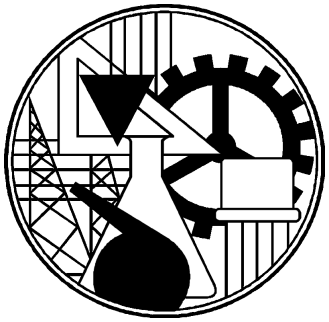


Instituto Superior de Engenharia de Lisboa

Área Departamental de Engenharia Electrónica e Telecomunicações e de Computadores

Sistema de recolha e execução de rotas (*GPS*) e partilha em rede social *RecGPS*

Rodrigo Matias Carvalho Silva
(Licenciado)

Trabalho de projecto realizado para a obtenção do grau de mestre em Engenharia Informática e de Computadores

Orientador:

Mestre Fernando Miguel Santos Lopes de Carvalho

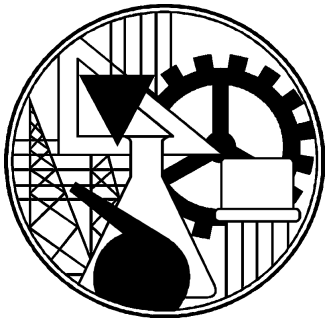
Júri:

Presidente: Professor Coordenador Fernando Manuel Gomes de Sousa

Vogal: Mestre Paulo Alexandre Leal Barros Pereira

Vogal: Mestre Fernando Miguel Santos Lopes de Carvalho

13 de Dezembro de 2010



Instituto Superior de Engenharia de Lisboa

Área Departamental de Engenharia Electrónica e Telecomunicações e de Computadores

Sistema de recolha e execução de rotas (*GPS*) e partilha em rede social *RecGPS*

Rodrigo Matias Carvalho Silva
(Licenciado)

Trabalho de projecto realizado para a obtenção do grau de mestre em Engenharia Informática e de Computadores

Orientador:

Mestre Fernando Miguel Santos Lopes de Carvalho

Aluno:

Rodrigo Matias Carvalho Silva

---

13 de Dezembro de 2010

Orientação

Trabalho realizado sob a orientação do

Professor Fernando Miguel Santos Lopes de Carvalho

Equiparado a Professor Adjunto da

Área Departamental de Engenharia Electrónica e Telecomunicações e de Computadores do

Instituto Superior de Engenharia de Lisboa

Resumo

A popularização dos *PDA* (*Personal Data Assistant*) originou um aumento das suas capacidades e funcionalidades, nomeadamente a inclusão de sistemas de posicionamento global (*GPS - Global Positioning System*). Tendo como objectivo o aproveitamento das capacidades deste tipo de dispositivos surge o sistema *RecGPS*. O *RecGPS* é um pacote de aplicações para recolha, gestão, análise e partilha de percursos (rotas). A recolha e armazenamento das rotas é efectuada através de uma aplicação instalada no *PDA* com recurso à funcionalidade *GPS* do mesmo. A gestão e análise das rotas pode ser efectuada nas aplicações *PDA* e *web*, sendo a partilha efectuada através de aplicação *web* (portal/rede social). A colecção de rotas tanto pode ser armazenada no dispositivo como através da aplicação *web*.

Ao nível do *PDA* para além da criação de novas rotas é possível executar as rotas existentes, efectuando recolha de dados que permitem a comparação com as execuções anteriores. A rede social permite a partilha de rotas e fomenta a discussão dos utilizadores sobre as mesmas.

Para permitir a verificação das capacidades da aplicação para *PDA* são efectuados testes. Tendo por base um conjunto de amostras são obtidos resultados através da aplicação e efectuada a sua comparação.

Abstract

The PDA (Personal Data Assistant) popularity growth originated the increase of these devices capabilities and functionalities, like the Global Positioning System (GPS). With the goal of taking advantage of these capabilities the *RecGPS* idea appears. The *RecGPS* is a package of applications that records, manages, analyses and shares routes. The recording is made through an PDA installed application that makes use of the devices' GPS. The route management and analysis can be made through PDA and web applications and the sharing is accomplished through the web application (website/social network). The route collection can be stored on the device as well as in the web application.

Beyond the capability to create new routes the PDA allows to execute the existing routes, collecting data that allows the comparison to previous executions. The social network allows the route sharing and nurtures the user discussion around them.

In order to verify the PDA application capabilities tests are made. Based on a sample set the application is used to obtain results which are compared.

Agradecimentos

Quero deixar uma palavra de apreço aos que me apoiaram no decorrer de mais esta etapa da minha vida académica e pessoal:

- Aos familiares, amigos e namorada pelo apoio ao longo deste projecto.
- Ao orientador, Eng.º Fernando Miguel Carvalho, pela orientação dada.
- Aos colegas de curso pelas ideias e opiniões.
- E a todos os que aqui não se encontram referidos mas que de uma forma ou de outra contribuíram para o desenvolvimento deste projecto.

Índice

Lista de Figuras

Lista de Tabelas

Listagens de Código

Terminologia

latitude	-	distância de um ponto ao equador [1]
longitude	-	distância de um ponto da terra ao meridiano geral [1]
coordenadas	-	abcissa e ordenada dum ponto, linha ou superfície [1] (latitude e longitude)
PDA	-	*Personal Data Assistant*
.NET	-	implementação da *Microsoft* de camada de abstracção do sistema operativo
API	-	*Application Programming Interface*
framework	-	conjunto de classes de auxílio para a implementação de um sistema ou aplicação
tags	-	etiquetas
.NETcf	-	*.NET Compact Framework*
ASP.NET	-	*Active Server Pages* da plataforma *.NET*
ASP.NET MVC	-	*ASP.NET Model-View-Controller*
POI	-	*Point Of Interest*/Ponto de Interesse
SMS	-	*Short Message System*/sistema de mensagens curtas
SGBD	-	Sistema de Gestão de Bases de Dados
CRUD	-	*Create, Read, Update, Delete*
DAAB	-	*Data Access Application Block*
IOC	-	*Inversion Of Control*
DI	-	*Dependency Injection*
RML	-	*Route Management Library*
HTTP	-	protocolo para comunicação cliente-servidor

HTTP Get	-	pedido de um endereço(página) do cliente ao servidor
HTTP Post	-	submissão de uma página do cliente ao servidor
Overlay	-	camada; elemento que se posiciona sobre o mapa (*Google Maps*)
Marker	-	marcador de posição (*Google Maps*)
Polyline	-	conjunto de segmentos de recta tratados com uma única entidade (*Google Maps*)
InfoWindow	-	balão informativo (*Google Maps*)
SDK	-	*Software Development Kit*
singleton	-	padrão em que apenas um objecto da classe é permitido

Capítulo 1

Introdução

A passagem de conhecimento depende, entre outros factores, da capacidade de expressão dos intervenientes e do conhecimento do assunto que o receptor tem. Mais dificultada é quando a língua é um entrave. Quer seja um percurso diário, um caminho descoberto ao acaso ou uma viagem demorada existe sempre a dificuldade de transmitir a rota[1] efectuada, não só pelos factores inerentes à passagem de conhecimento, quer pelo seu tamanho, especificidade ou falta de conhecimento da área geográfica.

Com a popularização dos *PDAs* com funcionalidade *GPS* surge a oportunidade de desenvolver um sistema que, recorrendo aos recursos já existentes, simplifique a recolha de rotas (coordenadas e outros dados associados) para posterior partilha através de rede social. Os sistemas de recolha de rotas são hoje em dia comuns e até com interfaces bastante desenvolvidas/intuitivas, mas a adição de informações adicionais, as plataformas de discussão e comparação das rotas, incluindo os dados (adicionais) recolhidos ainda se encontram pouco exploradas, dificultando assim a passagem de conhecimento que se deseja fomentar.

1.1 Objectivos gerais

O *RecGPS* é uma solução que permite a recolha e execução de rotas através de dispositivos móveis com funcionalidade *GPS*. Além disso o *RecGPS* também permite anexar dados adicionais e ainda a partilha e discussão através de rede social. Uma das principais motivações

[1] *rota - s.f. - rumo; caminho* - [1]

para a implementação de um componente de recolha de rotas é a adição de dados adicionais que fomentem a discussão e, consequentemente, a partilha.

A implementação do sistema assenta sobre tecnologias *Microsoft*, nomeadamente: *.NET Compact Framework* para a implementação da aplicação móvel; *ASP.NET* e *framework ASP.NET MVC* para a implementação de aplicação *web* para partilha em rede social; *SQL Server 2008* para a persistência de dados; e *Enterprise Library 4.1* para acesso à persistência, através de *Data Access Application Blocks*.

1.1.1 Aplicação móvel

Para a recolha das coordenadas da rota é utilizada a funcionalidade *GPS* do dispositivo móvel[2], assim como a obtenção de informações adicionais (velocidade e outras). Com as coordenadas é gerado um ficheiro descritor da rota, assim como das informações adicionais. Durante a recolha da rota é também possível a inserção de dados por parte do utilizador (pontos de interesse, alojamento, notas adicionais e outros), sendo que no final é também permitida a sua inserção numa coordenada específica ou ao nível global da rota. São também associadas *tags* referentes à(s) categoria(s) em que a rota se enquadra (caminhada, corrida, bicicleta, automóvel, etc.).

Tendo uma rota é possível efectuar a sua execução - seguimento das coordenadas que compõem a rota - [3], durante a qual são recolhidas as informações adicionais. A execução é feita através da indicação da direcção da próxima coordenada e da distância a que esta se encontra.

1.1.2 Partilha em rede social

Através da rede social os utilizadores têm acesso a funcionalidades de inserção/criação, pesquisa e discussão. A inserção pode ser de rotas ou de dados de execuções das rotas existentes, enquanto que a criação consiste no desenho de uma rota através de *API* de mapas[4]. Para a pesquisa podem ser aplicados vários filtros, por exemplo, zona geográfica

[2] Através da cama de abstracção fornecida na plataforma *.NET.* que permite o consumo de dados *GPS* a várias aplicações em simultâneo
[3] *execução - s. f. - Cumprimento* -[1]
[4] Capítulo 6.2 - Desenvolvimentos futuros.

ou categorias de interesse. Tanto as rotas como as suas execuções têm associados tópicos para dúvidas, comentários e discussão de ideias.

Existem ainda grupos que têm associados tópicos de discussão próprios e permitem a partilha de rotas e execuções referentes a actividades do grupo[5].

1.2 Organização da dissertação

Este documento é formado por cinco capítulos. No decorrer do primeiro capítulo são apresentados os motivos que levam ao sistema *RecGPS* assim como os objectivos que este pretende atingir. É efectuada a descrição dos componentes do sistema e da organização do documento. A terminologia e convenções utilizadas são também incluídas neste capítulo.

No segundo capítulo efectua-se o enquadramento dos principais desafios decorrentes da implementação de cada um dos componentes do *RecGPS*.

Ao longo do terceiro capítulo trata-se a arquitectura alcançada, assim como a especificação associada à mesma. São ainda abordadas as plataformas *web* utilizadas para concretização de alguns dos conceitos propostos no segundo capítulo.

A implementação dos vários níveis da arquitectura proposta é descrita ao longo do quarto capítulo com uma abordagem *bottom-up*.

O quinto e último capítulo está reservado às criticas e desenvolvimentos futuros.

1.3 Convenções

Ao longo do documento as palavras que não façam parte do vocabulário da língua portuguesa estão em *itálico*, conceitos de língua estrangeira que sejam introduzidos estão incluídos na Terminologia e/ou acompanhados de nota de rodapé explicativa[6]. Excertos de código e outros formatos (por exemplo, *XML*) estão em fonte de `tamanho fixo`. As referências são remetidas para o capítulo correspondente através de parêntesis rectos e o respectivo número da referência (exemplo de referência de índice "n": [n])

[5] Capítulo 6.2 - Desenvolvimentos futuros.
[6] exemplo de nota de rodapé explicativa.

Capítulo 2

Enquadramento

A implementação da aplicação móvel apresenta como principais desafios a escolha do formato de armazenamento dos dados recolhidos (Capítulo 2.1.1), o cálculo da distância entre coordenadas (Capítulo 2.1.2), a definição de critérios de recolha de coordenadas (Capítulo 2.1.3) e ainda a comparação com outros sistemas existentes (Capítulo 2.1.4).

Para a partilha em rede social encontram-se desafios ao nível das estruturas de dados a utilizar (Capítulo 2.2.1), a persistência de rotas e execuções (Capítulo 2.2.2), as *API* de mapas consideradas (Capítulo 2.2.3), a discussão de rotas e execuções (Capítulo 2.2.4), a organização dos grupos (Capítulo 2.2.5) e ainda o controlo de acessos na integração em rede social (Capítulo 2.2.6).

Existe ainda o desafio da extensão ao formato *GPX* (Capítulo 2.3.1).

2.1 Aplicação móvel

2.1.1 Formato de armazenamento dos dados recolhidos

Através da *API GPS*[2] [3] disponibilizada pela *.NETcf* são obtidas as coordenadas através de campos das classes `Gps` e `GpsPosition`. Para o armazenamento destas coordenadas em memória secundária (não volátil) consideram-se duas opções de implementação: um formato próprio ou da utilização de um formato standard. Utilizando um formato próprio consegue-se reduzir ao máximo o tamanho da informação armazenada, tendo como grande

desvantagem limitar a utilização das coordenadas recolhidas ao próprio sistema, sendo apenas acessível a outros através de uma ferramenta que efectue a sua conversão para um formato conhecido/standard. Através da utilização de um formato standard baseado em *XML* não só se facilita a leitura por parte de humanos como se permite a utilização directa dos ficheiros gerados noutras plataformas.

Além da interoperabilidade oferecida por um formato *XML* standard é também possível a sua conversão para outros formatos standard através das ferramentas existentes[1]. Para os dados que não os das coordenadas, com recurso aos campos de extensão ao formato *GPX* consegue-se a sua portabilidade sem comprometer a interoperabilidade do ficheiro gerado.

```
<trkpt lat="39.09815270" lon="-9.26060557">
    <ele>30.000000</ele>
    <time>2009-11-12T18:03:50Z</time>
</trkpt>
```

Listagem de Código 2.1: *trackpoiont GPX* - 112char

```
    39.09815270;-9.26060557;30.000000;;
```

Listagem de Código 2.2: formato próprio minimalista - 35char

```
    39.09815270;-9.26060557;30.000000;2009-11-12T18:03:50Z;;
```

Listagem de Código 2.3: formato próprio com data - 56char

```
    -9.26060557,39.09815270,30.000000
```

Listagem de Código 2.4: coordenada KML - 33char

```
<Trackpoint>
    <Time>2009-11-12T18:16:44Z</Time>
    <Position>
        <LatitudeDegrees>39.09815270</LatitudeDegrees>
        <LongitudeDegrees>-9.26060557</LongitudeDegrees>
    </Position>
    <AltitudeMeters>30.000000</AltitudeMeters>
    <DistanceMeters>0.000</DistanceMeters>
</Trackpoint>
```

Listagem de Código 2.5: coordenada GarminCourse - 278char

[1] Exemplo de conversor gratuito em: *gpsbabel.org*[4].

```
1    -926060|3909815|POINT 1-1|0|
```

Listagem de Código 2.6: coordenada TomTom - 28char

Como se pode verificar nas listagens de código 2.1, 2.2, 2.3, 2.4, 2.5 e 2.6, para a indicação de uma coordenada a quantidade de caracteres necessários para a mesma informação entre um formato standard minimalista (listagem de código 2.4) e um formato próprio (listagem de código 2.2) com a mesma informação é mínima, enquanto que para os formatos mais completos aumenta significativamente. Esta diferença pode ser aumentada através da utilização de formatos binários em vez de caracteres *ASCII*. Com a utilização de um formato não standard, apesar da redução no armazenamento conseguida perde-se a interoperabilidade entre as rotas do *RecGPS* e os restantes sistemas.

O formato escolhido para o armazenamento de rotas e dados adicionais é o *GPX*[5]. Este formato além de ser um standard aberto é extensível, o que permite a inclusão dos dados adicionais recolhidos sem incompatibilizar o ficheiro para utilização noutras plataformas.

2.1.2 Cálculo da distância entre coordenadas

As coordenadas não são mais que os ângulos horizontal e vertical entre a posição actual e as linhas de referência (equador e semi-meridiano de *Greenwich*, com vértice no centro da terra). Tendo o planeta uma forma esférica o cálculo da distância entre duas coordenadas não deve ser efectuado pelo método Euclidiano, pois este apenas contempla superfícies planas. Devido à curvatura da superfície terrestre é necessário um método que tenha esta em conta, existindo para tal vários métodos.

Tendo:
θ_s, λ_s - latitude e longitude do ponto inicial
θ_f, λ_f - latitude e longitude do ponto de destino
r - raio da terra

Arco co-seno: Fórmula que permite o cálculo da distância entre duas coordenadas, mas que para distâncias reduzidas pode apresentar um erro considerável devido aos arredondamentos.

$$d = r \arccos(\sin\theta_s \sin\theta_f + \cos\theta_s \cos\theta_f \cos\Delta\lambda) \tag{2.1}$$

***Haversine*[6]**: Esta fórmula apenas apresenta erros de arredondamento para pontos especiais tais como coordenadas diametralmente opostas, sendo que para distâncias pequenas apresenta precisão suficiente.

$$d = 2r \arcsin\left(\sqrt{\sin^2(\frac{\Delta\theta}{2}) + \cos\theta_s \cos\theta_f \sin^2(\frac{\Delta\lambda}{2})}\right) \tag{2.2}$$

***Vicenty*[7]**: Esta fórmula é considerada precisa para todas as distâncias, apesar de ter um cálculo mais complexo.

$$d = r \arctan\left(\frac{\sqrt{(\cos\theta_f \sin\Delta\lambda)^2 + (\cos\theta_s \sin\theta_f - \sin\theta_s \cos\theta_f \cos\Delta\lambda)^2}}{\sin\theta_s \sin\theta_f + \cos\theta_s \cos\theta_f \cos\Delta\lambda}\right) \tag{2.3}$$

Visto o cálculo da distância ter que ser feito aquando da recolha e, tendencialmente, a distância entre coordenadas ser reduzida (ver secção 2.1.3) o método de cálculo de *Haversine* (fórmula 2.2) é o mais adequado, pois apresenta boa precisão para os valores em questão e um custo computacional menor face ao método de *Vicenty* (fórmula 2.3).

2.1.3 Recolha de coordenadas

Através da camada de abstracção de *GPS*[2][3] são disponibilizadas as coordenadas através de notificação por eventos. De todas as coordenadas obtidas são escolhidas as que integram a rota que se recolhe, sendo a decisão sobre o seu aproveitamento efectuada através de um conjunto de factores, tais como: o tipo de rota e suas características; a velocidade; a necessidade de detalhe que o utilizador deseja; Na tentativa de dar resposta a estas especificidades são disponibilizados vários modos de recolha de coordenadas:

- Precisão máxima
- Tempo mínimo
- Distância mínima
- Adaptativo

Precisão máxima: modo de recolha em que todas as coordenadas recolhidas são incluídas na rota (sensivelmente uma a cada segundo). Este modo gera os ficheiros de rota com o maior tamanho e em situações de velocidade baixa tem tendência à redundância de coordenadas pois estas vão ser muito próximas (coordenadas sequenciais repetidas descartadas).

Tempo mínimo: modo de recolha em que o utilizador indica qual o período de recolha desejado, sendo efectuada a recolha da próxima coordenada disponível findo cada período. Para períodos grandes ou velocidades elevadas o detalhe da rota pode ficar comprometido.

Distância mínima: modo de recolha em que o utilizador indica qual a distância mínima a que deseja que seja recolhida a próxima coordenada. Em rotas sinuosas, com distâncias elevadas perde-se o detalhe pois a distância entre coordenadas não considera o percurso efectuado mas a variação das coordenadas.

Adaptativo: modo de recolha em que são tidos em conta os principais factores que implicam maior ou menor frequência de recolha de coordenadas de maneira a efectuar uma recolha fiel à rota efectuada. Os principais factores considerados são a velocidade instantânea, a variação de velocidade, a mudança de direcção, a altitude e a distância espacial e temporal à ultima coordenada recolhida.

Durante a recolha de uma rota é possível a comutação de modo de recolha, assim como a pausa da mesma. É também possível a inserção de informações adicionais a coordenadas através da criação de pontos de interesse (`POI`).

2.1.4 Comparação com outros Sistemas

Existindo já um conjunto considerável de sistemas para recolha e partilha de rotas é então crucial efectuar a comparação entre estas e o sistema proposto. Com esta comparação entre sistemas pretende-se verificar que não se trata apenas da replicação de sistemas já existentes, mas sim de uma solução com valor acrescentado. Na comparação são destacadas as principais características e também diferenças de relevo para com o *RecGPS* caso as existam (ver Tabela 2.1).

TrackMe[8][9]: Sistema de partilha de localização. Este sistema publica através de uma aplicação móvel a posição dos seus membros numa rede social e permite a recolha da rota efectuada, assim como a inclusão de informações adicionais. A rota pode ser vista pelo utilizador directamente em sistemas públicos de mapas ou através do visualizador próprio (*Web Viewer*) em tempo real ou em diferido (modo *offline*). Entre utilizadores apenas é permitida a consulta de localização.

Comparativamente com o *RecGPS* a rede social não disponibiliza a partilha de rotas nem dá suporte à discussão sobre as mesmas.

MapMyWalk[10]: Sistema que disponibiliza a criação de rotas através de aplicação *web* /rede social, com suporte a partilha e discussão. É também possível a importação de ficheiros com coordenadas (em formatos standard) para a criação de rotas. A recolha de dados associados a execuções de rotas é efectuada através através de uma aplicação para *iPhone*. O *MapMyWalk* é apenas vocacionado para rotas associadas a exercício físico (bicicleta, caminhada, corrida, canoagem e outras).

Comparativamente com o *RecGPS* está limitado a apenas um dispositivo móvel, assim como a um reduzido leque de categorias de rotas.

Tracky[11]: Sistema que permite a recolha e execução de rotas. Disponibiliza vários modos de visualização (mapas, bússola) e permite a comparação com a execução inicial da rota. Utiliza ficheiros *GPX* para a importação e exportação de rotas. A sua utilização está sujeita ao pagamento de licença. Não disponibiliza plataforma para partilha e discussão de rotas e execuções (rede social).

Marathon[12]: Sistema composto por aplicação móvel e aplicação *web*/rede social. Ao nível da aplicação móvel é possível a recolha e execução de rotas, com informações adicionais (velocidade, elevação e outras). Na aplicação *web* é possível a partilha de rotas e execuções, não existindo lugar a comentários/discussão dos mesmos. Através da aplicação *web* é também possível o descarregamento das rotas em formatos standard (*GPX, KML e CSV*). O envio de rotas para a applicação *web* apenas é possível através da aplicação móvel proprietária e que está sujeita ao pagamento de licença.

Global Navigator[13]: Sistema para dispositivos móveis com funcionalidade semelhante a um produto *GPS* dedicado, mas com utilização das *API* de mapas de da *Google*. Permite a troca de localização entre utilizadores através de *SMS*. Dá suporte à georreferenciação de imagens e vídeos e criação de rotas, não permitindo no entanto a recolha. A utilização do *Global Navigator* está sujeita ao pagamento de licença.

GPS Tuner[14]: Sistema para dispositivos móveis com mapas 2D e 3D incorporados,

assim como a adição de mapas 2D por parte do utilizador. Permite também a recolha e extracção de rotas em formato *GPX*, tal como a comparação de execuções. Funcionalidade de situações de emergência, em que, quando activada, é enviada a localização para um contacto previamente definido. Os mapas estão organizados em pacotes cuja utilização está sujeita a pagamento de licença.

GPSises[15]: Sistema para partilha de rotas através de aplicação *web*/rede social. Disponibiliza ferramentas para criação de rotas e para importação através de ficheiros em formatos standard (leque abrangente de formatos suportados). Permite a visualização das rotas através de várias *API* de mapas, assim como o descarregamento das suas coordenadas em vários formatos. O sistema permite a adição de comentários às rotas, apesar de não ser dado destaque a esta funcionalidade. A comparação de execuções também não é suportada, assim como a criação de grupos de utilizadores.

SportTracks[16]: Sistema composto por uma aplicação *desktop* para registo de exercício físico através de coordenadas recolhidas por *GPS*. Permite a criação manual ou por ficheiro (em formato de coordenadas standard) de rotas e a comparação de dados da execuções. É disponibilizada a funcionalidade de adição de dados adicionais, tais como batimento cardíaco, recolhido por dispositivo próprio. O *SportTracks* faz também a gestão do equipamento e da preparação física do utilizador. A solução apresentada não demonstra ter objectivos relativamente à ferramentas para recolha e execução de rotas ou para a partilha e discussão em rede social.

EveryTrail[17]: Sistema de partilha e discussão de rotas através de aplicação *web*/rede social. Permite a criação de rotas manualmente, através da *API* de mapas *Google* ou através da importação de ficheiros (em formato standard) com as coordenadas da mesma. Apresenta um conjunto de categorias abrangente, sendo que cada rota apenas pode ter associada uma categoria. Permite também a adição de informação adicional (textual) da rota. O sistema não dá suporte à execução de rotas. O *EveryTrail* contêm ainda uma funcionalidade que permite a obtenção de um caminho até à localização de cada rota (através da *API* de mapas *Google*).

	TrackMe	*MapMyWalk*	*Tracky*	*Marathon*	*Global Navigator*	*GPS Tuner*	*GPSies*	*SportTracks*	*Every Trail*	*RecGPS*
Aplicação Móvel	x	x	x	x	x	x				x
Recolha de rota	x			x						x
Informação adicional rota	x			x				x	x	x
Recolha dados execução		x		x						x
Comparação execuções			x							x
Formatos standard	x	x	x	x		x	x	x	x	x
Rede social/ Aplicação *web*	x	x		x			x		x	x
Partilha localização	x				x					
Partilha rotas				x			x			x
Discussão rotas							x			x
Partilha execuções								x		x
Discussão execuções										x
Criação (*online*) de rotas		x					x	x	x	
Importação rotas		x						x	x	x
Georreferenciação imagens e vídeo					x					
Comunicação por *SMS*					x					
Mapas 3D						x				
Modo *SOS*						x				
Aplicação *Desktop*								x		
Interdisciplinar							x		x	x
Gestão equipamento								x		
Obtenção caminho para rota									x	
Utilização gratuita	x	x					x	x	x	x

Tabela 2.1: Principais características dos sistemas comparados

2.2 Partilha em rede social

2.2.1 Estruturação de dados

Com a utilização da tecnologia *ASP.NET MVC* para a implementação da aplicação de partilha seguem-se as boas práticas para esta recomendadas[18][19], sendo por isso utilizado o padrão de repositório. A implementação deste padrão consiste na definição e implementação de um conjunto de *interfaces* para as entidades e outro para as operações *CRUD* a realizar sobre a persistência de dados.

São então definidas *interfaces* para as operações que representam as principais categorias de entidades do sistema, ou seja, ao nível de utilizadores, rotas, execuções e discussão. Ao nível das entidades são definidas desde a granularidade mais baixa através da composição.

2.2.2 Persistência de rotas e execuções

Para a persistência dos ficheiros *GPX* associados às rotas e às suas execuções recorre-se ao SGBD utilizado como repositório dos restantes dados. Com recurso ao tipo de dados *XML* consegue-se o armazenamento dos ficheiros na própria base de dados, evitando assim a gestão de ficheiros em memória secundária.

2.2.3 Integração das *API* de mapas

Para a integração de mapas no sistema *RecGPS* existe um variado leque de *API* disponíveis. De entre as opções equacionadas o modo de utilização é semelhante, assim como as funcionalidades oferecidas, sendo por isso escolhida a *API* de mapas do *Google* devido à sua disseminação, cobertura dos mapas, disponibilidade e facilidade de utilização.

As *API* consideradas na escolha foram:

- *Google Maps* [20]
- *Bing Maps* [21]
- *Yahoo! LOCAL Maps* [22]
- *OpenStreetMap* [23]
- *SAPO Mapas* [24]

2.2.4 Discussão de rotas e execuções

Para as rotas e execuções é disponibilizada funcionalidade de discussão. A discussão tem como intervenientes os membros (utilizadores registados, autenticados e autorizados) e pode ser feita ao nível da rota ou de uma execução. A visualização dos comentários é condicionada pelo seu grau de acesso, sendo que no seu grau mais baixo um visitante (utilizador sem registo) consegue visualizar e no grau mais elevado é necessário ter sessão aberta (*login*) e pertencer ao grupo para poder visualizar. Ao nível da escrita, apenas membros com sessão aberta têm autorização.

2.2.5 Organização dos grupos

A pesquisa e inscrição em grupos é da responsabilidade de cada utilizador. Para cada grupo a que o utilizador pertence é permitida a criação de rotas e execuções, sendo sobre estas permitidas as operações de discussão (tópicos e comentários). A criação de grupos apenas é disponibilizada a utilizadores com o *role* de administrador. Os membros do grupo podem abandonar por iniciativa própria ou através de expulsão por parte do criador do grupo[2].

2.2.6 Controlo de acessos na integração em rede social

A implementação de uma rede social de raiz tem sempre dificuldade em obter visibilidade, assim sendo, para o *RecGPS* optou-se pela abordagem de integrar com uma rede social já existente. Com esta integração consegue-se a inserção num meio com aceitação e, consequentemente, um número elevado de membros que são potenciais interessados no sistema que se disponibiliza.

Para a integração em rede social foi escolhido o *Facebook* visto ser uma das maiores redes sociais [3]. A utilização da suas *APIs* permite a partilha de identidade dos membros com o *RecGPS* evitando assim ao utilizador a memorização de mais um conjunto de credenciais.

Apesar da integração em rede social não é obrigatório que os utilizadores do sistema sejam membros da rede social. Assim sendo é permitida a criação de registo no *RecGPS* sendo posteriormente permitida a fusão com um registo da rede social[4]. Outra funcionalidade

[2]Capítulo 6.2 - Desenvolvimentos futuros.
[3]Mais de 350 000 000 de utilizadores [25]
[4]Capítulo 6.2 - Desenvolvimentos futuros.

também disponibilizada é a de separação de contas[5], pois caso um membro do *RecGPS* elimine o registo na rede social não deve ser impossibilitado de aceder aos seus dados no sistema de recolha e execução de rotas.

2.3 Outros desafios

2.3.1 Extensão ao formato *GPX*

Para os dados específicos da implementação do *RecGPS* recorre-se à extensão ao formato *GPX*. A definição deste formato já contempla extensões, sendo para tal necessária a definição de um *schema* que permita a validação de ficheiros recebidos.

Com a possibilidade de incluir dentro do próprio ficheiro *GPX* evita-se a utilização de ficheiros separados para a inclusão dos dados específicos de rotas e execuções (pontos de interesse, acumulado e outras).

[5]Capítulo 6.2 - Desenvolvimentos futuros.

Capítulo 3

Concepção

3.1 Desenho da Arquitectura

No desenho da arquitectura (figura 3.1) são consideradas especificidades tais como a necessidade dos utilizadores da aplicação móvel poderem ser taxados pelo tráfego efectuado. Por isso, neste caso é preferida a utilização de um *web service* face à aplicação *web* pois consegue-se dispensar a transferência de dados de *user interface*. A escolha desta abordagem obriga por isso ao descarregamento prévio de uma aplicação cliente para o *PDA*.

Na solução desenhada é também considerada a utilização da *API* disponibilizada pela rede social *Facebook* para a integração de aplicações [26]. Com a utilização da *API* consegue-se responder a requisitos como a verificação da identidade dos utilizadores e a obtenção dos seus dados.

Ao nível do acesso a dados é utilizado o padrão repositório, sendo este concretizado através de *Data Access Application Blocks* da *Enterprise Library 4.1*.

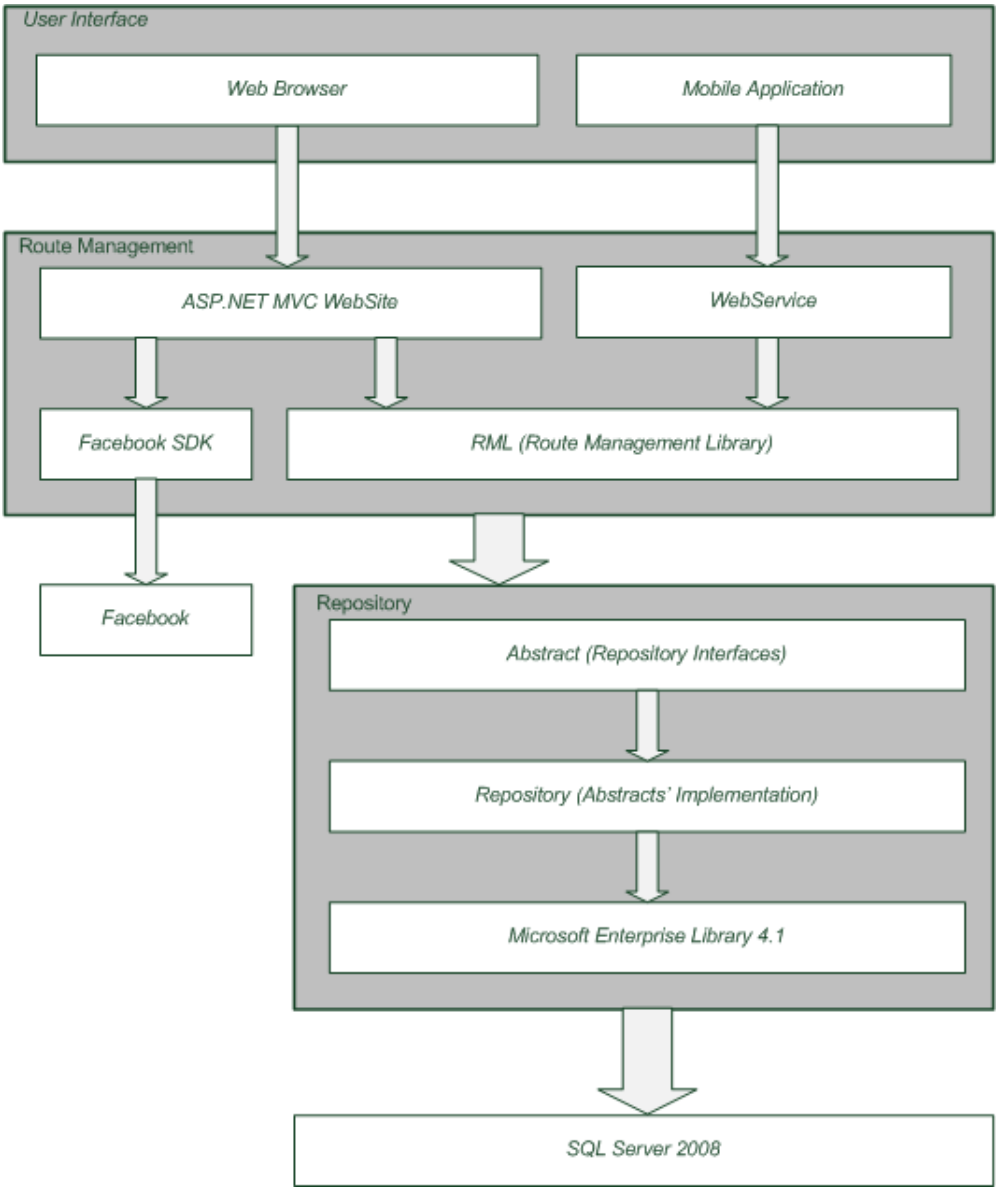


Figura 3.1: Arquitectura do *RecGPS*

3.2 Especificação

Com base na arquitectura (figura 3.1) estão definidos conjuntos de *interfaces* para as entidades de negócio e para o repositório. As entidades têm como objectivo simplificar a implementação e manutenção da lógica de negócio pois definem os blocos de informação que circulam entre as camadas do sistema, ou seja, independentemente dos campos que formam cada entidade, do ponto de vista da lógica de negócio a semântica desse bloco mantêm-se. O repositório tem como objectivo a abstracção por parte das camadas superiores do meio de persistência utilizado, ou seja, disponibiliza as operações lógicas necessárias à manipulação (criação, obtenção, alteração e remoção) das entidades de negócio sem que essas estejam dependentes da forma de persistência utilizada.

3.2.1 Entidades de negócio

Sendo as entidades de negócio as unidades de informação do sistema, a utilização de composição na sua definição permite, conforme necessário, a consulta dos dados com uma granularidade mais ou menos elevada. Por exemplo, a um nível mais elevado, a pesquisa de rotas pelas suas posições inicial e final, ou com menor granularidade, a consulta posição à posição de uma rota.

Na declaração das entidades foram tomadas opções de ocultação do *setter* de algumas das propriedades consideradas críticas. Esta opção prende-se com a necessidade de garantir que campos que sejam identificadores - por exemplo, `RouteId` - não possam ser alterados acima da camada de dados, por serem específicos ao domínio dos dados, sendo que é neste que a sua geração deve ocorrer. Também em propriedades cuja afectação tenha especificidades, como no caso de entidades enumeráveis - `IEnumerable<T>` -, o *setter* não é disponibilizado, sendo por isso necessário recorrer ao repositório para conseguir a sua afectação. Também propriedades que sejam calculadas a partir de outras - `Duration`, diferença temporal entre `StartPosition` e `EndPosition` - têm o *setter* omisso.

```
public interface IRoute
{
    int RouteId                                    { get; }
    string Name                                    { get; set; }
    int UserId                                     { get; set; }
    IRouteRecMode RouteRecMode                     { get; set; }
    DateTime Created                               { get; set; }
    IEnumerable<IRouteType> Types                  { get; }
    IPosition StartPosition                        { get; set; }
    IPosition EndPosition                          { get; set; }
    decimal Length                                 { get; set; }
    decimal TotalAscent                            { get; set; }
    decimal TotalDescent                           { get; set; }
    double Duration                                { get; }
    IEnumerable<IPosition> Positions               { get; }
    IEnumerable<IPointOfInterest> PointsOfInterest { get; }
}
```

Listagem de Código 3.1: *IRoute Interface*

Como resultado da composição de entidades para manipulação, as dependências existentes são:

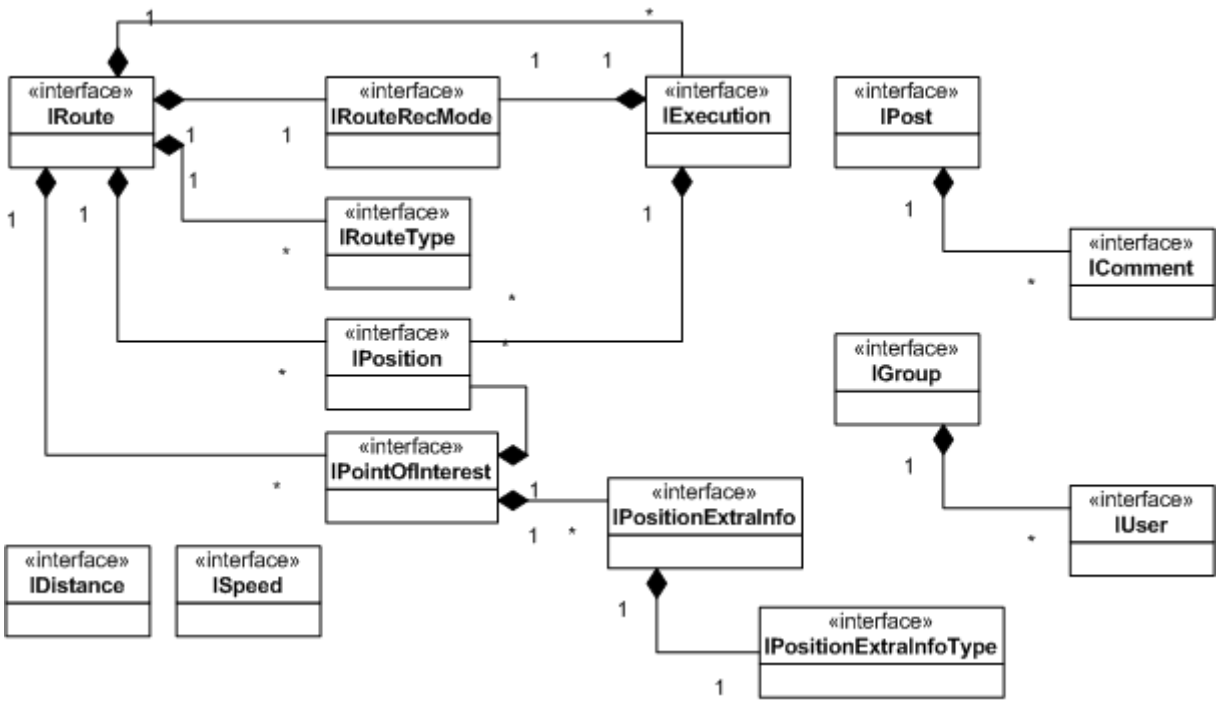


Figura 3.2: Interfaces das entidades definidas e relações

3.2.2 Repositório

Tipicamente a utilização desta técnica disponibiliza uma *interface* por cada entidade de negócio existente, mas dada a composição de entidades e o consequente cruzamento de componentes optou-se pela criação de *interfaces* por categoria, ou seja, os dados do sistema são divididos em quatro grupos lógicos: utilizadores, rotas, execuções e discussão.

Para cada um destes existe uma *interface* que declara o conjunto de operações de manipulação disponíveis (Figura 3.3).

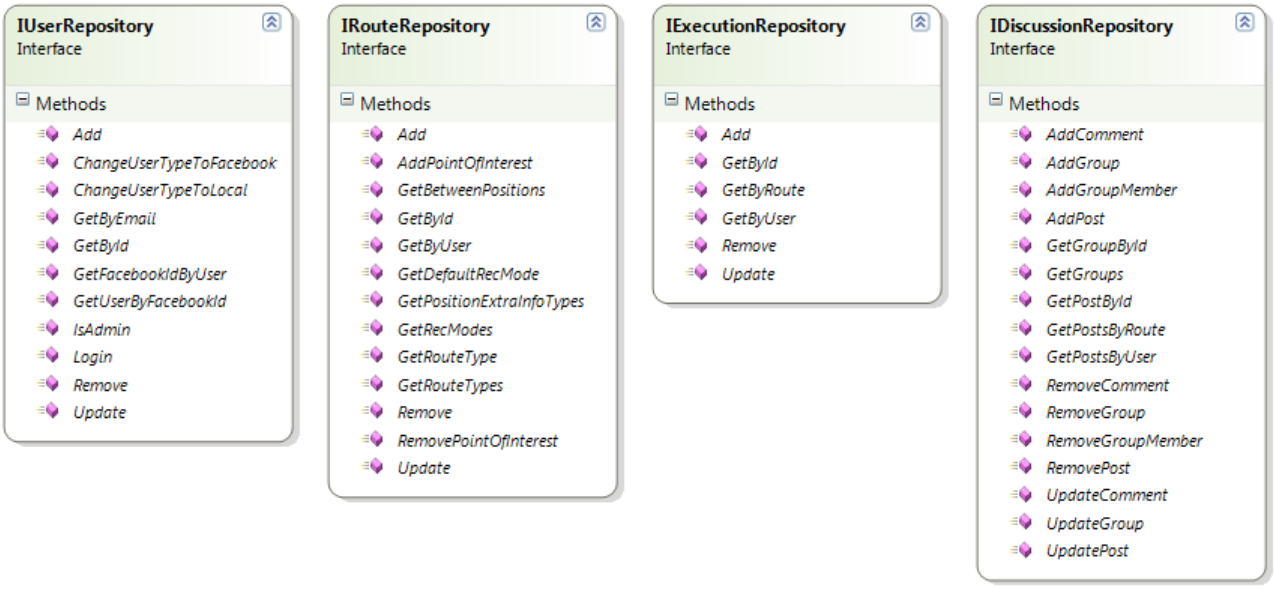


Figura 3.3: Interfaces do repositório

Apesar de não existirem entidades específicas para a origem de autenticação de um utilizador, ao nível do repositório tem que se efectuar distinção entre utilizadores locais e *Facebook* pois a sua autenticação efectua-se de modos distintos. Enquanto que um utilizador local tem que efectuar *login* com as suas credenciais - *email* e *password* - um utilizador com conta de *Facebook* não efectua autenticação perante o *RecGPS* mas, tendo a garantia por parte do *Facebook* de que este se encontra autenticado, verifica-se se o seu identificador de utilizador *Facebook* se encontra associado a um registo no *RecGPS*.

Declara-se ainda uma *interface* de fábrica de modo a que na utilização se consigam obter com facilidade instâncias das entidades de negócio necessárias. Além da obtenção de instâncias é também desejada a simplificação de preenchimento dos contentores de coordenadas,

sendo para tal declarados métodos de preenchimento destes (Figura 3.4).

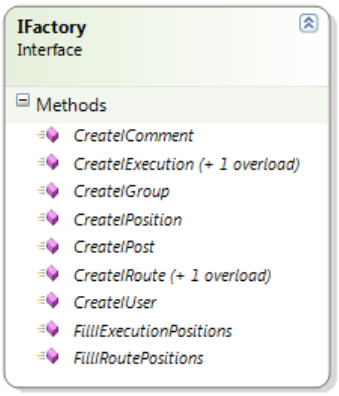


Figura 3.4: Interface da fábrica

3.2.3 Base de Dados

A base de dados consiste na representação das entidades definidas (Figura 3.6). Nas entidades compostas, alguns dos componentes têm representação diferente consoante a sua função, por exemplo, na representação de uma `Route` (Figura 3.5) os campos de posição de início e fim (`StartPosition` e `EndPosition`) são expostos directamente sob a forma de colunas dos tipos que correspondem às suas propriedades - `Latitude`, `Longitude`, `Altitude` e `Date`, enquanto que o conjunto de posições que definem a `Route` (`Positions`) são armazenados num único campo do tipo *XML*. Esta diferenciação deve-se à necessidade de efectuar pesquisa de rotas pela posição inicial e final, sendo que ao armazenar estas em vários campos se permite que com apenas uma interrogação se obtenham os registos que satisfazem a pesquisa. Caso se optasse pelo armazenamento destas posições em *XML* a pesquisa apenas seria possível após a sua interpretação.

Column Name	Data Type	Allow Nulls
RouteId	int	☐
Name	nvarchar(100)	☐
UserId	int	☐
RouteRecModeId	int	☐
Created	datetime	☐
StartLatitude	decimal(10, 8)	☐
StartLongitude	decimal(11, 8)	☐
StartAltitude	decimal(6, 2)	☑
StartDate	datetime	☑
EndLatitude	decimal(10, 8)	☐
EndLongitude	decimal(11, 8)	☐
EndAltitude	decimal(6, 2)	☑
EndDate	datetime	☑
Length	decimal(12, 2)	☐
TotalAscent	decimal(7, 2)	☑
TotalDescent	decimal(7, 2)	☑
Positions	xml	☐

Figura 3.5: Tabela `Route`

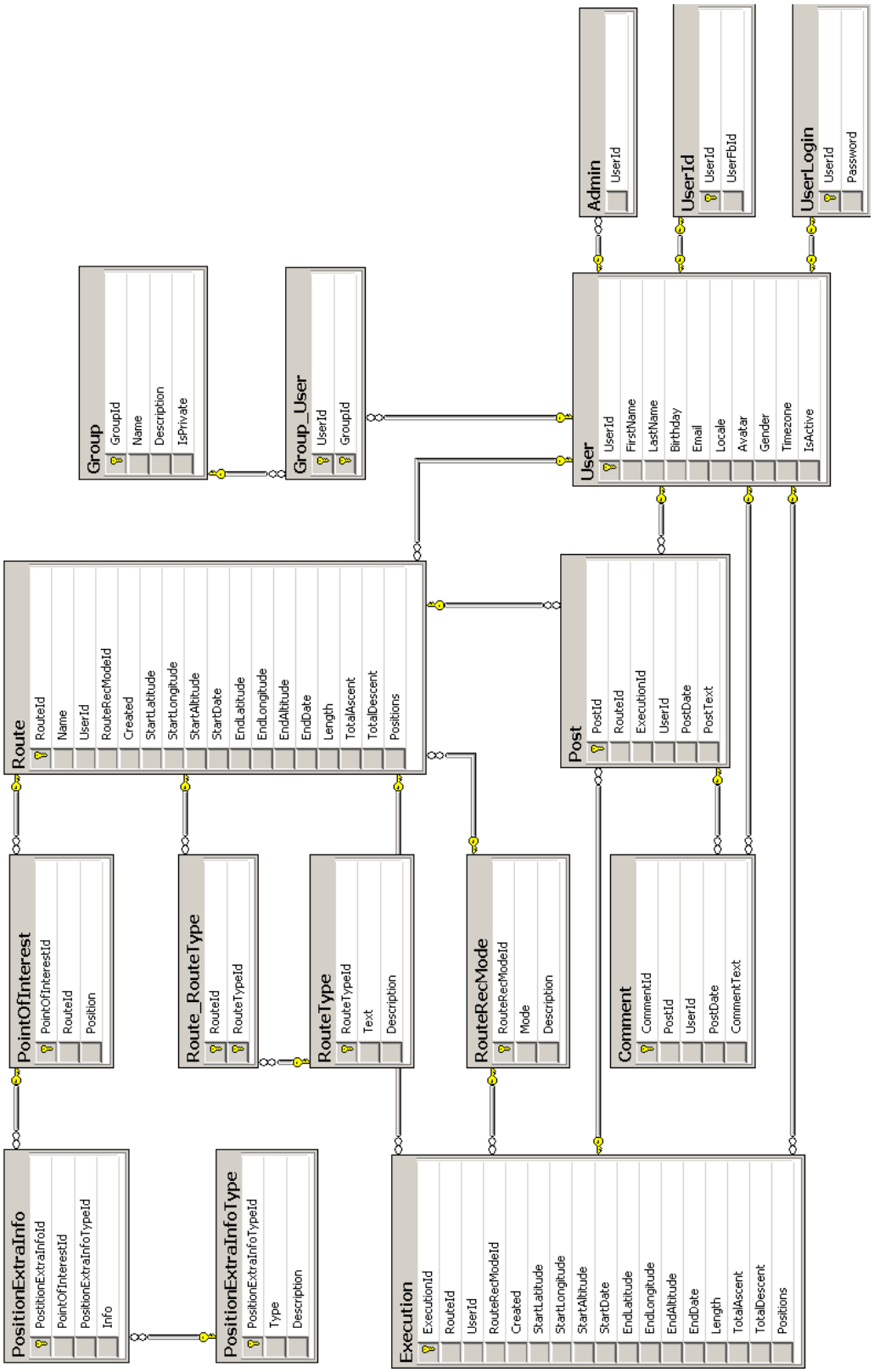


Figura 3.6: Diagrama da Base de Dados

3.3 Plataformas *web* utilizadas - integração

Para se conseguir a concretização de alguns dos conceitos propostos no enquadramento - Capítulo 2 - é necessário recorrer a ferramentas externas. A integração destas ferramentas - mapas e rede social - requer o conhecimento dos seu principais conceitos.

3.3.1 Mapas - *Google Maps*

Estando a utilização de mapas prevista para a aplicação *web*, de entre as vertentes disponíveis [27], a que se enquadra com o meio - aplicação *web* - e características de utilização - refrescamento assíncrono[1] - é a *Google Maps JavaScript API*[28].

A *API* é formada por um conjunto de tipos *Javascript* que têm como centro o `GMap2` que é o responsável pela representação do mapa. A sua implementação permite a configuração de opções de visualização, resposta a interacções do utilizador, adição e remoção de conteúdo através de objectos de classes igualmente incluídas na *API*.

Ao ser implementada, para o lado do cliente, através de um conjunto de tipos que se encarregam na totalidade da comunicação com o serviço de mapas (servidor), a *Google Maps JavaScript API* permite ao programador abstrair-se dos factores de transferência de dados tais como a geração de pedidos e tratamento de respostas.

3.3.2 Rede social - *Facebook*

Estando proposta para a aplicação *web* a interacção com a rede social *Facebook* é então necessário recorrer às ferramentas por esta disponibilizadas. De modo a permitir a integração com esta rede social em aplicações *web* são disponibilizados um *SDK* - *Javascript SDK*[29] - para acesso aos dados dos utilizadores e uma extensão ao formato *HTML*[2][30] - o *XFBML*[31] - para a representação de elementos gráficos.

Por limitações de segurança os *browsers* não permitem que, para domínios distintos - *RecGPS* e *Facebook* - sejam efectuados pedidos assíncronos através de *Javascript* - *cross domain communication* - com origem na mesma página. Para permitir a sua utilização em contextos assíncronos o *SDK* define um mecanismo que permite ultrapassar tal

[1] Evitar o refrescamento total da página durante a utilização do mapa
[2] *XHTML*

limitação[32]. O seu funcionamento consiste na criação, na aplicação *web*, de um ficheiro (`xd_receiver.htm`) através do qual o *SDK* faz os pedidos ao domínio *Facebook*, viabilizando assim a utilização de pedidos assíncronos através de *Javascript*, para um segundo domínio, sem que a limitação de segurança imposta pelos *browsers* se faça sentir.

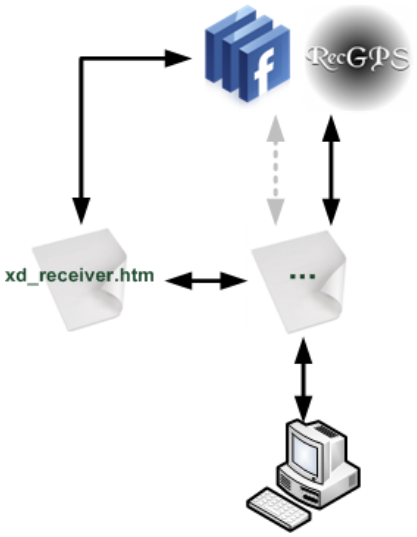


Figura 3.7: *Cross domain communication*

Através das ferramentas disponibilizadas pelo *Facebook* é possível ao *RecGPS* aceder aos dados de registo dos utilizadores da rede social, desde que estes o permitam, assim como a publicação na conta de *Facebook* de conteúdos provenientes do *RecGPS*.

Todas as interacções se iniciam com um estímulo por parte do utilizador no *RecGPS*, sendo que do lado da rede social apenas se efectua acesso ao *RecGPS* através de conteúdos por este publicados.

Capítulo 4

Implementação

A implementação do sistema segue uma abordagem *bottom-up*, sendo por isso a sua primeira secção respeitante às considerações na implementação da base de dados (Capítulo 4.1). O segundo nível trata da implementação do *Repository* de acesso a dados (Capítulo 4.2) e o seguinte,*Route Management*, respeita à gestão de rotas (Capítulo 4.3). Por último são abordadas as *User Interfaces* do sistema (Capítulo 4.4).

4.1 Base de Dados

Para as interacções com a base de dados recorrem-se a procedimentos armazenados pois, em caso de alterações à base de dados, estes tendem a permitir que através do seu reajuste o acesso a dados se processe de igual forma (Apêndice A.7 - Listagem de procedimentos armazenados).

Em relações *1-N* em que as tabelas *N* representam campos de entidades compostas[1] é utilizada a regra *Delete Cascade* de modo a que a remoção de uma instância de uma entidade seja eficiente (apenas um comando *SQL*).

Na operação de remoção de um `User`, por este ser chave estrangeira de várias tabelas , a remoção consiste na alteração de um campo do tipo `bit` que indica se este está activo/visível. Em alternativa poderia ser efectuada a alteração de propriedade de uma `Route` para um `User` especial ou ser permitida a existência de campos `null` na chave estrangeira com

[1] *Tags* de uma `Route`, `Comment` de um `Post` e `User` de um `Group`.

a utilização da regra *Delete Set Null* em `User`. Com esta abordagem perder-se-ia a possibilidade de recuperar a conta desse utilizador, mantendo todo o histórico da sua passagem pelo *RecGPS*.

Ao nível do repositório de discussão as actualizações de `Comment` e `Post` apenas permite a alteração do texto. Dada a afinidade de um `Post` e respectivos `Comment` a uma rota ou execução não é aceitável que se possa alterar essa afinidade.

4.2 Repository

4.2.1 *Data Access Application Block*

A ligação do código *.NET* à base de dados é feita com *Data Access Application Block*. Este *application block* disponibiliza uma série de objectos que permitem a utilização do *ADO.NET* seguindo um conjunto de boas práticas. Algumas destas boas práticas são de especial interesse para o nosso caso:

- *Pool* de conexões: permite a reutilização controlada de um recurso essencial como as conexões;
- Fecho de conexões automático: abre e fecha uma conexão por cada comando, excepto quando os resultados são retornados sobre a forma de um `DataReader`. Neste caso, a conexão é fechada quando o objecto `DataReader` é fechado. Também se comporta de forma diferenciada quando se utilizam transacções (ver ponto a seguir);
- Reutilização da conexão numa transacção: de forma a evitar a promoção para uma transacção distribuída, quando se utiliza mais que uma conexão numa transacção, a execução de comandos através do *application block* detecta se o comando está a ser executado dentro de um `TransactionScope` e reutiliza a conexão até ao fim dessa mesma transacção.

É também necessário ter em atenção que a invocação de procedimentos que não retornem tuplos[2] necessita da passagem de um parâmetro adicional de retorno, para que se consiga obter o número de linhas afectado[3].

[2] `ExecuteNonQuery`.

[3] Na versão 4.1 do *Enterprise Library - DAAB* o valor de retorno da chamada ao procedimento não é correcto.

4.2.2 Carregamento Diferido

Para as propriedades das entidades que na base de dados estão representadas através de relações *1-N* a obtenção de valores é efectuada de modo diferido. Esta opção prende-se com a possibilidade de essas propriedades nunca serem acedidas, motivando assim a que o seu carregamento se dê o mais tarde possível. Além destas propriedades também as `Position` das entidades `Route` e `Execution` têm o carregamento diferido, pois ao estarem armazenadas sob o formato *XML* para serem representadas em entidades é necessária a sua conversão. Por exemplo, em pesquisa de rotas por área geográfica, é expectável que nem todas as rotas seleccionadas sejam visualizadas em detalhe, não sendo por isso necessário o carregamento dos `POI` (*1-N*) e `Positions` (*XML*) de todas elas.

Para minimizar a dependência entre a implementação das entidades e o repositório são aplicadas as técnica de fábrica, de *Inversion Of Control* (*IOC*) e *Dependency Injection* (*DI*). A entidade que implementa o *IOC* indica no seu construtor um tipo `delegate` com retorno igual ao da propriedade cujo carregamento é diferido. No primeiro acesso dá-se a execução do `delegate`, ficando assim disponíveis os dados da propriedade, de forma transparente a quem ela acede.

```
public class Route : Abstract.IRoute
{
  private Func<IRoute, IEnumerable<IPosition>> loadPositions;
  private IEnumerable<IPosition> positions;
...
  public Route(Func<IRoute, IEnumerable<IPosition>> loadPositions, ...)
  {
    this.loadPositions = loadPositions;
    Positions = new List<IPosition>().AsQueryable();
    ...
  }

  public IEnumerable<Abstract.IPosition> Positions
  {
    get{  if (positions.Count().Equals(0) && !RouteId.Equals(-1))
            positions = loadPositions(this);
          return positions;  }
    set { positions = value; }
  }
...
}
```

Listagem de Código 4.1: `Positions` - Carregamento diferido

4.2.3 *Schema* para validação de ficheiros *GPX*

Com a utilização de extensões ao formato *GPX*[5] é necessária a definição de um *schema* pelo qual se guie a geração e consumo de ficheiros no *RecGPS*. O formato *GPX* na zona de `metadata` define o tipo `extensions`, cujo propósito é a inclusão de dados que não os contemplados pelo *standard*.

No contexto do *RecGPS* é necessário que um ficheiro *GPX* contenha dados suficientes para a instanciação de uma entidade que representa esse mesmo percurso (figura 4.1). De entre o conjunto apresentado existe um sub-conjunto que dispensa representação, sendo ele formado pelos identificadores, pois apenas têm significado no contexto do *RecGPS* e pela duração que é calculada pela diferença de tempo entre as posições iniciais e finais.

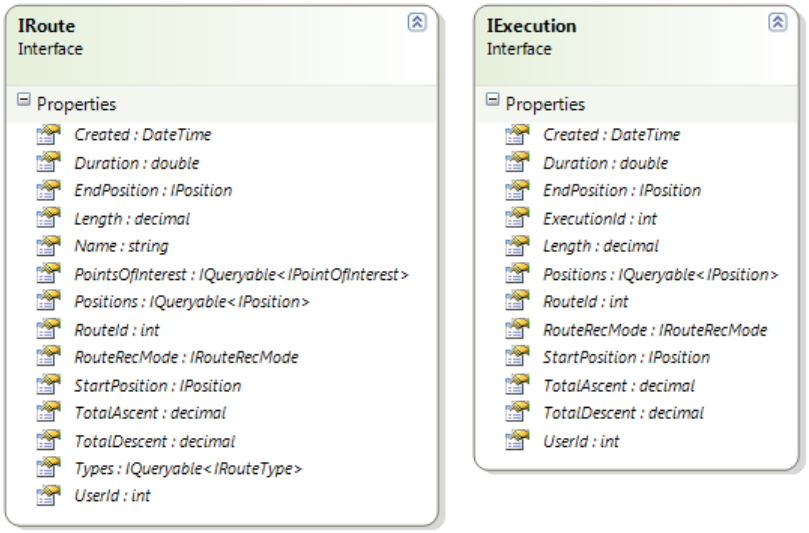


Figura 4.1: Propriedades de `IRoute` e `IExecution`

Com o *GPX* é possível satisfazer os campos de `Name`, `Created` e `Positions`, tendo para os restantes que ser utilizada a extensão ao formato. Para os campos cujo tipo tem representação nos tipos *built-in* do *schema* são declarados os elementos correspondentes (`RouteRecMode`, `RouteType`, `Length`, `TotalAscent` e `TotalDescent`). Os restantes são definidos através da composição de tipos primitivos e pela definição de tipos complexos.

`StartPosition` e `EndPosition` - Por serem representações de `Position` é utilizado a mesma formatação que o *GPX* utiliza para uma posição, mas incluída em tipos complexos com os respectivos nomes.

```
<recgps:startPosition>
    <trkpt lat="39.09815270" lon="-9.26060557">
        <ele>30.000000</ele>
        <time>2009-11-12T18:03:50Z</time>
    </trkpt>
</recgps:startPosition>
```

Listagem de Código 4.2: `StartPosition` - Exemplo

`PointsOfInterest` - A entidade que representa um `POI`, para uma mesma posição, permite a existência de um conjunto de informações adicionais. Para tal é definido um tipo complexo que além de uma posição contem um conjuntos de informações adicionais. A representação dos pontos de interesse de uma rota é efectuada através de uma sequência destas construções.

```
<recgps:pointOfInterest>
  <trkpt lat="39.09815270" lon="-9.26060557">
    <ele>30.000000</ele>
    <time>2009-11-12T18:03:50Z</time>
  </trkpt>
  <recgps:positionExtraInfo>
    <recgps:peiType>JustInfo</recgps:peiType>
    <recgps:peiInfo>Informação extra 1</recgps:peiInfo>
  </recgps:positionExtraInfo>
  <recgps:positionExtraInfo>
    <recgps:peiType>JustInfo</recgps:peiType>
    <recgps:peiInfo>Informação extra 2</recgps:peiInfo>
  </recgps:positionExtraInfo>
</recgps:pointOfInterest>
<recgps:pointOfInterest>
...
```

Listagem de Código 4.3: `PointOfInterest` - Exemplo

O *schema* de extensão ao *GPX* encontra-se no Apêndice A.1.

4.2.4 Leitura e escrita de *GPX*

A manipulação *GPX* está assente sobre *Linq to XML*. Na conversão para entidades é efectuada a validação do *GPX* recebido com o *schema* definido (capítulo 4.2.3), de modo a garantir que o mesmo está íntegro. De seguida são extraídos os dados através da manipulação da árvore *XML* que o *parser* fornece. Ao consumir as coordenadas os elementos do mesmo tipo são mantidos pela ordem do ficheiro, ou seja, é garantida a ordenação na

leitura dos pontos de interesse e das posições.

Para a formação do *GPX* a partir das entidades é gerada uma árvore de elementos *XML*, mantendo a ordem de inserção de pontos de interesse e posições.

4.3 Route Management

4.3.1 `Route Management Library`

Para permitir a separação entre a lógica de gestão de rotas e o seu armazenamento está definida a `Route Management Library` - `RML` - (Figura 3.1 - arquitectura). Esta biblioteca disponibiliza concretizações das *interfaces* definidas para o repositório (Capítulo 3.2.2 - Repositório) e entidades de negócio (Capítulo 3.2.1 - Entidades de negócio). São ainda disponibilizados métodos com o objectivo de simplificar a utilização do repositório quer pela combinação de funcionalidades (por exemplo, `ChangePassword`) quer pela substituição dos parâmetros (por exemplo, `CreateUser`). No caso de `ChangePassword` efectua-se a alteração da *password* de um utilizador, sendo que para tal é necessária a obtenção do seu `IUser`, validação da sua *password* e alteração desta pela nova dentro de contexto transaccional. Para a inserção de um utilizador é necessária uma instância de `IUser` preenchida com os dados do utilizador e submeter ao repositório, sendo esses passos efectuados através de `CreateUser`.

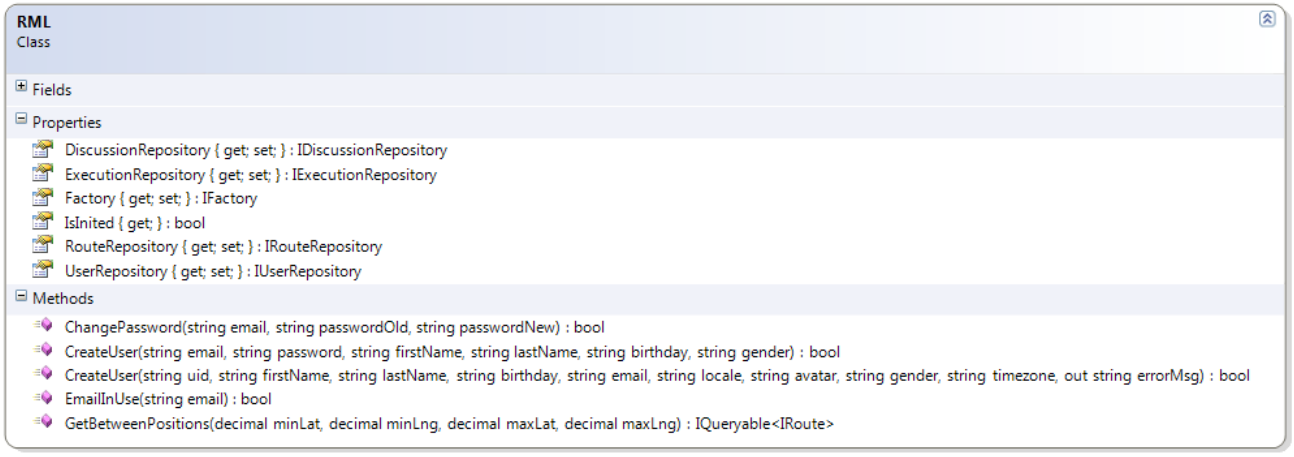


Figura 4.2: *Wrappers*

A decisão das implementações a utilizar na `RML` é independente desta pois a mesma aplica

o padrão de *Inversion Of Control*. Para ser possível a sua utilização é necessário afectar as propriedades referentes à implementação do repositório:

```
IFactory Factory { get{...} set{...} }
IUserRepository UserRepository { get{...} set{...} }
IRouteRepository RouteRepository { get{...} set{...} }
IExecutionRepository ExecutionRepository { get{...} set{...} }
IDiscussionRepository DiscussionRepository { get{...} set{...} }
```

Listagem de Código 4.4: Propriedades da `RML`

Sendo a utilização da `RML` efectuada através do *ASP.NET MVC Web Site* a sua parametrização efectua-se no arranque do mesmo.

4.3.2 Aplicação *web*

ASP.NET MVC - Estrutura da implementação

A utilização da tecnologia *ASP.NET MVC* encoraja a que as páginas (`view`) seja agrupadas por funcionalidade através de classes (`controller`) que lhes dão suporte. Os métodos dos `controller` efectuam o atendimento de pedidos *HTTP Get* ou *Post* do cliente. Nesses métodos é implementada a lógica necessária ao correcto preenchimento das `view` (*HTTP Get*), assim como o tratamento dos dados submetidos (*HTTP Post*). Para simplificar o consumo de dados dos formulários são ainda definidas classes de dados (`model`) que efectuam, por convenção - *Model Binding*[18] -, a ligação entre os campos do formulário (na `view`) e propriedades de `model` - parâmetro de um método do `controller`.

Nos `controller` além da implementação da lógica a utilizar nas `view` é efectuado o atendimento de pedidos assíncronos (*AJAX*). Em resposta a estes pedidos são seriados objectos para o formato *JSON*[33][34] de modo a que junto do utilizador (*Browser*) os mesmos possam ser reconvertidos em objectos *Javascript*[35] e a interface com o utilizador actualizada.

Google maps API[28]

Para visualização em mapas recorre-se à *API* fornecida pela *Google*. Para ser possível a sua utilização é necessário a criação de uma chave[36] que permite ao *Google* verificar qual o domínio - *RecGPS* - através do qual os pedidos estão a ser efectuados. A utilização da

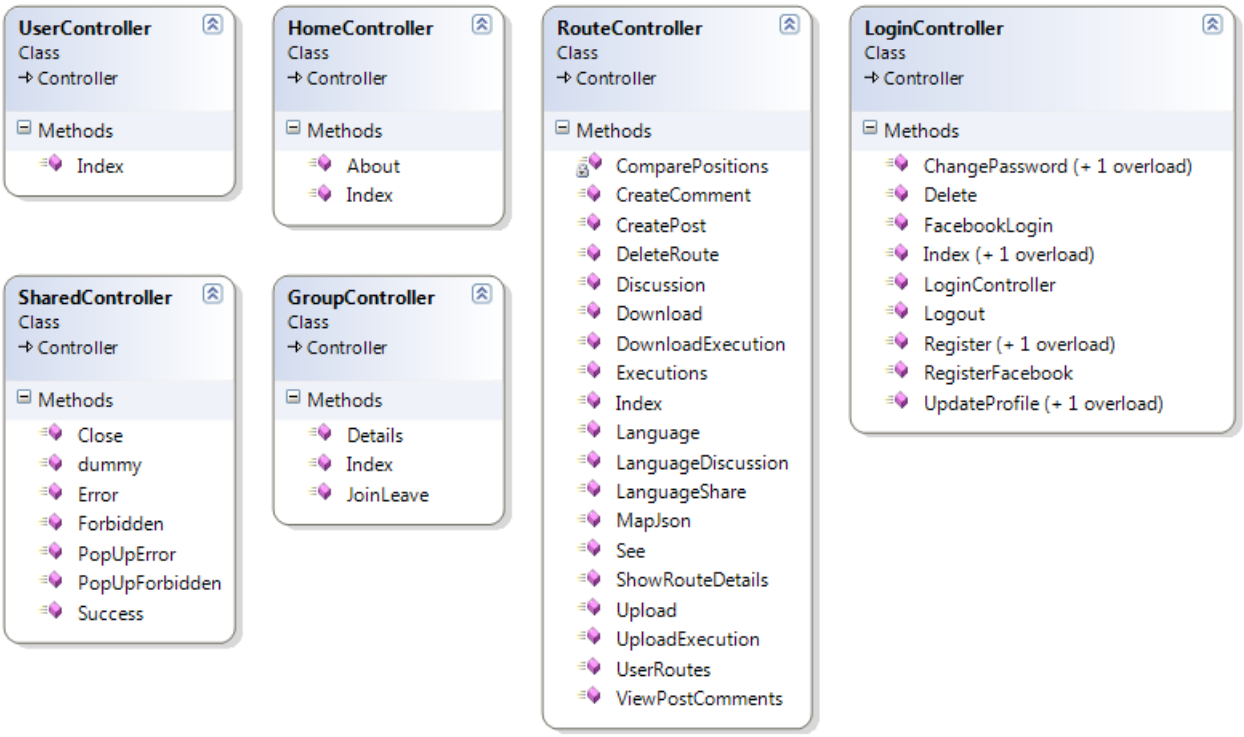


Figura 4.3: `Controllers`

API, por ser efectuada através de *JavaScript*, acontece no lado do cliente, iniciando-se pela inclusão do ficheiro de *script* que permite o carregamento das *APIs Google*[37].

No carregamento da *API* de mapas é necessário indicar qual a versão a utilizar (*Version 2*), devendo também ser prevenidas as fugas de memória[38] (referência circulares que o tratamento de eventos da *API* pode gerar). A função a invocar no final do carregamento é também indicada, de modo a que a configuração do mapa se possa dar.

Após o carregamento é então possível mostrar o mapa ao utilizador, assim como sobrepor conteúdo (`Overlay`). Os principais elementos adicionados ao mapa são os `Markers` e as `Polylines`. Os `Markers` indicam uma posição no mapa, enquanto as `Polylines` representam um percurso e/ou uma forma geométrica. Em ambos os elementos são disponibilizados balões - `InfoWindow` - para apresentação de informações.

Ao nível da interacção com o utilizador, a lógica é implementada através dos eventos disponibilizados pela *API* [39].

```
//create a V2 map
google.load("maps", "2");

//prevent memory leaks
document.body.setAttribute("onunload", "GUnload()");

//call initialize and put the whole thing moving
google.setOnLoadCallback(initialize);

//creates the map and makes the initial request
function initialize()
{
    if (GBrowserIsCompatible())
    {
        //create map
        map = new GMap2(document.getElementById("map"), { size: new GSize(700,500) } );
        //set map center
        map.setCenter(new google.maps.LatLng(39.019584, -9.151955), 10);
        //scroll zoom
        map.enableScrollWheelZoom();
        //add controls
        map.addControl(new google.maps.SmallMapControl());
        map.addControl(new google.maps.MapTypeControl());
    }
}
```

Listagem de Código 4.5: Carregamento *Google Maps JavaScript API*

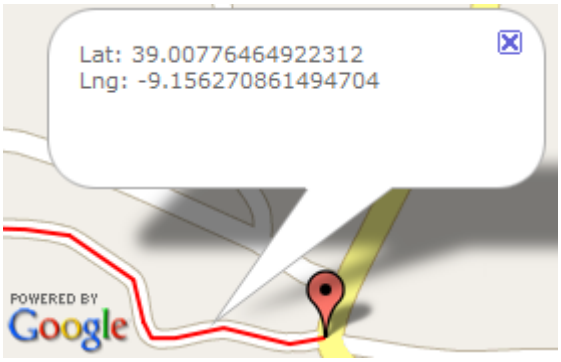


Figura 4.4: `Marker`, `Polyline` e `InfoWindow`

***Facebook Connect* [40]**

À semelhança das *APIs* de mapas, a integração em rede social de uma aplicação *web* baseia-se na utilização de funcionalidades disponibilizadas através de *Javascript*. No caso do *Facebook* além dessas funcionalidades - *Javascript SDK* [29] - é ainda disponibilizada uma extensão ao formato *HTML*, denominada de *XFBML* [31], cuja representação é gerada através de *Javascript*.

Cada aplicação que se integre com o *Facebook* necessita de registar o seu endereço base de modo a que lhe seja atribuída uma chave. Além do registo é necessária a inclusão na aplicaçao de um ficheiro de *Cross Domain Communication*[32] para que a utilização de funcionalidades do *Facebook* seja permitida pelo *browser* através da aplicação que o utiliza.

Para que o *XFBML* seja reconhecido pelo *browser*[41] as páginas que o utilizem têm que indicar serem do tipo *XHTML*, assim como incluir o *namespace* que define o *XFBML*.

```
<import xmlns="http://www.w3.org/1999/xhtml" xmlns:fb="http://www.facebook.com/2008/fbml" />

<fb:login-button onlogin="loginThroughFbConnect()" length="long">ClickMe</fb:login-button>

<script type="text/javascript" src="http://static.ak.connect.facebook.com/js/api_lib/v0.4/FeatureLoader.js.php"></script>
<script type="text/javascript">
    FB.init("18fe525f5daa7744d417c567d2aee340", "/xd_receiver.htm");
</script>
```

Listagem de Código 4.6: *Tag* de `login` em *XFBML*

Figura 4.5: Representação da *tag* de `login` em *XFBML*

A conjugação do *XFBML* com o *SDK* permite a obtenção dos dados de utilizador necessários ao registo e autenticação (*login*), assim como o acesso de escrita à conta do utilizador (criação de mensagens na página do utilizador), de modo a que este possa partilhar na rede social informações do *RecGPS*.

```
function loginThroughFbConnect()
{
    var api = FB.Facebok.apiClient;
    var ses = api.get_session();
    if (ses == null)
    {
        alert("no login");
        return;
    }
    var uid = ses.uid;
    window.location = "/Login/FacebookLogin/?fbId=" + uid;
}
```

Listagem de Código 4.7: Obtenção da sessão de utilizador *Facebook*

Rotas

Com a utilização de *API* de mapas[28] para a pesquisa e visualização de rotas é necessária a utilização de linguagem *JavaScript*. Com recurso à linguagem são também efectuados pedidos assíncronos (*AJAX*) à aplicação *web* de modo a que não hajam interrupções na navegação por carregamento total da página.

Para a pesquisa de rotas é utilizado um `model` específico - `RouteToJson` - que consiste na condensação de um `IRoute`. Essa redução é efectuada com o objectivo de minimizar o tráfego, pois na pesquisa apenas são utilizados campos como as coordenadas de início e fim, não sendo por isso necessário transferir propriedades de `IRoute` como `Positions` e `PointsOfInterest` que, dado a sua natureza, tendem a conter muito mais dados que as restantes propriedades de `IRoute`. Quando na pesquisa é seleccionada uma rota é efectuado o pedido à aplicação que serie para a resposta a `IRoute` correspondente, permitindo assim a criação de linhas e pontos de interesse no mapa, após a avaliação do *JSON*.

```
public class RouteToJson
{
  public int Id { get; set; }
  public string Name { get; set; }
  public decimal StartLat { get; set; }
  public decimal StartLng { get; set; }
  public decimal EndLat { get; set; }
  public decimal EndLng { get; set; }
  public decimal Length { get; set; }
  public decimal TotalAscent { get; set; }
  public decimal TotalDescent { get; set; }
  public string[] Types { get; set; }
}
```

Listagem de Código 4.8: Classe `RouteToJson`

Sendo uma rota submetida por um utilizador apenas este tem a capacidade de a remover. Para tal, através de um pedido assíncrono é verificada na aplicação *web* a idoneidade do pedido e enviado na resposta *JSON* com o resultado da operação.

Para a criação o utilizador faz o envio de um ficheiro com o qual se tenta a criação de uma rota. O conteúdo do ficheiro é validado com o *schema* definido para o *RecGPS* e em caso de sucesso a resposta para o utilizador contêm informações sobre a rota criada: nome; modo de recolha; data de recolha; subida acumulada; descida acumulada; duração da recolha;

número de coordenadas; número de pontos de interesse.

Partilha no *Facebook*

A partilha de execuções no *Facebook* é conseguida através de uma chamada ao *SDK* fornecido pela rede social. Como parâmetros são fornecidos a mensagem a publicar e a função que consome o resultado.

```
function publishOnFacebook()
{
    var post =
    {
        message:    strings.Header,
            attachment: {
                name: route.Name,
                description: strings.Description,
                href: (strings.DescriptionLink + route.RouteId)
        },
        action_links:
        [
            {
                href: strings.RecGPSLink,
                text: strings.RecGPS
            }
        ],
        user_message_prompt: strings.Prompt
    };
    FB.publish(post,
                function(published_post)
                {
                    if (published_post)
                        alert(strings.Success);
                    else
                        alert(strings.Fail);
                }
    );
}
```

Listagem de Código 4.9: Partilha no *Facebook*

Visualização e criação de execuções

Para as rotas existentes é permitida a submissão de execuções. Após a validação do ficheiro submetido pelo utilizador contra o *schema* definido para o *RecGPS* é verificada a compatibilidade entre a execução e a rota para a qual se faz a submissão.

As execuções de cada rota podem ser visualizadas através da aplicação *web*, vendo apenas a informação reduzida (data e duração da recolha) ou descarregadas pelo utilizador para consulta com maior detalhe[4].

Visualização e inserção de `Post` e `Comment`

Ao aceder aos detalhes de uma rota é também possível aceder aos *posts* da mesma. Na visualização são mostrados os *posts* existentes para a rota e, opcionalmente, os comentários associados. São também disponibilizadas a todos os utilizadores autenticados as funcionalidades de inserção de *posts* e comentários.

(a) Versão reduzida

(b) Versão aumentada - sem inserção de *comments*

Figura 4.6: Visualização e inserção de `Post` e `Comment`

Controlo de acessos

De modo a conjugar a gestão rotas com uma plataforma de acesso público, para a sua partilha, é necessária a adopção de mecanismos que previnam o acesso indevido à informação. Para a implementação desse controlo de acessos foram consideradas várias opções: Utilização do controlo de acessos da plataforma *ASP.NET*; Implementação de um mecanismo de controlo de acessos; Adopção de um mecanismo de controlo de acessos.

[4] Através da aplicação móvel, por exemplo.

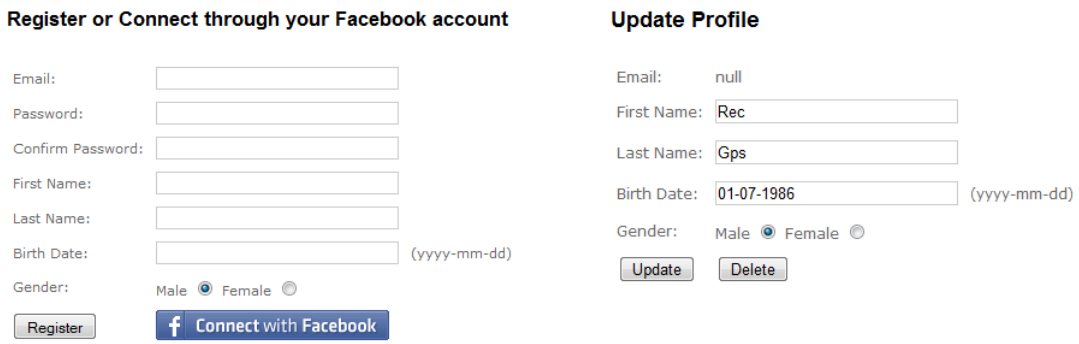
Sendo o controlo de acesso necessário ao nível da aplicação *web* e com regras que podem mudar optou-se por manter o controlo de acessos a este nível, evitando por isso a adaptação do modelo de dados do *RecGPS* aos requisitos da plataforma de controlo de acessos da plataforma *ASP.NET* ou à criação de um modelo de dados específico para este. Mantendo o controlo de acessos local a escolha para a persistência das regras de acesso recaiu sobre um ficheiro *XML* local à aplicação (`Authorization.config`). Para o consumo e aplicação das regras tanto a implementação de uma solução de raiz como a adopção de uma solução existente necessitam de ser independentes e genéricas o suficiente para que se moldem à arquitectura. A escolha recaiu sobre um sistema já existente[42].

A utilização deste mecanismo tem como princípios a utilização dos eventos de autenticação e autorização - `AuthenticateRequest` e `AuthorizeRequest` - do ciclo de vida da página. Com base no ficheiro *XML* são verificadas as condições de acesso do utilizador ao `controller`. O ficheiro descreve para cada `controller` os níveis de acesso (*roles*) a considerar e para cada `action` quais os *roles* com autorização.

```
<controllers>
    <controller name="User">
        <roles>
            <role>View</role>
            <role>User</role>
            <role>Admin</role>
        </roles>
        <actions>
            <action name="Index">
                <roles>
                    <role>User</role>
                    <role>Admin</role>
                </roles>
            </action>
        </actions>
    </controller>
</controllers>
```

Listagem de Código 4.10: Excerto de ficheiro de configuração*XML* do mecanismo de controlo de acessos

Na implementação da aplicação *web* apenas existe necessidade dos níveis de visitante (`View`), utilizador registado (`User`) e administrador (`Admin`). Caso os requisitos do sistema evoluam de modo a que o número de níveis de acesso se altere, além da sua inclusão no ficheiro *XML*, é necessária a alteração do *Login* da aplicação *web* pois é aí que se faz a



(a) Formulário de registo

(b) Formulário de alteração de registo

Figura 4.7: Formulários de registo e alteração de registo

atribuição do nível de acesso ao utilizador.

Tal como sugerido pelo autor do mecanismo de controlo de acessos[42] a estrutura tem flexibilidade suficiente para ser alterada a fonte de configuração de ficheiro *XML* para base de dados ou outra, permitindo assim acompanhar a evolução de requisitos do sistema, caso necessário.

Criação, actualização e remoção de contas de utilizador

A criação de conta de utilizador é permitida através do preenchimento de um formulário ou pela associação de uma conta *Facebook* ao *RecGPS*. Na criação por formulário é verificada a presença do *email* fornecido no sistema de modo a que se garanta que para cada utilizador existe um contacto directo e único. Ao utilizar uma conta *Facebook* para a criação de uma conta o endereço de email não é verificado, pois o mesmo pode não ser partilhado pelo titular da conta *Facebook*. Apesar da sua indisponibilidade consegue-se um canal de contacto com o utilizador, caso necessário, através do identificador de utilizador no *Facebook*.

Após o registo, independentemente do modo utilizado, é permitida a alteração dos dados pessoais (à excepção do endereço de *email*), assim como a remoção da conta.

Login

O *login* no *RecGPS* pode ser efectuado internamente, através das credenciais fornecidas aquando do registo - *email* e *password* - ou externamente, através da verificação da validade de sessão de um conta *Facebook* que esteja associada ao *RecGPS*.

-Interno:

Estando definidos três níveis de acesso - `View`, `User` e `Admin` - no início de sessão de cada utilizador é-lhe atribuído o nível mínimo. Caso o utilizador forneça credenciais válidas é-lhe atribuído um dos níveis superiores, conforme o resultado da consulta à `RML`. Com a validação das credenciais passam a circular entre o utilizador e a aplicação *web* o identificador para a manutenção de sessão - `FormsAuthenticationTicket` - juntamente com o *email* e nome do utilizador.

-Externo:

O *login* externo é efectuado através da obtenção de uma sessão *Facebook* válida para um utilizador registado no *RecGPS*. Caso não exista sessão válida são pedidas ao utilizador as credencias para que seja iniciada uma sessão no *Facebook* e com isso o utilizador consiga acesso ao *RecGPS*.

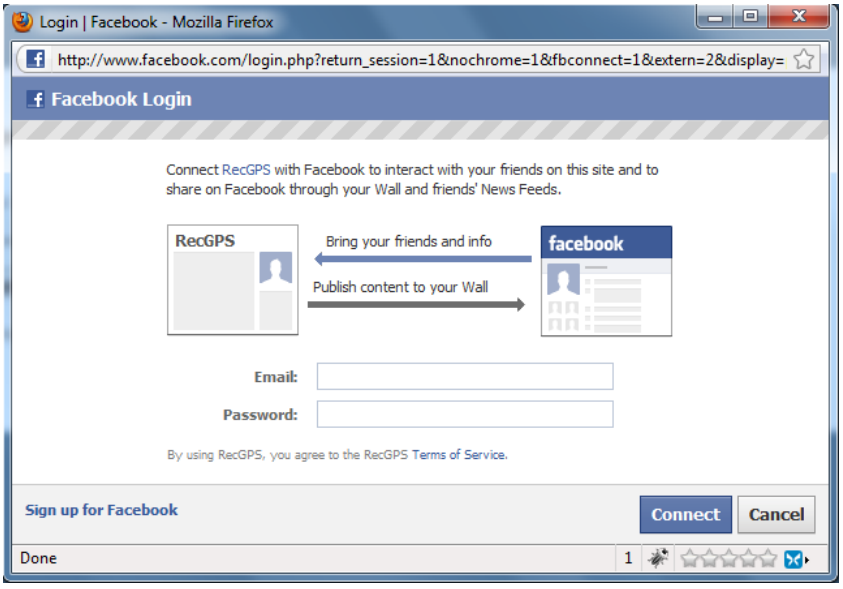


Figura 4.8: *Login Facebook*

Ficheiro de `resources`

Ao nível dos conteúdos escritos da aplicação *web* optou-se pela centralização dos mesmos num ficheiro de `resources`. Esse ficheiro é visto em toda a aplicação como sendo uma classe pública em que cada uma das entradas corresponde a uma propriedade. Caso seja necessário efectuar correcções ou alterações evita-se a pesquisa ficheiro a ficheiro.

Também os conteúdos escritos mostrados através de *Javascript* se encontram no ficheiro de `resources`. Para evitar o carregamento total ou individual dos conteúdos os mesmos são divididos em conjuntos específicos às funcionalidades que os utilizam: mapa[5]; discussão[6]; partilha[7].

4.4 User Interface

Ao bloco denominado de *User Interface* correspondem um *Web browser* para acesso ao *ASP.NET MVC WebSite* e uma aplicação móvel - Mobile Application; Figura 3.1: Arquitectura do *RecGPS*.

O *Web Browser* tem como objectivo a representação dos conteúdos gerados no *Web Site*, assim como a execução do *Javascritp* associado.

A implementação da aplicação móvel corresponde à implementação de um sub-sistema completo, pois apesar de interagir com o nível de *Route Management* - Capítulo 4.3 - não necessita deste para o seu funcionamento, ou seja, a aplicação móvel permite a recolha, armazenamento e execução de rotas sem que seja necessária qualquer interacção com a restante implementação do sistema - *standalone*. Na sua implementação destacam-se os seguintes passos, que de seguida se aprofundam:

- Estrutura
- Execução de rotas
- Recolha de rotas
- Comparação de Execuções
- Comunicação com *Route Management*

4.4.1 Estrutura

Tal como uma aplicação de *Windows Forms* em *.NET*, uma aplicação *.NETcf* com representação gráfica - janela - necessita ter uma classe que estenda de `Form`[8]. A instanciação

[5] `StandaloneWebApp.Controllers.RouteController.Language(string rand)`
[6] `StandaloneWebApp.Controllers.RouteController.LanguageDiscussion(string rand)`
[7] `StandaloneWebApp.Controllers.RouteController.LanguageShare(string rand)`
[8] `System.Windows.Forms.Form`

de uma janela tem um custo de alocação dos recursos necessários, sendo ainda necessário gerar o conteúdo a mostrar, tipicamente controlos.

Gestão de conteúdos

De modo a que os recursos utilizados pela aplicação se mantenham num nível reduzido é utilizado um mecanismo de controlo de conteúdo. Esse mecanismo baseia-se num conjunto de regras para a transição de conteúdo de modo a que uma alteração na ordem ou, no conteúdo mostrado tenha o menor impacto possível na restante aplicação. Para tal na classe `Main` além de uma propriedade que disponibiliza o objecto único - *singleton*[9] - do seu tipo, disponibiliza métodos estáticos para que conteúdo seja afixado, removido e para a inclusão de menu.

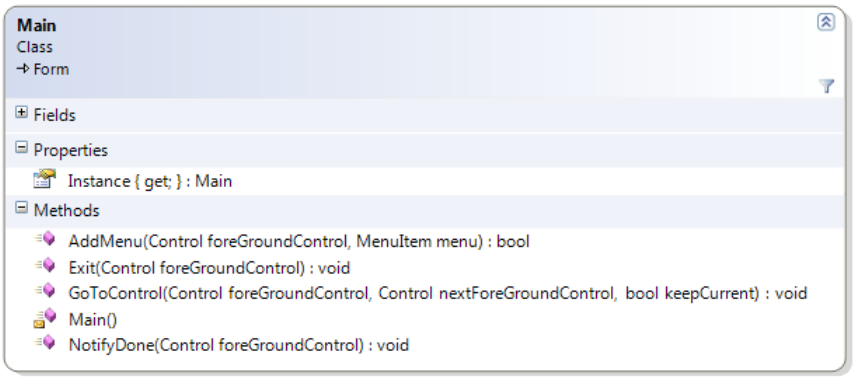


Figura 4.9: Classe `Main`

De modo a garantir que o controlo que efectua o pedido de mudança de conteúdo é o que está a a ser visualizado, os métodos de alteração têm como parâmetro um objecto do tipo `Control`, permitindo assim a verificação da condição.

`AddMenu`: A adição de *menu* - um objecto `MenuItem` - é efectuada pela chamada a `AddMenu`, no contexto do evento de notificação de alteração da propriedade `Parent`[10] - `ParentChanged` - e quando esta não esteja nula.

`Exit`: A classe `Main` remove e invoca `Dispose`[11] de todos os seus conteúdos.

[9] O modo de arranque de uma aplicação *forms* não permite que a classe a carregar seja estática ou *singleton* puro, daí a existência de um construtor `internal`. Ver Apêndice A.2.

[10] Indicação do `Control` em que o `Control` corrente está incluído.

[11] Notificação para libertação de recursos.

`GoToControl`: Adição de novo conteúdo, com indicação se o corrente é para ser mantido ou libertado.

`NotifyDone`: Indicação para libertação de recurso, voltando ao anterior.

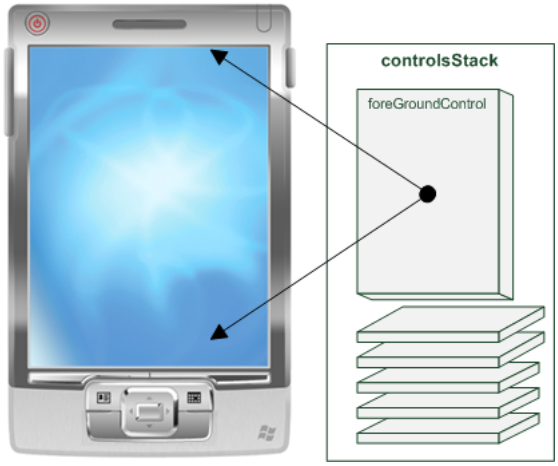


Figura 4.10: Exemplo da pilha de conteúdos - `controlsStack`

Criação de conteúdo

Os conteúdo criados têm como requisito estender de `Control` e no construtor ter como parâmetros, pelo menos, as dimensões máximas, em pixeis, que estes podem tomar[12]. Devem também implementar `IDisposable` sempre que estejam registados em eventos e/ou utilizem recursos que necessitem de libertação[13].

Suporte multilíngua

Com o intuito de aumentar o leque de possíveis utilizadores é necessário que todo o texto apresentado seja proveniente de ficheiro de `resources`. Com a adição de ficheiro localizados consegue-se a alteração da língua de vizualização da aplicação em tempo de execução.

Paralelismo com o `Repository`

Sendo o objectivo da ponte entre a aplicação móvel e o *Route Management* a troca de rotas, sobre um formato *standard*, através de *web service*, não há necessidade de implementar

[12] Altura e Largura, respectivamente, `Heigth` e `Width`.
[13] Por exemplo, *Streams*.

parte de *Abstract* - Capítulo 3, Concepção - para a representação de rotas. Apesar de o modelo estar bem definido, ao nível da aplicação móvel é mais conveniente a utilização de representações mais simplistas[14] por não ser necessário respeitar um modelo relacional de armazenamento de dados. Esta abordagem permite também a agregação de alguns alguns elementos que deixam de ser necessários (`PocketPointOfInterest` agrega as *interfaces* `IPointOfInterest`, `IPositionExtraInfo` e `IPositionExtraInfoType` de *Abstract*).

[14] `PocketRouteShort` e `PocketRoute`, sendo a primeira uma simplificação da segunda e esta de `IRoute` de *Abstract*.

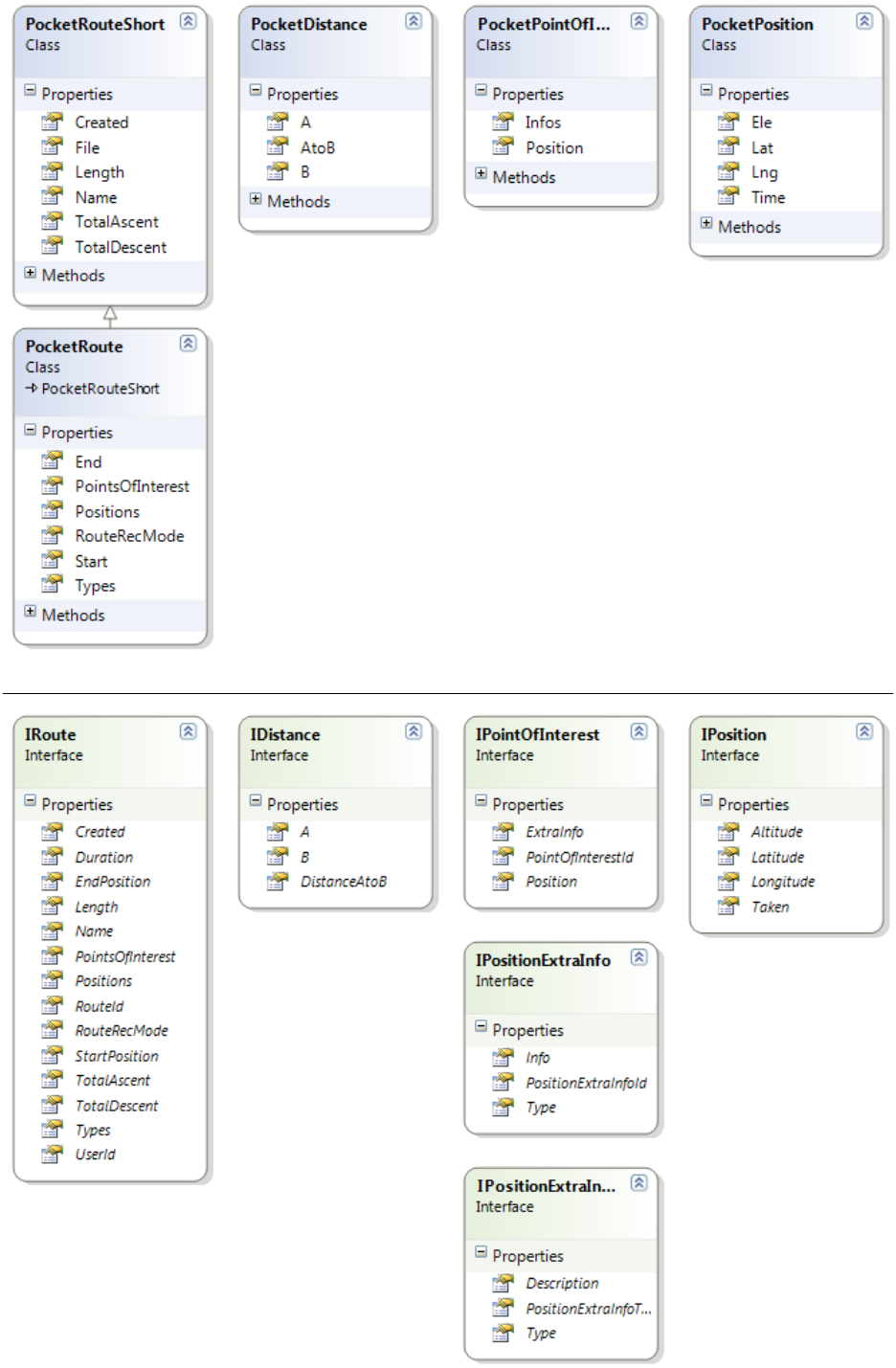


Figura 4.11: Entidades: paralelismo entre aplicação móvel e `Abstract`

4.4.2 Execução de rotas

Com a execução de rotas justifica-se a existência de dois tipos - `PocketRouteShort` e `PocketRoute` - para a representação de um rota. Para a escolha de qual a rota a executar são listadas as rotas existentes na directoria respectiva[15] e mostradas informações sobre as mesmas. Ao ser necessário obter informação sobre todas as rotas, sendo que apenas uma necessita de conter todos os dados presentes no ficheiro é então feita a distinção entre carregamento parcial para escolha e carregamento total para execução.

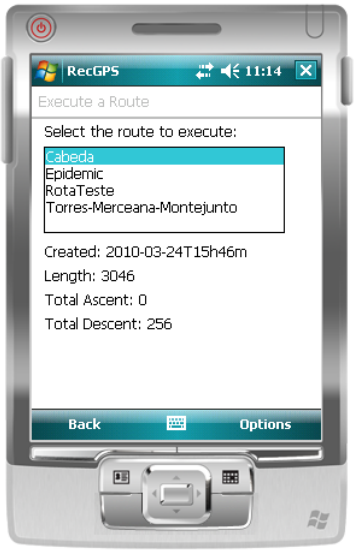


Figura 4.12: Escolha de rota a executar

Tanto para a escolha como a execução de rotas o carregamento dos dados da rota é efectuado através da classe `XmlTextReader`. Para a execução, além da rota, é utilizado um objecto da classe `Gps` para a obtenção da posição, assim como um `Timer` para o refrescamento da rota.

Sendo a classe `Gps` baseada em eventos é necessário a adição de *handlers* para os eventos de mudança de posição e estado[16]. Visto estes eventos não serem de *user interface*, no seu atendimento não é possível alterar a mesma[17]. Para tal é utilizado o mecanismo de `Invoke` com um `delegate` genérico para que o fio de execução correcto proceda à alteração dos controlos de *user interface*.

[15] `/Program Files/PocketApp/Routes`
[16] `LocationChanged` e `DeviceStateChanged`.
[17] Tentativa de alterar a *user interface* que não pelo fio de execução que a detém gera excepção.

Também a execução do *handler* associado ao `Timer` está impedido de invocar métodos específicos de *user interface*, sendo por isso utilizada uma estratégia semelhante.

```
public delegate void ControlUpdater(Control c, string s);

public static void ControlUpdate(Control c, string s)
{
    c.Text = s;
}


public delegate void UIRefresher(Control caller);

public static void UIRefresh(Control caller)
{
    caller.Refresh();
}
```

Listagem de Código 4.11: Mecanismo de actualização de *user interface*

Com o *handler* associado ao `Timer` provoca-se o refrescamento de todo o conteúdo para actualização da posição actual face à rota que se executa. O refrescamento tem que ser de todo o conteúdo pois apenas no evento de `Paint` se tem acesso ao objecto `Graphics` que permite efectuar o desenho.

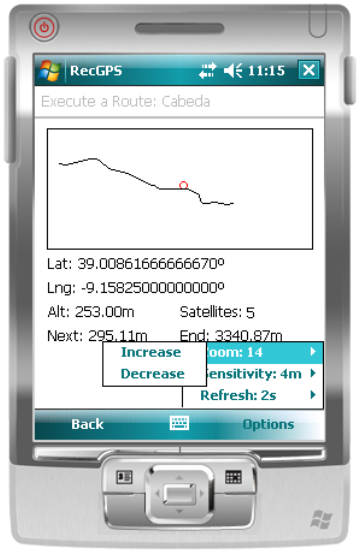


Figura 4.13: Rota em execução

A representação gráfica da rota é feita pela diferença de latitude e longitude das suas coordenadas ao centro da área de desenho (posição do utilizador). É também aplicado um

factor de escala - *zoom* - para permitir visualizar com mais detalhe ou, uma maior extensão da rota.

```
...
pos = route.Positions[i];

//Longitude - East&West
x = (int)((pos.Lng - lng) * zoom);
x += mapCenterX;

//Latitude - North&South
//Reversed difference because negative->south & positive->north | screen top->0 bottom->Heigth
y = (int)(-(pos.Lat - lat) * zoom);
y += mapCenterY;
...
```

Listagem de Código 4.12: Cálculo de posição relativa de um posição da rota

Durante a execução além da visualização dos dados da rota é possível a configuração do período de refrescamento do mapa, da distância máxima a que se pode estar da próxima posição para que esta seja assumida como cumprida e ainda o nível de *zoom* a aplicar ao mapa.

Sendo a execução definida como a repetição das posições de uma rota, conforme se executa uma rota vai-se alterando no objecto `PocketRoute` a hora de cada posição, conforme esta é alcançada, de modo a que no final o utilizador tenha a possibilidade de persistir essa mesma execução. Essa persistência consiste na criação de um ficheiro *GPX* através da rota alterada/executada.

4.4.3 Recolha de rotas

Para a recolha de rotas é prevista a existência de vários modos de recolha - Capítulo 2.1.3, Recolha de coordenadas - entre os quais se pode alternar. De modo a garantir a compatibilidade entre todos os modos de recolha é imposta uma *inteface* que estes têm que implementar. Nesta *interface* constam as propriedades que permitem obter o nome, descrição e existência de ecrã (conteúdo) de configuração - `Name`, `Description` e `HasConfigControl` -, assim como os métodos de obtenção de ecrã de configuração (`ConfigControl`) e de existência de nova posição para gravação(`Record`). Este último é invocado a cada nova posição disponível, sendo que apenas é efectuada a gravação caso os requisitos impostos pelo modo de recolha sejam cumpridos.

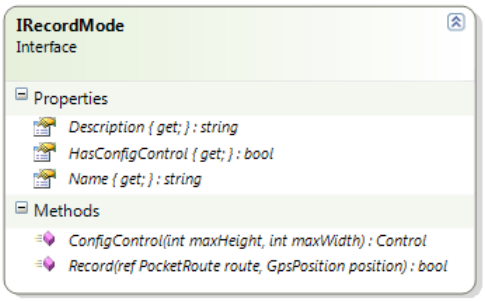


Figura 4.14: *Interface* `IRecordMode`

Além do modo de recolha é também utilizada a classe `Gps`[2] [3] para a obtenção da posição actual do utilizador. Durante a recolha é possível ao utilizador pausar a mesma, sendo isso equivalente a descartar a notificação de mudança de localização (evento `LocationChanged`).

```
private void gps_LocationChanged(object sender, LocationChangedEventArgs args)
{
  if (paused)
    return;
...
```

Listagem de Código 4.13: Pausa na recolha de rota

A criação de pontos de interesse durante a recolha consiste na passagem a um ecrã em que são preenchidos o tipo e a informação para o ponto de interesse. A posição a utilizar para esse ponto é obtida por acesso em exclusão ao campo que tem a última posição obtida. O cuidado nos acessos ao campo deve-se à sua actualização assíncrona através do evento de `PositionChanged`. Garantindo que todos os acessos ao campo são efectuados em exclusão garante-se a integridade dos valores obtidos. Na obtenção da posição para a criação de ponto de interesse adiciona-se a mesma à rota, podendo no entanto o ponto de interesse não ser inserido. Apesar de não respeitar os critérios de inserção do modo de recolha é desejado que cada ponto de interesse seja coincidente com uma posição da rota, sendo o custo de mais uma posição reduzido e evitando-se a verificação na inserção do ponto de interesse se essa mesma posição faz parte e caso não faça inserir na posição devida[18].

[18] A posições de uma rota têm que ser sequenciais. Entre a indicação de querer criar um ponto de interesse e a sua inserção - tempo de preenchimento - o evento de `Gps` continua a ser disparado, podendo por isso existir inserção de novas posições.

```
PocketPosition pos;
lock (lastPosLock)
{
    pos = lastPos.ToPocketPosition();
    route.Positions.Add(pos);
    //under the lock guarantees that the route has the position even if the POI is
    //not inserted into the route - not problematic and less expensive than confirming after
}
RecordRoute_AddPOI toAdd = new RecordRoute_AddPOI(Height, Width, pos, route.PointsOfInterest);
Main.Instance.GoToControl(this, toAdd, true);
```

Listagem de Código 4.14: Obtenção de última posição para preenchimento de ponto de ponto de interesse

4.4.4 Comparação de execuções

A comparação de duas execuções de uma rota consiste na visualização da evolução das posições de ambas as execuções face aos seus momentos de início. O desenho é efectuado com base na rota a que as execuções se referem.

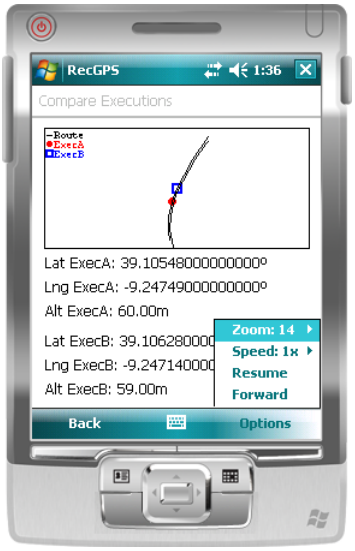


Figura 4.15: Comparação de execuções

Tal como na execução de rotas - Capítulo 4.4.2, Execução de Rotas - o desenho da rota é efectuado através de um `Timer` que desencadeia o evento de `Paint`. Além do desenho da rota, tendo em conta a posição mais avançada de ambas as execuções, são desenhados os identificadores de cada execução, assim como a legenda. Durante a execução é permitida a pausa, recuo e alteração de velocidade de execução.

Tendo as execuções a comparar horas de início distintas a sua comparação tem que ser

feita através de uma diferença temporal (comum a ambas as execuções) à hora de início. Assim sendo, o avanço na execução equivale a, findo cada período do `Timer`, avançar ou recuar com a diferença temporal. A regulação da velocidade de visualização[19] corresponde à manipulação do período utilizado. O motivo para a alteração do período ao invés da porção de avanço[20] é a prevenção de 'saltos' devido a deslocamentos elevados entre refrescamentos.

```
private void TimerAction(object dummy)
{
  if (paused) return;

  //convert from ticks to seconds
  if (forward) diff += new TimeSpan(elapsedTimeAdvance * 10000000);
  else diff -= new TimeSpan(elapsedTimeAdvance * 10000000);

  if (diff <= tsZero) diff = new TimeSpan(0);

  if ((routeIdx + 2 >= route.Positions.Count) || (execAIdx + 2 >= execA.Positions.Count) || (execBIdx + 2 >= execB.Positions.Count))
  {
    refreshTimer.Dispose();
    MessageBox.Show(Language.CompareEnded, Language.CompareExecutions, MessageBoxButtons.OK, MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
    UnHook();
    Invoke(dn, this);
  }

  refresh = forward ? Advance() : Reverse();

  if (refresh)
  {
    try {
      Invoke(uir, this);
    } catch (ObjectDisposedException) { UnHook(); }
  }
}
```

Listagem de Código 4.15: Rotina de refrescamento do `Timer`

Para a escolha da rota e das execuções a comparar são disponibilizados os respectivos ecrãs. Para passar da rota às execuções tem-se como requisito a existência de pelo menos duas execuções. Em ambos os casos a obtenção dos elementos disponíveis é feita através do varrimento das respectivas pastas[21].

A escolha da rota leva à passagem para o ecrã seguinte de um objecto `PocketRouteShort`,

[19] Aumento e redução com factor 2.

[20] Utiliza-se 1 segundo, pois é o período típico de obtenção de coordenadas nos dispositivos *GPS*

[21] `/Program Files/PocketApp/Routes` e `Program Files/PocketApp/Executions`

(a) Escolha de rota (b) Escolha de execuções (c) Gestão de execuções

Figura 4.16: Escolha de rota, escolha execuções a comparar e gestão de execuções

sendo no ecrã seguinte escolhidas as execuções e então passadas à visualização a rota e as execuções (três objectos `PocketRoute`).

Tendo como ponto de partida a escolha de rotas na (Figura 4.16) é possível aceder a um ecrã que lista todas a execuções e permite a sua remoção.

4.4.5 Comunicação com `Route Management`

Sendo as rotas e execuções presentes no *RecGPS* associadas a um utilizador, para a publicação destas através da aplicação móvel - via *Web Service* - são necessário mecanismos que permitam verificar a identidade do utilizador (autenticação). Para os utilizadores com *login* interno - Capítulo 4.3.2, Aplicação *web* - basta o fornecimento das suas credenciais (*email* e *password*) para que no `Route Management` se consiga verificar a sua veracidade. Para os utilizadores cujo *login* é externo (através da rede social *Facebook*) é necessária a obtenção do seu identificador.

Facebook for Mobile Apps

Para a obtenção do identificador de utilizador é disponibilizada pela rede social uma variante do método de autenticação utilizado para aplicações *web*, vocacionada para aplicações

móveis[43]. A utilização deste método dá-se também através de *browser*, sendo por isso utilizada a classe `WebBrowser`[22][44] e consiste num conjunto de três interacções[23].

-Primeira interacção

Figura 4.17: Interacções para obtenção de *Facebook ID*, #1.

O utilizador é convidado a autenticar-se e a autorizar a aplicação a aceder aos seus dados, caso ainda não o tenha feito. De seguida é redireccionado de volta com um código que permite validar, perante o *Facebook*, que o utilizador autorizou a aplicação.

-Segunda interacção

Figura 4.17: Interacções para obtenção de *Facebook ID*, #2.

Com o código obtido na primeira interacção inicia-se a segunda, tendo como resposta um *access token*. Este *token* permite aceder aos dados de utilizador através da *Graph API*[45].

-Terceira interacção

Figura 4.17: Interacções para obtenção de *Facebook ID*, #3.

Com recurso à *Graph API* é efectuada a terceira interacção, sendo no pedido requisitado o identificador do utilizador. Na resposta é então obtido o identificador a utilizar para autenticação perante o *RecGPS*.

[22] `System.Windows.Forms.WebBrowser`

[23] Entenda-se: um pedido e uma resposta/redireccionamento.

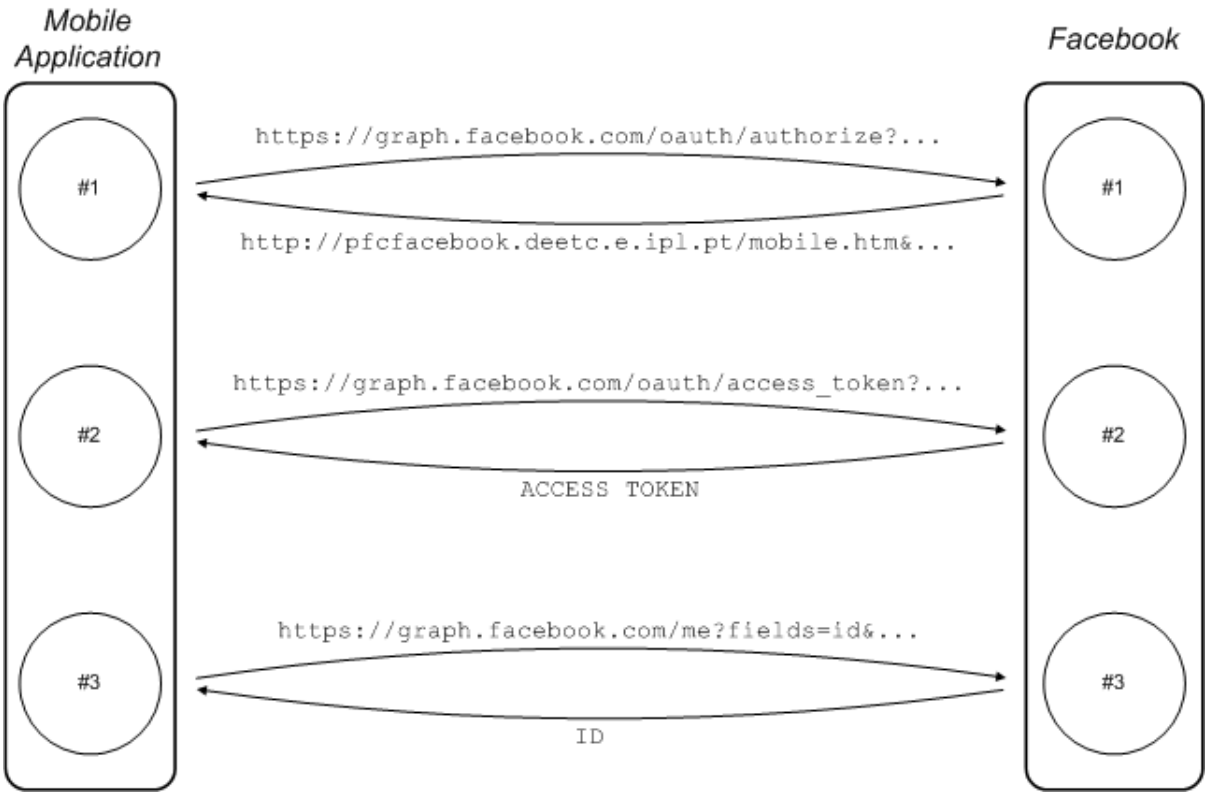


Figura 4.17: Interacções para obtenção de *Facebook ID*

(a) Escolha de método (b) Inserção de credenciais (c) Resultado da publicação

Figura 4.18: Passos para publicação através de conta *Facebook*

Web Service

Sendo o *ABC* - *Address*, *Binding* e *Contract* - a forma de descrever um *Web Service*, para a comunicação entre a aplicação móvel e o `Route Management`, utiliza-se um endereço - *Address* - da aplicação *Web*[24] e um *Binding* suportado pela *.NETcf*, o `BasicHttpBinding` - Apêndice A.3, *Binding* do *Web Service* `RouteManagementWebService`. Ao nível do contrato - *Contract* - são definidas as operações que permitem a inserção de rotas e exeuções tanto para utilizadores locais como *Facebook*, de acordo com a Listagem de Código 4.16: Contrato do *Web Service* `RouteManagementWebService`.

```
[ServiceContract]
public interface IRouteManagementWebService
{
    [OperationContract]
    KeyValuePair<int, string> PublishRouteFbUser(decimal fbId, string routeTxt);

    [OperationContract]
    KeyValuePair<int, string> PublishRouteLocalUser(string email, string passMD5, string ↩
        routeTxt);

    [OperationContract]
    KeyValuePair<int, string> PublishExecutionFbUser(decimal fbId, string routeTxt, string ↩
        executionTxt);

    [OperationContract]
    KeyValuePair<int, string> PublishExecutionLocalUser(string email, string passMD5, string ↩
        routeTxt, string executionTxt);
}
```

Listagem de Código 4.16: Contrato do *Web Service* `RouteManagementWebService`

Na implementação do serviço são validadas as credenciais fornecidas pelo utilizador e efectuadas as operações pedidas. O retorno das operações inclui o identificador da inserção - rota ou execução - e a mensagem associada. Caso a inserção seja de um elemento que já se encontre no sistema é retornado o identificador já existente e a mensagem indica tal situação.

Quer ao nível de rotas quer de execuções é verificada a existência prévia das mesmas, sendo que no retorno é fornecido o identificador das mesma, assim como uma mensagem de resultado, permitindo notificar o utilizador que a rota[25] ou execução[26] já se encontra

[24] `http://pfcfacebook.deetc.e.ipl.pt/RouteManagementWebService.svc?wsdl`
[25] Apêndice A.5, Inserção de rota em `RouteManagementWebService`
[26] Apêndice A.6, Inserção de execução em `RouteManagementWebService`

inserida ou a ocorrência de erro na inserção/autenticação.

Capítulo 5

Aferição de resultados

Este capítulo tem como objecto verificar as capacidades da aplicação móvel desenvolvida. A aferição de resultados compreende os seguintes passos:

- 5.1 - Pressupostos[1]
- 5.2 - Recolha de amostras
- 5.3 - Obtenção de resultados
- 5.4 - Comparação de resultados

5.1 Pressupostos

O cenário de teste desejado compreende a repetição de uma rota, em condições de teste semelhantes e a diferentes velocidades, de modo a comparar os resultados obtidos com os métodos de recolha existentes na aplicação móvel.

A aferição de resultados é efectuada através de automóvel numa estrada nas imediações de Sobral de Monte Agraço na distância aproximada de 2900 metros. O motivo da escolha prende-se com o percurso algo sinuoso e com zonas mais e menos rápidas, nomeadamente a passagem pela localidade de Cabêda onde a estrada é estreita. Além do traçado o reduzido tráfego dessa estrada facilita a obtenção de resultados em diferentes velocidades sem o transtorno de veículos a diferentes velocidades.

[1] *pressuposto - s.m. - projecto; plano* - [1]

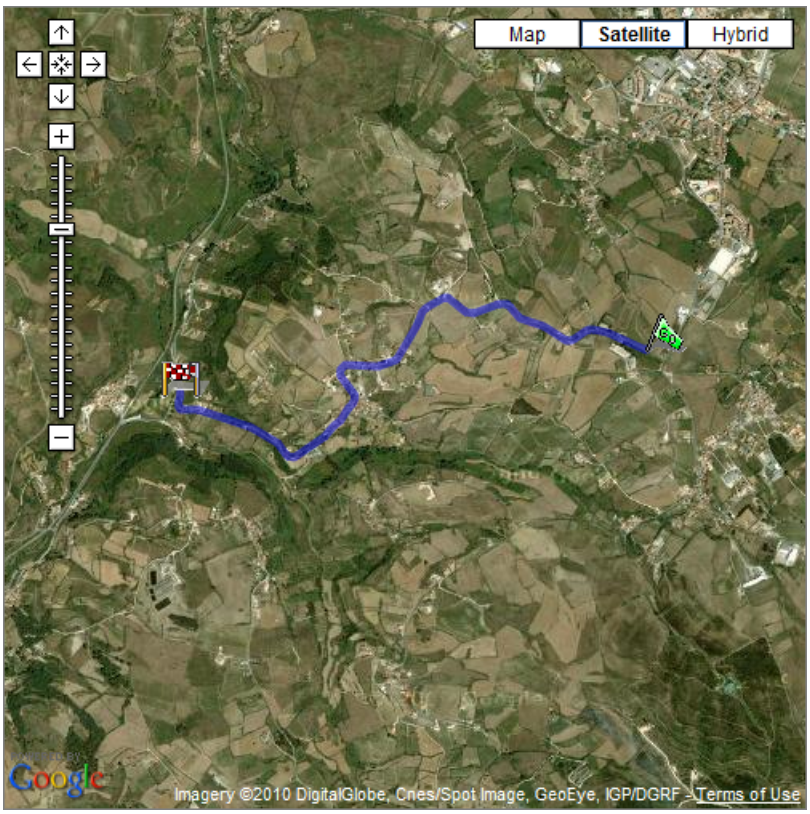


Figura 5.1: Percurso base de teste

Devido à indisponibilidade de um *PDA* com capacidade para executar a *.NETcf*, para a qual a aplicação móvel se encontra desenvolvida, é definido um cenário alternativo. Ao invés da execução, no local, da aplicação, utiliza-se um dispositivo com funcionalidade *GPS* e capacidade de recolha dos seus dados em bruto (sem tratamento). Os dados obtidos são fornecidos a uma aplicação - *FakeGPS* [46] - que posteriormente, no emulador de *PDA*, simula a funcionalidade *GPS* e permite a obtenção de dados de posicionamento através da *API GPS* da *.NETcf*.

5.2 Recolha de amostras

A recolha das amostras de dados *GPS* foi efectuada através da aplicação *Race Chrono* [47] que se executava num telemóvel *Nokia 5800 XpressMusic*[48]. Os dados (NMEA [49] [50]) foram recolhidos em três passagens consecutivas pelo percurso de teste.

Com recurso à ferramenta *GPS Babel* [4] efectuou-se a conversão do formato *NMEA* para

o *GPX*. O resultado da conversão é utilizado na aplicação *Web GPSies* [15] para obtenção de dados base das amostras, nomeadamente distância percorrida e representação em mapa das posições obtidas (Figuras: 5.2; 5.3; 5.4).

	Amostra nº1	Amostra nº2	Amostra nº3
Data de recolha	11-09-2010	11-09-2010	11-09-2010
Hora de início	18:30:53	18:39:37	18:47:44
Hora de fim	18:35:09	18:42:44	18:50:23
Duração	4:16	3:07	2:39
Distância (m)	2875	2844	2989
Vel. média (Km/h)	40.43	54.75	67.68

Tabela 5.1: Informações gerais das amostras recolhidas

Na representação em mapa é possível observar que com o aumento da velocidade a distância entre cada posição recolhida também aumenta, assim como o aparecimento de zonas em que o dispositivo *GPS* é incapaz de fornecer posições válidas (Amostra 3).

Figura 5.2: Amostra 1

Figura 5.3: Amostra 2

Figura 5.4: Amostra 3

5.3 Obtenção de resultados

Devido à utilização da ferramenta *FakeGPS* constatou-se que apenas os dados recolhidos são repetidos, não sendo o tempo total dos mesmos respeitado. Assim sendo são considerados a distância total, o número de posições recolhidas e o tamanho do ficheiro gerado.

Modo de recolha	Dado recolhido	Amostra nº1	Amostra nº2	Amostra nº3
Precisão máxima	Distância (m)	2792.07	2830.55	2976.82
	Nº Posições	222	175	146
	Ficheiro (KB)	31.7	25.2	21.3
Distância mínima (5m)	Distância (m)	2839.63	2813.16	2984.23
	Nº Posições	210	165	134
	Ficheiro (KB)	30.1	23.9	19.7
Tempo mínimo (5s)	Distância (m)	2766.17	2755.07	2803.32
	Nº Posições	48	36	28
	Ficheiro (KB)	7.8	6.31	5.2
Adaptativo	Distância (m)	2773.95	2764.13	
	Nº Posições	49	43	38
	Ficheiro (KB)	8.1	7.3	6.5

Tabela 5.2: Dados recolhihos

Visualmente podem-se observar as variações presentes na Tabela 5.2 nas Figuras: 5.5; 5.6; 5.7; 5.8 e Apêndices: A.8; A.9; A.10; A.11; A.12; A.13; A.14; A.15.

Figura 5.5: Amostra 1 - Precisão máxima

Figura 5.6: Amostra 1 - Distância mínima

Figura 5.7: Amostra 1 - Tempo mínimo

Figura 5.8: Amostra 1 - Adaptativo

5.4 Comparação de resultados

A comparação dos resultados obtidos é efectuada, para cada modo de recolha, ao nível das distâncias totais , número de posições que geram essa distância e ainda o tamanho do ficheiros *GPX* gerados.

5.4.1 Distâncias

Tendo como distâncias de referência as obtidas na Tabela 5.1 as distâncias recolhidas para o conjunto das vinte e quatro recolhas efectuadas encontram-se tipicamente abaixo do valor de referência e numa proporção inferior a 5% (em destaque as recolhas que não respeitam).

	Amostra nº1 2875m (100%)	Amostra nº2 2844m (100%)	Amostra nº3 2989m (100%)
Precisão máxima	2792.07 (97.12%)	2830.55 (98.15%)	2980 (99.59%)
Distância mínima	2839.63 (98.77%)	2813.16 (98.91%)	2984.23 (99.84%)
Tempo mínimo	2766.17 (96.21%)	2755.07 (96.87%)	2803.32 (**93.79%**)
Adaptativo	2773.95 (96.49%)	2764.13 (97.19%)	2900.13 (97.03%)

Tabela 5.3: Comparação de distâncias recolhidas

5.4.2 Posições

A comparação do número de posições obtidas toma como referência o valor mais alto obtido através do modo de recolha de precisão máxima, pois este armazena todas as posições sequenciais não repetidas.

	Amostra nº1 Posições (%)	Amostra nº2 Posições (%)	Amostra nº3 Posições (%)
Precisão máxima	222 (100%)	175 (100%)	146 (100%)
Distância mínima	210 (94.59%)	165 (94.29%)	134 (91.78%)
Tempo mínimo	48 (21.62%)	36 (20.57%)	28 (19.18%)
Adaptativo	49 (22.07%)	43 (24.57%)	38 (26.03%)

Tabela 5.4: Comparação de número de posições recolhidas

Tendo em consideração um percurso de estrada feito de automóvel nota-se a tendência de a recolha com base na distância mínima tender para o número de posições do modo mais preciso derivado a uma configuração de distância mínima reduzida (5m). No método baseado em tempo (5s), para velocidades mais elevadas, nota-se a tendência a recolher menos posições face ao método adaptativo.

5.4.3 Ficheiros gerados

Ao nível dos ficheiros gerados também a comparação de resultados toma como referência o maior ficheiro gerado para as recolhas efectuadas sobre cada uma das amostras.

	Amostra nº1 Tamanho KB (%)	Amostra nº2 Tamanho KB (%)	Amostra nº3 Tamanho KB (%)
Precisão máxima	31.7 (100%)	25.2 (100%)	21.3 (100%)
Distância mínima	30.1 (94.95%)	23.9 (94.84%)	19.7 (92.49%)
Tempo mínimo	7.88 (24.86%)	6.31 (25.04%)	5.18 (24.32%)
Adaptativo	8.05 (25.39%)	7.30 (28.97%)	5.62 (26.38%)

Tabela 5.5: Comparação do tamanho do ficheiro gerado

Capítulo 6

Conclusões

6.1 Conclusões e críticas

Foram atingidos estados funcionais dos dois principais elementos do projecto, a aplicação *web* e a aplicação móvel.

Na aplicação *web* é permitido ao utilizador a visualização e discussão de rotas, assim como a sua partilha em rede social. O utilizador pode ainda efectuar o seu registo a nível interno ou externo. Ao nível dos grupos não foi possível implementar a totalidade das funcionalidades desejadas devido a limitações no desenho do modelo de dados.

A aplicação móvel permite a recolha e execução de rotas de modo autónomo, assim como a publicação destas na aplicação *web* quando com ligação à *Internet*. Na aplicação móvel são também contemplados os utilizadores internos e externos, pois é disponibilizada a autenticação dos externos através da rede social.

Com a montagem de um cenário de testes (Capítulo 5) foi possível aferir a qualidade de recolha da aplicação móvel pois são apresentados resultados de distância próximos ao de outra solução (*web*) aceite pela comunidade (mais de 500 000 rotas submetidas [51]). Foi também possível aferir que os modos de recolha implementados permitem a recolha de rotas, com maior ou menor nível de detalhe, sem comprometer a semelhança ao percurso efectuado.

Devido ao cenário de teste utilizado as recolhas têm por base um conjunto de amostras reduzido. Face a um cenário real os testes efectuados não permitem a confirmação da

qualidade/semelhança entre recolhas distintas de uma mesma rota. Caso se utilizasse um dispositivo real com funcionalidade *GPS* não existiriam duas amostras iguais, permitindo assim a confirmação destas características.

Ambas as aplicações atingiram os principais objectivos propostos, já que, em conjunto, permitem a recolha, execução, partilha em rede social e discussão de rotas.

6.2 Desenvolvimentos futuros

Durante o desenvolvimento do *RecGPS* não foi possível a implementação de algumas das funcionalidades desejadas.

O desenho de rotas na aplicação *web*, com recurso a *API*, de mapas necessitaria de um investimento temporal significativo face ao tempo disponível e às restantes funcionalidades a implementar. Visto, desde o primeiro momento do projecto, estar prevista a implementação de uma aplicação móvel com capacidade de recolha de rotas optou-se por deixar o desenho através da aplicação *web* para versões futuras do *RecGPS*.

A discussão e partilha associada a grupos requereria a repetição das funcionalidades de discussão e partilha gerais já existentes na aplicação *web*, assim como a alteração do modelo de dados e do repositório. Estas alterações seriam necessárias pois no desenho destes últimos não foi considerada a existência de rotas e *posts* específicos ao grupo.

Também ao nível dos grupos a disponibilização na aplicação *web* de funcionalidades de criação e gestão estava prevista. A mesma não foi disponibilizada devido às limitações existente ao nível de discussão e partilha, estando no entanto prevista ao nível do modelo de dados e do repositório.

Permitir aos utilizadores a fusão de tipos de conta (interna e externa) também seria um exercício interessante do ponto de vista de usabilidade pois, por exemplo, permitiria ao utilizador o acesso ao *RecGPS* de um computador público pela sua conta de *Facebook* caso este não se recordasse dos dados para *login* interno e vice versa.

Seria também interessante permitir aos utilizadores efectuarem a separação de contas pois caso um utilizador elimine a sua conta *Facebook* não deve ser impedido de aceder à sua conta no *RecGPS*. Tanto a fusão como a separação de contas são permitidas pelo modelo

de dados e repositório existentes, não tendo sido disponibilizadas devido às limitações temporais existentes.

A utilização de um grafismo mais apelativo, assim como o melhoramento em termos de usabilidade e acessibilidade seriam uma mais-valia nas aplicações *web* e móvel.

6.3 Notas finais

A realização deste trabalho de projecto permitiu acompanhar o ciclo de desenvolvimento de um projecto desde o desenvolvimento da sua ideia até à sua concretização. Foi ainda possível, através do Capítulo 5, constatar que a funcionalidade de recolha, para o cenário de testes montado, apresenta bons resultados.

Apêndice A

Listagens e diagramas

A.1 *Schema* de extensão ao *GPX*

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://pfcfacebook.deetc.e.ipl.↩
    pt/xsd" xmlns:gpx="http://www.topografix.com/GPX/1/1" targetNamespace="http://pfcfacebook.↩
    deetc.e.ipl.pt/xsd" elementFormDefault="qualified">
    <xs:import namespace="http://www.topografix.com/GPX/1/1"/>
    <xs:complexType name="extensions">
        <xs:choice minOccurs="1" maxOccurs="1">
            <xs:element ref="routeRecMode" minOccurs="0" maxOccurs="1"/>
            <xs:element ref="routeType" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="totalLength"/>
            <xs:element ref="totalAscent"/>
            <xs:element ref="totalDescent"/>
            <xs:element name="startPosition">
                <xs:complexType>
                    <xs:choice minOccurs="1" maxOccurs="1">
                        <xs:element name="trkpt">
                            <xs:complexType>
                                <xs:choice minOccurs="1" maxOccurs="1">
                                    <xs:element name="ele" type="xs:decimal" />
                                    <xs:element name="time" type="xs:dateTime" />
                                </xs:choice>
                                <xs:attribute name="lat" type="xs:decimal" />
                                <xs:attribute name="lon" type="xs:decimal" />
                            </xs:complexType>
                        </xs:element>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="endPosition">
                <xs:complexType>
                    <xs:choice minOccurs="1" maxOccurs="1">
                        <xs:element name="trkpt">
```

```
                            <xs:complexType>
                                <xs:choice minOccurs="1" maxOccurs="1">
                                    <xs:element name="ele" type="xs:decimal" />
                                    <xs:element name="time" type="xs:dateTime" />
                                </xs:choice>
                                <xs:attribute name="lat" type="xs:decimal" />
                                <xs:attribute name="lon" type="xs:decimal" />
                            </xs:complexType>
                        </xs:element>
                    </xs:choice>
                </xs:complexType>
            </xs:element>
            <xs:element name="pointOfInterest" >
                <xs:complexType>
                    <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element name="trkpt">
                            <xs:complexType>
                                <xs:choice>
                                    <xs:element name="ele" type="xs:decimal" />
                                    <xs:element name="time" type="xs:dateTime" />
                                </xs:choice>
                                <xs:attribute name="lat" type="xs:decimal" />
                                <xs:attribute name="lon" type="xs:decimal" />
                            </xs:complexType>
                        </xs:element>
                        <xs:choice minOccurs="1" maxOccurs="unbounded">
                            <xs:element name="positionExtraInfo">
                                <xs:complexType>
                                    <xs:sequence minOccurs="1" maxOccurs="unbounded">
                                        <xs:element name="peiType" type="xs:string" />
                                        <xs:element name="peiInfo" type="xs:string" />
                                    </xs:sequence>
                                </xs:complexType>
                            </xs:element>
                        </xs:choice>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:choice>
    </xs:complexType>
    <xs:element name="routeRecMode" type="xs:string"/>
    <xs:element name="routeType" type="xs:string"/>
    <xs:element name="totalLength" type="xs:decimal"/>
    <xs:element name="totalAscent" type="xs:decimal"/>
    <xs:element name="totalDescent" type="xs:decimal"/>
    <xs:element name="extensions" type="extensions"/>
</xs:schema>
```

Listagem de Código A.1: *Schema* de extensão ao GPX

A.2 Implementação de *singleton* na classe `Main`

```
public partial class Main : Form
{
    /// <summary>
    /// Private fields
    /// </summary>
    private static Main instance = null;
    private static object instanceLock = new object();
    ...

    /// <summary>
    /// So that a cast (from Parent property) can be avoided when a screen change is needed
    /// </summary>
    public static Main Instance
    {
        get
        {
            lock(instanceLock)
            {
                if (Equals(instance, null))
                    instance = new Main();
                return instance;
            }
        }
    }

    /// <summary>
    /// Shows the welcome splash screen
    /// </summary>
    internal Main()
    {
        if (!Equals(instance, null))
            throw new ApplicationException("The Singleton pattern is not beeing respected");
        ...
    }

    ...
}
```

Listagem de Código A.2: Implementação de *singleton* na classe `Main`

A.3 *Binding* do *Web Service* RouteManagementWebService

```
<system.serviceModel>
    <bindings>
        <basicHttpBinding>
            <binding name="myBindConfig">
                <security mode="None">
                    <transport clientCredentialType="None"/>
                </security>
                <readerQuotas maxStringContentLength="10485760"/>
            </binding>
        </basicHttpBinding>
    </bindings>
    <behaviors>
        <serviceBehaviors>
            <behavior name="StandaloneWebApp.RouteManagementWebServiceBehavior">
                <serviceMetadata httpGetEnabled="true" />
                <serviceDebug includeExceptionDetailInFaults="false" />
            </behavior>
        </serviceBehaviors>
    </behaviors>
    <services>
        <service behaviorConfiguration="StandaloneWebApp.RouteManagementWebServiceBehavior"
                 name="StandaloneWebApp.RouteManagementWebService">
            <endpoint address=""
                      binding="basicHttpBinding"
                      bindingConfiguration="myBindConfig"
                      contract="StandaloneWebApp.IRouteManagementWebService">
                <identity>
                    <dns value="localhost" />
                </identity>
            </endpoint>
            <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
        </service>
    </services>
</system.serviceModel>
```

Listagem de Código A.3: *Binding* do *Web Service* RouteManagementWebService

A.4 Classes da aplicação móvel

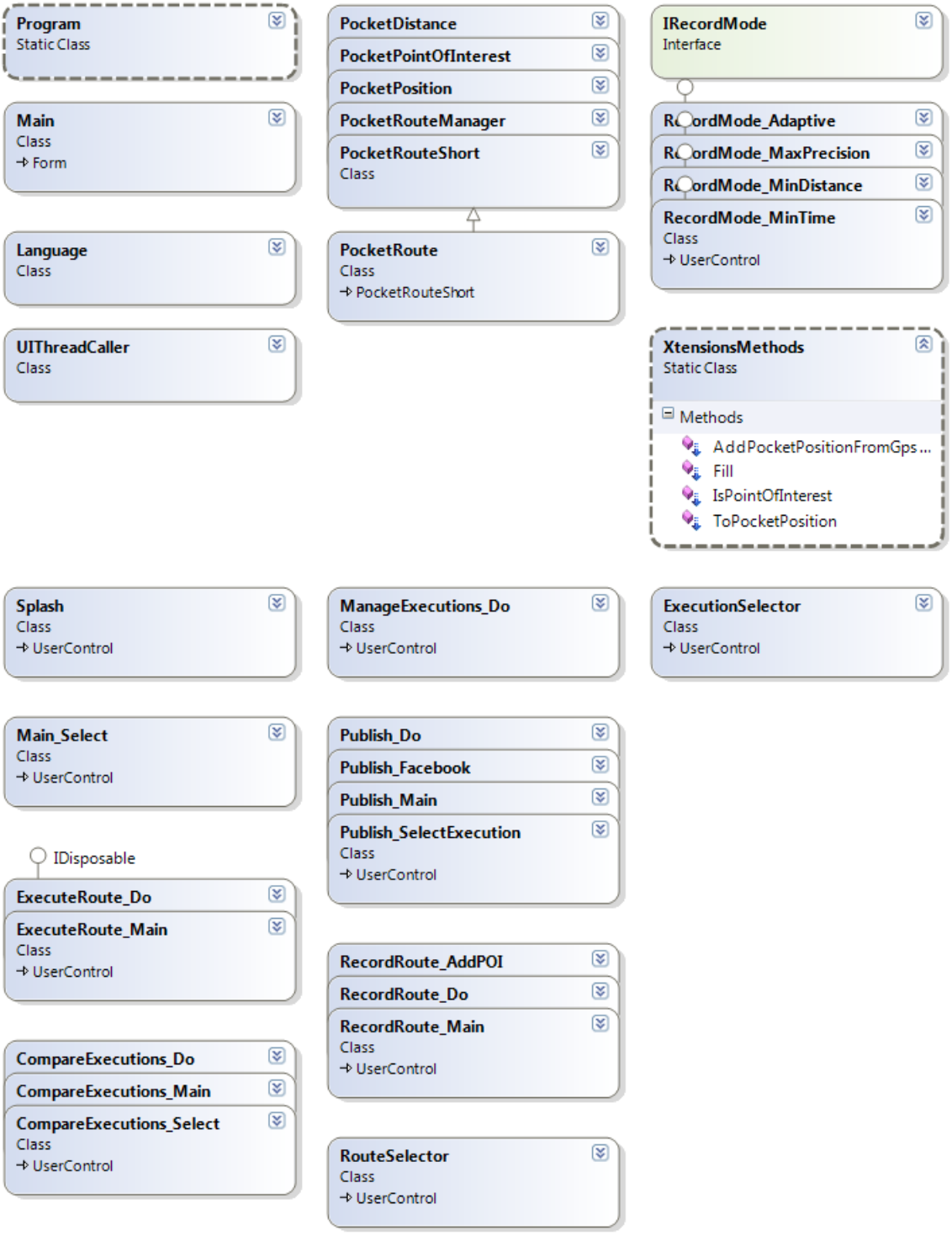


Figura A.1: Classes da aplicação móvel

A.5 Inserção de rota em RouteManagementWebService

```
1  private KeyValuePair<int, string> InsertRoute(IUser user, string routeTxt)
   {
       IRoute route = RML.Factory.CreateIRoute(routeTxt);
       decimal minLat = route.StartPosition.Latitude < route.EndPosition.Latitude ? route.↩
           StartPosition.Latitude : route.EndPosition.Latitude;
       decimal maxLat = route.StartPosition.Latitude > route.EndPosition.Latitude ? route.↩
           StartPosition.Latitude : route.EndPosition.Latitude;
6      decimal minLng = route.StartPosition.Longitude < route.EndPosition.Longitude ? route.↩
           StartPosition.Longitude : route.EndPosition.Longitude;
       decimal maxLng = route.StartPosition.Longitude > route.EndPosition.Longitude ? route.↩
           StartPosition.Longitude : route.EndPosition.Longitude;
       IEnumerable<IRoute> routesBetween = RML.GetBetweenPositions(minLat, minLng, maxLat, maxLng↩
           );

       IEnumerable<IRoute> routesMatch = from rt in routesBetween
11         where Equals(rt.StartPosition.Latitude, route.StartPosition.Latitude)
           && Equals(rt.StartPosition.Longitude, route.StartPosition.Longitude)
           && Equals(rt.EndPosition.Latitude, route.EndPosition.Latitude)
           && Equals(rt.EndPosition.Longitude, route.EndPosition.Longitude)
           select rt;
16
       if (Equals(routesMatch.Count(), 0))
       {
           route.UserId = user.UserId;
           IRoute routeAdded = RML.RouteRepository.Add(route);
21         return new KeyValuePair<int, string>(routeAdded.RouteId, Language.RouteAdded);
       }

       routesMatch = from rt in routesMatch
           where Equals(rt.Created, route.Created)
26         && Equals(rt.Duration, route.Duration)
           && Equals(rt.Name, route.Name)
           && Equals(rt.TotalAscent, route.TotalAscent)
           && Equals(rt.TotalDescent, route.TotalDescent)
           select rt;
31
       if (Equals(routesMatch.Count(), 0))
       {
           IRoute routeAdded = RML.RouteRepository.Add(route);
           return new KeyValuePair<int, string>(routeAdded.RouteId, Language.RouteAdded);
36     }

       routesMatch = from rt in routesMatch
           where Equals(rt.Positions.Count(), route.Positions.Count())
           && Equals(rt.PointsOfInterest.Count(), route.PointsOfInterest.Count())
41         select rt;

       if (Equals(routesMatch.Count(), 0))
       {
           IRoute routeAdded = RML.RouteRepository.Add(route);
46         return new KeyValuePair<int, string>(routeAdded.RouteId, Language.RouteAdded);
```

```
    }

    //POIs should be ordered too
    IPointOfInterest poiA;
    IPointOfInterest poiB;
    bool match;
    List<IRoute> routesMatch2 = new List<IRoute>();
    foreach (IRoute rt in routesMatch)
    {
        match = Equals(rt.PointsOfInterest.Count(), 0) ? Equals(route.PointsOfInterest.Count()↩
            , 0) : true;
        for (int i = 0; i < rt.PointsOfInterest.Count(); i++)
        {
            poiA = rt.PointsOfInterest.ElementAt(i);
            poiB = route.PointsOfInterest.ElementAt(i);
            if (!Equals(poiA.Position.Latitude, poiB.Position.Latitude) || !Equals(poiA.↩
                Position.Longitude, poiB.Position.Longitude))
            {
                match = false;
                break;
            }
        }
        if (match)
            routesMatch2.Add(rt);
    }

    if (Equals(routesMatch2.Count(), 0))
    {
        IRoute routeAdded = RML.RouteRepository.Add(route);
        return new KeyValuePair<int, string>(routeAdded.RouteId, Language.RouteAdded);
    }

    //routesMatch = routesMatch2.AsQueryable();
    IPosition posA;
    IPosition posB;
    List<IRoute> routesMatch3 = new List<IRoute>();
    foreach (IRoute rt in routesMatch2)
    {
        match = Equals(rt.Positions.Count(), 0) ? Equals(route.Positions.Count(), 0) : true; ↩
            match = true;
        for (int i = 0; i < rt.Positions.Count(); i++)
        {
            posA = rt.Positions.ElementAt(i);
            posB = route.Positions.ElementAt(i);
            if (!Equals(posA.Latitude, posB.Latitude) || !Equals(posA.Longitude, posB.↩
                Longitude))
            {
                match = false;
                break;
            }
        }
        if (match)
            routesMatch3.Add(rt);
    }
```

```
    if (Equals(routesMatch3.Count(), 0))
    {
        IRoute routeAdded = RML.RouteRepository.Add(route);
        return new KeyValuePair<int, string>(routeAdded.RouteId, Language.RouteAdded);
    }

    ////At this point only one should match, but if more any suffices??
    IRoute routeMatch = routesMatch3.First();
    return new KeyValuePair<int, string>(routeMatch.RouteId, Language.RouteAlreadyExisted);
}
```

Listagem de Código A.4: Inserção de rota em `RouteManagementWebService`

A.6 Inserção de execução em RouteManagementWebService

```
private KeyValuePair<int, string> InsertExecution(IUser user, IRoute route, string ↩
    executionTxt)
{
    try
    {

        IExecution ex = RML.Factory.CreateIExecution(executionTxt);
        IEnumerable<IExecution> execs = RML.ExecutionRepository.GetByRoute(route);

        IEnumerable<IExecution> matches = from e in execs
            where Equals(ex.Created, e.Created)
            && Equals(ex.Duration, e.Duration)
            && Equals(ex.Length, e.Length)
            && Equals(ex.RouteRecMode.Mode, e.RouteRecMode.Mode)
            && Equals(ex.TotalAscent, e.TotalAscent)
            && Equals(ex.TotalDescent, e.TotalDescent)
            && Equals(ex.Positions.Count(), e.Positions.Count())
            && Equals(ex.StartPosition.Latitude, e.StartPosition.Latitude)
            && Equals(ex.StartPosition.Longitude, e.StartPosition.Longitude)
            select e;

        if (Equals(matches.Count(), 0))
        {
            ex.RouteId = route.RouteId;
            ex.UserId = user.UserId;
            IExecution added = RML.ExecutionRepository.Add(ex);
            return new KeyValuePair<int, string>(added.ExecutionId, Language.ExecutionAdded);
        }

        return new KeyValuePair<int, string>(matches.First().ExecutionId, Language.↩
            ExecutionAlreadyExisted);
    }
    catch (ArgumentException)
    {
        return new KeyValuePair<int, string>(-1, Language.ExecutionError);
    }
}
```

Listagem de Código A.5: Inserção de execução em RouteManagementWebService

A.7 Listagem de procedimentos armazenados

```
[dbo].[AddComment]
    @PostId int,
    @UserId int,
    @PostDate datetime,
    @CommentText nvarchar(max),
    @return_value int output

[dbo].[AddExecution]
    @RouteId int,
    @UserId int,
    @RouteRecModeId int,
    @Created datetime,
    @StartLatitude decimal(10,8),
    @StartLongitude decimal(11,8),
    @StartAltitude decimal(6,2),
    @StartDate datetime,
    @EndLatitude decimal(10,8),
    @EndLongitude decimal(11,8),
    @EndAltitude decimal(6,2),
    @EndDate datetime,
    @Length decimal(12,2),
    @TotalAscent decimal(7,2),
    @TotalDescent decimal(7,2),
    @Positions xml,
    @return_value int output

[dbo].[AddGroup]
    @Name nvarchar(50),
    @Description nvarchar(max),
    @return_value int output

[dbo].[AddGroupUser]
    @GroupId int,
    @UserId int,
    @return_value int output

[dbo].[AddLogin]
    @UserId int,
    @Email nvarchar(50),
    @Password nvarchar(200),
    @return_value int output

[dbo].[AddPointOfInterest]
    @RouteId int,
    @Position nvarchar(170),
    @return_value int output

[dbo].[AddPositionExtraInfo]
    @PointOfInterestId int,
    @PositionExtraInfoTypeId int,
    @Info nvarchar(max),
```

```
    @return_value int output

[dbo].[AddPost]
    @RouteId int,
    @ExecutionId int,
    @UserId int,
    @PostDate datetime,
    @PostText nvarchar(max),
    @return_value int output


[dbo].[AddRoute]
    @Name nvarchar(100),
    @UserId int,
    @RouteRecModeId int,
    @Created datetime,
    @StartLatitude decimal(10,8),
    @StartLongitude decimal(11,8),
    @StartAltitude decimal(6,2),
    @StartDate datetime,
    @EndLatitude decimal(10,8),
    @EndLongitude decimal(11,8),
    @EndAltitude decimal(6,2),
    @EndDate datetime,
    @Length decimal(12,2),
    @TotalAscent decimal(7,2),
    @TotalDescent decimal(7,2),
    @Positions xml,
    @return_value int output

[dbo].[AddRouteRouteType]
    @RouteId int,
    @RouteTypeId int,
    @return_value int output

[dbo].[AddRouteRouteType]
    @RouteId int,
    @RouteTypeId int,
    @return_value int output

[dbo].[AddRouteType]
    @TypeName nvarchar(50),
    @Description nvarchar(max),
    @return_value int output

[dbo].[AddUser]
    @FirstName nvarchar(50),
    @LastName nvarchar(50),
    @Birthday datetime,
    @Email nvarchar(50),
    @Locale nvarchar(5),
    @Avatar nvarchar(250),
    @Gender bit,
    @Timezone int,
```

```
    @return_value int output

[dbo].[AddUserFbId]
    @UserId int,
    @FacebookId decimal(15,0),
    @return_value int output


[dbo].[CheckForEmail]
    @UserId int,
    @Email nvarchar(50),
    @return_value int output


[dbo].[DeleteComment]
    @CommentId int,
    @return_value int output

[dbo].[DeleteExecution]
    @ExecutionId int,
    @return_value int output

[dbo].[DeleteGroup]
    @GroupId int,
    @return_value int output

[dbo].[DeleteGroupUser]
    @GroupId int,
    @UserId int,
    @return_value int output

[dbo].[DeletePointOfInterest]
    @PointOfInterestId int,
    @return_value int output

[dbo].[DeletePost]
    @PostId int,
    @return_value int output

[dbo].[DeleteRoute]
    @RouteId int

[dbo].[DeleteUser]
    @UserId int,
    @return_value int output


[dbo].[GetComment]
    @CommentId int

[dbo].[GetCommentsByPostId]
    @PostId int

[dbo].[GetDefaultRecMode]
```

```
160
    [dbo].[GetExecutionById]
        @id int

165 [dbo].[GetExecutionPositionsXML]
        @ExecutionId int,
        @return_value xml output

    [dbo].[GetExecutionsByRouteId]
170     @id int

    [dbo].[GetExecutionsByUserId]
        @id int

175 [dbo].[GetFacebookIdByUser]
        @UserId int,
        @return_value decimal(15) output

    [dbo].[GetGroupById]
180     @GroupId int

    [dbo].[GetGroupMembersByGroupId]
        @GroupId int

185 [dbo].[GetGroups]

    [dbo].[GetPointsOfInterestByRouteId]
        @RouteId int

190 [dbo].[GetPositionExtraInfoTypes]

    [dbo].[GetPostById]
        @PostId int

195 [dbo].[GetPostByRouteId]
        @RouteId int

    [dbo].[GetPostByUserId]
        @UserId int
200
    [dbo].[GetRecModes]

    [dbo].[GetRouteBetweenPositions]
        @ALat decimal(10,8),
205     @ALon decimal(11,8),
        @BLat decimal(10,8),
        @BLon decimal(11,8)

    [dbo].[GetRouteById]
210     @id int

    [dbo].[GetRouteByUser]
        @userId int
```

```
[dbo].[GetRoutePositionsXML]
    @RouteId int,
    @return_value xml output

[dbo].[GetRouteType]
    @TypeName nvarchar(50)

[dbo].[GetRouteTypes]

[dbo].[GetRouteTypesByRouteId]
    @RouteId int

[dbo].[GetUserByEmail]
    @Email nvarchar(50)

[dbo].[GetUserByFacebookId]
    @UserFbId decimal(15,0)

[dbo].[GetUserById]
    @UserId int

[dbo].[GetUserGroups]
    @UserId int

[dbo].[TryAddPositionExtraInfoType]
    @Type nvarchar(50),
    @Description nvarchar(max),
    @return_value int output

[dbo].[UpdateComment]
    @CommentId int,
    @CommentText nvarchar(max),
    @return_value int output

[dbo].[UpdateExecution]
    @ExecutionId int,
    @RouteId int,
    @UserId int,
    @RouteRecModeId int,
    @Created datetime,
    @StartLatitude decimal(10,8),
    @StartLongitude decimal(11,8),
    @StartAltitude decimal(6,2),
    @StartDate datetime,
    @EndLatitude decimal(10,8),
    @EndLongitude decimal(11,8),
    @EndAltitude decimal(6,2),
    @EndDate datetime,
    @Length decimal(12,2),
    @TotalAscent decimal(7,2),
    @TotalDescent decimal(7,2),
    @Positions xml,
    @return_value int output
```

```
[dbo].[UpdateGroup]
    @GroupId int,
    @Name nvarchar(50),
    @Description nvarchar(max),
    @return_value int output

[dbo].[UpdateLogin]
    @UserId int,
    @Email nvarchar(50),
    @Password nvarchar(200),
    @return_value int output


[dbo].[UpdatePost]
    @PostId int,
    @PostText nvarchar(max),
    @return_value int output

[dbo].[UpdateRoute]
    @RouteId int,
    @Name nvarchar(100),
    @UserId int,
    @RouteRecModeId int,
    @Created datetime,
    @StartLatitude decimal(10,8),
    @StartLongitude decimal(11,8),
    @StartAltitude decimal(6,2),
    @StartDate datetime,
    @EndLatitude decimal(10,8),
    @EndLongitude decimal(11,8),
    @EndAltitude decimal(6,2),
    @EndDate datetime,
    @Length decimal(12,2),
    @TotalAscent decimal(7,2),
    @TotalDescent decimal(7,2),
    @Positions xml,
    @return_value int output

[dbo].[UpdateUser]
    @UserId int,
    @FirstName nvarchar(50),
    @LastName nvarchar(50),
    @Birthday datetime,
    @Email nvarchar(50),
    @Locale nvarchar(5),
    @Avatar nvarchar(250),
    @Gender bit,
    @Timezone int,
    @return_value int output

[dbo].[UpdateUserFbId]
    @UserId int,
    @FacebookId decimal(15,0),
```

```
    @return_value int output



[dbo].[ValidateUserAdmin]
    @UserId int,
    @return_value int output

[dbo].[ValidateUserLogin]
    @UserId int,
    @Email nvarchar(50),
    @Password nvarchar(200),
    @return_value int output
```

Listagem de Código A.6: Listagem de procedimentos armazenados

A.8 Amostra 2 - precisão máxima

Figura A.2: Amostra 2 - precisão máxima

A.9 Amostra 2 - distância mínima

Figura A.3: Amostra 2 - distância mínima

A.10 Amostra 2 - tempo mínimo

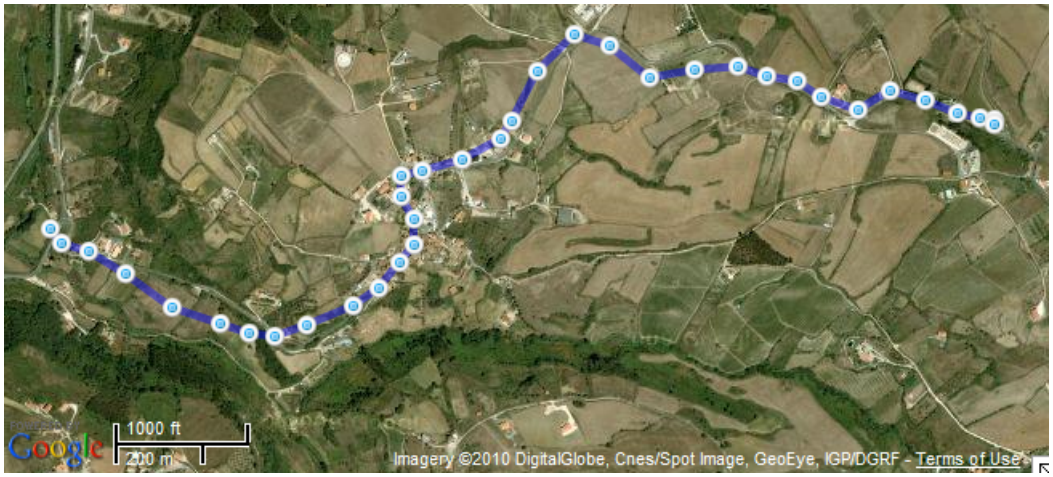


Figura A.4: Amostra 2 - tempo mínimo

A.11 Amostra 2 - adaptativo

Figura A.5: Amostra 2 - adaptativo

A.12 Amostra 3 - precisão máxima

Figura A.6: Amostra 3 - precisão máxima

A.13 Amostra 3 - distância mínima

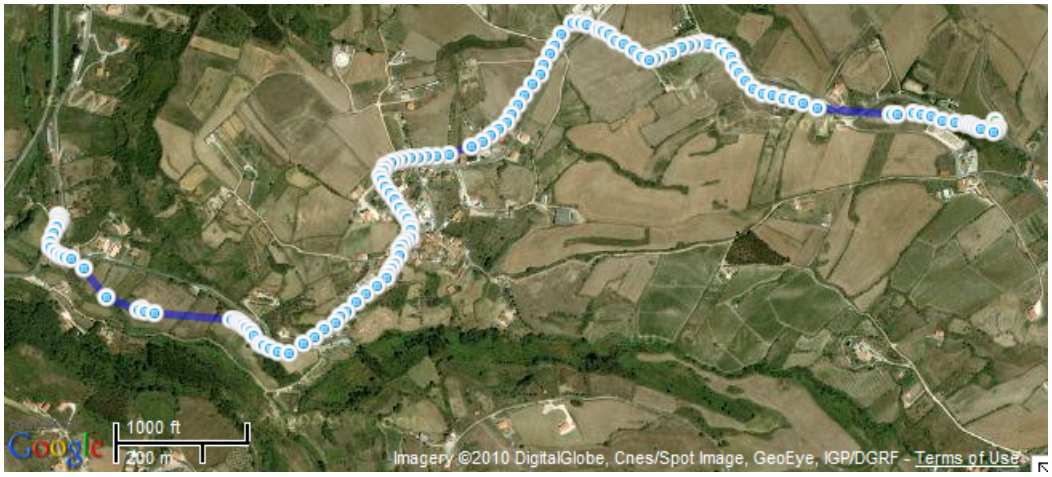


Figura A.7: Amostra 3 - distância mínima

A.14 Amostra 3 - tempo mínimo

Figura A.8: Amostra 3 - tempo mínimo

A.15 Amostra 3 - adaptativo

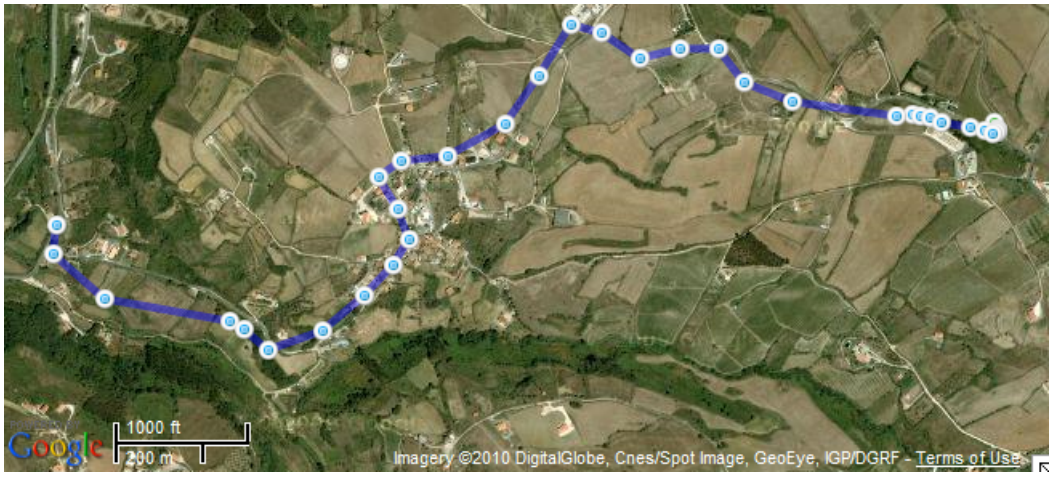


Figura A.9: Amostra 3 - adaptativo

A.16 Planeameto

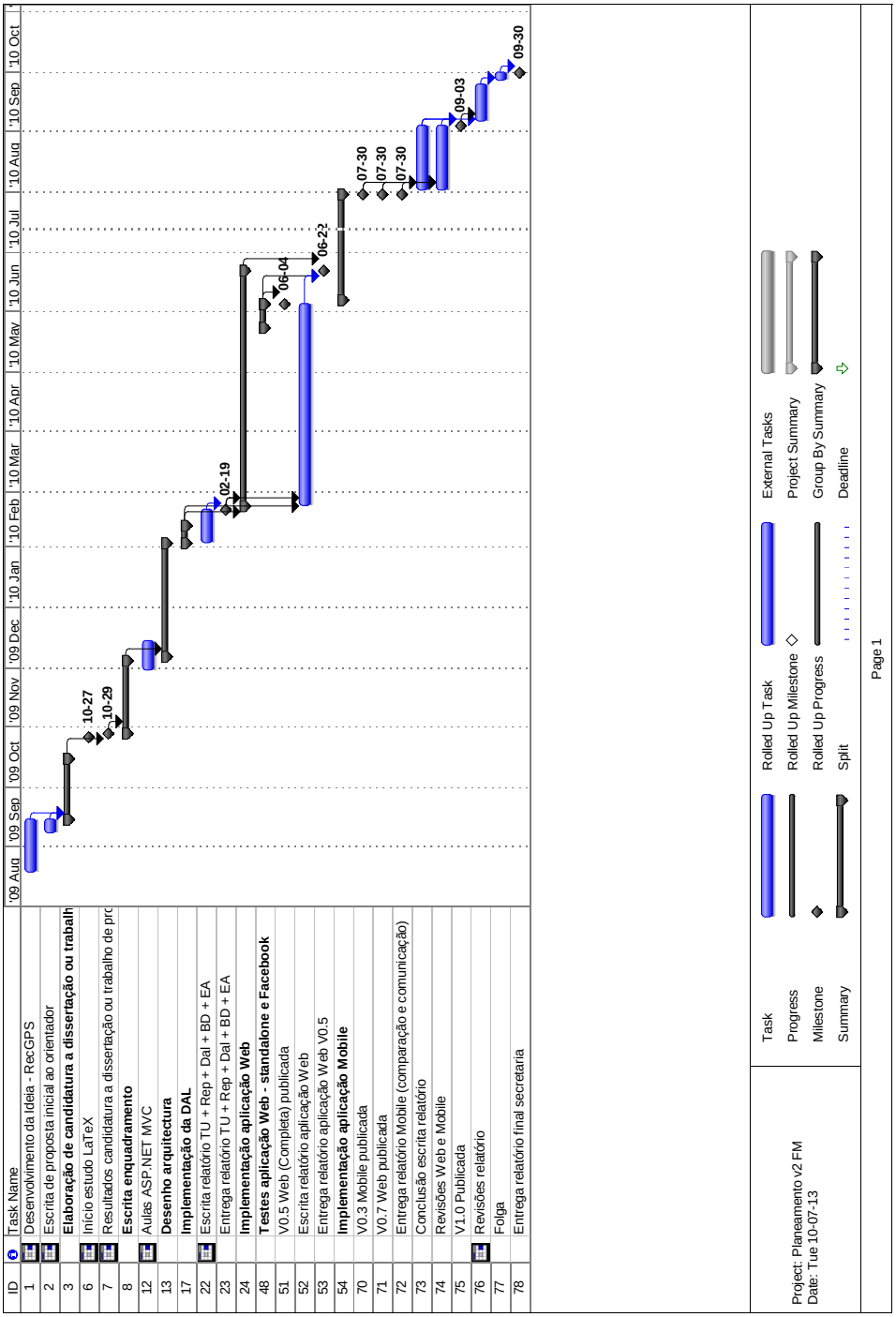


Figura A.10: Planeamento

A.17 CD de projecto

Referências

[1] Priberam Informática S.A. *Dicionário Priberam de Língua Portuguesa.* priberam.pt

[2] Microsoft Corporation. *GPS Intermediate Driver*, 2009. msdn.microsoft.com/en-us/library/bb202086.aspx

[3] Maarten Struys. *GPS Intermediate Driver to Retrieve Location Information*, July 2008. msdn.microsoft.com/en-us/netframework/cc719033.aspx

[4] Robert Lipe. *GPS Babel*, June 2010. gpsbabel.org

[5] Dan Foster. *GPX: The GPS Exchange Format*, January 2010. topografix.com/gpx.asp

[6] R.W. Sinnott. *Virtudes of the Haversine.* Sky and Telescope, 1984.

[7] Thaddeus Vicenty. *Direct and Inverse Solutions of Geodesics on the Ellipsoid with Application of Nested Equations.* Directorate of Overseas Surveys, April 1975. www.ngs.noaa.gov/PUBS_LIB/inverse.pdf

[8] Luis Espinosa. *TrackMe*, 2004. trackme.com

[9] Luis Espinosa. *TrackMe tutorial.* luisespinosa.com/trackme_eng.html

[10] Inc MapMyFitness. *MapMyWakl*, 2005. mapmywalk.com

[11] JMD Software Solutions. *Tracky*, 2009. trackthisout.com

[12] Marathon. *Marathon*, 2005. marathon-mobile.net

[13] Pdafun.net. *Global Navigator*, 2007. pdafun.net

[14] GPS Tuner Kft. *GPS Tuner Atlas*, 2009. gpstuner.com

[15] Klaus Bechtold. *GPSies*, 2009. gpsies.com

[16] Zone Five Software. *SportTracks*, 2009. zonefivesoftware.com/SportTracks

[17] GlobalMotion Media Inc. *EveryTrail*, 2009. www.everytrail.com

[18] Steven Sanderson. *Pro ASP.NET MVC Framework*. Apress, April 2009.

[19] Martin Fowler. *Patterns of Enterprise Application Architecture*. Addison Wesley, November 2002.

[20] Google. *Google Maps*. maps.google.com

[21] Microsoft. *Bing Maps*. bing.com/maps

[22] Yahoo! Inc. *Yahoo! LOCAL Maps*. maps.yahoo.com

[23] MediaWiki. *Open Street Map*. openstreetmap.org

[24] SAPO. *SAPO Mapas*. mapas.sapo.pt

[25] Mark Zuckerberg. *An Open Letter from Facebook Founder Mark Zuckerberg*, December 2009. blog.facebook.com/blog.php?post=190423927130

[26] DanielSchaffer. *.NET Facebook API Client*, December 2009. facebook.codeplex.com/

[27] Google. *Google Maps API Family*, June 2010. code.google.com/apis/maps/index.html

[28] Google. *Google Maps API Reference*, March 2010. code.google.com/apis/maps/documentation/reference.html

[29] Facebook. *Facebook Javascript SDK*, April 2010. developers.facebook.com/docs/reference/javascript/

[30] W3C. *XHTML2 Working Group Home Page*, April 2010. www.w3.org/MarkUp/

[31] Facebook. *Facebook XFBML*, April 2010. wiki.developers.facebook.com/index.php/XFBML

[32] Facebook. *Cross Domain Communication*, April 2010.
wiki.developers.facebook.com/index.php/Cross_Domain_Communication

[33] JSON. *JSON*, March 2010. www.json.org/

[34] D. Crockford. *JSON rfc*, March 2010.
www.ietf.org/rfc/rfc4627.txt?number=4627

[35] JSON. *JSON in JavaScript*, March 2010. www.json.org/js.html

[36] Google. *Google AJAX API Sign Up*, March 2010.
code.google.com/apis/maps/signup.html

[37] Google. *Google AJAX API Loader*, March 2010.
code.google.com/apis/maps/documentation/index.html#AJAX_Loader

[38] Google. *Google Maps API Concepts*, August 2010.
code.google.com/apis/maps/documentation/javascript/v2/basics.html

[39] Google. *Google Maps API Events*, March 2010.
code.google.com/apis/maps/documentation/events.html

[40] Facebook. *Facebook Connect*, April 2010.
wiki.developers.facebook.com/index.php/Facebook_Connect

[41] Facebook. *Facebook Connect Site Set Up*, April 2010.
wiki.developers.facebook.com/index.php/Connect/Setting_Up_Your_Site

[42] Jigar Desai. *Authorization in ASP.Net MVC using XML Configuration.*, March 2010.
www.jigar.net/articles/viewhtmlcontent324.aspx

[43] Facebook. *Facebook for Mobile Apps*, July 2010.
developers.facebook.com/docs/guides/mobile

[44] Microsoft. *WebBrowser Class*, July 2010.
msdn.microsoft.com/en-us/library/5d67hf8a.aspx

[45] Facebook. *Facebook Graph API*, July 2010.
developers.facebook.com/docs/api

[46] Microsoft. *FakeGPS*, September 2010.
msdn.microsoft.com/en-us/library/bb158722.aspx

[47] RaceChrono Oy. *Race Chrono*, September 2010. racechrono.com

[48] Nokia. *Nokia 5800 XpressMusic*, September 2010.
nokia.pt/produtos/todos-os-modelos/nokia-5800-xpressmusic

[49] National Marine Electronics Association. *NMEA 0183 Standard*, 2009.
www.nmea.org/content/nmea_standards/nmea_083_v_400.asp

[50] Dale DePriest. *NMEA Data*, September 2010.
gpsinformation.org/dale/nmea.htm

[51] Klaus Bechtold. *GPSies Statistics*, September 2009. gpsies.com/analytics.do