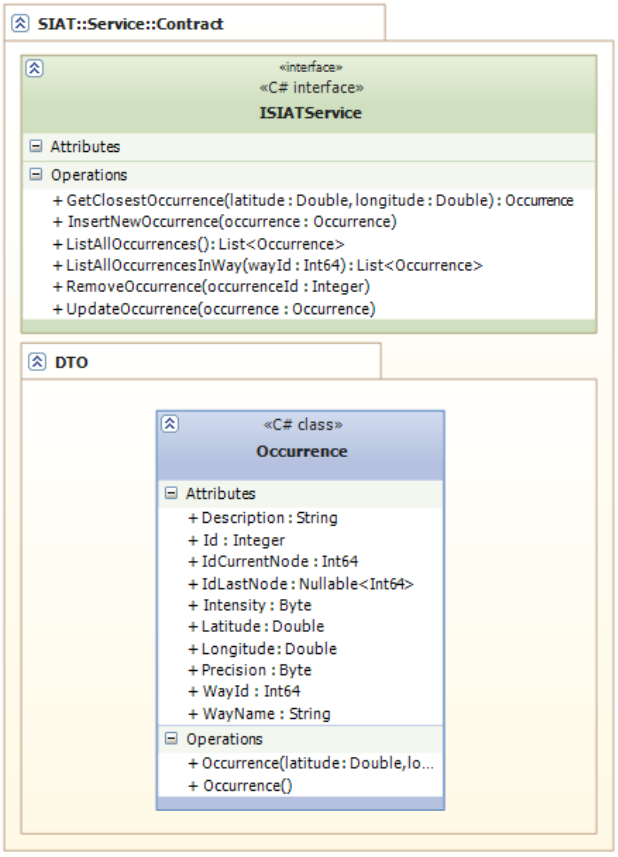

Figura 5.1: Diagrama UML dos contratos do serviço ***SIAT Querys***.

- **Id** - Identificador único;
- **Latitude** - Latitude do ponto da ocorrência;
- **Longitude** - Longitude do ponto da ocorrência;
- **Intensity** - Valor que permite perceber a gravidade da ocorrência;
- **Precision** - Usado para identificar a veracidade da ocorrência;
- **Description** - Descrição textual do tipo de ocorrência;
- **IdCurrentNode** - Identificador da entidade ***Node***, representa o ponto da ocorrência;
- **IdLastNode** - Identificador da entidade ***Node***, representa o último ponto anterior à ocorrência;
- **IdWay** - Identificador da entidade ***Way***, identifica a estrada da ocorrência.
- **Geo** - Campo do tipo geográfico onde é criado um objecto do tipo **Point**, que representa a coordenada geográfica da ocorrência, permitindo fazer pesquisas espaciais mais facilmente.

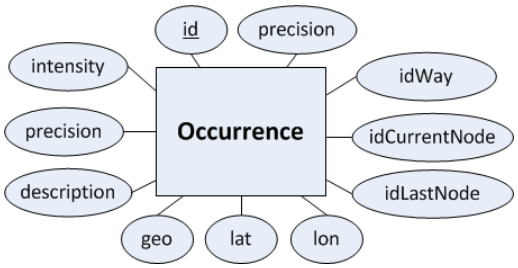

Figura 5.2: Modelo *EA* do componente ***SIAT Querys***.

Uma questão que pode ser levantada da análise à estrutura desta entidade, é a necessidade de existirem dois identificadores para a entidade ***Node***. Esta necessidade surge com o objectivo de cumprir uma funcionalidade deste projecto: identificar o sentido da ocorrência.

Em relação aos campos ***intensity*** e ***precision***, uma explicação mais aprofundada do seu significado pode ser encontrada na Secção 5.3.3.

Os campos ***IdCurrentNode***, ***IdLastNode*** e ***IdWay*** são identificadores de entidades externas que não são armazenadas nesta base de dados, logo, estes campos não são, pelo menos formalmente, chave estrangeira.

Funções

Foi criada uma *stored procedure* para encapsular a consulta que permite obter qual a ocorrência mais próxima. A *stored procedure* criada tem o nome de ***GetClosestOccurrence***, utiliza o campo geográfico e pode ser consultada na Listagem 4. Esta consulta utiliza a função *STDistance* para calcular a distância entre pontos, no entanto, esta é uma opção temporária porque as novas versões do *SQL Server* já terão funções mais optimizadas para este tipo de consultas.

```
1  create procedure GetClosestOccurrence @lat float, @lon float
2  as
3  declare @h geography = geography::STPointFromText(
4          'Point(' + CONVERT(varchar,@lat) + ' ' + CONVERT(varchar, @lon)+ ')', 4326)
5  select Top(1) *
6  from Occurrence
7  where geo.STDistance(@h) < 100
8  order by geo.STDistance(@h) asc
9  go
```

Listagem 4: *Query SQL* que obtém a Ocorrência mais próxima.

Índices

Para permitir um melhor desempenho das pesquisas mais criticas, foi criado um novo índice. Neste modelo a pesquisa mais utilizada é a que está contida na *stored procedure* ***GetClosestOccurrence*** logo, para melhorar a sua *performance*, foi criado um índice espacial no campo geográfico (***geo***). Como esta entidade é fortemente actualizada não convém criar muitos índices, sob pena de deteriorar

o desempenho dessas alterações. No caso do índice criado, o campo em questão não sofre modificações portanto não existe problema.

5.1.2 Camada de acesso a dados

Mais uma vez, foi criada uma camada de acesso a dados, que permite manipular a informação guardada segundo o modelo da Figura 5.2. Esta camada de acesso a dados é semelhante à criada para o componente ***OSM Querys***, tanto ao nível da sua arquitectura como dos padrões utilizados. Também neste caso, esta *DAL* é disponibilizada na forma de uma biblioteca *.Net*.

Nesta implementação, a classe que disponibiliza a *DAL* tem o nome de ***SIATDataAccessLayer***, e é composta por uma propriedade que representa a entidade ***Occurrence***, a qual deve ser usada para aceder aos seus dados.

Optou-se novamente por impedir a criação de dependências tecnológicas através da utilização de um conjunto de padrões. Não entrando novamente em demasiado detalhe, importa apenas acrescentar que, nesta implementação, existe apenas um *data mapper* que mapeia as ocorrências (***IOccurrencesDataMapper***) e uma classe *DTO* (***Occurrence***), que representa também a entidade ocorrência e que, mais uma vez, já tinha sido criada, estando presente no *assembly* partilhado com os contratos deste serviço.

Implementação utilizando a *ADO.Net Entity Framework*

A semelhança entre estas duas *DAL's* acontece também ao nível das tecnologias usadas no acesso a dados, portanto, foi novamente utilizada a *ADO.Net Entity Framework* para esse efeito.

Mais uma vez foi utilizado o padrão *POCO*, portanto, o *DTO* é directamente utilizado como representante da entidade ***Occurrence*** e o contexto que permite aceder a esses dados foi criado manualmente (***SIAT Entities***).

A classe ***OccurrencesDataMapper*** implementa a interface ***IOccurrencesDataMapper***, e utiliza o contexto criado (***SIAT Entities***) para aceder aos dados. Para além das operações *CRUD*, é ainda disponibilizada uma outra operação, que permite obter a ocorrência mais próxima de uma coordenada geográfica.

O diagrama *UML* presente na Figura 5.3 apresenta o conjunto de classes que compõem esta *DAL*.

5.1.3 Serviços

Por fim, as funcionalidades requisitadas no contrato ***ISIATService*** (apresentado na Figura 5.1) são disponibilizadas aos utilizadores através de um serviço construído em *WCF*, seguindo o mesmo método já utilizado no componente ***OSM Querys***. Este serviço não contém qualquer tipo de lógica de negócio, limitando-se a fazer consultas à base de dados.

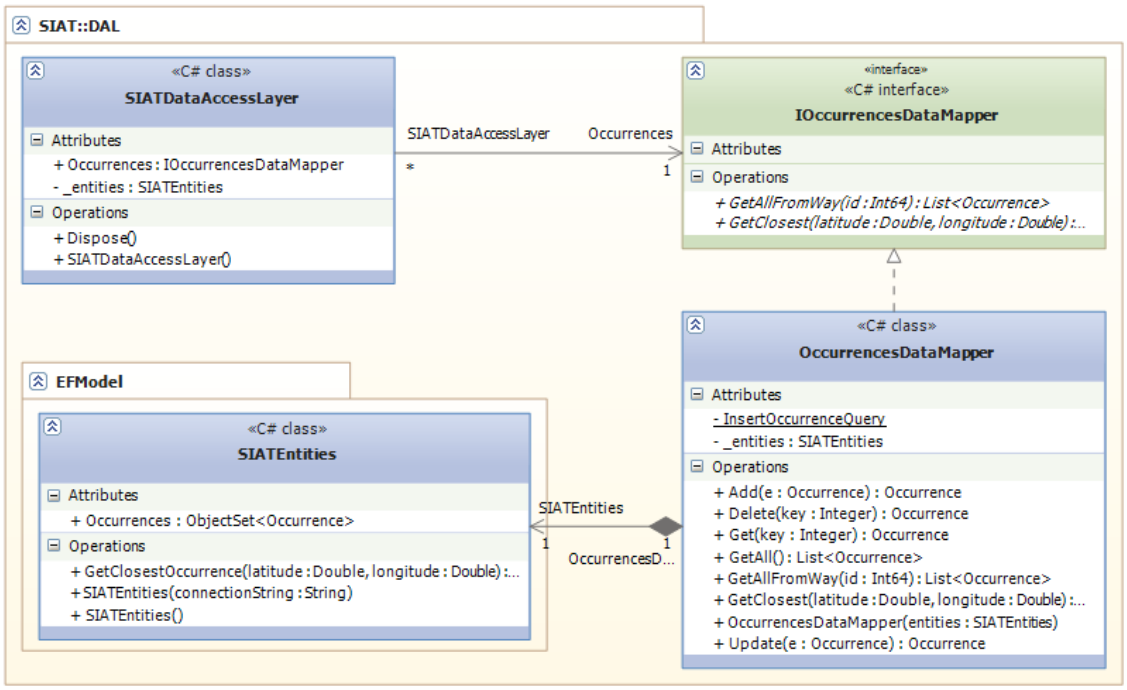

Figura 5.3: *UML* da *DAL* do componente ***SIAT Querys***.

Como este componente lida apenas com a entidade ocorrência, as operações disponibilizadas por este serviço incidem apenas sobre esta entidade. Para além das operações *CRUD*, são ainda disponibilizadas operações que permitem listar as ocorrências mediante alguns parâmetros (como, por exemplo, filtradas por estrada), ou obter a ocorrência mais próxima de um determinado ponto.

Configurações

Mais uma vez este serviço utiliza o protocolo *WS-Discovery* para se anunciar na rede local, cumprindo um dos requisitos da solução. A configuração desta funcionalidade no serviço é muito simples, e ocorre apenas no ficheiro de configuração, bastando adicionar uma nova secção no *behavior* do serviço e um novo *endpoint*, usado na descoberta do mesmo, tal como é apresentado na Listagem 5.

Em termos de outras configurações do *WCF*, por omissão este serviço utiliza *wsHttpBinding* e está à escuta no *endpoint http://localhost:18088/SIATService*, no entanto esta última configuração pode ser alterada, sem grande prejuízo, porque a descoberta do endpoint é feita, do lado do cliente, utilizando o protocolo previamente apresentado. Do lado das configurações do *binding*, é preciso ter em conta que o tipo utilizado na descoberta deste serviço é o *wsHttpBinding*, como tal, se for feita essa alteração do lado do serviço este deixa de estar disponível. Importa acrescentar que em termos de comportamento, o serviço está configurado para funcionar em modo *Single*, ou seja, com uma única instância mas com pedidos concorrentes.

```
<services>
        <service name="SIAT.Service.SIATService" behaviorConfiguration="SIATServiceBehavior">
                <!-- Discovery Endpoints -->
                <endpoint kind="udpDiscoveryEndpoint" />
                ...
        </service>
        ...
</services>

<behavior name="SIATServiceBehavior">
        <serviceDiscovery>
                <announcementEndpoints>
                        <endpoint kind="udpAnnouncementEndpoint"/>
                </announcementEndpoints>
        </serviceDiscovery>
        ...
</behavior>
```

Listagem 5: Configuração do serviço para ser descoberto através do protocolo *WS Discovery*.

5.2 Analyze Positions Worker - APW

Figura 5.4: Arquitectura do componente APW.

O componente ***Analyze Positions Worker*** foi construído com o objectivo de resolver um problema de desempenho, verificado no algoritmo que analisa a informação recebida dos utilizadores (mais informação acerca deste algoritmo será dada na Subseção 5.3.3).

Este problema, faz com que seja impensável que essa análise possa ser efectuada directamente na chamada à operação que disponibiliza esse serviço. Inclusivamente, o utilizador não necessita de nenhum feedback relativo ao envio destes dados, logo, o seu retorno é irrelevante para o mesmo.

A primeira solução para este problema consiste em executar este algoritmo num fio de execução à parte, mas mesmo esta hipótese não parece fazer muito sentido, porque sobrecarregaria o processo que devia estar preocupado em dar uma resposta rápida aos utilizadores.

A solução encontrada passa por delegar a execução deste algoritmo, num processo separado, o qual é notificado sobre os novos dados a analisar através de uma fila de mensagens. Desta forma, podem existir vários processos a consumir os dados dessa fila em paralelo, sendo que estes processos podem

ser criados ou destruídos ao longo do tempo, consoante a carga de mensagens pendentes na fila, o que torna o sistema mais dinâmico.

Ao componente criado para fazer esse processamento deu-se o nome de ***Analyze Positions Worker*** (***APW***), foi desenvolvido em *.Net* como um processo com output para a consola do sistema. O seu comportamento consiste em, ao receber novas mensagens lançar uma nova *Task* (classe que integra a *Task Parallel Library - TPL* [17]) que, assincronamente, faz o processamento dessa informação recorrendo ao algoritmo que está inserido no componente ***SIAT Operations***.

Foi utilizada uma fila de mensagens do tipo *Microsoft Message Queuing* (*MSMQ*) [18] no endereço *"./private/AnalyzePositionsWorker"*.

5.3 SIAT Operations

O ***SIAT Operations*** é um componente fulcral nesta solução. Para além de agregar a maioria da lógica de negócio, funciona ainda como cliente do ***SIAT Core***. Aliás, uma das funcionalidades deste componente é, precisamente, a descoberta dos restantes serviços que compõem o ***SIAT Core***.

Este componente é disponibilizado na forma de um *assembly .Net*, que depois é utilizado pelos restantes componentes que necessitam de aceder às funcionalidades do ***SIAT Core***.

A descoberta dos serviços do ***SIAT Core*** é feita recorrendo ao protocolo *WS-Discovery* anteriormente apresentado, o que permite que este componente seja altamente dinâmico, não necessitando de qualquer tipo de configuração desde que os serviços estejam disponíveis na rede local.

São disponibilizadas um conjunto de operações e, cada uma delas utiliza um ou vários dos serviços que compõem o ***SIAT Core***, das quais destaca-se a operação que analisa a informação recebida dos dispositivos móveis. O conjunto de operações disponibilizado pode ser consultado no *UML* presente na Figura 5.5.

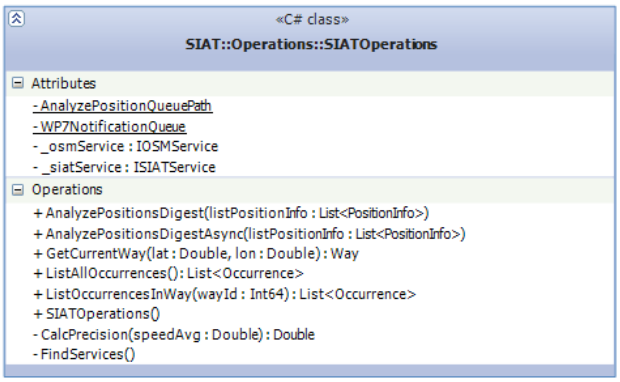

Figura 5.5: Diagrama *UML* da classe ***SIAT Operations***.

5.3.1 Descoberta dos serviços do *SIAT Core*

A arquitectura desenvolvida para esta solução é baseada em *SOA* e, numa arquitectura deste tipo, uma das principais funcionalidades é a descoberta de serviços. Nesta solução, essa funcionalidade é simplificada através da utilização do protocolo *WS-Discovery*.

Para descobrir os serviços que compõem o ***SIAT Core*** utilizando este protocolo, o componente ***SIAT Operations*** utiliza, como critério na descoberta, as interfaces dos diferentes serviços. Como em ambos os serviços (***OSMQuerys*** e ***SIAT Querys***) as suas interfaces são disponibilizadas em *assemblys* à parte, este componente referencia esses *assemblys* para, *à posteriori*, utilizar essas interfaces na descoberta dos serviços na rede local, tal como é apresentado na Listagem 6.

```
1  Binding binding = new WSHttpBinding();
2  var client = new DiscoveryClient(new UdpDiscoveryEndpoint());
3  var criteria = new FindCriteria(typeof (IOSMService));
4  criteria.Duration = TimeSpan.FromSeconds(2);
5  criteria.MaxResults = 1;
6  Collection<EndpointDiscoveryMetadata> services = client.Find(criteria).Endpoints;
7
8  if (services.Count == 0)
9  {
10         Console.WriteLine("\nOSM Service - No services were found.");
11 }
12 else
13 {
14         Console.WriteLine("\nOSM Service - Found.");
15         _osmService = ChannelFactory<IOSMService>.CreateChannel(binding, services[0].Address);
16 }
```

Listagem 6: Obter proxy para os serviços usando o protocolo *WS Discovery*.

A obtenção do *proxy* para os serviços é feita imediatamente no construtor da classe ***SIATOperations***, logo, só depois de concluído este processo é possível utilizar as operações desta classe. Quando algum dos *proxys* fica indisponível, é desencadeada uma nova procura, na tentativa de encontrar um substituto do mesmo.

5.3.2 Operações e lógica de negócio

O componente ***SIAT Operations*** é um ponto central na arquitectura do sistema, como tal, faz sentido que seja este componente a disponibilizar as operações do mesmo.

As operações disponibilizadas por este serviço podem ser divididas em duas categorias: *input* e *output*. As operações da categoria *input* são as que fornecem informação ao sistema e as de *output* utilizam o sistema para informar os utilizadores.

Neste momento existem apenas três operações que fornecem *input* ao sistema: duas operações para analisar a informação recolhida nos utilizadores, uma síncrona e outra assíncrona; e outra operação que permite enviar um alerta explicito de uma ocorrência. Quanto às duas primeiras, na operação síncrona o processamento dos dados é executado imediatamente naquele fio de execução, na assíncrona, a informação é colocada numa fila de mensagens e é tratada posteriormente por uma instância

do componente ***APW***. O algoritmo usado será explicado detalhadamente na Subseção 5.3.3.

As operações de *output* também são apenas três: duas operações para listar as ocorrências, uma listagem total e outra que filtra por estrada; e uma operação que permite obter a estrada mais próxima, mediante a apresentação das coordenadas geográficas. Esta última foi desenvolvida apenas para melhorar a qualidade da informação prestada ao utilizador, uma vez que lhe pode ser útil saber qual a estrada em que se encontra. No futuro, serão criadas mais operações para listar as ocorrências, com mais hipóteses de filtragem, e removida a operação que lista a totalidade das ocorrências.

A implementação destes métodos tem em consideração as obrigações e objectivos de cada um dos restantes componentes do ***SIAT Core***. Essa implementação deve utilizar, para cada método, o componente mais adequado de modo a oferecer as funcionalidades para que foi criado.

5.3.3 Algoritmo de análise da informação recolhida

O algoritmo que analisa a informação recebida dos dispositivos móveis é o mais complexo de toda a solução. Este algoritmo lida sobretudo com o tipo ***Occurrence***, adicionando e removendo objectos deste tipo e manipulando alguns dos seus campos, mais especificamente o ***precision*** e o ***intensity***.

Inicialmente, este algoritmo começa por ler toda a informação recebida determinando se existe um padrão, que denuncie uma possível ocorrência.

Em caso afirmativo verifica-se se a ocorrência já existe, caso isso aconteça os valores da precisão e intensidade são actualizados. No caso da ocorrência ainda não existir, uma nova é criada e inserida no sistema. Esta nova ocorrência é criada com um valor de precisão médio-baixo, para que não seja imediatamente considerada como uma ocorrência válida e, simultaneamente, possa ser removido do sistema com alguma facilidade, caso se verifique que é um falso positivo.

Quando uma nova ocorrência é detectada, esta é, simultaneamente, inserida no sistema de dados central e enviada para os sistemas de notificação dos dispositivos móveis (este assunto será abordado com maior detalhe na Subseção 5.4.3).

Mesmo quando a informação recebida não denuncia uma nova ocorrência, essa informação não pode ser descartada, ela é necessária para provar o contrário, ou seja, quando uma ocorrência fica solucionada ou quando é um falso positivo. Quando este caso sucede, os valores da precisão e intensidade são novamente actualizados e a ocorrência pode vir a ser removida do sistema.

Interpretar os valores da precisão e da intensidade

Os campos ***precision*** e ***intensity*** representam, respectivamente, a precisão e intensidade da ocorrência. Estes campos são actualizados ao longo do tempo e essas actualizações poderão provocar outras mudanças. Ambos os campos são armazenados num único *byte* (que pode representar números inteiros entre 0 e 255). É apenas na lógica de negócio que estes campos passam a ter algum

significado, e são criadas escalas de valor para cada um deles. Passa-se a explicar o significado dos valores de cada um destes campos.

Intensidade A intensidade é o valor médio entre as velocidades (em m/s, por esta ser a unidade do sistema internacional que mede a velocidade) registadas. Para calcular a média com precisão seria necessário manter um histórico de todos os dados recebidos, o que não faz sentido. Para eliminar esta necessidade foi decidido que este valor seria actualizado continuamente, sendo que as novas entradas teriam um peso de $\frac{1}{5}$ no valor final, chegando-se à formula apresentada em 5.1. Quando a ocorrência é criada tem o valor da intensidade igual à velocidade da informação que a gerou, caso o valor da intensidade exceda um determinado valor a ocorrência é removida.

$$I = \frac{(Ia \times 4) + In}{5} \text{ onde } Ia = \text{ Intensidade actual e } In = \text{ Nova intensidade} \tag{5.1}$$

Precisão A precisão representa o grau de veracidade de uma ocorrência. Este valor varia consoante os dados que chegam dos dispositivos móveis, caso esses dados confirmem a ocorrência o grau de precisão aumenta, no entanto, se os dados indicarem o oposto esse valor diminui. Surge assim uma questão: de que forma os dados confirmam, ou não, a ocorrência? Neste ponto, tomou-se o seguinte pressuposto, quando dois ou mais veículos passam com velocidades semelhantes (assumindo que as mesmas são baixas ou nulas) no mesmo trajecto então a probabilidade de existir uma ocorrência é grande. Caso se verifique que outros veículos percorrem esse mesmo trajecto com velocidades diferentes, então a ocorrência começa a perder veracidade.

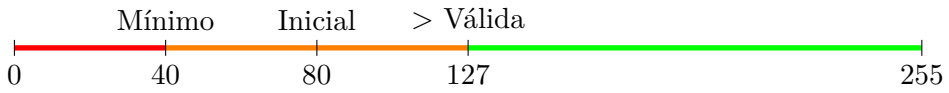

Figura 5.6: Escala com os diferentes valores de precisão.

A Figura 5.6 apresenta a escala dos valores dados ao campo precisão. Nesta escala, existem três categorias: não valida; em validação e validada. Quando a precisão toma um valor inferior a 40 considera-se não válida; caso tome um valor superior a 127 é valida; no intervalo compreendido entre estes dois valores a ocorrência é classificada como em validação. Todas as ocorrências são criadas com o campo precisão tomando o valor 80, pois este é um valor intermédio na escala de ocorrências em validação, o que permite, simultaneamente, uma rápida detecção de um falso positivo ou de uma confirmação da ocorrência.

Como foi apresentado anteriormente o campo intensidade armazena um valor médio das velocidades registadas. Utilizando este valor é possível fazer a diferença entre este e o valor da intensidade que está a ser analisado naquele momento ($|\Delta Intensidade|$). Seguindo o pressuposto previamente apresentado, então quanto mais este valor tender para zero maior será a veracidade da ocorrência.

Para actualizar o valor da precisão foi criada uma função linear baseada nesta conclusão, na qual o x é a diferença das velocidades e y é o valor da actualização. Nesta função, quanto mais o valor de x tender para zero maior é o valor de y, sendo que no ponto em que $x = 7$ (cerca de 25 km/h) o valor de y é 0. Considera-se este como sendo o ponto de equilíbrio, a partir do qual não há confirmação da ocorrência, mas o inverso. A fórmula encontrada é apresentada em Equação 5.2.

$$y = \frac{30}{-7} \times x + 30 \text{ onde } x = |\Delta Intensidade| \tag{5.2}$$

5.4 Módulo *mobile*

O módulo *mobile* é um conjunto de três componentes responsáveis por disponibilizar o serviço SIAT a utilizadores de dispositivos móveis, mais concretamente a dispositivos que possuam o sistema operativo *Windows Phone 7*. Na Figura 5.7 são visáveis os três componentes distintos, onde um deles é a aplicação móvel desenvolvida para este sistema operativo (***SIAT Phone App***) e os restantes são componentes responsáveis por disponibilizar o serviço SIAT da forma mais adequada.

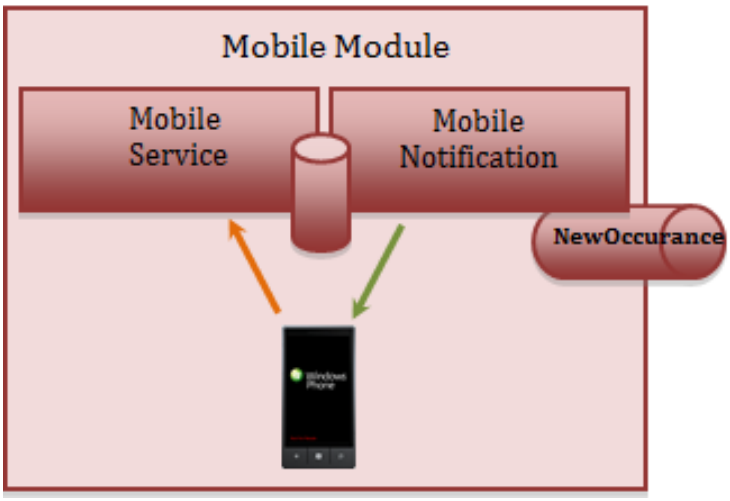

Figura 5.7: Arquitectura do módulo *mobile*.

A existência de dois componentes para fornecer este serviço, ao invés de um único, deve-se à utilização de notificações para informar os utilizadores da existência de novas ocorrências. Neste ponto existem duas soluções: ou o dispositivo faz *pooling* ao serviço, ou é informado acerca destas actualizações. A primeira opção é claramente mais ineficiente portanto optou-se por utilizar a segunda opção. Como nestes dispositivos não é permitida a comunicação directa (por exemplo através de *sockets*), o mecanismo pela plataforma para este tipo de comunicação chama-se *push notifications*.

Estes componentes foram desenvolvidos tendo em conta os requisitos da plataforma *WP7*, estando intrinsecamente comprometidos com a mesma, formando uma dependência indesejável. Foi equacionada a hipótese de construir este módulo abrangendo as restantes plataformas móveis, no entanto, o

estudo prévio realizado às mesmas (Secção 2.1) demonstrou que existem muitas diferenças entre estas. Uma solução deste género iria ter um grau de complexidade acrescido que neste momento não faz sentido. Assim sendo considera-se este ponto como uma possível funcionalidade no desenvolvimento futuro deste sistema.

A solução implementada, adiciona uma nova entidade, de nome ***Subscription*** (ver Figura 5.8), que representa a intensão de um dispositivo móvel em ser informado de cada vez que acontecer uma nova ocorrência, numa determinada estrada. Para persistir esta informação e de modo a que esta seja utilizada em paralelo por vários processos, recorreu-se a uma base de dados. Como o acesso aos dados é idêntico para ambos os componentes, foi posteriormente criada uma biblioteca (***WP7 Shared***) que agrega as funcionalidades que são partilhadas pelos dois componentes, neste caso a camada de acesso aos dados das subscrições.

5.4.1 WP7 Shared

Como existem várias funcionalidades partilhadas pelos dois componentes que fornecem o serviço aos dispositivos móveis com *WP7*, faz todo o sentido confinar a implementação dessas funcionalidades, numa única biblioteca partilhada por ambos.

A principal funcionalidade partilhada é o acesso aos dados das subscrições existentes. Para aceder a estes dados foi novamente criada uma *DAL*, desta vez um pouco mais simplista dada a natureza do problema também ser muito especifica. O desenvolvimento desta *DAL*, seguiu os mesmos parâmetros já utilizados noutros componentes do mesmo tipo, ou seja, foi utilizada a *ADO.Net Entity Framework* para mapear a entidade da base de dados num modelo, bem como para gerar as operações CRUD sobre essa entidade. Para não existirem dependências são novamente usados os padrões *DTO*, *Data Mapper* e *POCO*. A Figura 5.8 apresenta um diagrama *UML* com as classes criadas no âmbito desta *DAL*.

A entidade ***Subscription*** é composta apenas por dois campos: o identificador da estrada sobre a qual se deseja receber informação e um *uri*. Este ultimo é fornecido pelo serviço de *Push Notifications* gerido pela Microsoft, o qual será, posteriormente, utilizado para comunicar com este serviço e, consequentemente, com o dispositivo móvel.

5.4.2 WP7 SIAT Service

O componente ***WP7 SIAT Service*** é o ponto de entrada dos utilizadores, de dispositivos móveis com *WP7*, no serviço SIAT.

Para oferecer as funcionalidades do serviço, este componente disponibiliza as suas operações através de um *web service*, criado com recurso à *framework WCF*. Ao contrário dos outros serviços *WCF*, também criados nesta solução, este não é *self-hosted* numa aplicação do tipo consola. A classe que implementa este serviço foi criada num ficheiro com extensão *.svc*, podendo ser alojada em qualquer

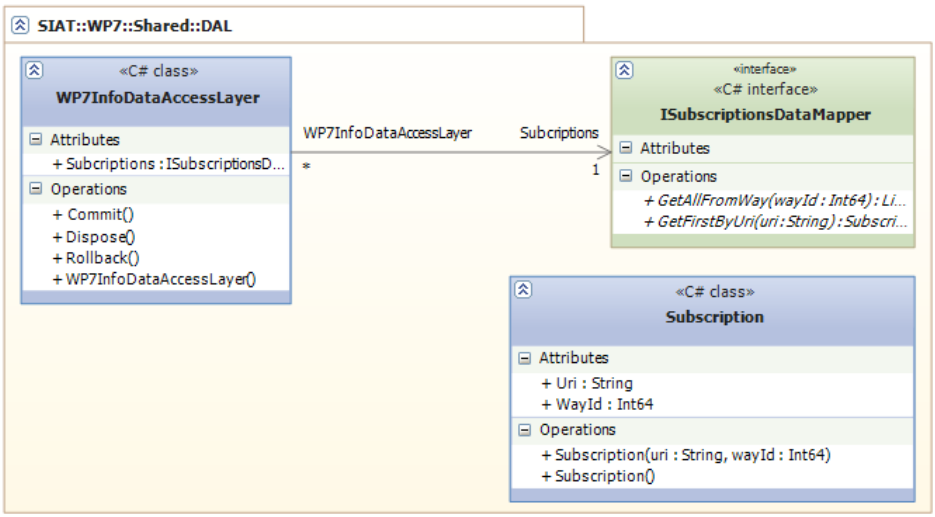

Figura 5.8: Diagrama *UML* da *DAL* que acede às subscrições.

servidor *web IIS*. Esta opção justifica-se pela maior facilidade em expor os serviços fora da rede local, através da utilização deste mecanismo, graças às potencialidades oferecidas pelo *IIS*.

Este componente é, primeiro que tudo, um cliente do serviço SIAT, como tal, utiliza o componente ***SIAT Operations*** para comunicar com este serviço. Entre as funcionalidades disponibilizadas incluem-se, não apenas, as que já estão implementadas nesse componente, mas também tudo o que está relacionado com a manipulação das subscrições. O contrato deste serviço é apresentado na Listagem 7.

```
1  [ServiceContract]
2  public interface ISIATServiceWP7
3  {
4          [OperationContract]
5          void SendPositionInformation(List<PositionInfo> listPosition);
6
7          [OperationContract]
8          void Alert(Alert alert);
9
10         [OperationContract]
11         List<Occurrence> ListAllOccurrences();
12
13         [OperationContract]
14         List<Occurrence> ListAllOccurrencesInWay(long wayId);
15
16         [OperationContract]
17         Way CurrentWay(double lat, double lon);
18
19         [OperationContract]
20         void CreateOrUpdateOccurrencesSubscription(Uri uri, long wayId);
21
22         [OperationContract]
23         void ChangeOccurrencesSubscriptionUri(Uri oldUri, Uri newUri);
24
25         [OperationContract]
26         void DeleteOccurrencesSubscription(Uri uri);
27 }
```

Listagem 7: Contrato do serviço SIATServiceWP7.

O conjunto de operações disponibilizadas por este serviço podem ser divididas em três categorias: input, output e as que gerem as subscrições. Tal como o nome sugere, as duas primeiras categorias funcionam apenas como fachada para a utilização dessas mesmas operações do componente ***SIAT Operations***.

5.4.3 WP7 Notification Service

As notificações assíncronas são um mecanismo muito importante de comunicação com dispositivos móveis, pois permite poupar os recursos dos mesmos em *poolings* desnecessários. O componente ***WP7 Notification Service*** é o responsável por estas notificações aos dispositivos com o sistema operativo WP7. Neste sistema operativo, o serviço de notificações tem o nome de *Push Notifications* e é utilizado da seguinte forma: o dispositivo móvel requisita um canal de mensagens ao serviço que por sua vez devolve um *uri*. Este deve ser utilizado como *endpoint* no envio das notificações, obedecendo a um protocolo específico.

Uma das limitações deste protocolo, é que a mensagem não pode ter mais de 1500 *bytes*, limitando a quantidade de informação que pode ser inserida na mensagem, uma vez que a mesma já contém cabeçalhos que, por si só, já ocupam muito espaço.

O modo de funcionamento deste componente consiste em receber actualizações de novas ocorrências, verificar se existe alguma subscrição para a estrada com esse identificador e depois envia uma notificação para o dispositivo. Dada a limitação previamente mencionada, a mensagem enviada não pode incluir o conteúdo da totalidade das ocorrências. A solução encontrada passa por enviar apenas o identificador da estrada, para que o dispositivo possa requisitar todas as ocorrências sobre a mesma. Esta solução é genérica para todas as situações mas, no limite, este sistema deveria funcionar de forma híbrida, ou seja, sempre que houvesse espaço suficiente seria enviada a totalidade da informação da ocorrência.

A informação sobre novas ocorrências é recebida através de uma fila de mensagens, alojada no *Microsoft Message Queuing* local com o nome *OccurrencesUpdate_WP7*. Como foi mencionado na Subseção 5.3.3, este algoritmo envia uma mensagem por cada vez que insere uma nova ocorrência. Ao receber um nova mensagem desencadeia-se o processo de notificação.

5.4.4 WP7 SIAT Application

O sistema operativo *Windows Phone 7* foi o escolhido para ser utilizado neste projecto, como tal, foi criada uma aplicação para este tipo de dispositivos, que utiliza as funcionalidades oferecidas. Esta aplicação é responsável por explorar e evidenciar todas as potencialidades deste projecto.

A aplicação terá de apresentar a informação mais actualizada das ocorrências e, em simultâneo, recolher dados para entregar ao sistema central. A aplicação tira partido de algumas capacidades avançadas deste tipo de dispositivos, nomeadamente, o acesso à localização (*GPS*) e à *internet*.

São também disponibilizadas algumas informações ao condutor, tais como a estrada e velocidade actuais, bem como algumas opções de configuração da aplicação.

Os dois modos de visualizar a informação das ocorrências são:

Lista Simples lista que apresenta o conjunto das ocorrências (Figura 5.9), e na qual é possível perceber o nome da estrada e a gravidade da situação, através de um esquema de cores;

Mapa O mapa, no qual são mostrados os alertas, é um controlo *Silverlight* já embutido na *framework* do *Windows Phone 7*, o que permitiu uma integração eficiente. Esta vista permite ter uma noção mais intuitiva da localização geográfica da ocorrência. Também nesta situação é usado um esquema de cores para dar a noção de gravidade da ocorrência (Figura 5.10).

Por fim, resta apenas um formulário no qual é possível o utilizador fazer algumas configurações (Figura 5.11).

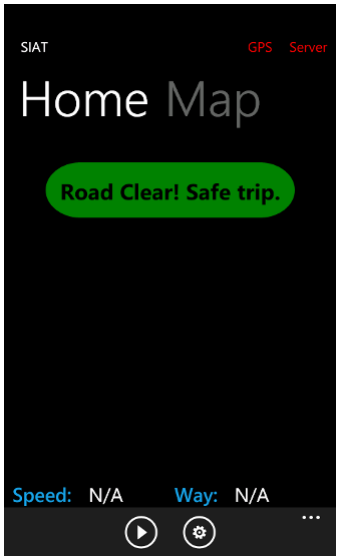

Figura 5.9: Listagem.

Figura 5.10: Mapa.

Figura 5.11: Configurações.

As configurações disponíveis são as seguintes:

- Escolha da métrica utilizada para quantificar a velocidade;
- Autorização para continuar a correr a aplicação quando o ecrã de bloqueio está activo;
- Autorização para impedir o ecrã de bloqueio de ser accionado automaticamente.

Estas duas autorizações são necessárias para que a aplicação seja aprovada no *MarketPlace*, uma vez que as respectivas acções têm um impacto negativo no tempo de vida da bateria, à qual o utilizador deve estar informado.

A ***WP7 SIAT Application*** está periodicamente a ser actualizada com os dados provenientes do *GPS* do dispositivo. Esses dados são armazenados e, posteriormente, enviados para o sistema central. Simultaneamente, é requisitada a estrada actual do veiculo, mediante a apresentação das suas mais recentes coordenadas geográficas. Esta informação irá ser necessária para obter as ocorrências de uma forma mais precisa, recebendo apenas a informação relativa à estrada actual.

Logo que é identificada a estrada actual do veiculo, é então criada uma subscrição no sistema de notificações. Daí para a frente, sempre que a estrada muda, é feita a correspondente alteração no sistema de notificações. Quando o telefone desliga ou a monitorização é terminada, por solicitação do utilizador, a subscrição é removida.

Por outro lado, sempre que exista uma nova ocorrência para a subscrição inscrita o telefone é informado através do canal de notificações (*Push Notifications*), previamente criado para o efeito.

As configurações da aplicação são guardadas de forma persistente, para que o utilizador não necessite de configurar a aplicação sempre que a utiliza.

5.5 Aplicação *web*

A aplicação *web* tem por objectivo ser mais uma forma de contacto com o utilizador deste sistema. Foi construída em *ASP.Net MVC 3* e encontra-se alojada no servidor *IIS*. Na arquitectura deste sistema, esta aplicação encaixa-se como mais um módulo externo (idêntico ao ***SIAT Mobile***), portanto também utiliza a componente ***SIAT Operations*** para comunicar com os restantes componentes.

Nesta aplicação *web*, o utilizador pode consultar as ocorrências das mesmas formas já utilizadas para expor essa informação na aplicação *mobile*, ou seja, através de uma lista ou de um mapa.

No modo em lista, as informações são mais detalhadas, pois é possível apresentar a descrição da ocorrência e a data da última actualização (ver Figura 5.12). O utilizador pode ainda filtrar as ocorrências por uma determinada estrada, bastando depois clicar no respectivo nome dessa estrada, e será redireccionado para outra página com informação mais precisa.

Na vista em mapa, as actualizações estão dispersas, através de *pushpins*, num mapa interactivo construído em *Silverlight*. A Figura 5.13 apresenta a vista das ocorrências, onde é possível identificar alguns *pushpins* a marcar ocorrências.

Nesta aplicação é também possível criar novas ocorrências, existindo uma página dedicada a esse efeito, a qual foi desenvolvida a pensar em serviços, como por exemplo as rádios, que possuem informações de trânsito e poderão partilha-las directamente neste sistema. Para que esta funcionalidade fosse correctamente implementada seria necessário um sistema de autenticação e autorização que ainda não existe nesta aplicação. Ainda assim, o sistema permite a criação de ocorrências desta forma, mas no futuro essa funcionalidade só estará reservada a utilizadores com um papel especial no sistema.

Figura 5.12: Aplicação Web - Lista das ocorrências.

Esta aplicação é composta apenas por dois *controllers*, o *Home* e o *Occurrence*. O primeiro não tem nada de especial, e as vistas mostram apenas informação estática. Já no segundo é onde se encontra a maioria da lógica, sendo o *controller* responsável por lidar com tudo o que esteja relacionado com ocorrências: listas, criar, ver detalhe, remover, Etc.

Para oferecer estas funcionalidades a aplicação utiliza o componente ***SIAT Operations***, tal como foi mencionado anteriormente. A instanciação de um objecto deste tipo requer algum tempo, pois nesse processo é feita a pesquisa pelos serviços do ***SIAT Core***. É preciso ter em atenção esta situação, uma vez que não faz qualquer sentido criar um objecto deste tipo directamente em cada *controller* pois estaria a ser criado constantemente, agravando o desempenho do sistema. A solução implementada cria uma única instância desta entidade no *Application_Start* e guarda essa mesma instância no *application state* global, tal como é mostrado na Listagem 8. Nos *controllers* ocorre o processo inverso, ou seja, obtém-se a instância do objecto e a partir daí utiliza-se normalmente (ver a Listagem 9). Como este componente não mantém estado partilhado, existem apenas dois *proxys* para aceder aos serviços, não sendo necessário proteger os acessos concorrentes.

```
1  protected void Application_Start()
2  {
3          ...
4
5          SIATOperations operations = new SIATOperations();
6          this.Application.Add("siat",operations);
7  }
```

Listagem 8: Iniciação da instância do tipo ***SIAT Operations***.

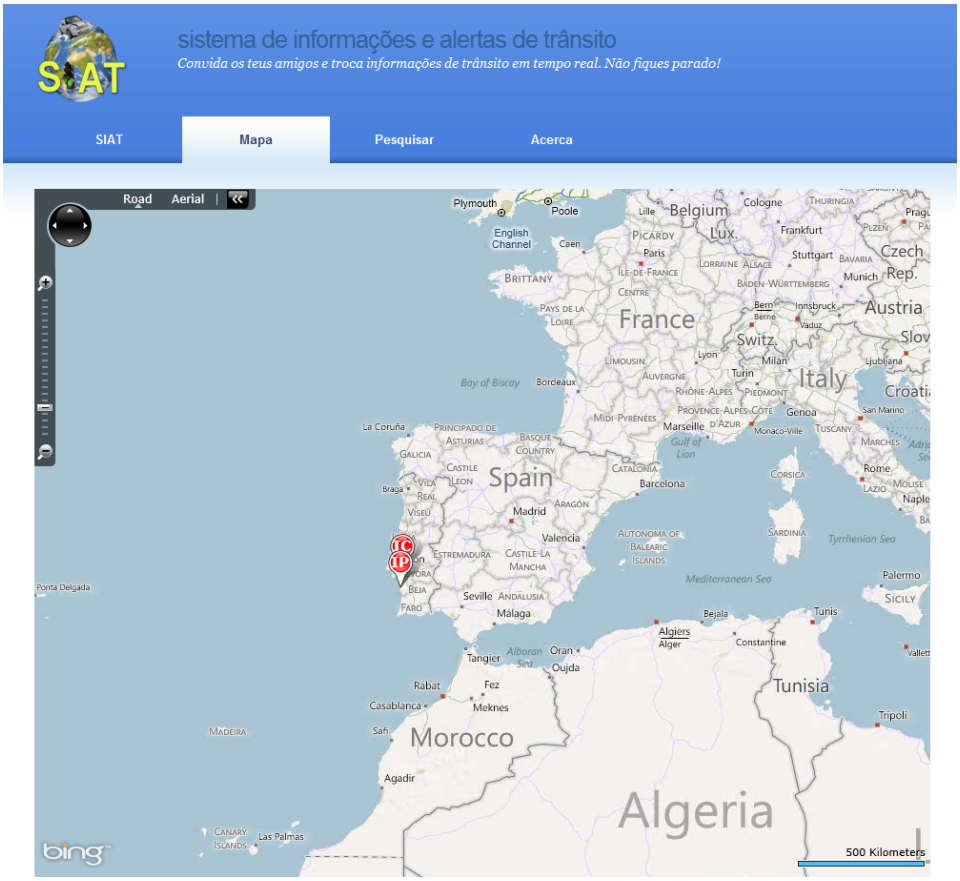

Figura 5.13: Aplicação Web - Vista em modo de Mapa.

```
1  public ActionResult Occurrences()
2  {
3          var siatOperations = (SIATOperations)this.HttpContext.Application["siat"];
4          var list = siatOperations.ListAllOccurrences().Select(o => new Occurrence()
5                  {
6                          WayName = o.WayName,
7                          Latitude = Convert.ToDecimal(o.Latitude),
8                          Longitude = Convert.ToDecimal(o.Longitude),
9                          Intensity = o.Intensity.ToString(),
10                         Id = o.Id
11                 });
12         return Json(list, JsonRequestBehavior.AllowGet);
13 }
```

Listagem 9: Uso do ***SIAT Operations*** no *controller*.

Esta página foi intencionalmente deixada em branco.

6 Conclusões

Este documento apresenta a arquitectura e implementação do Sistema de Informações e Alertas de Trânsito desenvolvido neste projecto. Com este sistema é possível proporcionar à comunidade uma plataforma de troca de informações de trânsito, em tempo-real, para que o utilizador possa tomar decisões o mais cedo possível.

O estudo inicial às plataformas já existentes foi relevante na elaboração deste projecto, uma vez que permitiu enriquecer o conhecimento nesta área, tanto a nível tecnológico como das características de negócio.

Uma das principais dificuldades no desenvolvimento deste projecto foi encontrar uma solução de *geocoding* que cumprisse, na totalidade, os requisitos necessários. Esta procura revelou-se infrutífera, no entanto, permitiu conhecer as diversas soluções existentes, o que se revelou decisivo para a solução final, uma vez que permitiu conhecer o *Open Street Maps*. No final, acabou por ser desenvolvida uma solução de raíz, o que não estava no planeamento inicial. A solução desenvolvida é autónoma e está disponibilizada à comunidade como um projecto *open-source*.

A principal funcionalidade deste sistema consiste em informar os seus utilizadores acerca de novas ocorrências. Uma ocorrência é tudo o que faça com que um automobilista circule a baixa velocidade, o que pode acontecer por vários motivos, tais como: acidente, avaria, obras ou trânsito intenso. Uma das características deste serviço é a capacidade de identificar o sentido do veiculo na estrada, permitindo que a informação disponibilizada seja mais acertiva.

Este sistema é composto por vários componentes, divididos em duas categorias: os do núcleo (***SIAT Core***) e os módulos externos.

Os componentes da primeira categoria comunicam entre si com o objectivo de oferecer as funcionalidades desejadas. Para facilitar a comunicação e integração entre estes vários módulos, a arquitectura desenvolvida baseia-se em *SOA*. Para ajudar a implementar uma solução deste tipo, recorreu-se ao protocolo *WS-Discovery*, que é utilizado no anuncio e na descoberta de serviços numa rede local. Este protocolo é um *standard* da *OASIS*, existindo uma implementação do mesmo na *framework*

WCF, que foi utilizada para construir os vários serviços deste conjunto de componentes.

Na segunda categoria incluem-se os componentes responsáveis por disponibilizar o serviço ao consumidor final. O sistema foca-se, sobretudo, em oferecer estas funcionalidades aos utilizadores de dispositivos móveis, como tal, o principal módulo externo é o ***SIAT Mobile***, existindo, no entanto, também um *website* no qual o utilizador pode consultar as ocorrências.

As principais funcionalidades, enumeradas nos objectivos gerais do projecto, foram implementadas com sucesso. Foi criada uma plataforma, que permite a um utilizador ser notificado de novas ocorrências e, em simultâneo, fornecer as suas informações a essa mesma plataforma. O sistema funciona de forma dinâmica, utilizando a informação recebida para, continuamente, validar ou invalidar as diversas ocorrências. Pode-se assim afirmar que este sistema é, apenas, uma prova de conceito de um futuro sistema mais complexo, que disponibilizará várias outras funcionalidades, como por exemplo: a criação de uma rede social de partilha de informações de trânsito ou sistema de sugestões de poupança baseado na condução do utilizador.

6.1 Análise crítica da solução

Concluída a fase de desenvolvimento deste projecto, e apesar de terem sido cumpridos os principais requisitos, é importante realizar uma análise crítica sobre todos os aspectos positivos e negativos, explicitando os pontos fortes da solução e os pontos que necessitam de uma abordagem futura.

Pontos fortes

Arquitectura *SOA* Um dos pontos fortes desta solução é o facto da sua arquitectura ser baseada nos padrões de desenho *SOA*. As decisões tomadas no desenho desta arquitectura permitem que, no futuro, a inserção de novos componentes possa ser feita de forma transparente. Para isso, contribuiu em muito a utilização da mais recente tecnologia, mais precisamente o *WCF Discovery* da *framework WCF*.

Utilização de padrões de desenho A utilização de padrões de desenho, no desenvolvimento desta solução, permite melhorar o desempenho da mesma, a legibilidade do código e, sobretudo, impede que existam algumas dependências das tecnologias. Esta característica está mais patente nos componentes que utilizam camadas de acesso a dados, onde foram utilizados vários padrões de desenho, tais como o *data mapper* ou *DTO*. Desta forma, no futuro, qualquer uma das camadas que integram estes componentes pode ser alterada, desde que mantenha a utilização de um conjunto de contratos, que as restantes camadas não serão afectadas.

Solução de *Geocoding* Foi desenvolvida, de raíz, uma solução que permite colmatar as necessidades de *geocoding* do projecto SIAT. Esta solução utiliza a tecnologia mais recente, tanto ao nível dos servidores de base dados como das *frameworks* utilizadas no desenvolvimento.

Tal como já foi mencionado, esta solução é totalmente autónoma e está disponível, para ser utilizada pela comunidade *OSM*, como mais uma solução de *geocoding*, idêntica ao *Nominatim*.

Análise à informação recebida pelo utilizador A solução implementada para analisar a informação recebida pelo utilizador é bastante interessante, pois revela a preocupação dada ao desempenho global do sistema.

Tendo em conta o problema de desempenho encontrado, a solução encontrada passa por executar assincronamente essa operação. Para isso, utiliza-se uma fila de mensagens para colocar todas as análises pendentes que, posteriormente, um processo dedicado irá proceder à execução das mesmas.

Esta solução tem a vantagem de ser altamente dinâmica, podendo ser criadas uma ou mais instâncias do mesmo processo consoante o nível de mensagens pendentes.

Pontos fracos

Algoritmo para balanceamento de carga O algoritmo usado para balanceamento de carga entre bases de dados, no componente ***OSM Querys***, é claramente insuficiente. Este ponto terá de ser alterado, criando um novo algoritmo que implemente esta funcionalidade com um outro grau de exigência.

Testes práticos O principal ponto fraco deste sistema é não ter sido suficientemente testado. Para testar um sistema deste género existem duas hipóteses: ou se vai para a estrada utilizar a aplicação ou é criada uma aplicação que simula essa utilização. A primeira hipótese foi implementada, mas apenas com um ou dois utilizadores, o que não é significativo. Quanto à segunda hipótese, infelizmente, o tempo não foi suficiente para o seu desenvolvimento.

6.2 Trabalho Futuro

Integração na plataforma *Windows Azure*

Uma das possíveis tarefas futuras é migrar toda a plataforma para a *cloud*, mais precisamente para o *Windows Azure*. A escolha deste fornecedor de serviços, em detrimento de outros, deve-se apenas à facilidade de integração, uma vez que toda a implementação deste projecto assenta em tecnologias *Microsoft*.

A migração desta plataforma já foi, em parte, testada, porque foi utilizada uma base de dados no *SQL Azure* para albergar o modelo de dados do *OSM*. Os testes realizados demonstram que, comparativamente, uma base de dados idêntica num servidor local tem um desempenho semelhante à solução com *SQL Azure*.

Quanto ao restantes componentes, eles são compatíveis com o *Windows Azure*, ficando apenas a dúvida se serão executados como *worker* ou *web roles* [19].

A integração deste sistema na plataforma *Windows Azure* faz todo o sentido porque, sendo este um sistema aberto às massas, é essencial algum tipo de dinamismo no que toca à capacidade de processamento.

Integração na rede social *Facebook*

Um dos objectivos mais imediatos deste sistema é criar uma rede social, que permita alguma interacção entre utilizadores na partilha de informações de trânsito. Sendo o *Facebook* uma rede social largamente implementada, e que disponibiliza uma *API* para que se possa obter alguma informação dela, a rede que venha a ser criada deverá integrar com essa rede social.

Desta integração, o importante para o sistema SIAT é ficar com a informação dos amigos de cada utilizador, de modo a verificar se esses amigos já utilizam esse mesmo sistema. Quando isso acontece, são disponibilizados mecanismos de troca directa de informação entre eles. Quando o utilizador assim o permitir, os seus amigos terão acesso à sua informação geográfica actualizada. A autenticação pode igualmente ser uma funcionalidade fornecida pelo *Facebook*, permitindo que utilizador recorra à mesma conta para aceder aos dois serviços.

O sistema deve, no entanto, manter a opção de contribuir de forma anónima, sendo que nestas situações o utilizador não usufrui, como é natural, do conjunto de funcionalidades criadas por esta integração.

Cliente para o sistema operativo *Android*

Sendo o sistema operativo *Android* o mais utilizado, actualmente, do mercado, deverá ser criado um cliente do serviço SIAT para este vasto número de utilizadores.

A aplicação criada deverá ser, de certa forma, idêntica à que já existe para a plataforma *Windows Phone 7*. Nessa adaptação, é importante manter os mesmos níveis de qualidade exigidos na aplicação para *WP7*, no que toca ao uso das mais recentes tecnologias, permitindo poupar os recursos destes dispositivos.

Data Warehouse **com o histórico da informação recolhida**

Outra das propostas de trabalho futuro que se poderão considerar no âmbito deste projecto, é o desenvolvimento de um *data warehouse* que armazene a informação histórica recebida dos vários utilizadores.

Uma ferramenta deste género irá permitir vários tipos de funcionalidades sendo que, a mais imediata, é a possibilidade de prever, *à priori*, possíveis ocorrências, através da identificação de zonas que, historicamente, costumam concentrar ocorrências em determinados períodos do dia. Seria possível, inclusive, identificar possíveis pontos negros das estradas.

Tamanha quantidade de informação deste género pode ainda ser útil para vários estudos estatísticos, tanto a nível ambiental como antropológico.

Esta página foi intencionalmente deixada em branco.

Referências

[1] ANACOM, "Serviços móveis - informação estatística - 2 semestre," ANACOM, Tech. Rep., 2010.

[2] IDC. (2010) European mobile phone tracker. [Online]. Available: http://www.idc.pt/press/pr_2010-09-13.jsp

[3] T. Tom. (2007) How tomtoms hd traffic and iq routes data provides the very best routing. [Online]. Available: http://www.tomtom.com/lib/doc/download/HDT_White_Paper.pdf

[4] C. Int. (2008) Finally: Traffic data that you can trust. [Online]. Available: http://cellint.com/traffic_data/TrafficSenseBrochure.pdf

[5] G. Inc. (2011) Android. [Online]. Available: http://www.android.com/

[6] D. Bornstein. (2011) Dalvik virtual machine insights. [Online]. Available: http://www.dalvikvm.com/

[7] Google. (2011) Google maps. [Online]. Available: http://code.google.com/intl/en/apis/maps/

[8] Microsoft. (2011) Bing maps. [Online]. Available: http://www.microsoft.com/maps/developers/

[9] Sapo. (2011) Sapo mapas. [Online]. Available: http://api.mapas.sapo.pt/

[10] OSM. (2011) Open street maps. [Online]. Available: http://www.openstreetmap.org/

[11] ——. (2011) Mapnik. [Online]. Available: http://wiki.openstreetmap.org/wiki/Mapnik

[12] MapQuest. (2011) Nominatim. [Online]. Available: http://wiki.openstreetmap.org/wiki/Nominatim

[13] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2003.

[14] T.Brisco. (1995) Rfc 1794 - dns suport for load balancing. [Online]. Available: http://tools.ietf.org/html/rfc1794

[15] T. Earl, *SOA - Principles of Service Design*. Prentice Hall, 2007.

[16] Microsoft. (2009) Discovery find and findcriteria. [Online]. Available: http://blogs.msdn.com/b/discovery/archive/2009/06/09/discovery-find-and-findcriteria.aspx

[17] ——. (2011) Task parallel library - tpl. [Online]. Available: http://msdn.microsoft.com/en-us/library/dd537609.aspx

[18] ——. (2011) Msmq. [Online]. Available: http://msdn.microsoft.com/en-us/library/ms711472.aspx

[19] C. Hay and B. Prince, *Azure in Action*. Manning, 2010.