INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrónica

e Telecomunicações e de Computadores

Observador de Mobilidade Pessoal

EDISON DE MELO SPENCER LOPES DOS SANTOS

(*Licenciado*)

Trabalho de Projeto realizado para obtenção do grau de

Mestre em Engenharia Informática e de Computadores

Orientador:

Doutor Porfírio Pena Filipe, ISEL

Júri:

Presidente:

Mestre Pedro Alexandre Seia Cunha Ribeiro Pereira, ISEL

Vogais:

Doutor João Carlos Amaro Ferreira, ISEL

Doutor Porfírio Pena Filipe, ISEL

Dezembro de 2013

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Área Departamental de Engenharia Eletrónica

e Telecomunicações e de Computadores

Observador de Mobilidade Pessoal

EDISON DE MELO SPENCER LOPES DOS SANTOS

(*Licenciado*)

Trabalho de Projeto realizado para obtenção do grau de

Mestre em Engenharia Informática e de Computadores

Orientador:

Doutor Porfírio Pena Filipe, ISEL

Júri:

Presidente:

Mestre Pedro Alexandre Seia Cunha Ribeiro Pereira, ISEL

Vogais:

Doutor João Carlos Amaro Ferreira, ISEL

Doutor Porfírio Pena Filipe, ISEL

Dezembro de 2013

Resumo

Atualmente existe uma oferta variada de meios de transporte, contudo, a escolha do transporte mais adequado, para uma determinada viagem, depende de diversos fatores, e.g. económicos (e.g. preço dos combustíveis) ou ambientais (e.g. emissão de dióxido de carbono).

Sendo assim, pretende-se no âmbito deste projeto, desenvolver um sistema computacional para gerir (recolher e manter) a informação sobre os hábitos de mobilidade dos viajantes. Este sistema privilegia a utilização de dispositivos computacionais móveis, preferencialmente *smartphones*, considerando abordagens tecnológicas emergentes.

Este projeto tem como principal objetivo a recolha de informação antes, durante e após uma determinada viagem, com a intenção de registar os hábitos de mobilidade de um viajante, tentando minimizar os recursos computacionais e humanos exigidos. A informação recolhida é gerida centralmente com o propósito de ser disponibilizado seletivamente a quem interessar (e.g. operadores ou autoridades de transporte) ou a ser processada (e.g. gerar sugestões de transporte).

Para a implementação deste projeto, é utilizada a tecnologia Android para o desenvolvimento de uma aplicação móvel destinada à recolha de experiências de mobilidade caraterizadas por dados obtidos pelos sensores existentes nos dispositivos móveis. Adicionalmente, foi desenvolvido um serviço *web* para suportar a submissão das experiências de mobilidade que posteriormente podem ser processadas para, por exemplo, determinar o transporte envolvido nas experiências.

Palavras-Chave: Mobilidade sustentável; dispositivo móvel; mobilidade pessoal; viagem; Android; serviço *web*.

Abstract

Nowadays, there is a huge variety of transport means, however, the choice of the most suitable transport mean, for a particular trip, depends on a variety of factors, e.g. economical (e.g. the price of fuel) or environmental (e.g. carbon dioxide emissions).

With that in mind, this project, aims to develop a computational system to manage (retrieve and store) the information about the mobility habits of travellers. This system will favour the usage of mobile devices, such as smartphones, since lately these have been considered emerging technological approaches due to the high penetration in society.

The main goal of this project will be the information retrieval before, during and after a mobility experience. The information retrieved will later be stored automatically, thus reducing the human and computational resources needed for such tasks. The information retrieved is also processed centrally at the repository, and afterwards made available selectively to third parties (e.g. transport operators or transport authorities) or to be processed (e.g. create transport suggestions or path studies).

For the development of this project, Android technology is used to develop a mobile application for gathering mobility experiences, which are characterized by data obtained from the sensors that are present on mobile devices. In addition, a web service was developed to support submission of the mobile experiments, mobile experiments that will be later processed in order to, for example, determine the type of transport that was used in the mobility experiment.

Keywords: Sustainable mobility; mobile devices; personal mobility; trip; Android; web service.

Agradecimentos

Ao meu orientador Porfírio Filipe, pela disponibilidade dedicada, assim como pelos momentos de conversa proporcionados de modo a esclarecer as ideias.

Aos meus pais, Manuel e Deolinda, por todo o apoio dado nesta reta final e por fazerem de mim o que sou hoje.

À minha namorada, Joanette Eveline, por todos os momentos dispensados e pela força dada.

Acrónimos

AR	*Activity Recognition*
CSV	*Comma Separated Values*
DM	*Data Mining*
FK	*Foreign Key*
GPS	*Global Positioning System*
KML	*Keyhole Markup Language*
k-NN	*k-Nearest Neighbour*
MVC	*Model View Controller*
OHA	*Open Handset Alliance*
PK	*Primary Key*
SVM	*Support Vector Machine*
UML	*Unified Modeling Language*
XML	*eXtensible Markup Language*
XSD	*XML Schema Definition*
XSLT	*eXtensible Stylesheet Language Transformation*

Convenções Tipográficas

De seguida, apresentam-se as convenções tipográficas adotadas na escrita deste documento:

i. Aplica-se ao texto normal a fonte *Times New Roman*, com o tamanho 12 e o espaçamento entre linhas de 1,5 cm

   Exemplo: Texto normal;

ii. Aplica-se (parêntesis curvos) a referências bibliográficas

   Exemplo: 2000 milhões de viaturas (1);

iii. Aplica-se o texto em *Itálico* a designações em inglês

   Exemplo: *Activity Recognition*;

iv. Aplica-se o texto em **Negrito** para a definição de acrónimos e siglas

   Exemplo: *Keyhole Markup Language* (**KML**).

Índice de Conteúdos

Índice de Figuras

Índice de Tabelas

1. Prólogo

Com uma sociedade moderna consumista e comodista, estima-se que o número de veículos existentes nos próximos anos atinja facilmente os 2000 milhões (1). Esses números podem ser considerados alarmantes, uma vez que atualmente, com um número inferior, estima-se que já são perdidas cerca de 38 horas por ano no trânsito (2), o equivalente a uma semana de trabalho perdido.

Com um tão elevado número de veículos nas estradas, se as infraestruturas que os suporta não forem otimizadas, isso traduzir-se-á em maiores tempos de espera, maior números de atrasos e uma maior ineficiência no geral. O que levará a uma maior frustração por parte dos cidadãos.

Esta tendência natural que se tem verificado com o aumento do número de veículos, tem contrariado a contínua subida do preço dos combustíveis, porque é cada vez maior a quantidade de pessoas que opta por ter um veículo próprio para as deslocações diárias, uma vez que isso lhes garante liberdade de movimento.

No Brasil, por exemplo, apesar de ter havido um crescimento demográfico de 13% entre 2003 e 2010 (3), o número de carros em circulação no mesmo período cresceu cerca de 66% (3). Isto porque, na cultura brasileira, o carro está associado a um certo estatuto social, o que não acontece em certos países europeus pois, apesar dos carros atribuírem um certo estatuto social, a consciencialização sobre as alterações climáticas são maiores, o que faz com que os transportes públicos ganhem um maior papel de destaque na sociedade.

O fato de se basear amplamente no nível de conforto como principal fator na escolha do meio de transporte para a realização de um determinado trajeto demonstra que nem sempre se escolhe o melhor meio de transporte para um determinado percurso. Isso porque podem existir outros meios de transporte que não sejam tão confortáveis, mas que permitam ser tanto ou mais eficientes, quer a nível da duração da viagem, como a nível dos efeitos nocivos para com o ambiente.

1.1. Motivação

Para que o utilizador, quando pretende efetuar um determinado trajeto, consiga escolher de entre todos os meios de transporte disponíveis para esse trajeto, o que melhor se adequa, este tem de ter consigo toda a informação necessária para tomar a decisão correta.

Contudo, a informação necessária para a tomada de tais decisões, não se encontra disponibilizada num único lugar. Assim sendo, demonstra-se ser de certa importância a sua existência de forma centralizada, o que permite que a tomada de decisões como, qual o meio de transporte a ser utilizado para um determinado percurso, seja feita de forma simples e rápida pelo utilizador.

1.2. Objetivos

O objetivo deste trabalho consiste na implementação e disponibilização de uma solução que efetue a monitorização de dados sensoriais dos hábitos de mobilidade dos utilizadores recorrendo a abordagens tecnológicas emergentes no âmbito das plataformas móveis, nomeadamente, os *smartphones*.

Os dados sensoriais recolhidos dos *smartphones* contendo informações sobre os hábitos de mobilidade dos utilizadores são posteriormente submetidos para um repositório central onde são analisados. Os resultados dessas análises são posteriormente utilizados para a criação de um sistema que possibilite auxiliar os seus utilizadores na tomada de decisões aquando da realização de novas experiências de mobilidade. Os resultados, provenientes da análise dos dados submetidos, são ainda disponibilizados a entidades que estejam interessadas, quer seja para o reprocessamento dos dados submetidos, quer seja para fazer uma análise das conclusões já efetuadas pelo sistema.

1.3. Organização

No capítulo 1, efetua-se uma introdução e descreve-se a motivação e os objetivos do trabalho proposto. No final é apresentada a organização do documento. Ao longo do capítulo 2, é efetuada uma descrição sobre o estado da arte. De seguida, no capítulo 3,

será descrito o estado atual da solução proposta, designadamente as decisões tomadas durante a realização da aplicação móvel e do repositório. No capítulo 4, será descrito o resultado dos testes efetuados a alguns utilizadores de teste de modo a tentar verificar se o algoritmo implementado no repositório determina corretamente os meios de transporte suportados. Por fim, no capítulo 5 serão apresentadas as conclusões e trabalho futuro a desenvolver no seguimento deste projeto.

2. Estado da arte

Neste capítulo, são abordados temas e conceitos relacionados com o desenvolvimento deste projeto.

2.1. Android

Desde a introdução do primeiro *smartphone* em 1992 (4), intitulado Simon, que o mercado dos dispositivos móveis tem vindo a observar uma evolução exponencial. Dados recentes indicam que, apesar do recente aparecimento de tais dispositivos, mais de 1000 milhões (5) de pessoas já possuem um *smartphone*, o que equivale a aproximadamente 14% da população mundial (6). Isto verifica-se uma vez que os dispositivos móveis possuem um alto nível de penetração na sociedade, pois permitem não somente uma comunicação simplificada, fácil personalização, como também facilitam certos aspetos da vida quotidiana (e.g. anotar eventos, gerir contatos, partilhar experiências) tudo num único lugar.

Os dispositivos móveis eram inicialmente comercializados com um sistema operativo proprietário. Assim, caso fosse necessário desenvolver aplicações para dispositivos produzidos por diversos fabricantes teria de ser realizado desenvolvimento à medida.

Contudo, desde Setembro de 2008 (7) que, com o lançamento do primeiro *smartphone* equipado com o sistema operativo Android (8) da Google (9), em parceria com a *Open Handset Alliance* (**OHA**) (10), que esta tendência sofreu uma gradual alteração.

Sendo o Android um sistema operativo *open-source*, este permite adaptações por parte de terceiros, tendo por isso vindo a ganhar grande aceitação por parte dos utilizadores e dos operadores móveis. Sensivelmente no mesmo período de surgimento do Android, surgiu também o iOS (11), sistema operativo da Apple (12) que, apesar de não ser *open-source*, disponibiliza componentes que permitem o desenvolvimento de novas

funcionalidades, o que lhe permitiu igualmente ser um protagonista de relevo no mercado dos *smartphones*.

Sistema Operativo	Número Vendas 3º Trimestre 2011	Quota de Mercado 3º Trimestre 2011	Número Vendas 3º Trimestre 2012	Quota de Mercado 3º Trimestre 2012	Crescimento Anual
Android	71,0	57,4%	136,0	75,1%	91,5%
iOS	17,1	13,8%	26,9	14,9%	57,3%
Blackberry	11,8	9,5%	7,7	4,3%	-34,3%
Symbian	18,1	14,6%	4,1	2,3%	77,3%
Windows Phone 7	1,5	1,2%	3,6	2,0%	140,0%
Linux	4,1	3,3%	2,8	1,5%	-31,7%
Others	0,1	0,1%	0,0	0,0%	-100,0%
Total	123,7	100,0%	181,1	100,0%	46,4%

Tabela 1 - Quota de mercado dos Sistemas Operativos (milhões)

Como se pode ver pela Tabela 1, que ilustra a percentagem dos sistemas operativos existentes no mercado, o Android e o iOS possuíam, no terceiro trimestre de 2012, 90% do mercado. Apesar do iOS ser um sistema operativo com muita aceitação à semelhança do Android, os preços praticados pela Apple são mais elevados, pelo que, de momento, tem garantido uma vantagem substancial ao Android pois, os preços são mais baixos e existe uma maior diversidade de dispositivos, o que garante maior liberdade de escolha aos consumidores.

Atendendo à cota de mercado, considerou-se como opções mais relevantes os sistemas operativos Android e iOS, contudo, devido ao curto tempo disponível para a realização deste projeto, optou-se pelo Android uma vez que, somente este, representa mais de 75% dos *smartphones* existentes atualmente no mercado.

Após esta escolha, o passo seguinte passa por determinar qual a versão do sistema operativo Android permite obter o mesmo efeito, pois, desde o seu surgimento em 2008, este tem vindo a sofrer várias atualizações.

Para tal, foram escolhidas as versões do Android a partir da versão 2.3 até à atual. Esta decisão resulta de dois fatores, o primeiro porque, como se pode verificar pela Tabela 2 (13), estas permitem abranger um total de 86,6% dos dispositivos Android existentes no mercado.

Versão	Nome de Código	Quota de Marcado
1.5	Cupcake	0.1%
1.6	Donut	0.3%
2.1	Eclair	2.7%
2.2	Froyo	10.3%
2.3.-	Gingerbread	0.2%
2.3.2		
2.3.3		50.6%
2.3.7		
3.1	Honeycomb	0.4%
3.2		1.2%
4.0.3	Ice Cream Sandwich	27.5%
4.0.4		
4.1	Jelly Bean	5.9%
4.2		0.8%

Tabela 2 - Quota de mercado das diversas versões Android

O segundo fator que contribuiu para a determinação da versão a partir da qual a aplicação deve ser compatível, são os sensores suportados por cada uma das versões Android. Uma vez que a aplicação a desenvolver necessita de recolher dados do maior número possível de fontes sensoriais presentes nos *smartphones*, escolheu-se a versão 2.3, pois, como se pode constatar através da Tabela 3 (14), esta é a primeira versão que possui suporte para um maior número de sensores, e.g. barómetro, gravidade, aceleração linear entre outros (7).

Sensor	Android 4.0	Android 2.3	Android 2.2	Android 1.5
TYPE_ACCELEROMETER	Sim	Sim	Sim	Sim
TYPE_AMBIENT_TEMPERATURE	Sim	**N/A**	**N/A**	**N/A**
TYPE_GRAVITY	Sim	Sim	**N/A**	**N/A**
TYPE_GYROSCOPE	Sim	Sim	**N/A**	**N/A**
TYPE_LIGHT	Sim	Sim	Sim	Sim
TYPE_LINEAR_ACCELERATION	Sim	Sim	**N/A**	**N/A**
TYPE_MAGNETIC_FIELD	Sim	Sim	Sim	Sim
TYPE_ORIENTATION	Sim	Sim	Sim	Sim
TYPE_PRESSURE	Sim	Sim	**N/A**	**N/A**
TYPE_PROXIMITY	Sim	Sim	Sim	Sim
TYPE_RELATIVE_HUMIDITY	Sim	**N/A**	**N/A**	**N/A**
TYPE_ROTATION_VECTOR	Sim	Sim	**N/A**	**N/A**
TYPE_TEMPERATURE	Sim	Sim	Sim	Sim

Tabela 3 - Disponibilidade de sensores nas versões Android

O Android como se referiu previamente, é um sistema operativo *open-source*, construído sobre o Kernel Linux (15) (16) e desenvolvido com o intuito de possibilitar a criação de aplicações móveis que consigam tirar o máximo proveito do sistema operativo sobre o qual foram desenvolvidos. Por esta razão, este ganhou uma grande parte da quota de mercado dos dispositivos móveis num curto espaço de tempo.

Para que seja possível aos dispositivos com baixo poder computacional e gráfico, serem equipados com o sistema operativo Android, ao contrário dos seus concorrentes diretos, e.g. o iOS da Apple que só é disponibilizado em dispositivos com um alto poder computacional e gráfico desenvolvidos especificamente para esse fim, a Google teve uma abordagem diferente na medida em que desenvolveu uma máquina virtual denominada Dalvik (17) que, sobre o resultado do código Java compilado, é aplicado ainda mais um nível de otimização por parte do Dalvik antes da instalação das aplicações nos dispositivos móveis, o que permite um melhor desempenho a nível das aplicações e do sistema operativo nos dispositivos que sejam mais limitados a nível da memória e do processador.

2.2. Reconhecimento de atividade

O reconhecimento de atividade, originalmente denominado em inglês por *Activity Recognition* (**AR**), é uma área de investigação, onde o principal objetivo é o de determinar a atividade realizada por um determinado utilizador, monitorizando somente os seus movimentos.

Este tipo de monitorização de atividades que estão sendo realizadas pelos utilizadores, passa a permitir que atividades como, conduzir, andar, correr, andar de bicicleta, aspirar, subir escadas, andar de elevador bem como um variado leque de outras atividades sejam determinadas com certo nível de precisão (18) (19).

Apesar do nível de precisão na determinação da atividade em execução ser elevado, a forma como esses dados são recolhidos pode ser por vezes um pouco invasiva, como acontece na experiência de monitorização de Bao & Intille (19), em que 5 (cinco) sensores são colocados em pontos estratégicos do corpo do utilizador, por forma a conseguir determinar as atividades com um maior nível de precisão. Nesta experiência,

abdicando do conforto do utilizador, e utilizando os cinco sensores, Bao & Intille conseguiram, identificar com certo nível de precisão, 20 (vinte) atividades distintas.

Contudo, nas últimas décadas, com a exponencial evolução tecnológica, e consequente miniaturização dos transístores, foi possível diminuir o tamanho de todos os componentes utilizados nos computadores. Acompanhando esse processo de miniaturização, estiveram os sensores, cujo tamanho foi possível também diminuir significativamente. Isso passou a permitir que estes passassem a ser integrados mais facilmente em todo o tipo de dispositivos, nomeadamente, nos dispositivos móveis, o que abriu um novo espetro de campos de investigação.

Aproveitando esta nova gama de sensores, atualmente existente na maioria dos dispositivos móveis comercializados, como telemóveis e mesmo iPods e iPod Touch da Apple (12), Kwapisz *et al.* (20) tentam recolher destes, o mesmo tipo de informações que Bao & Intille de uma forma menos intrusiva, pois estes são tipos de dispositivos que atualmente acompanham as pessoas na maioria das suas tarefas diárias. Kwapisz *et al*. (20) propõem a utilização destes sensores, mais concretamente do acelerómetro, que recolhe informações sobre a orientação do dispositivo através do seu eixo triaxial que é sensível à força gravítica da terra, permitindo assim saber, em cada instante, a velocidade que se encontra a ser aplicada a cada um dos eixos.

Toda esta informação recolhida é posteriormente analisada, permitindo, com base em padrões, retirar conclusões sobre a eventual atividade que se encontra a ser realizada pelo utilizador. Para efetuar esta análise sobre os dados recolhidos pelos sensores dos dispositivos móveis, existe uma série de algoritmos comumente utilizados, que foram estudados por X. Wu *et al.* (21), e permitem obter melhores resultados nos processos de *data mining* (**DM**) (22), não obstante do fato de poderem ser também aplicados algoritmos modificados, que permitam ter um maior nível de certeza sobre os dados a que serão aplicados.

De entre estes algoritmos mais utilizados nos processos de DM, destacam-se principalmente os de classificação, nomeadamente, o C4.5 (23), o *Support Vector Machines* (**SVM**) (24), o *k-Nearest Neighbour* (**k-NN**) (25) ou o Naive Bayes (26), isto porque, regra geral, o principal objetivo dos algoritmos de DM, é tentar determinar a classe a que um determinado objeto pertence (27) levando em consideração os seus atributos, e.g. com base em atributos de uma pessoa como pressão sanguínea, batimento

cardíaco, colesterol, nível de glicemia, determinar se a pessoa tem o diagnóstico positivo ou negativo face a uma determinada doença.

2.3. Algoritmos de classificação em árvore

Os algoritmos de classificação em árvore são uma forma simples e fiável de prever e explicar a relação existente entre uma determinada classe e os seus atributos (28). Ao contrário dos algoritmos de aprendizagem não supervisionados, em que o algoritmo de classificação aprende sozinho as classes possíveis com base nos atributos recebidos, os algoritmos de classificação em árvore enquadram-se no grupo dos algoritmos de aprendizagem supervisionados, uma vez que os valores possíveis que a classe pode tomar são conhecidas à partida, e este tem somente de determinar as relações existentes entre os atributos, que permitem chegar mais rapidamente à classe pretendida.

Estes tipos de algoritmos, os de classificação em árvore, encontram-se entre os mais populares e os mais utilizados de entre todos os algoritmos que são passíveis de serem utilizados nos métodos de aprendizagem e DM (23) (21). Como exemplo, destaca-se o ID3 (29) e o seu sucessor, o C4.5 (23), ambos desenvolvidos por J. Ross Quinlan em 1979 e 1993 respetivamente.

O funcionamento destes algoritmos em árvore consiste na criação de uma árvore de decisão binária, constituída por nós. Na árvore construída existe um nó raiz, e os restantes nós se encontram subdivididos em nós de decisão e nós folha. Para a eleição de qual o nó raiz, é escolhido o nó cujo ganho de informação seja o maior possível, ou seja, o nó que permite separar o maior número de casos e, consequentemente, chegar mais rapidamente à classe à qual o dado a ser analisado pertence. Após a determinação do nó raiz, podemos visualizar os dois ramos do nó raiz como novas árvores, em que, dos nós restantes, são eleitos os dois que apresentam o maior ganho de informação, sendo que este processo é repetido recursivamente até não haver mais nós restantes. Os nós que apresentarem menor ganho de informação serão os nós folha, que representarão as classes existentes.

Esta árvore de decisão gerada pelo algoritmo é considerada um modelo previsivo que, com base nos dados do conjunto de testes que o geraram, servirá para efetuar a classificação de novos dados que venham a ser recebidos. Durante a construção do

modelo previsivo, de modo a assegurar que este não sofre de sobre aprendizagem, ou seja, que este classifica corretamente o conjunto de teste com uma eficácia de 100%, este tem de passar por um processo de poda, processo este que consiste na substituição de alguns ramos da árvore gerada por um nó folha, o que garante que não existe perda de informação, mas apenas uma eventual descida da taxa de sucesso da classificação (30).

Durante o processo de poda da árvore, começando pelos nós folha, e avançando em direção ao nó raiz, se as estimativas da árvore indicam que, com a remoção de um determinado nó, a taxa de precisão na árvore passa a ser maior então esse nó é efetivamente eliminado.

Outro processo que se pode aplicar de modo a evitar a sobre aprendizagem, consiste em limitar o tamanho da árvore aquando da sua construção, contudo, segundo Quinlan, isto traz resultados irregulares no processo de classificação (23), razão pela qual passou a adotar e aconselhar o processo de poda, face ao processo de limitação do tamanho da árvore.

2.4. Precisão e cobertura

A precisão e cobertura, também designada em inglês como *precision and recall*, é uma medida de desempenho utilizada para avaliar os algoritmos de classificação binários utilizados no estudo de reconhecimento de padrões ou recuperação de informação (31).

Para a determinação da percentagem da precisão e cobertura existem duas fórmulas.

$$precisão = \frac{|\{\text{número de casos corretamente catalogados}\}|}{|\{\text{número total de casos catalogados}\}|}$$

A percentagem de precisão consiste na razão existente entre o número de casos corretamente catalogados e o número total de casos catalogados.

$$cobertura = \frac{|\{\text{número de casos corretamente catalogados}\}|}{|\{\text{número total de casos existentes}\}|}$$

O cálculo da percentagem de cobertura consiste na razão existente entre o número de casos corretamente catalogados e o número total de casos existentes.

3. Projeto

Neste capítulo descreve-se a arquitetura geral de referência, assim como as arquiteturas da aplicação móvel e do repositório.

Em conformidade com os objetivos deste projeto, desenvolveu-se uma aplicação para a recolha de dados e sua posterior submissão para um repositório central de modo a facilitar que, posteriormente, estes dados venham a ser processados e eventualmente disponibilizados a entidades que neles tenham interesse. Assim sendo, para que não exista dependência entre o repositório, a aplicação móvel e as entidades que pretendam obter os dados, o repositório disponibiliza um serviço *web* para facilitar o desacoplamento, tal como é apresentado na arquitetura geral ilustrada na Figura 1.

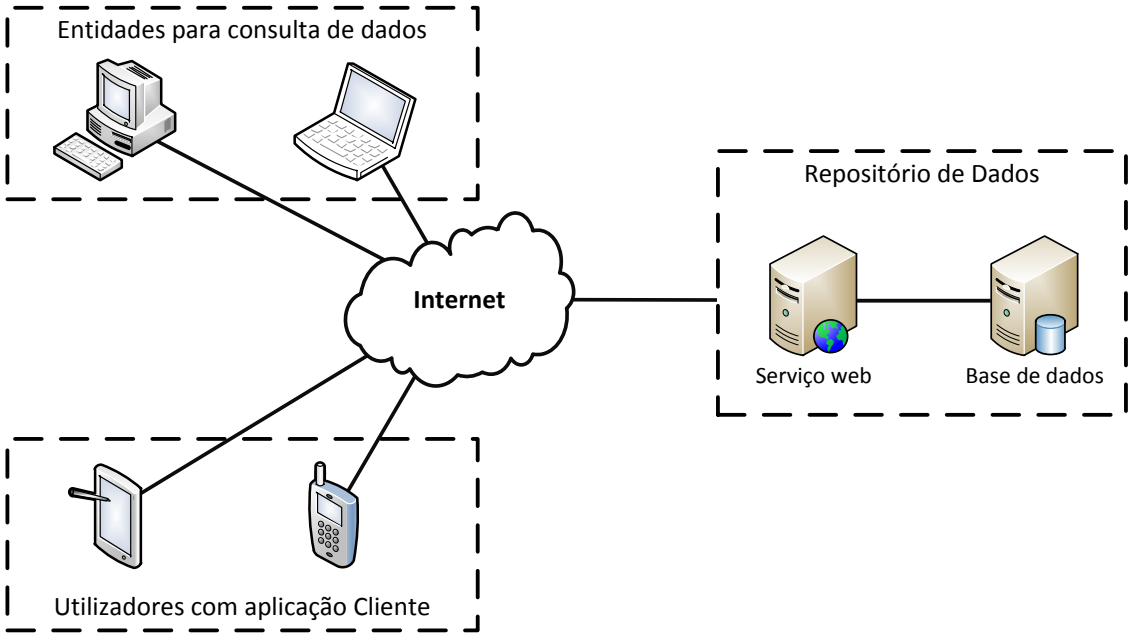

Figura 1 - Arquitetura geral do projeto

A arquitetura apresentada na Figura 1 é modular na medida em que, a aplicação cliente e as entidades que queiram ter acesso ao repositório não possuem um acoplamento forte, ou seja, toda a comunicação é efetuada através da API disponibilizada pelo serviço *web* do repositório.

3.1. Arquitetura da aplicação móvel

De modo a melhor realizar o primeiro objetivo deste projeto, nomeadamente desenvolver uma aplicação que recolha dados obtidos dos sensores presentes num *smartphone* e efetue o envio dessa informação para um repositório central, foram criados os seguintes diagramas de sequência por forma a compreender quais as possíveis interações do utilizador com o sistema, bem como uma possível interface gráfica que melhor se adeque às necessidades do utilizador para realizar tais ações.

Assim sendo, os diagramas de sequência seguintes representam os cenários resultantes da interação do utilizador com o sistema através dos vários ecrãs existentes na aplicação.

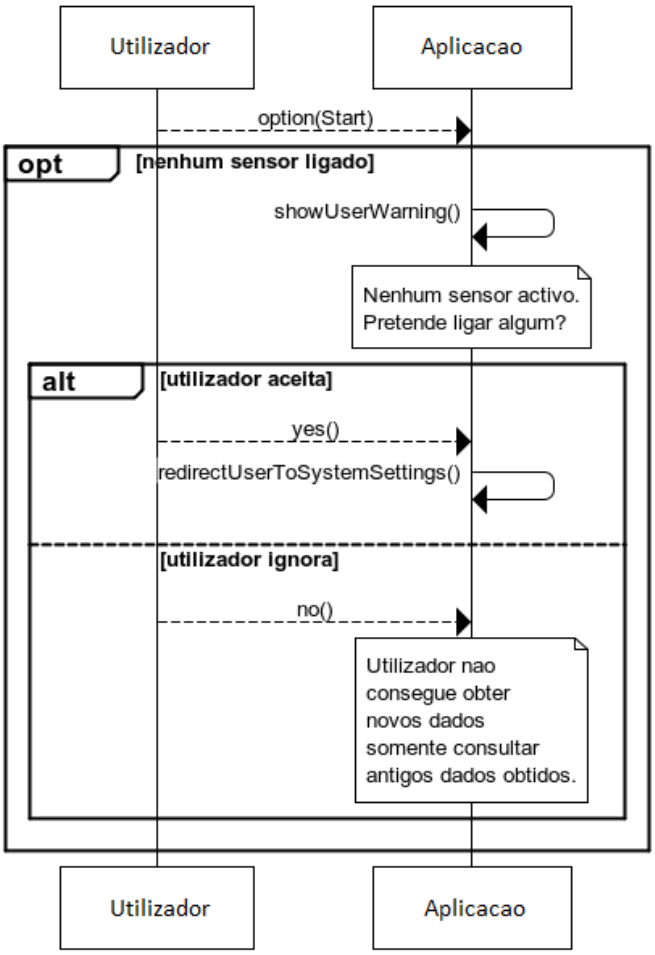

Figura 2 - Diagrama de sequência sem sensores ativos

Referente ao diagrama de sequência do ecrã principal, presente na Figura 2, é possível verificar que, quando o utilizador pretende efetuar a recolha dos dados, mas não existe nenhum sensor do sistema ativo, o utilizador é informado de tal fato pela aplicação e,

caso pretenda ativar algum sensor, é reencaminhado para as definições do sistema onde poderá ativar os sensores pretendidos.

No segundo caso, onde os sensores já se encontram ativos, como é possível verificar pela Figura 3, quando o utilizador manifesta à aplicação a intenção de proceder à recolha de dados, esta recorre às definições da aplicação de modo a verificar quais os sensores que o utilizador permitiu a aplicação utilizar. Uma vez obtida a lista de sensores permitidos, a aplicação começa a recolha de informação sensorial disponibilizada pelos mesmos.

Ainda na Figura 3 é possível verificar que, caso o utilizador assim o pretenda, no ecrã principal, é possível interromper a captura de novos dados sensoriais.

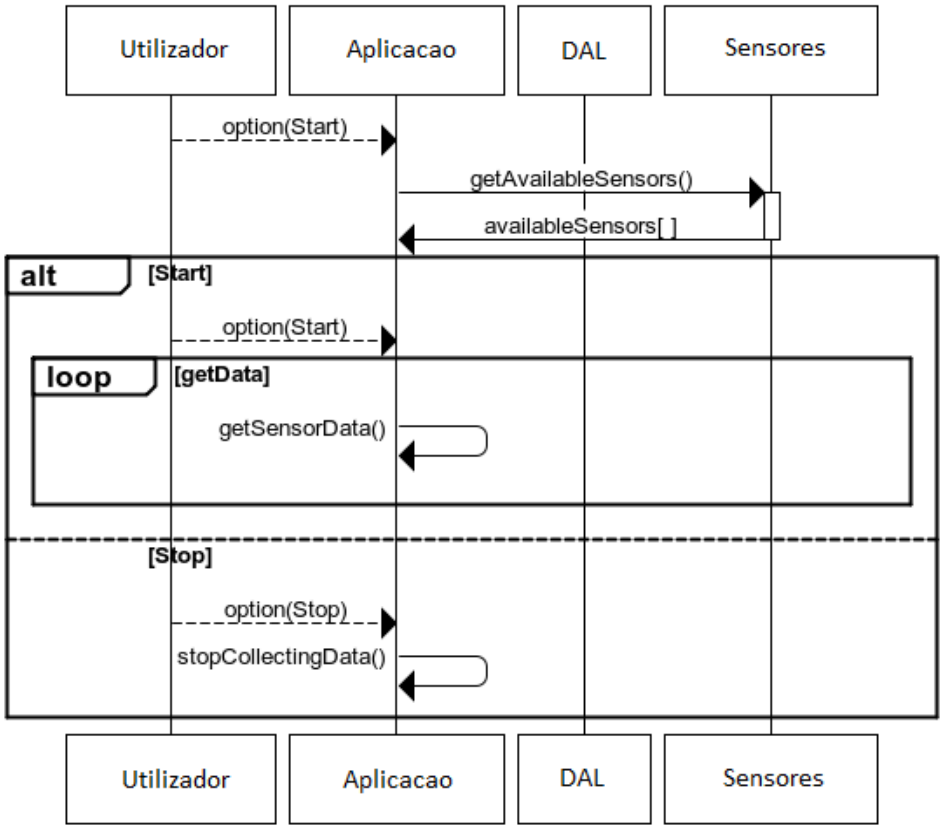

Figura 3 - Diagrama de sequência com sensores ativos – Parte 1

Como é retratado no diagrama de sequência presente na Figura 4, que representa ainda interações que o utilizador pode efetuar sobre o ecrã principal, o utilizador consegue ainda, efetuar operações como, limpar os dados recolhidos até à data, prosseguir para os ecrãs onde são apresentadas as experiências de mobilidade salvas ou as definições da aplicação.

Ainda referente à Figura 4, quando o utilizador pretende salvar uma experiência cujos dados se encontram a ser recolhidos, a aplicação para imediatamente a recolha dos dados, sendo que, de seguida, procede ao armazenamento dos mesmos.

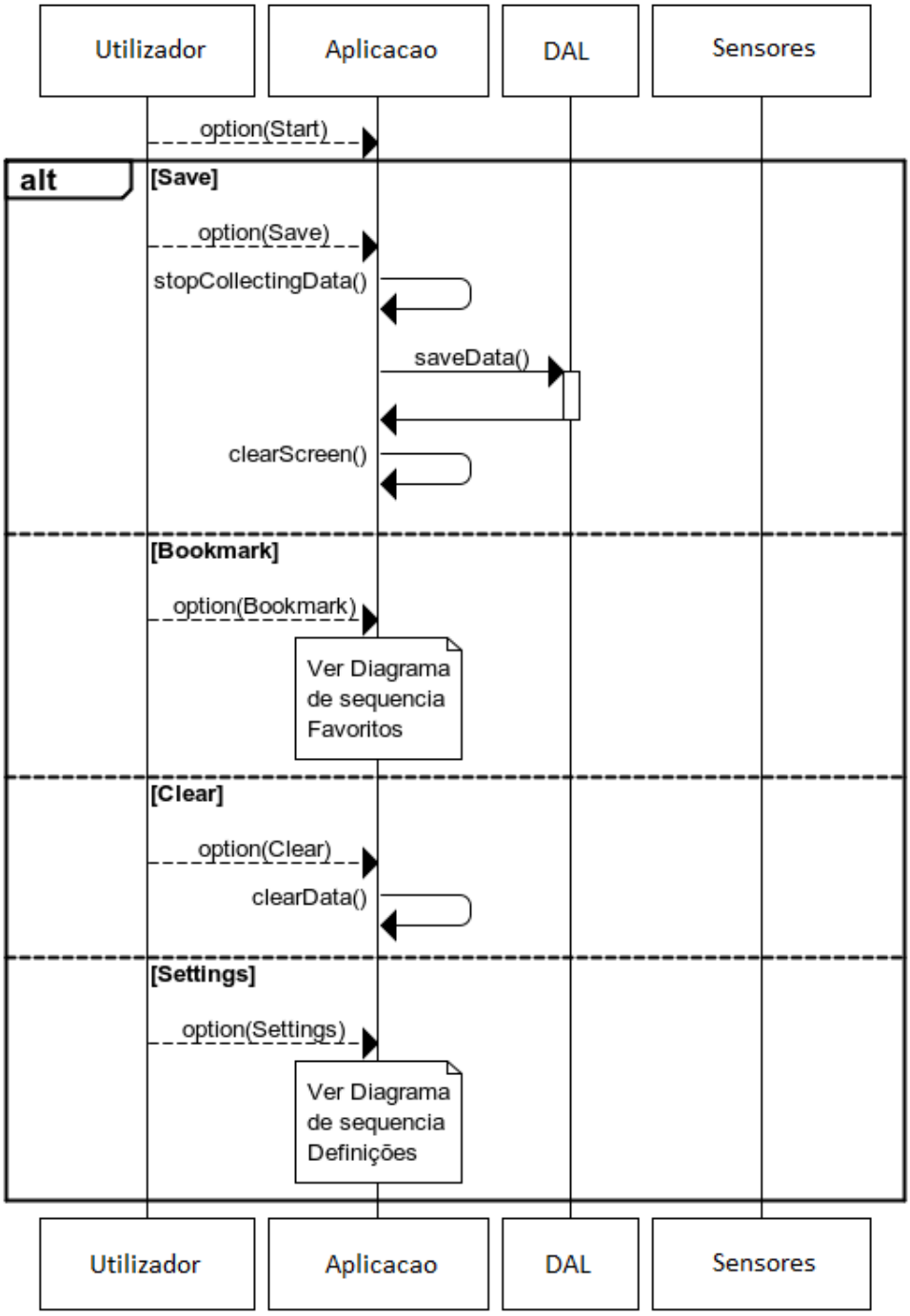

Figura 4 - Diagrama de sequência com sensores ativos - Parte 2

Efetuando a mesma análise sobre as interações possíveis do utilizador com os ecrãs de favoritos e de definições, os respetivos diagramas de sequência foram criados.

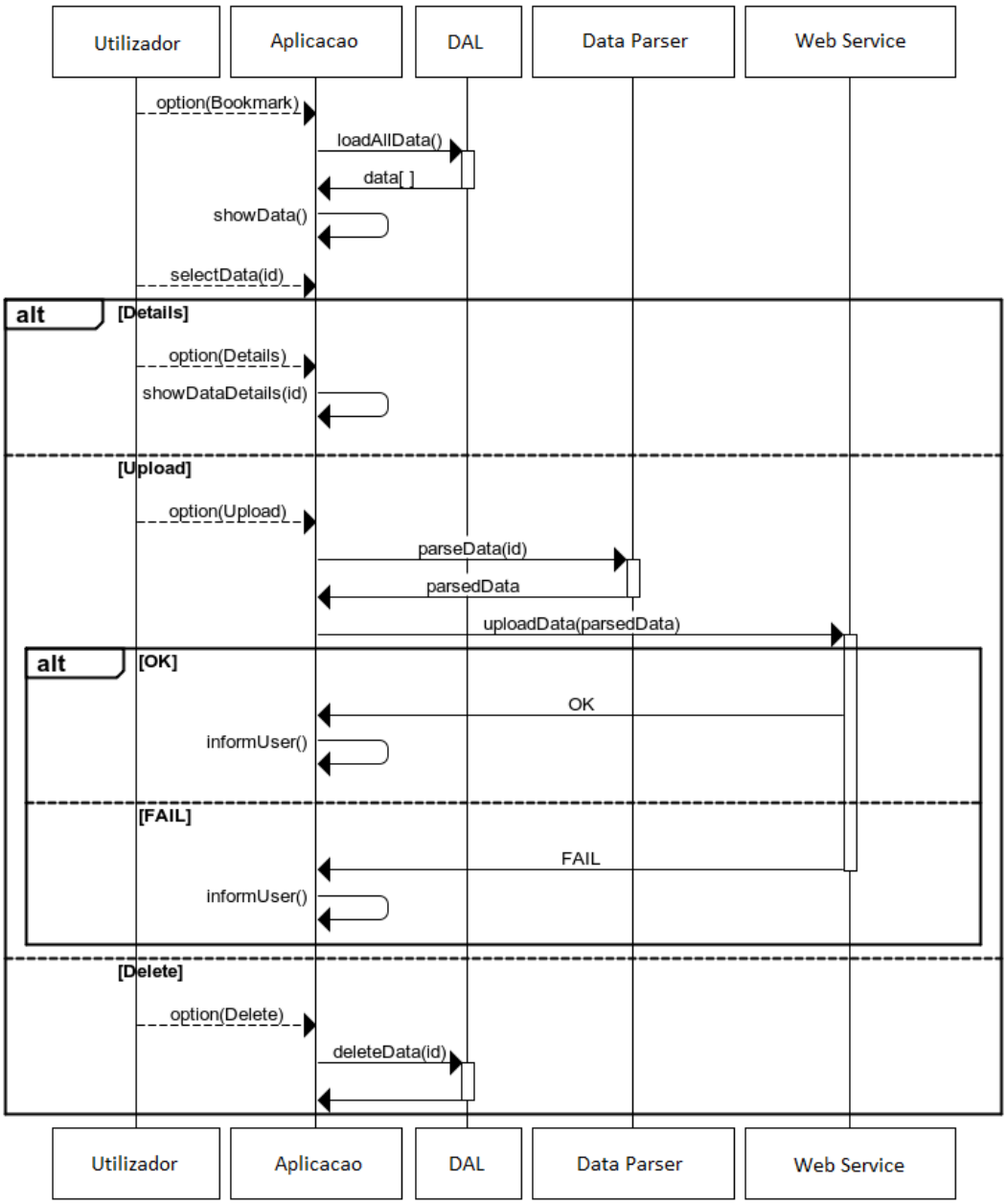

Figura 5 - Diagrama de sequência do ecrã de favoritos

O diagrama de sequência apresentado na Figura 5, representa o ecrã de favoritos, ecrã onde é apresentado ao utilizador as experiências de mobilidade previamente efetuadas e salvas por ele. Como se pode verificar pela Figura 5, as interações que o utilizador pode efetuar após selecionar uma das experiências apresentadas passam por, apagar a experiência selecionada, ver os detalhes da experiência, nomeadamente os dados sensoriais recolhidos durante a realização da experiência de mobilidade selecionada, ou ainda efetuar a submissão da experiência selecionada para o repositório sendo que, mediante a resposta do repositório, a aplicação informa o utilizador se a submissão da experiência ocorreu com ou sem erros.

No ecrã de definições, onde são apresentadas as definições da aplicação, o utilizador indica à aplicação quais os sensores que pretende que este tenha acesso, bem como a cadência da recolha dos dados sensoriais. O diagrama de sequência elaborado pode ser consultado na Figura 6.

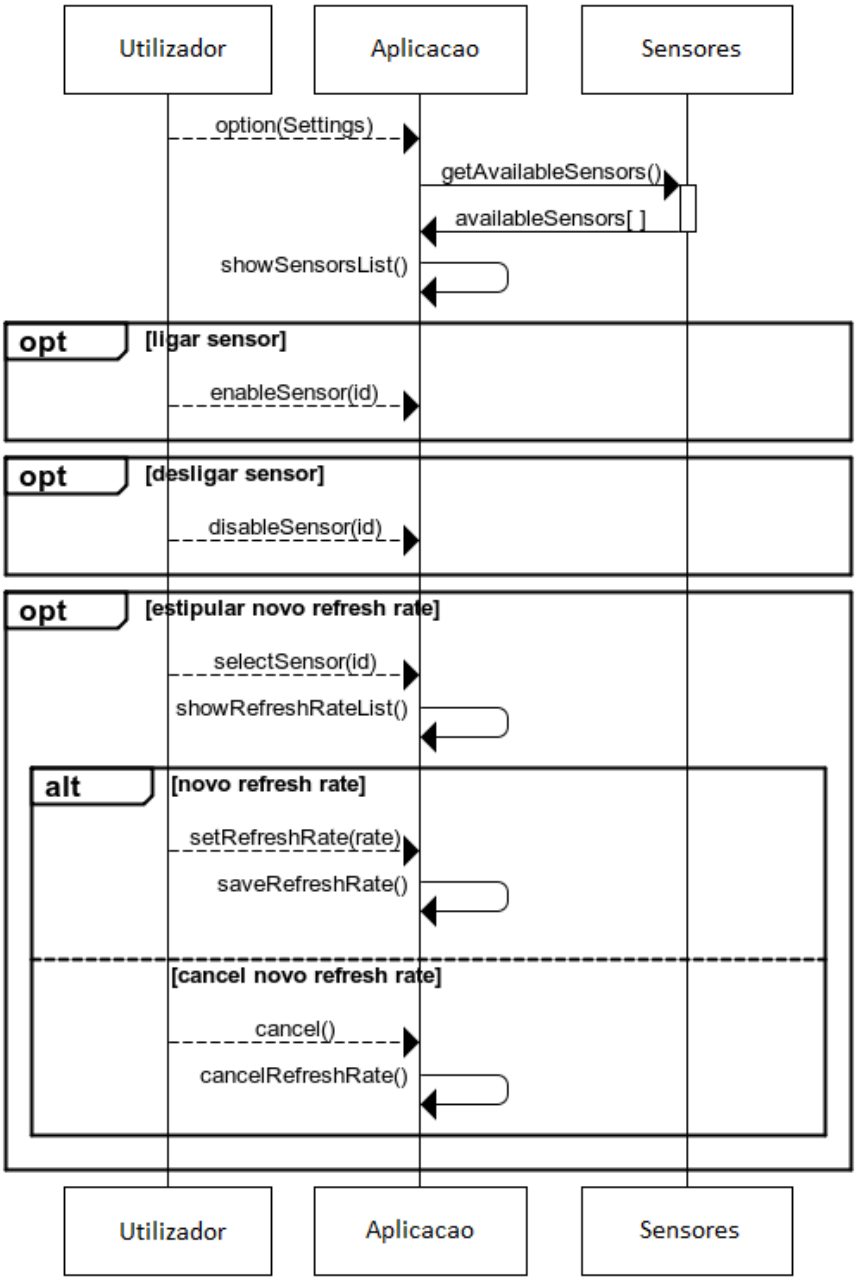

Figura 6 - Diagrama de sequência do ecrã de definições

No ecrã de definições, de modo a que seja possível apresentar ao utilizador a listagem dos sensores existentes no *smartphone*, recorre-se ao sistema de modo a obter essa informação. Uma vez obtida essa informação, é apresentado ao utilizador em forma de

lista, todos os sensores que este dispõe para efetuar a recolha de informação sensorial durante a realização das suas experiências de mobilidade.

Sobre as opções apresentadas ao utilizador, este pode, sobre um sensor pretendido, dar ou não à aplicação a permissão de recolher dados do mesmo, bem como estipular a cadência com o qual os dados serão obtidos do sensor em questão.

Após o levantamento das possíveis interações do utilizador com a aplicação cliente, passou-se à elaboração de uma arquitetura que pudesse dar suporte a essa interação. Para que a arquitetura tivesse igualmente certa modularidade e separação de funcionalidades, decidiu-se aplicar o padrão *Model-View-Controller* (**MVC**) (32) (33).

O MVC é um padrão atualmente muito utilizado na engenharia de *software* que permite a separação entre três camadas distintas, nomeadamente a camada de apresentação, a camada de lógica e a camada de armazenamento dos dados. A ideia de separação de funcionalidades surgiu pela primeira vez em 1979 através de um engenheiro norueguês chamado *Trygve Reenskaug* que afirma que, numa aplicação, deveria existir uma separação entre os *Models*, responsáveis pela representação da informação, os *Controllers*, responsáveis pela interação entre o utilizador, o sistema e os *Views*, responsáveis pela apresentação visual da informação contida no *Model*, atuando assim como uma espécie de filtro, pois este é que teria a decisão de escolher quais os atributos do *Model* apresentar ou ocultar (34).

Ao aplicar o padrão MVC na elaboração da solução para a aplicação móvel, chegou-se à seguinte arquitetura como pode ser visto na Figura 7 que se segue.

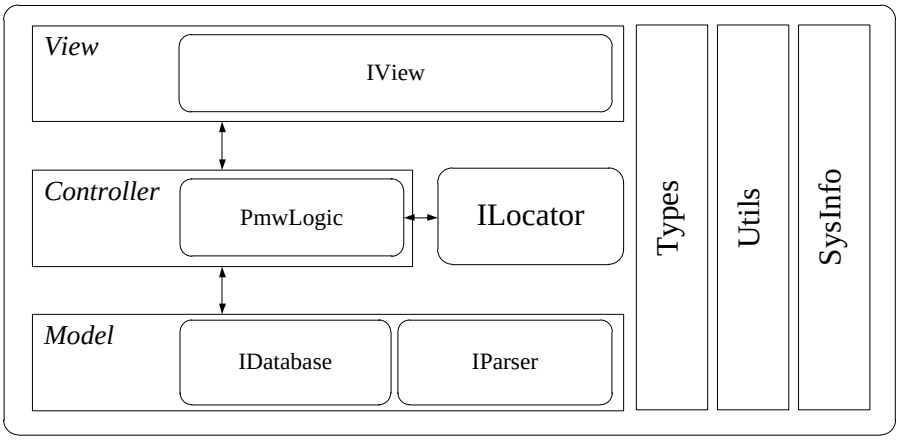

Figura 7 - Arquitetura da Aplicação Móvel

Da arquitetura apresentada, destacam-se as seguintes entidades:

i. *IView* – interface a ser implementada por qualquer classe que seja criada com o intuito de apresentar ao utilizador do sistema alguma representação da informação presente na camada *Model*;
ii. *PmwLogic* – é a única entidade existente na camada *Controller* e é a responsável por recolher a interação do utilizador com o sistema;
iii. *ILocator* – esta interface deve ser implementada por todas as classes que permitam à aplicação obter qualquer tipo de dados do sistema enquanto o utilizador mantiver a aplicação em funcionamento;
iv. *IDatabase* – esta interface é implementada pelas classes da camada *Model* cuja funcionalidade seja a de preservar os dados recolhidos pela aplicação;
v. *IParser* – esta interface, à semelhança da interface *IDatabase*, presente na camada *Model*, é implementada pelas classes que são responsáveis por efetuar o *parsing* dos dados armazenados, e.g. transformar dados presentes na base de dados local num documento *eXtensible Markup Language* (**XML**) (35), uma linguagem que permite descrever dados num formato que seja legível tanto para humanos como para máquinas ou *Keyhole Markup Language* (**KML**) (36), linguagem que tem por base o XML e é utilizada para codificar e representar linhas, polígonos, texto, entre outros.

Transversal a estas três camadas, existem ainda três entidades denominadas por *Types*, *Utils* e *SysInfo* que são responsáveis por:

i. *Types* – esta entidade tem definidos os tipos que são manipulados por esta aplicação;
ii. *Utils* – esta entidade encapsula qualquer tipo de função que sirva para efetuar operações que possam ser reutilizadas em diferentes partes da aplicação;
iii. *SysInfo* – esta entidade é semelhante à entidade *Utils*, no entanto, as funções existentes nesta classe servem especificamente para permitir a aplicação obter informação sobre o estado do dispositivo.

3.1.1. Interface da aplicação móvel

Tendo em conta o tipo de informação suportado pela aplicação, a maneira como esta seria apresentada ao utilizador, bem como a maneira como o utilizador interagiria com a aplicação, foram definidas as seguintes funcionalidades que, no conjunto, permitiram definir a interface da aplicação.

Na Figura 8, é apresentado o ecrã principal da aplicação no qual é possível iniciar o processo de obtenção de dados. Esta ação é feita quando o utilizador recorre à primeira opção do menu denominada *Start*. Uma vez iniciado o processo de obtenção de dados, a primeira opção do menu passa a disponibilizar a opção *Stop*, permitindo assim ao utilizador parar o processo de recolha de dados. Este ecrã apresenta a lógica previamente descrita no diagrama de sequência presente na Figura 2.

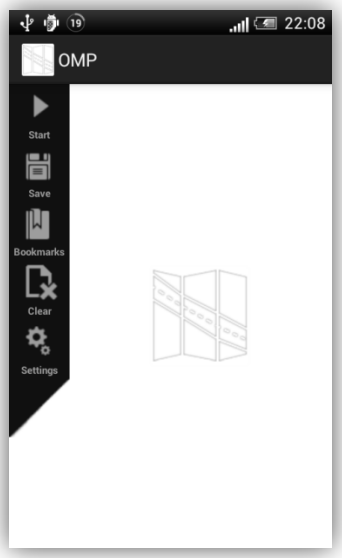

Figura 8a – Ecrã principal

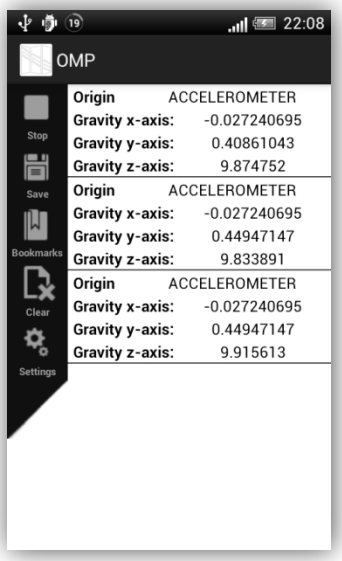

Figura 8b – Ecrã principal obtendo dados

Figura 8 - Ecrã principal da aplicação para obtenção dos dados

Todas as outras opções do menu possuem um comportamento idempotente, ou seja, independentemente do número de vezes que for escolhida a opção, o seu comportamento mantem-se o mesmo, permitindo assim ao utilizador fazer determinadas ações ou navegar para outras partes da aplicação, *e.g.*, se o utilizador escolher a opção *Clear*, a lista atual dos dados já obtidos é apagada. Se se escolher a opção *Bookmarks*, é apresentado o ecrã presente na Figura 9, que permitirá ao utilizador verificar quais as

experiências que foram previamente salvas. Existe ainda a possibilidade do utilizador escolher a opção *Settings*, que reencaminhará o utilizador para o ecrã de definições e cuja interface é apresentada posteriormente na Figura 10.

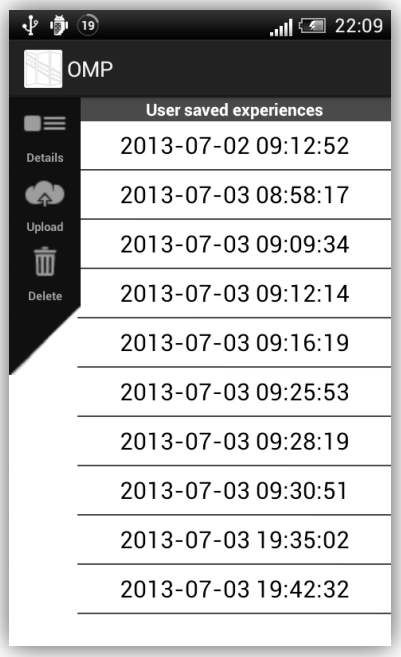

Figura 9a – Lista de todas as experiências salvas

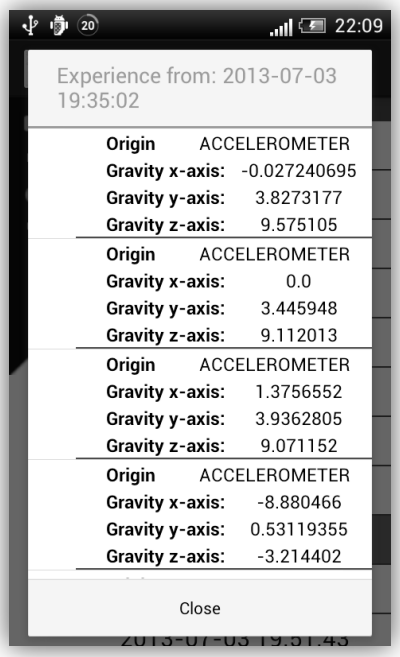

Figura 9b – Detalhe de uma experiência salva

Figura 9 - Ecrã de favoritos

Como se pode ver pela Figura 9b referente ao ecrã de favoritos, onde são apresentadas as experiências de mobilidade previamente salvas pelo utilizador, este pode selecionar uma determinada experiência e, sobre esta, poderá ver os detalhes da experiência (opção *Details*), efetuar o seu *upload* para o repositório (opção *Upload*) ou apagar a mesma (opção *Delete*). É de notar ainda que a lista das experiências previamente salvas pode encontrar-se dividida em duas categorias, "*User saved experiences*" e "*Auto saved experiences*", isto porque a primeira refere-se às experiências que foram salvas pelo utilizador e a segunda refere-se às experiências salvas automaticamente pela aplicação, que acontece quando o utilizador escolhe efetuar o *upload* de uma determinada experiência, mas a operação falha por falta de conetividade com o repositório. Este ecrã apresenta a lógica previamente descrita no diagrama de sequência presente na Figura 5.

Por fim, no ecrã de definições, é dado ao utilizador a possibilidade de configurar os sensores sobre os quais pretende recolher a informação sensorial, ilustrado na Figura 10a, bem como a cadência com que a aplicação obtém esses dados, Figura 10b.

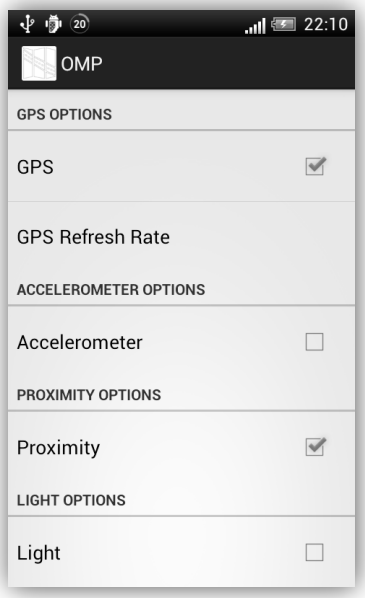

Figura 10a – Listagem de sensores disponíveis para recolha de dados

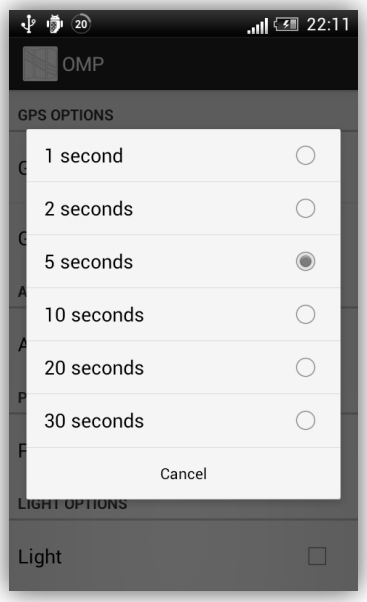

Figura 10b – Interface para estabelecer o ritmo com o qual os dados são recolhidos

Figura 10 - Ecrã de definições

À semelhança dos dois últimos ecrãs, este último representa a lógica previamente descrita no diagrama de sequência presente na Figura 6.

3.1.2. Implementação da aplicação móvel

Após a definição da arquitetura inicial da aplicação, bem como da interface a ser adotada pela aplicação tendo em conta os requisitos que este terá de cumprir, foi desenvolvido o *Unified Modeling Language* (**UML**) (37) presente na Figura 11.

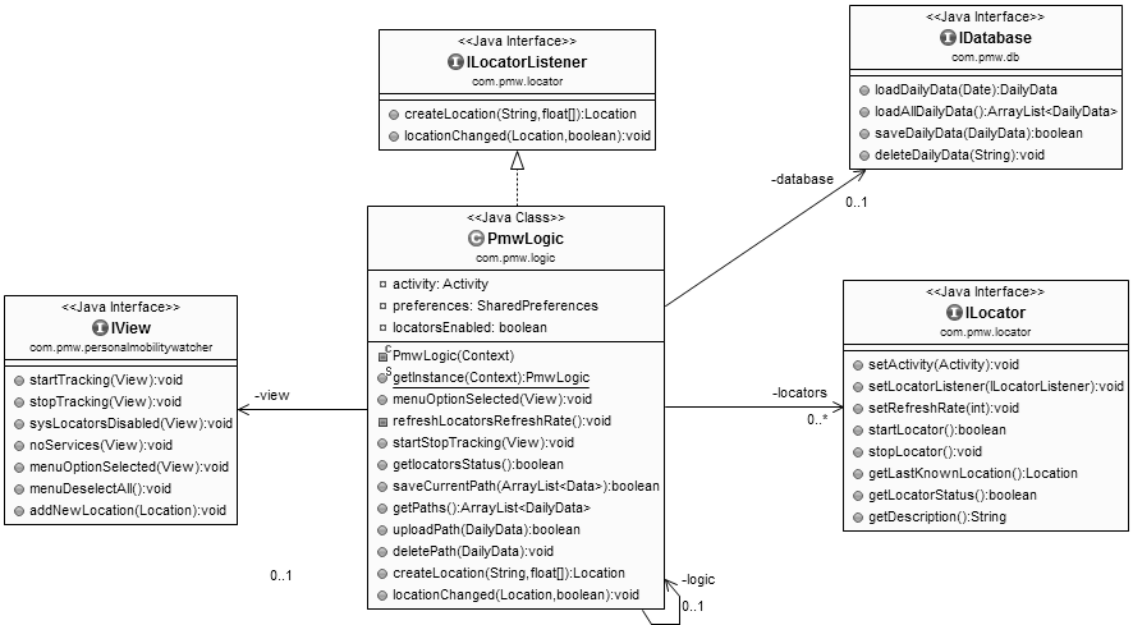

Figura 11 – Diagrama de classes da Aplicação Móvel

De modo a simplificar o UML, foram contempladas na Figura 11 somente as interfaces existentes bem como a classe *PmwLogic* que possui a lógica da aplicação.

De modo a manter as camadas de apresentação, lógica e armazenamento de dados o mais separados possível, seguindo o padrão MVC previamente descrito, a classe *PmwLogic* é a classe que encapsula toda a lógica da aplicação, ou seja, qualquer interação do utilizador com a interface gráfica, resultará numa chamada a uma das funções desta classe, que processará o pedido, e chamará a respetiva função presente na interface *IView*, a classe que implementar esta interface, e.g. o *PersonalMobilityWatcher* ou o *BookmarksActivity*, será o responsável por apresentar ao utilizador, de forma visual, o resultado da sua interação com a aplicação.

Uma vez que esta aplicação consiste ainda na recolha de dados de diversas fontes sensoriais do *smartphone*, criou-se uma interface denominada *ILocator* que deve ser implementada por qualquer classe cuja responsabilidade seja a de recolher dados de fontes sensoriais.

A classe *PmwLogic* implementa ainda a interface *ILocatorListener*, pois, uma vez que este encapsula toda a lógica da aplicação, esta classe registar-se-á perante as classes que implementem a interface *ILocator* de modo a receber notificações destas sobre novos dados obtidos por parte dos sensores do *smartphone*.

A interface *IDatabase* é a interface que é implementada por todas as classes que sejam responsáveis pelo armazenamento dos dados obtidos pela aplicação. Como exemplo, temos a classe *SqliteDatabase* que é a responsável por guardar os dados numa base de dados local SQLite.

Embora o Android possibilite outras formas de armazenamento e persistência de dados, nomeadamente a possibilidade de armazenamento num ficheiro localizado na memória interna do *smartphone* ou num cartão de memória, a escolha recaiu sobre uma base de dados SQLite, uma vez que este proporciona certas vantagens como, facilidade de interrogação dos dados armazenados, o tempo de vida da base de dados, bem como o seu conteúdo ser gerido exclusivamente pela aplicação, ao contrário do armazenamento num cartão de memória, em que o utilizador pode apagar inadvertidamente através do sistema de ficheiros.

Quanto ao formato sobre o qual os dados são submetidos para o repositório para posterior armazenamento e processamento, escolheu-se o formato XML.

Esta escolha foi feita, pois, uma vez que a aplicação móvel tem de ser modular, i.e., permite que o utilizador efetue a escolha de quais os sensores a partir dos quais pretende que a informação seja recolhida, como também tem de suportar diversos sensores, cujos dados sensoriais recolhidos podem estar presentes numa estrutura que pode variar de sensor para sensor, a escolha de enviar os dados num formato XML permite com que estes estejam numa estrutura similar, independentemente do tipo de sensor a partir do qual foram recolhidos.

A estrutura do XML enviado ao repositório, bem como do documento *XML Schema Definition* (**XSD**) (38) utilizado para a validação, no repositório, do XML criado pela aplicação móvel, podem ser consultados nos anexos presentes nas páginas 59 e 60 respetivamente.

O documento XML, quando validado pelo XSD, permite a definição de um elemento raiz denominado por *experience*, sendo que este por sua vez pode ter vários elementos do tipo *data*. O tipo *data* é o elemento que contém a informação referente aos dados lidos do sensor, e para isso, possui um *timestamp* do momento em que o dado foi recolhido, bem como as informações que este recolheu do sensor, e.g. a leitura em cada um dos eixos, a informação referente à localização geográfica, entre outros.

Caso o documento XML a ser submetido passe na validação efetuada com o documento XSD, essa informação será adicionada ao repositório, e o seu conteúdo processado, caso contrário, a informação submetida será descartada.

3.1.2.1. Extensibilidade

Uma vez que, dependente da versão do sistema operativo Android, os sensores disponíveis variam, é imperativo para esta aplicação móvel detetar quais os sensores que se encontram disponíveis, permitindo assim efetuar a eventual recolha dos dados.

Assim sendo, todos os sensores atualmente suportados por qualquer uma das versões do Android é suportado pela aplicação móvel desenvolvida. A classe responsável por efetuar tal suporte é a classe *Sensor*. Nesta classe, encontram-se definidos todos os sensores existentes, bem como o nome a atribuir a cada um dos eixos dos sensores, informação esta que é utilizada aquando da apresentação dos dados sensoriais recolhidos ao utilizador.

Caso passe a existir um novo sensor suportado pelo Android numa versão futura do sistema operativo, será necessário somente disponibilizar uma nova versão da aplicação, após ter sido definido nesta classe as informações necessárias sobre o novo sensor suportado, nomeadamente, o nome do sensor, a unidade de medida utilizada, o nome a atribuir a cada eixo sensorial, entre outros.

3.1.2.2. Analytics

Posteriormente, esta aplicação será fornecida a sujeitos de teste de modo a recolher e armazenar informação sensorial de diversas experiências de mobilidade. Isso permitirá não só testar a capacidade de recolha dos dados, como permitirá igualmente a possibilidade de testar a capacidade de classificação do algoritmo de determinação do meio de transporte descrito posteriormente na secção *3.2.2.2 Classifier*.

Apesar do fato de a distribuição ser acompanhada de um manual de utilizador, permitindo assim explicar ao utilizador, em que consiste a aplicação, como esta deve ser utilizado, entre outros fatores, há sempre o risco da aplicação não ser corretamente utilizada. Assim sendo, esta foi integrada com o *Flurry Analytics* (39), de modo a monitorizar e tentar perceber a forma como o utilizador navega na aplicação.

O *Flurry Analytics* é uma biblioteca que permite efetuar o *log*, não só de aplicações móveis como de *websites*. Os *logs* criados permitem monitorizar diversos fatores decorrentes da utilização da aplicação por parte do utilizador, como por exemplo, o percurso do utilizador na aplicação, exceções que possam ocorrer durante a utilização da aplicação, e até algumas informações sobre o utilizador, e.g. nome do utilizador, idade, país de onde se encontra a aceder à aplicação, entre outros.

Assim sendo, complementarmente ao teste de recolha dos dados por parte da aplicação e consequente teste do algoritmo de classificação, a integração com o *Flurry Analytics* permitirá obter outros dados que permitirão confirmar se o utilizador encontra-se a utilizar a aplicação da maneira como foi desenvolvida, ou se existem alguns aspetos que poderiam ser melhorados.

Todos os dados recolhidos pelo *Flurry Analytics* são disponibilizados posteriormente no seu *website*[1] para visualização.

3.2. Arquitetura do repositório

Após a elaboração da aplicação móvel que é responsável por efetuar a recolha e submissão dos dados recolhidos durante a realização de uma experiência do utilizador, centra-se agora na elaboração da parte do repositório que será responsável pelo armazenamento, catalogação e eventual disponibilização dos dados recebidos.

À semelhança da arquitetura desenvolvida para a aplicação móvel, durante a elaboração da arquitetura do repositório, separou-se pelas três camadas do MVC os componentes existentes no repositório de acordo com as suas responsabilidades, originando assim, a arquitetura presente na Figura 12.

[1] http://www.flurry.com/

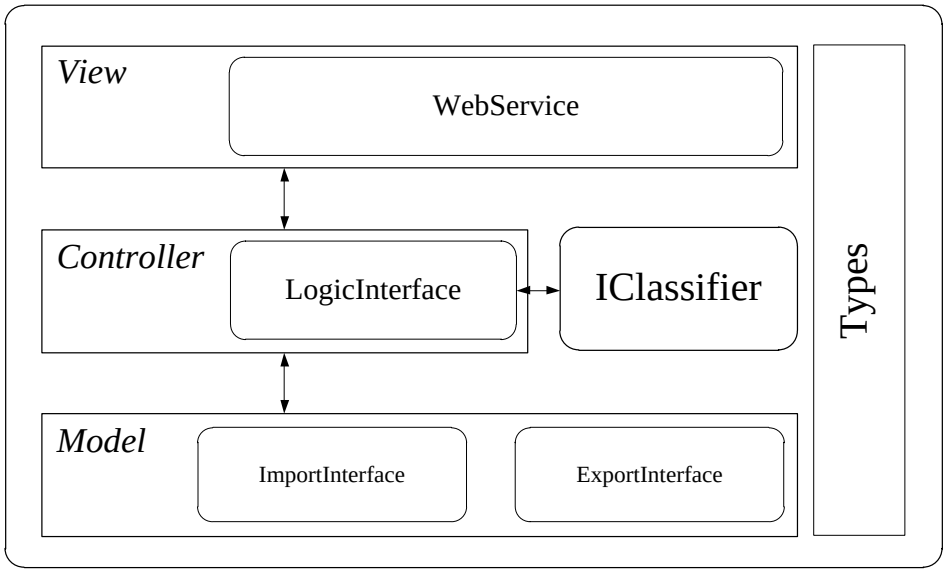

Figura 12 - Arquitetura do Repositório

As entidades apresentadas na Figura 12 são responsáveis por:

i. *WebService* – esta entidade é a responsável por apresentar ao utilizador as várias operações possíveis de se efetuar sobre o repositório, nomeadamente, nos serviços *web* de *Import* e *Export*;
ii. *LogicInterface* – esta interface deve ser implementada por todas as classes que sejam criadas com o intuito de associar lógica ao repositório;
iii. *IClassifier* – interface a ser implementada por qualquer classe que seja criada com o intuito de implementar um novo algoritmo de classificação para a determinação do meio de transporte utilizado numa determinada experiência;
iv. *ImportInterface* – esta interface deve ser implementada pelas classes que sejam implementadas com o intuito de processar as experiências enviadas pelos clientes usando a aplicação móvel;
v. *ExportInterface* – contrariamente à *ImportInterface*, esta interface deve ser implementada pelas classes cuja funcionalidade seja a disponibilização dos dados existentes a terceiros.

3.2.1. Interface do repositório

Nesta secção proceder-se-á a uma descrição mais detalhada dos métodos existentes em cada uma das interfaces do repositório. As funções descritas em cada uma das interfaces

foram criadas com o intuito de facilitar uma eventual interface gráfica que pudesse permitir ao utilizador, de forma fácil e intuitiva, a possibilidade de consulta e manipulação dos dados existentes no repositório relativo às experiências por ele submetidas.

3.2.1.1. ImportExperience

Como referido anteriormente, a interface *ImportExperience*, é a responsável por receber os pedidos enviados pela aplicação móvel, assim sendo, esta apresenta somente uma função intitulada *ImportUserExperience*.

Esta função tem como parâmetros:

i. *Username* – o nome de utilizador que se encontra a submeter a experiência. Este dado permite com que as várias experiências de um determinado utilizador possam ser agrupadas;
ii. *Description* – esta descrição é opcional e permite ao utilizador efetuar uma breve descrição sobre a experiência que se encontra a ser submetida, e.g. "corrida matinal";
iii. *Transports* – este parâmetro possui a informação de qual o meio de transporte utilizado. Uma vez que o repositório determinará qual o meio de transporte associado à experiência submetida, numa primeira fase, o valor presente neste campo servirá como termo de comparação de modo a perceber-se a eficiência do algoritmo utilizado para a determinação do meio de transporte;
iv. *Experience* – este parâmetro recebe a experiência em formato XML. A validade do documento XML pode ser confirmado através do XSD que se encontra disponibilizado pelo repositório. Contudo, uma vez que o documento XML submetido é gerado somente pela aplicação móvel, tem-se a garantia que este se encontra sempre bem formado.

3.2.1.2. ExportExperience

A interface *ExportExperience* foi feita de modo a permitir que as experiências existentes no repositório sejam disponibilizadas a terceiros. Caso estes enviem para os serviços do repositório o resultado da autenticação de um utilizador nos serviços deles,

é permitido a estes obterem informações específicas de um determinado utilizador, e.g. todas as experiências submetidas por um dado utilizador em concreto, caso contrário, é possível somente aceder às experiências submetidas ao repositório sem saber quem foi o utilizador que as submeteu.

Assim sendo, esta interface disponibiliza as seguintes funções para a consulta dos dados armazenados:

i. *ExportExperienceGuids* – esta função disponibiliza uma listagem de todos os identificadores das experiências existentes no repositório;
ii. *ExportRawExperiences* – esta função disponibiliza todas as experiências existentes no repositório no mesmo formato em que foram submetidos, formato XML;
iii. *ExportUserExperience* – esta função permite obter uma determinada experiência no formato pretendido;
iv. *ExportUserTransports* – esta função permite obter a informação de qual os meios de transporte utilizados por um determinado utilizador.

A função *ExportExperienceGuids* recebe como parâmetro somente um *username* que representa o nome do utilizador, sobre o qual se pretende obter os identificadores de todas as experiências por ele submetidas. Caso não seja inserido o nome de um utilizador válido, é retornada a lista de todos os identificadores das experiências existentes no repositório. Deste modo, possibilita-se a consulta dos dados sem saber qual o utilizador a que se encontra associado, garantindo assim a sua confidencialidade.

A função *ExportRawExperiences* não possui nenhum parâmetro uma vez que, quando invocado, esta disponibiliza todas as experiências existentes no repositório no mesmo formato que foram submetidas para o repositório. Esta função possui este comportamento, pois podem existir entidades que queiram disponibilizar recursos computacionais para efetuar o processamento das experiências existentes no repositório de modo a comparar os resultados ou simplesmente dar uma interpretação diferente aos mesmos.

A função *ExportUserExperience*, como referido anteriormente, disponibiliza uma determinada experiência de mobilidade num formato diferente do de submissão para o repositório. Assim sendo, este recebe como parâmetro um *experienceId*, identificador da

experiência que se pretende obter, e um *format*, que é uma extensão de três carateres que representa o formato no qual se pretende obter os dados. De momento os formatos que o repositório suporta para a disponibilização dos dados são XML, *Comma Separated Values* (**CSV**) (40) e KML. Uma vez que o formato KML serve para a representação de pontos geográficos, caso a experiência que se pretende obter não tenha informação de coordenadas de *Global Positioning System* (**GPS**) (41), a resposta devolvida pelo repositório trata-se de um documento KML vazio.

A função *ExportUserTransports* recebe o *username* de um determinado utilizador e retorna todos os meios de transporte que este alguma vez utilizou, bem como o número de vezes que este os utilizou. À semelhança do método *ExportExperienceGuids*, caso seja passado um nome de utilizador que não exista no repositório, de modo a garantir a confidencialidade dos outros utilizadores, é retornado todos os meios de transporte que alguma vez foram utilizados por todos os utilizadores do sistema.

3.2.2. Implementação do repositório

Após a realização do diagrama de alto nível previamente retratado na Figura 12 e das funções que fazem parte de cada uma das interfaces disponibilizadas pelo repositório, foi feito um diagrama UML que se encontra dividido entre a Figura 13 e a Figura 14, e representa as relações entre os diversos módulos retratados.

Na parte do diagrama UML presente na Figura 13, é de realçar que as classes *Import* e *Export*, presentes no *namespace WebService*, são as responsáveis por atender aos pedidos dos utilizadores. Para tal, estas duas classes recorrem a um ponto central denominado por *Logic*, que como o nome indica, contém toda a lógica associada ao repositório no que toca ao processamento e disponibilização das experiências a terceiros. Posteriormente, na secção *3.2.2.1 Lógica*, falaremos em maior detalhe sobre os aspetos da implementação desta classe.

Ainda na Figura 13, é possível verificar que as classes *Import* e *Export*, implementam as interfaces *ImportInterface* e *ExportInterface* respetivamente. Estas duas interfaces, como referido anteriormente, são as que definem a assinatura que as classes responsáveis por efetuar a receção e disponibilização dos dados têm de cumprir.

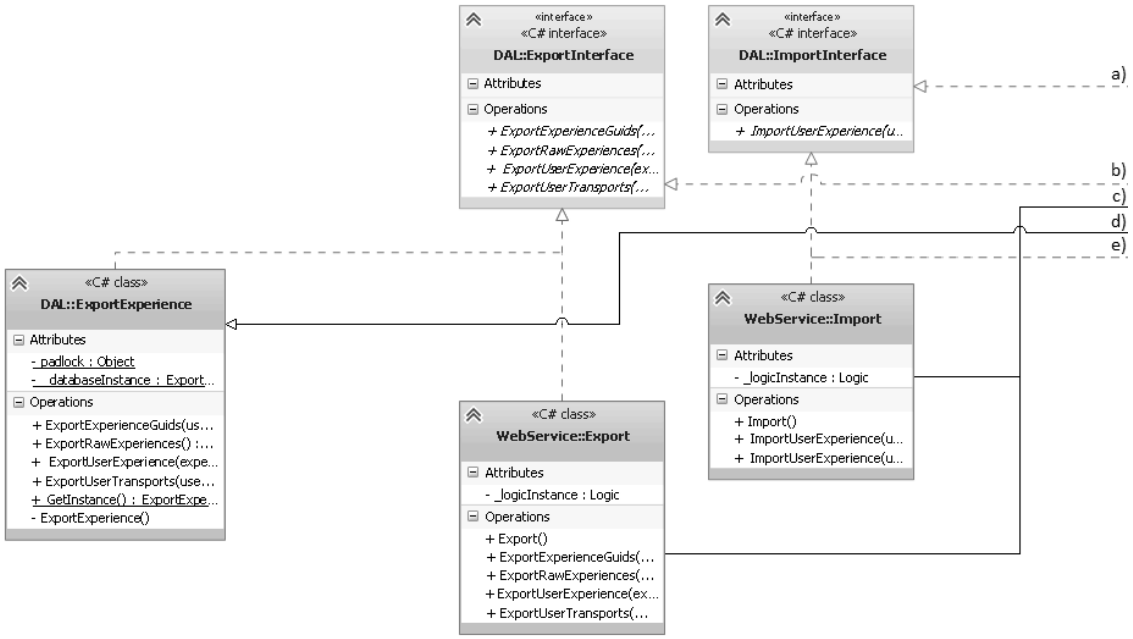

Figura 13 - UML do repositório - Parte 1

Referente à segunda parte do diagrama UML do repositório, presente na Figura 14, temos presente as classes *Logic*, responsável por toda a lógica existente no repositório, e a classe *RuleClassifier*, responsável pelo processo de análise dos dados recebidos e consequente classificação do meio de transporte utilizado nas experiências de mobilidade.

As classes *ExportExperience*, retratada na Figura 13, e a *ImportExperience*, retratada na Figura 14, são as responsáveis por efetuar o acesso ao modelo de dados onde são persistidas as experiências de mobilidade submetidas pelos utilizadores.

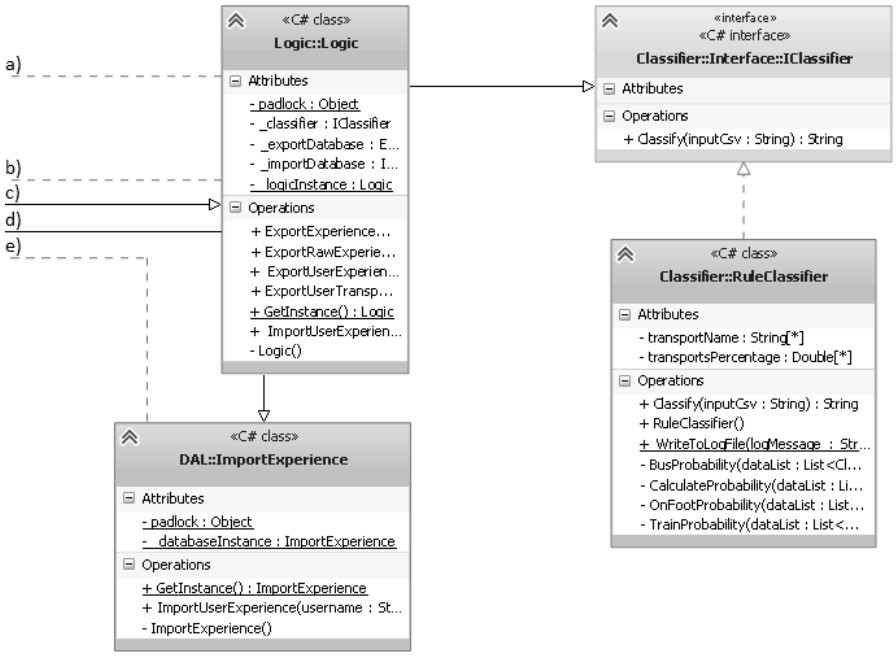

Figura 14 - UML do repositório - Parte 2

3.2.2.1. Lógica

A classe *Logic*, considerando a Figura 12 apresentada previamente, encontra-se enquadrada no módulo *Controller*, ou seja, implementa a interface *LogicInterface* e é por intermédio desta que os serviços *web* efetuam a manipulação dos dados existentes na camada *Model*.

Esta classe, sendo a única que implementa a interface *LogicInterface*, é a única que possui não somente toda a lógica associada ao processamento das experiências quando submetidas pela aplicação móvel, como também toda a lógica responsável pela disponibilização de tais experiências.

Tal solução representa um ponto de falha único para o repositório, pois, todas as ações efetuadas sobre o repositório têm de passar por esta classe. Contudo, apesar deste fato, esta classe é a única responsável por efetuar todas as operações relacionadas com o processamento, armazenamento e leitura das experiências existentes, isso permite garantir que as experiências se encontrem sempre num estado consistente uma vez que, em momento algum, existe o risco de escrita concorrente sobre os dados armazenados.

Qualquer chamada a uma das funções dos serviços *web* reflete numa chamada direta a um método da classe *Logic*, sendo que este efetua as ações necessárias de modo a processar e devolver o quanto antes a resposta ao pedido recebido por parte do utilizador, quer seja para submissão, quer seja para a consulta de experiências.

É na classe *Logic* que, ao se efetuar a receção de uma nova experiência de mobilidade por parte da aplicação móvel se efetua, com base nos dados recebidos, a determinação do meio de transporte utilizado. Para que essa classificação seja efetuada, a classe *Logic* recorre a uma classe que implemente a interface *IClassifier* de modo a que seja possível efetuar tal classificação. Esta classe é descrita de seguida em mais detalhe na secção *3.2.2.2 Classifier*.

3.2.2.2. Classifier

A classe *Classifier*, é uma classe que se encontra enquadrada no módulo *IClassifier* da Figura 12 previamente ilustrada, e foi desenvolvida de modo a permitir efetuar a determinação do meio de transporte utilizado aquando da recolha dos dados presentes na experiência submetida.

A implementação da interface *IClassifier* e do consequente método existente nesta, denominado por *Classify*, permite que a classe *Logic* não possua um acoplamento forte com as classes que permitam determinar o meio de transporte que se encontrava a ser utilizado quando os dados da experiência submetida foram recolhidos.

A necessidade de criar a interface *IClassifier*, que deve ser implementada por todas as classes desenvolvidas com o intuito de determinar o meio de transporte, surgiu na medida em que existe uma série de algoritmos de classificação, que, com maior ou menor nível de certeza, permitem a determinação do meio de transporte. Assim sendo, a determinação do meio de transporte recorrendo a esta interface possibilita alterar facilmente o algoritmo em uso, caso seja necessário, sem grandes alterações a nível do código.

O algoritmo utilizado durante a implementação do repositório para a determinação do meio de transporte utilizado é o *J48* (42), que se trata de uma implementação *open source* do algoritmo *C4.5* para a ferramenta de *data mining* intitulada *weka* (43).

O algoritmo *J48* é um algoritmo de decisão em árvore que, após a observação dos dados de teste, gera uma árvore de regras que são utilizadas na posterior classificação de novos dados submetidos. Esta árvore de regras pode ser facilmente traduzida em instruções *if-else* para uma mais fácil implementação.

Este algoritmo foi o escolhido em detrimento de outros algoritmos de decisão em árvore como o *J48Graph* ou o *RandomForest* uma vez que, para o mesmo conjunto de dados, este apresentava melhores resultados na classificação dos meios de transporte existentes. A análise comparativa entre os mesmos pode ser consultada na Tabela 4.

Percurso	% dados correctamente classificados		
	J48	J48Graph	RandomForest
Pedestre	**97,29%**	95,98%	88,62%
Autocarro	**91,50%**	90,99%	90,46%
Comboio	**90,26%**	88,00%	87,53%
Média	**93,02%**	91,66%	88,87%

Tabela 4 - Tabela comparativa dos algoritmos

Como é possível verificar pela Tabela 4, os algoritmos utilizados tentam classificar as experiências submetidas como tendo sido realizadas num percurso pedestre, de autocarro ou de comboio. Sendo que, para cada um destes, apresenta uma taxa de sucesso de classificação superior a 90% no caso do algoritmo J48.

Apesar dos algoritmos demonstrarem alta capacidade em determinar cada um dos percursos de forma eficiente quando aplicado ao conjunto de treino, o mesmo já não acontece com um conjunto de dados reais recebidos da aplicação móvel, como se pode verificar pela tabela Tabela 5.

Percurso	% de experiências correctamente classificadas
Pedestre	100,0%
Autocarro	71,4%
Comboio	73,7%
Média	81,7%

Tabela 5 - Taxa de sucesso na classificação das experiências

Como se pode verificar pela Tabela 5, as experiências realizadas num percurso pedestre são corretamente classificadas, contudo, o mesmo já não se verifica com os percursos realizados de comboio ou de autocarro.

A taxa de sucesso inferior a 100% na classificação das experiências realizadas de comboio e de autocarro reside no fato da orientação dos eixos dos sensores do dispositivo serem os mesmos, como se pode verificar pela Figura 15b. Assim sendo, caso haja alguma indecisão na determinação do meio de transporte no momento da submissão de uma experiência de mobilidade, este terá de optar entre os dois meios de transporte existentes que tenham essa orientação nos eixos sensoriais, o que faz com que a taxa de sucesso na determinação do meio de transporte seja inferior.

O mesmo já não acontece nas experiências realizadas num percurso pedestre, uma vez que há uma orientação distinta nos eixos do dispositivo móvel, como é retratado na Figura 15a.

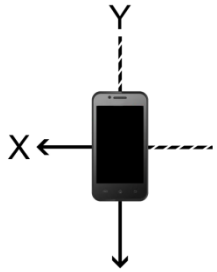

Figura 15a – Percurso “pedestre”

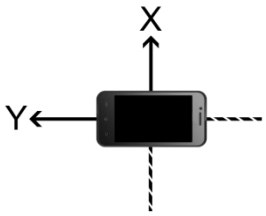

Figura 15b – Percurso “autocarro” e “comboio”

Figura 15 - Orientação do dispositivo móvel

Na Figura 15a, o eixo X é o eixo horizontal que aponta para o lado esquerdo, lado positivo do eixo. O eixo Y é o vertical e aponta para baixo, lado positivo do eixo. O eixo Z não é retratado, mas é uma linha perpendicular ao ponto de intersecção dos eixos X e Y e aponta para a parte posterior do dispositivo, lado positivo deste.

Na Figura 15b o eixo X passa a ser o eixo vertical, apontando para cima, o eixo Y passa a ser a horizontal que aponta para o lado esquerdo e relativamente ao eixo Z, este mantém as mesmas caraterísticas que a Figura 15a.

3.2.2.3. Persistência

Para a persistência dos dados submetidos pela aplicação móvel, uma vez que foi disponibilizada uma máquina virtual provida do sistema operativo *Windows 8*, optou-se por utilizar o *Microsoft SQL Server 2012*.

Como tal, e após uma análise dos dados submetidos e a serem armazenados no repositório, elaborou-se o seguinte diagrama entidade associação.

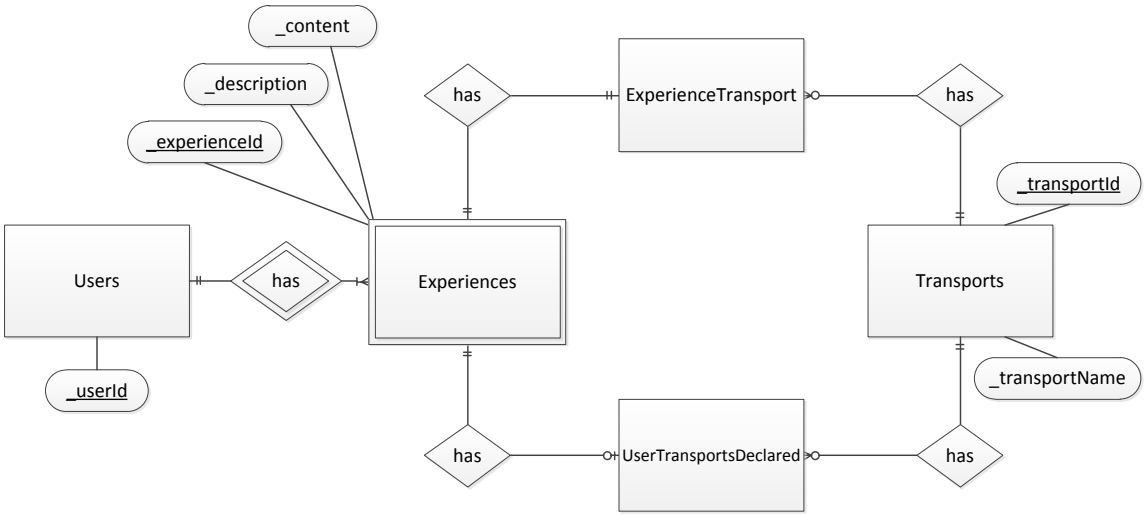

Figura 16 - Diagrama EA do repositório

No esquema relacional abaixo apresentam-se as relações existentes entre as chaves primárias e estrangeiras das mesmas.

Users (userId [**PK**])

Experiences (experienceId [**PK**], userId [**FK**], description, content)

Transports (transportId [**PK**], transportName)

ExperienceTransport (experienceId [**FK**], transportId [**FK**])

ExperienceTransportDeclared (experienceId [**FK**], transportId [**FK**])

Referente a este esquema relacional, as chaves primárias encontram-se assinaladas com [**PK**] (acrónimo da palavra inglesa *Primary Key*), e as chaves estrangeiras assinaladas com [**FK**] (acrónimo da palavra inglesa *Foreign Key*), sendo que estes referenciam a chave primária com o mesmo nome.

A tabela *Users*, como o nome indica, tem a responsabilidade de armazenar todos os utilizadores que alguma vez submeteram alguma experiência para o repositório. Uma vez que a autenticação não é contemplada, esta trata-se somente do nome do utilizador. Para minimizar o número de vezes que o nome do utilizador é inserido na aplicação móvel, o nome deste é memorizado pela aplicação móvel assim que o utilizador submete uma experiência pela primeira vez, caso contrário, a experiência submetida por este ficará associado a um utilizador anónimo.

É na tabela *Experiences* que encontram-se todos os dados recolhidos durante a experiência de mobilidade e que foi submetida pela aplicação móvel. A descrição, opcional, sobre a experiência por parte do utilizador é armazenada no campo *description*, e no campo *content* é guardado os dados recolhidos dos sensores. Este campo onde são armazenados os dados referentes à experiência de mobilidade é um campo XML, que alberga os dados no formato no qual são recebidos. Esta decisão foi tomada após levar em consideração alguns aspetos como:

i. Uma vez que os dados podem ser disponibilizados a terceiros em formato XML, KML e CSV, se este já se encontrar armazenado em formato XML, retiraria tempo de processamento na resposta;
ii. Se os dados recolhidos relativos à experiência já se encontrarem em formato XML, é possível a sua transformação em outros formatos, e.g. KML e CSV, recorrendo apenas a documentos de transformação *eXtensible Stylesheet Language Transformation* (**XSLT**) (44);
iii. Uma vez que a aplicação móvel dá a liberdade ao utilizador de escolher quais os sensores que este pretende utilizar durante a recolha dos dados, os valores que são submetidos para o repositório têm uma dimensão variável, o que faria o diagrama entidade associação do repositório mais complexo caso fosse necessário armazenar a informação em tabelas separadas;
iv. Com o armazenamento da informação submetida em tabelas separadas, o tempo de resposta aos pedidos do utilizador com intenção de obtenção de informação

seria maior, visto que seria necessário um maior número de instruções por forma a agregar e gerar o documento XML, KML ou CSV de resposta, referente à experiência pretendida.

Como foi referido anteriormente, no momento de submissão de uma determinada experiência de mobilidade para o repositório, é dado ao utilizador a possibilidade de escolher os meios de transporte que foram utilizados durante a recolha desses dados. A informação dos meios de transporte usados pelo utilizador é armazenada na tabela *ExperienceTransportDeclared* com a finalidade de poder verificar a eficiência do algoritmo de classificação implementado no repositório, comparando o resultado determinado pelo repositório com o que o utilizador afirma ter utilizado.

A tabela *ExperienceTransport* é similar à tabela *ExperienceTransportDeclared*, contudo, enquanto que a tabela *ExperienceTransportDeclared* guarda o meio de transporte que o utilizador afirma ter utilizado durante a recolha dos dados, a tabela *ExperienceTransport* guarda o meio de transporte que o repositório pressupõe que tenha sido utilizado, após analisar os dados recebidos por parte do utilizador. Assim sendo, se se efetuar uma comparação entre as duas tabelas, assumindo que o utilizador inseriu o meio de transporte correto aquando da recolha dos dados, pode-se verificar a eficiência do algoritmo presente no repositório.

4. Avaliação experimental

4.1. Descrição experimental

Para esta fase da avaliação experimental, a recolha dos dados necessários foi efetuado recorrendo ao dispositivo Android HTC Primo, também conhecido no mercado como HTC One V, que possui as seguintes caraterísticas:

Caraterísticas do Dispositivo	
Marca	HTC
Modelo	One V
Sistema Operativo	Android 4.0.3
Processador	1GHz
Memória interna	4GB
Sensores	A-GPS Acelerómetro Proximidade Luminosidade

Tabela 6 - Caraterísticas do HTC One V

De modo a proceder à recolha dos dados, reduzindo a influência dos fatores externos, e permitindo que os dados de todas as experiências realizadas sejam recolhidos da forma mais similar possível, o utilizador possui uma forma própria de orientar o dispositivo. Para tal, uma vez iniciada a recolha dos dados, o utilizador deve, enquanto se encontrar numa posição vertical efectuar os seguintes passos:

1. O mais brevemente possível, colocar o dispositivo no bolso direito das calças, ou o mais próximo possível da coxa direita;
2. O dispositivo deve estar orientado com o monitor virado para o utilizador e a parte superior do dispositivo, parte do altifalante, para cima.

Caso o utilizador pretenda efectuar um percurso pedestre, deve andar normalmente. Caso pretenda efectuar um percurso de autocarro ou de comboio, este deve colocar o dispositivo orientado de acordo com as indicações previamente dadas e sentar-se com a

cara virada no sentido do deslocamento do meio de transporte utilizado. As figuras abaixo demonstram a orientação correcta do dispositivo. Este deve permanecer nesta posição até que se pretenda interromper a gravação dos dados.

Figura 17a – Posição vertical para recolha de dados

Figura 17b – Posição sentada para recolha de dados

Figura 17 - Posição para recolha de dados

4.2. Recolha de dados e análise

De modo a assegurar que a aplicação desenvolvida é capaz de recolher os dados sensoriais dos dispositivos móveis, bem como efetuar a determinação do meio de transporte utilizado aquando da recolha dos dados, foi disponibilizada uma versão da aplicação a alguns utilizadores de teste juntamente com um manual de utilizador de modo a explicar a intenção da criação da aplicação e a forma como esta deve ser utilizada.

Como foi referido anteriormente na secção *3.1.2.2 Analytics*, esta aplicação de testes foi distribuída após efetuar a integração com o *Flurry Analytics*, permitindo assim verificar a navegação do utilizador na aplicação, o tempo despendido na mesma, o país de onde se encontram a aceder à aplicação, entre outros aspetos. Toda essa informação é utilizada para complementar a informação sensorial extraída dos dispositivos, de modo a perceber se os utilizadores de teste encontram-se a utilizar a aplicação do modo como foi pensado.

De seguida, é exemplificado o processo de recolha e catalogação de duas experiências de mobilidade, de um utilizador do sistema para efeitos de demonstração.

Na Figura 18 e Figura 19, são apresentadas graficamente nos mapas da Google as informações sensoriais recolhidas das referidas experiências de mobilidade. A Figura 18 apresenta uma experiência realizada de autocarro entre as zonas do Hospital Amadora-Sintra e Algés, enquanto que a Figura 19 representa uma experiência realizada num percurso pedestre na zona do Saldanha, em Lisboa.

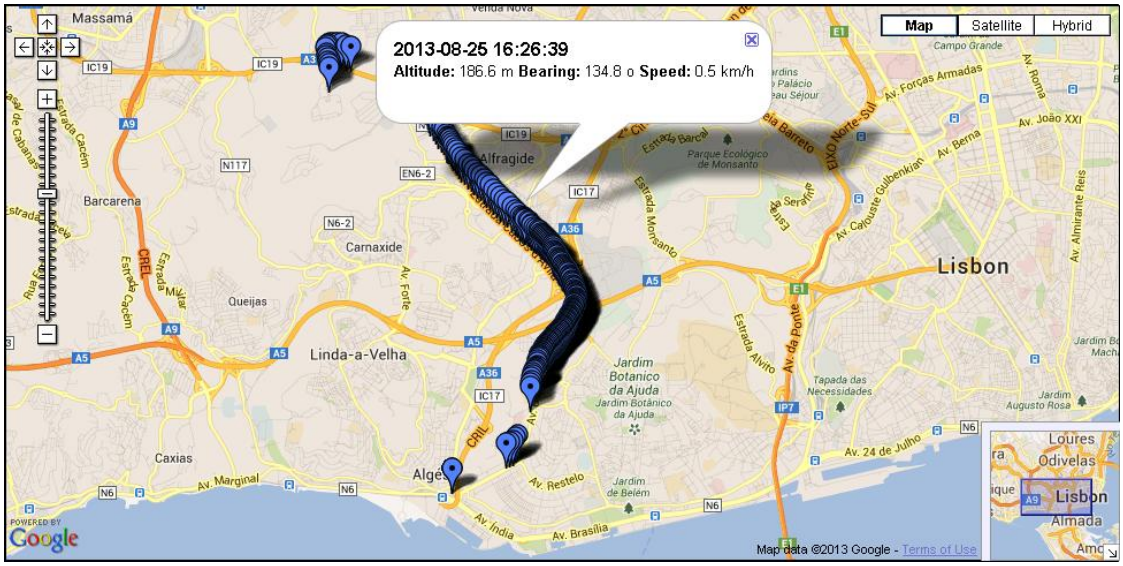

Figura 18 - Experiência realizada entre o Hospital Amadora-Sintra e Algés

Na Figura 18, é apresentado um exemplo da informação recolhida do sensor num ponto específico, mais concretamente, é apresentada a informação da altitude relativa ao nível do mar, que é de 186.6 metros, a orientação de 134.8º em relação ao Norte magnético e a velocidade 0.5 km/h uma vez que o autocarro se encontrava a sair de uma paragem.

Figura 19 - Experiência realizada na zona do Saldanha, Lisboa

Na Figura 19, analisando um exemplo da informação que é recolhida durante a realização de uma experiência de mobilidade, é possível verificar que a altitude relativa ao nível do mar, que é de 148.4 metros, a orientação de 0º em relação ao Norte magnético e a velocidade 0.0 km/h, pois o utilizador encontrava-se na altura, parado.

Ambos os percursos foram efetuados recolhendo a informação do sensor GPS e acelerómetro, sendo que os dados obtidos do GPS são utilizados para a apresentação visual do percurso, enquanto que os dados obtidos do acelerómetro são utilizados para a determinação do meio de transporte utilizado.

Como é possível verificar nas figuras, à exceção de alguns troços em que não foi possível obter o sinal de GPS de modo a ser apresentada a informação no mapa, a maioria do percurso encontra-se visível. A informação apresentada no mapa é o resultado da recolha dos dados e consequente geração de um documento KML que foi submetido a um *website*[2] em que este, quando recebe o documento KML, apresenta os dados nos mapas da Google.

No momento de submissão destas duas experiências para o repositório, o utilizador informa que as experiências foram recolhidas num percurso de autocarro e a pé respetivamente. Como foi dito anteriormente, esta informação será utilizada numa fase inicial, de modo a poder verificar a taxa de sucesso do algoritmo implementado no repositório e que é responsável por efetuar a classificação do meio de transporte utilizado aquando da recolha dos dados.

No lado do repositório, aquando da receção de uma determinada experiência de mobilidade, as informações enviadas pela aplicação cliente são todas persistidas na base de dados, como se pode verificar na Figura 20, onde se destacam as experiências da tabela cujos *_experienceId* são o 52 (destacado com a linha preta contínua) e o 73 (destacada com a linha preta tracejada), referentes às experiências realizadas de a pé e de autocarro respetivamente.

[2] http://display-kml.appspot.com/

	_experienceId	_userId	_description	_content
51	51	edison.spencer	test 18f	<experience xmlns="http://www.w3schools.com" xml...
52	52	edison.spencer	test 19f	<experience xmlns="http://www.w3schools.com" xml...
53	53	edison.spencer	test 13t	<experience xmlns="http://www.w3schools.com" xml...
		...		
73	73	edison.spencer	test 25b g+a	<experience xmlns="http://www.w3schools.com" xml...
74	74	edison.spencer	test 26b g+a	<experience xmlns="http://www.w3schools.com" xml...

Figura 20 - Tabela *Experiences*

Paralelamente a este processo de persistência da informação na tabela *Experiences*, o repositório efetua a análise da informação presente no campo *_content*, ou seja, o conteúdo XML que é enviado pela aplicação cliente e que possui a informação sensorial recolhida.

O resultado da análise a este campo, será armazenada na tabela *ExperienceTransport*, presente na Figura 21, e que efetua a associação da experiência recebida ao meio de transporte determinado pelo algoritmo de classificação do repositório.

	_experienceId	_transportId
51	51	9
52	52	9
53	53	6
	...	
73	73	6
74	74	6
75	75	6

Figura 21 - Tabela *ExperienceTransport*

Se se efetuar uma análise à Figura 22, será possível que, tal decisão foi tomada, uma vez que o resultado da execução do algoritmo sobre as experiências recebidas, obteve uma classificação maior para um determinado tipo de transporte face aos outros dois que atualmente são igualmente suportados. Nomeadamente, é possível verificar que, para a experiência realizada a pé, o algoritmo determinou com 96,7% de probabilidade, que o percurso foi realizado a pé, face aos restantes 1,02% e 2,25% de probabilidade de ter sido realizado de comboio ou de autocarro respetivamente.

O mesmo acontece com a experiência realizada de autocarro em que, com uma percentagem de 94,6% o algoritmo determinou como tendo sido realizado de autocarro,

face aos restantes 0,5% e 4,83% de probabilidade de ter sido realizado a pé ou de comboio respetivamente.

	timestamp	entry
154	2013-08-22 00:10:13.157	on foot > 96,7213114754088
155	2013-08-22 00:10:13.160	train > 1,02459016393443
156	2013-08-22 00:10:13.160	bus > 2,25409836065574
157	2013-08-22 00:11:18.800	on foot > 1,12903225806452
...		
214	2013-09-08 18:02:32.310	on foot > 0,506186726659...
215	2013-09-08 18:02:33.730	train > 4,83689538807649
216	2013-09-08 18:02:33.753	bus > 94,6569178852663

Figura 22 - Tabela *Log*

Na Figura 22, com a linha preta contínua se apresenta o resultado do algoritmo de classificação sobre a experiência realizada a pé, e com a linha preta tracejada o resultado referente à experiência realizada de autocarro.

Quando comparada a informação resultante da execução do algoritmo de classificação com a Figura 23, onde é armazenado o meio de transporte que o utilizador afirma ter utilizado (tabela *ExperienceTransportDeclared*), pode-se verificar que realmente as experiências foram efetivamente realizadas num percurso a pé e de autocarro respetivamente.

	_experienceId	_transportId
49	51	9
50	52	9
51	53	5
...		
64	73	6
65	74	6
66	75	6

Figura 23 - Tabela *ExperienceTransportDeclared*

Os meios de transporte equivalentes a cada um dos *_transportId* apresentados nas figuras acima podem ser consultados na Figura 24.

	_transportId	_transportName
1	1	unknown
2	2	car
3	3	bike
4	4	bycicle
5	5	train
6	6	bus
7	7	tram
8	8	taxi
9	9	on foot

Figura 24 - Tabela *Transports*

Para complementar a informação recolhida dos sensores do dispositivo e que foi voluntariamente submetida pelo utilizador da aplicação, temos ainda a informação do *Flurry Analytics* que é submetida automaticamente pela aplicação e que permite aceder a alguns outros dados da aplicação que não são suportados nativamente por este. Nomeadamente, consultando a informação recolhida por este no *website* do *Flurry Analytics*, pode-se verificar que, como é ilustrado na Figura 25 referente a um gráfico de navegação presente no *website* e tendo por base os dados recebidos, após o inicio do programa, 50% dos utilizadores acedem ao ecrã de definições, enquanto que 20% destes preferem aceder ao ecrã de favoritos, e os restantes 30% encontram-se divididos entre o inicio e fim da captura de informação dos sensores.

Figura 25 - Gráfico de navegação do *Flurry Analytics*

É ainda possível verificar que, apesar dos utilizadores de teste efetuarem vários acessos de pouca duração, cerca de 0 a 30 segundos, possivelmente com a intenção de explorar a aplicação numa primeira vez, a maioria deles efetuou acessos com uma duração média de 30 segundos a 10 minutos, desta feita assumindo que o acesso foi efetuado com o intuito de efetuar a recolha de dados.

4.3. Avaliação de precisão e cobertura

Uma vez recolhida uma séria de experiências de mobilidade, foi efetuada uma análise de precisão e cobertura de modo a avaliar o desempenho do algoritmo utilizado no repositório para a determinação dos meios de transporte.

Analisando as experiências de mobilidade submetidas no repositório, verificam-se, como é apresentada na segunda coluna da Tabela 7, que os utilizadores submeteram 19 experiências de mobilidade efetuadas no comboio, 28 experiências efetuadas no autocarro e 23 a pé.

Percurso	número de experiências submetidas	número de experiências classificadas	número de experiências corretamente classificadas
Comboio	19	24	14
Autocarro	28	26	20
Pedestre	23	26	23
Indeterminado	9	3	3
Total	79	79	60

Tabela 7 - Número de experiências

Efetuando ainda uma análise sobre o resultado do algoritmo responsável pela determinação do meio de transporte, que pode ser observado também na terceira coluna da Tabela 7, é possível verificar que, 24 das experiências submetidas foram classificadas como sendo pertencentes a experiências realizadas no comboio, 26 no autocarro e 26 num percurso pedestre.

Levando em consideração que, dessas experiências classificadas, foram corretamente classificadas 14 experiências para os percursos efetuados no comboio, 20 para os percursos efetuados no autocarro e 23 para os pedestres, se aplicarmos as fórmulas de precisão e cobertura, obtemos as seguintes percentagens para cada meio de transporte.

Percurso	Precisão	Cobertura
Comboio	58,3%	73,7%
Autocarro	76,9%	71,4%
Pedestre	88,5%	100,0%
Desconhecido	100,0%	33,3%
Média	80,9%	69,6%

Tabela 8 - Determinação da precisão e cobertura

Como é possível verificar pela Tabela 8, o algoritmo utilizado na determinação do meio de transporte possui, em média, uma cobertura de 69,9% e uma precisão de 80,9%.

5. Epílogo

Neste capítulo, procederemos à apresentação das conclusões deste projeto, assim como perspetivas de desenvolvimentos futuros.

5.1. Conclusão

Para a realização deste projeto, foi proposta a elaboração de um sistema que efetue a monitorização dos dados sensoriais dos hábitos de mobilidade dos utilizadores.

Uma vez que os dispositivos móveis encontram-se atualmente em franco crescimento no seio da população mundial, e possuem uma vasta gama de sensores, propôs-se a utilização destes, por forma a obter os dados sensoriais que são gerados durante a experiência de mobilidade.

Os dados sensoriais recolhidos foram posteriormente submetidos para um repositório central onde, estes foram analisados, armazenados e resultados extraídos dos mesmos, o que permitiu ir adquirindo informação sobre quais os meios de transporte mais utilizados entre os utilizadores, bem como para cada utilizador em particular. Estes resultados obtidos da análise dos dados sensoriais são utilizados para auxiliar os utilizadores na tomada de decisões aquando da realização de novas experiências de mobilidade. Os resultados são igualmente disponibilizados de forma pública, sem revelar informação sensível sobre o utilizador, permitindo assim que estes possam ser utilizados para fins diversos.

Para o desenvolvimento deste projeto, não foi necessária a aprendizagem de nenhuma linguagem de programação nova, uma vez que as tecnologias necessárias para o desenvolvimento do mesmo foram Java e C#, tecnologias estas que já tinham sido lecionadas durante o curso. Contudo, a elaboração deste projeto permitiu um primeiro contato com a implementação de uma aplicação móvel para o sistema operativo Android, desenvolvimento de um serviço *web* que dê suporte ao funcionamento do

mesmo, bem como o conhecimento de alguns aspetos a serem levados em conta para que seja efetuada a correta comunicação entre as duas partes.

Embora os testes efetuados sobre o sistema, permitiram verificar a existência de experiências de mobilidade em que não foi possível determinar o meio de transporte utilizado, o sistema conseguiu, em 94,5% dos casos, determinar com exatidão o meio de transporte utilizado.

5.2. Desenvolvimentos futuros

Com a conclusão deste projeto, foi possível observar que a maioria dos objetivos propostos inicialmente foram alcançados, contudo, ficaram ainda por realizar alguns aspetos que poderiam dar mais qualidade ao projeto em questão, nomeadamente:

i. **Submissão automática dos dados**. Uma vez que a aplicação móvel foi desenvolvida especificamente para a recolha dos dados, o utilizador está consciente, aquando da instalação da aplicação, da funcionalidade principal desta. Assim sendo, uma vez que uma experiência de mobilidade seja dada como terminada, a aplicação poderia, na presença de uma ligação ativa à internet, efetuar a submissão automática da experiência de mobilidade, perguntado ao utilizador simplesmente o seu nome de utilizador caso este ainda não tenha sido especificada;
ii. **Detecção de um maior número de meios de transporte**. Melhorar o algoritmo existente atualmente no repositório, permitindo assim que este passe a reconhecer não somente as experiências de mobilidade que tenham sido efetuadas num percurso pedestre, de comboio ou de autocarro, mas que reconhecesse igualmente outros meios de transporte, como por exemplo, carro, elétrico, bicicleta, entre outros;
iii. **Criação de um sistema de administração**. Este sistema de gestão permitiria ao responsável pela informação presente no repositório, efetuar a gestão da informação armazenada efetuando ações como a adição/remoção de utilizadores, manipulação das experiências submetidas pelos utilizadores, entre outros;
iv. **Criação de um *website* para visualização de informação própria**. Criação de um *website* que permitisse de uma forma mais interativa, o acesso aos dados

disponibilizados pelos serviços *web*, nomeadamente permitir que um utilizador autenticado aceda aos dados por ele submetidos, e a uma entidade aceder aos dados existentes no repositório mas sem comprometer os dados confidenciais do utilizador que os submeteu;

v. **Integração com a rede social Integra**. Como foi idealizado inicialmente, as informações recolhidas recorrendo a esta aplicação seriam posteriormente integradas com a rede social Integra. Contudo, este não foi possível devido ao curto espaço existente para o estudo da arquitetura da rede social e respetiva alteração de modo a permitir tal integração. Não obstante, este é um ponto que ajudaria a dar grande visibilidade ao projeto, na medida em que permitiria ao utilizador ver posteriormente numa rede social o seu percurso realizado, bem como efetuar a partilha dessa experiência de mobilidade.

Referências

1. **Sperling, D. e Gordon, D.** Two Billion Cars: Driving Toward Sustainability. s.l. : Oxford University Press, 2009.

2. **Lomax, Tim, Schrank, David e Eisele, Bill.** Inconsistent Traffic Conditions Forcing Texas Commuters to Allow Even More Extra Time. *Urban Mobility Information.* [Online] [Citação: 19 de Setembro de 2013.] http://d2dtl5nnlpfr0r.cloudfront.net/tti.tamu.edu/documents/tti-umr.pdf.

3. **Brasil, Agência.** Estudo indica melhores meios de transporte para grandes cidades do Brasil. *Folha de São Paulo.* [Online] 1 de Setembro de 2012. [Citação: 9 de Fevereiro de 2013.] http://folha.com/no1146996.

4. **Sager, Ira.** Before IPhone and Android Came Simon, the First Smartphone. *Businessweek.* [Online] Boomberg Businessweek Technology, 29 de Junho de 2012. [Citação: 26 de Janeiro de 2013.] http://www.businessweek.com/articles/2012-06-29/before-iphone-and-android-came-simon-the-first-smartphone.

5. **Rushton, Katherine.** Number of smartphones tops one billion. *The Telegraph.* [Online] 17 de Outubro de 2012. [Citação: 28 de Janeiro de 2013.] http://www.telegraph.co.uk/finance/9616011/Number-of-smartphones-tops-one-billion.html#.

6. **Bureau, United States Census.** World POPClock Projection. [Online] [Citação: 28 de Janeiro de 2013.] http://www.census.gov/population/popclockworld.html.

7. **Spicestellar.** A Brief History of Android. *Visual.ly.* [Online] Spicestellar. [Citação: 27 de Janeiro de 2013.] http://visual.ly/brief-history-android.

8. **Google.** Discover Android. [Online] [Citação: 27 de Janeiro de 2013.] http://www.android.com/about/.

9. —. About Google. [Online] [Citação: 11 de Fevereiro de 2013.] https://www.google.pt/intl/en/about/company/.

10. Open Handset Alliance. [Online] Open Handset Alliance. [Citação: 14 de Setembro de 2013.] http://www.openhandsetalliance.com/index.html.

11. **Apple.** iOS 6. [Online] [Citação: 27 de Janeiro de 2013.] http://www.apple.com/ios/.

12. —. About. [Online] [Citação: 27 de Janeiro de 2013.] http://www.apple.com/.

13. **Google.** Platform Versions. [Online] [Citação: 05 de Janeiro de 2013.] http://developer.android.com/about/dashboards/index.html.

14. —. Sensors Overview. [Online] [Citação: 05 de Janeiro de 2013.] http://developer.android.com/guide/topics/sensors/sensors_overview.html.

15. **Alliance, Open Handset.** Android Overview. [Online] Open Handset Alliance. [Citação: 14 de Setembro de 2013.] http://www.openhandsetalliance.com/android_overview.html.

16. The Linux Kernel Archives. [Online] Linux Kernel Organization, Inc. [Citação: 14 de Setembro de 2013.] https://www.kernel.org/category/about.html.

17. **Google.** dalvik. *Google Code*. [Online] [Citação: 14 de Setembro de 2013.] https://code.google.com/p/dalvik/.

18. **Brush, A., Krumm, John e Scott, James.** Activity Recognition Research: The Good, the Bad and the Future. *Pervasive 2010 Workshop: How to do good research in activity recognition.* s.l. : Microsoft Research, 2010.

19. **Bao, Ling e Intille, Stephen S.** Activity Recognition from User-Annotated Acceleration Data. Massachusetts : Lecture Notes Computer Science, 2004. Vol. 3001, 1-17.

20. **Kwapisz, Jennifer R., Weiss, M. Gary e Moore, A. Samuel.** Activity Recognition using Cell Phone Accelerometers. s.l. : Fordham University.

21. **Wu, Xindong, et al., et al.** Top 10 algorithms in data mining. s.l. : Springer-Verlag, 2007. Vol. 14, 1-37. 10.1007/s10115-007-0114-2.

22. Data Mining: What is Data Mining? *Anderson UCLA*. [Online] [Citação: 3 de Setembro de 2013.] http://www.anderson.ucla.edu/faculty/jason.frand/teacher/technologies/palace/datamining.htm.

23. **Salzberg, Steven L.** C4.5: Programs for Machine Learning. Boston : Kluwer Academic Publishers, 1994.

24. **Press, William H., et al., et al.** Section 16.5. Support Vector Machines. *Numerical Recipes: The Art of Scientific Computing.* s.l. : Cambridge University Press, 2007. 978-0-521-88068-8.

25. **Cover, T. e Hart, P.** Nearest neighbor pattern classification. *IEEE Transactions on.* 1967. Vol. 13, 21-27. 0018-9448.

26. **Rish, I.** An empirical study of the naive Bayes classifier. s.l. : T.J. Watson Research Center.

27. **Ozer, Patrick.** Data Mining Algorithms for Classification. 2008.

28. **Rokach, Lior e Maimon, Oded.** *Data Mining with Decision Trees.* s.l. : World Scientific Publishing Co. Pte. Ltd., 2008. ISBN-13 978-981-277-171-1.

29. **Quinlan, J. Ross.** Discovering rules by induction from large collections of examples. Edinburgh : Edinburgh University Press, 1979.

30. **Schaffer, C.** Deconstructing the digit recognition problem. *Proceedings of the Ninth International Machine Learning Workshop.* San Mateo : Morgan Kaufmann, 1992. pp. 394-399.

31. **Powers, David.** Evaluation: From precision, recall and f-measure to ROC, informedness, markedness & correlation. s.l. : Journal of Machine Learning Technologies, 2011. Vol. 2, 1. 2229-3981.

32. **Microsoft.** Model-View-Controller. [Online] [Citação: 05 de Janeiro de 2013.] http://msdn.microsoft.com/en-us/library/ff649643.aspx.

33. **Burbeck, Steve.** How to use Model-View-Controller. *The Smalltalk Archive.* [Online] 04 de Março de 1997. [Citação: 07 de Janeiro de 2013.] http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html.

34. **Reenskaug, Trygve.** Models - Views - Controllers. *University of Oslo.* [Online] 10 de Dezembro de 1979. [Citação: 07 de Janeiro de 2013.] http://heim.ifi.uio.no/~trygver/1979/mvc-2/1979-12-MVC.pdf.

35. **Bray, Tim, et al., et al.** Extensible Markup Language. *World Wide Web Consortium - W3C.* [Online] 26 de Novembro de 2008. [Citação: 07 de Janeiro de 2013.] http://www.w3.org/TR/REC-xml/.

36. **Holmes, Chris.** OWS-5 KML Engineering Report. 2008. OGC 07-124r2.

37. **Group, Object Management.** UML Resource Page. *UML.* [Online] Object Management Group. [Citação: 19 de Agosto de 2013.] http://www.uml.org/.

38. **Rouse, Margaret.** XSD (XML Schema Definition). *Search SOA.* [Online] Setembro de 2005. [Citação: 19 de Setembro de 2013.] http://searchsoa.techtarget.com/definition/XSD.

39. **Flurry.** Industry-Leading App Analytics for free. [Online] [Citação: 27 de Agosto de 2013.] http://www.flurry.com/flurry-analytics.html.

40. **Shafranovich, Y.** [Online] SolidMatrix Technologies, Inc., Outubro de 2005. [Citação: 13 de Setembro de 2013.] http://www.ietf.org/rfc/rfc4180.txt.

41. **Garmin.** What is GPS? [Online] [Citação: 13 de Janeiro de 2013.] http://www8.garmin.com/aboutGPS/.

42. **Arora, Rohit e Suman.** Comparative Analysis of Classification Algorithms on Different Datasets using WEKA. India : International Journal of Computer Applications, Setembro, 2012. Vol. 54, 13.

43. **Waikato, University of.** Weka 3: Data Mining Software in Java. [Online] [Citação: 19 de Agosto de 2013.] http://www.cs.waikato.ac.nz/ml/weka/.

44. **W3C.** XSL Transformations (XSLT). [Online] 16 de Novembro de 1999. [Citação: 14 de Setembro de 2013.] http://www.w3.org/TR/xslt.

45. **Chapman, Cameron.** The History of the Internet in a Nutshell. *Six Revisions.* [Online] 15 de Novembro de 2009. [Citação: 10 de Janeiro de 2013.] http://sixrevisions.com/resources/the-history-of-the-internet-in-a-nutshell/.

46. **Llamas, Ramon, Restivo, Kevin e Shirer, Michael.** Android Marks Fourth Anniversary Since Launch. *International Data Corporation.* [Online] 01 de Novembro de 2012. [Citação: 10 de Janeiro de 2013.] https://www.idc.com/getdoc.jsp?containerId=prUS23771812.

47. **Alliance, Wi-Fi.** Discover and Learn. [Online] [Citação: 13 de Janeiro de 2013.] http://www.wi-fi.org/discover-and-learn.

48. **Integra.** Integra Social Network. [Online] [Citação: 11 de Janeiro de 2013.] http://integra.isel.pt.

49. **Pereira, Pedro.** Observador de Mobilidade Sustentável. [Online] [Citação: 11 de Janeiro de 2013.] http://start.isel.pt/pdfs/OMS.pdf.

50. **Lane, Nicholas D., et al., et al.** A survey of Mobile Phone Sensing. *Ad Hoc and Sensor Networks.* s.l. : Dartmouth College.

51. **Arora, Alka, et al., et al.** Classification Using Decision Trees. *Data Mining Techniques and Tools for Knowledge Discovery in Agricultural Datasets.* New Delhi : Indian Agricultural Statistics Research Institute.

Anexos

Anexo 1 – Estrutura do documento XML

```
<experience xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://start.isel.pt:5004/schemas/xsd/experience.xsd">
  <data>
    <timestamp>2013-08-25 19:45:47</timestamp>
    <sensor origin="accelerometer">
      <x-axis>-9.765789</x-axis>
      <y-axis>-0.87170225</y-axis>
      <z-axis>-3.0645783</z-axis>
      <w-axis>-1.0</w-axis>
    </sensor>
  </data>
  <!-- … -->
  <data>
    <timestamp>2013-08-25 19:45:47</timestamp>
    <provider origin="gps">
      <latitude>38.709827</latitude>
      <longitude>-9.215343</longitude>
      <altitude>125.7</altitude>
      <bearing>0.0</bearing>
      <speed>0.0</speed>
    </provider>
  </data>
</experience>
```

Anexo 2 – Estrutura do documento XSD

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3schools.com" targetNamespace="http://www.w3schools.com"
elementFormDefault="qualified">
  <xs:attributeGroup name="origin">
    <!-- definition of attributes -->
    <xs:attribute name="origin" type="xs:string" />
  </xs:attributeGroup>
  <!-- definition of complex elements -->
  <xs:element name="provider">
    <xs:complexType>
      <xs:all>
        <xs:element name="latitude" type="xs:float" />
        <xs:element name="longitude" type="xs:float" />
        <xs:element name="altitude" type="xs:float" />
        <xs:element name="bearing" type="xs:float" />
        <xs:element name="speed" type="xs:float" />
      </xs:all>
      <xs:attributeGroup ref="origin" />
    </xs:complexType>
  </xs:element>
  <xs:element name="sensor">
    <xs:complexType>
      <xs:all>
        <xs:element name="x-axis" type="xs:float" />
        <xs:element name="y-axis" type="xs:float" />
        <xs:element name="z-axis" type="xs:float" />
        <xs:element name="w-axis" type="xs:float" />
      </xs:all>
      <xs:attributeGroup ref="origin" />
    </xs:complexType>
  </xs:element>
  <xs:element name="timestamp" type="xs:string" />
<xs:element name="data">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="timestamp" minOccurs="1" maxOccurs="1" />
        <xs:choice>
          <xs:element ref="provider" />
          <xs:element ref="sensor" />
        </xs:choice>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <!-- definition of main type -->
  <xs:element name="experience">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="data" minOccurs="0" maxOccurs="unbounded" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Anexo 3 – Documento de transformação XSLT (XML para KML)

```
<xsl:stylesheet xmlns:xs="http://www.w3schools.com"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:output method="xml" indent="yes" encoding="utf-8" />
   <xsl:template match="/">
      <kml>
         <Document>
            <xsl:for-each select="xs:experience/xs:data/xs:provider">
               <Placemark>
                  <name>
                     <xsl:value-of select="../xs:timestamp" />
                  </name>
                  <description>
                     <xsl:text disable-output-escaping="yes"><![CDATA[</xsl:text>
<b>Altitude:</b>
<xsl:value-of select="xs:altitude"/>
m
<b>Bearing:</b>
<xsl:value-of select="xs:bearing"/>
o
<b>Speed:</b>
<xsl:value-of select="xs:speed"/>
km/h
<xsl:text disable-output-escaping="yes">]]></xsl:text>
                  </description>
<Point>
                     <coordinates>
                        <xsl:value-of select="xs:longitude" />
,
<xsl:value-of select="xs:latitude" />
,
<xsl:value-of select="xs:altitude" /></coordinates>
                  </Point>
               </Placemark>
            </xsl:for-each>
         </Document>
      </kml>
   </xsl:template>
</xsl:stylesheet>
```

Anexo 4 – Documento de transformação XSLT (XML para CSV)

```
<xsl:stylesheet xmlns:xs="http://www.w3schools.com"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <xsl:output method="text" />
   <xsl:template match="/">
timestamp,origin,reading0,reading1,reading2,reading3,reading4
<xsl:for-each select="xs:experience/xs:data"><xsl:value-of select="./xs:timestamp" />
,
<xsl:choose><xsl:when test="./xs:sensor"><xsl:value-of select="./xs:sensor/@origin" />
,
<xsl:for-each select="./xs:sensor/*"><xsl:value-of select="." /><xsl:if
test="not(position()=last())">,</xsl:if></xsl:for-
each><xsl:text></xsl:text></xsl:when><xsl:otherwise><xsl:value-of
select="./xs:provider/@origin" />
,
<xsl:for-each select="./xs:provider/*"><xsl:value-of select="." /><xsl:if
test="not(position()=last())">,</xsl:if></xsl:for-
each><xsl:text></xsl:text></xsl:otherwise></xsl:choose></xsl:for-each></xsl:template>
</xsl:stylesheet>
```