Mestrado em Engenharia Informática
Estágio
Relatório Final

Desenvolvimento de Aplicativo Móvel para Definição de Rotas Turísticas

Bruno Jorge Rodrigues Castanheira
bjrod@student.dei.uc.pt

Orientador:
Pedro Abreu
Data: 4 de Setembro de 2013

Co-Orientador:
Daniel Silva
Data: 4 de Setembro de 2013

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

O território patrimonial português é bastante descentralizado, dificultando a elaboração manual de guias turísticos e a divulgação de pólos atractivos de caminhadas e passeios de btt. Estes problemas podem ser minimizados através de aplicações móveis que permitem registar diversas informações sobre os locais e divulga-las nas redes sociais que possuem milhões de utilizadores.

Neste estágio implementou-se um sistema que tira partido das tecnologias de dispositivos móveis para recolher dados de pontos de interesse e que pretende aumentar a sua divulgação nas redes sociais. Os dados dos utilizadores anónimos permitem ainda criar uma fonte de dados detalhada dos actuais pontos de interesse e das rotas utilizadas, sendo complementada por eventos inseridos através de uma aplicação web. Todos os dados anónimos podem disponibilizados para outros utilizadores através da aplicação móvel andoid.

O sistema é constituído por uma aplicação android com boa usabilidade, um servidor e uma aplicação desktop. Aplicação android tem algumas semelhanças com outras já existentes pois permite o registo de localização, tirar fotografias e notas sobre pontos de interesse, sendo possível a sua divulgação nas redes sociais. Também melhora alguns aspectos das existentes tais como: a não violação de privacidade do utilizador, o envio de dados para um servidor seguro e a minimização de custos para o utilizador. Os dados recebidos no servidor são validados pela autarquia e permitem criar um mapeamento das zonas turísticas e fornecer aos utilizadores uma função pesquisa de rotas, pontos de interesse e eventos. A aplicação desktop pode ser utilizada para criar cópias dos dados das rotas e pontos de interesse e também permite gerar diversos gráficos dos dados das rotas.

O concelho de Amarante será a primeira região a acolher este sistema pois contém diversos pontos de interesse dispersos pelo seu enorme território. Esta contém também vários pontos de aceso WiFi nas suas localidades permitindo aos utilizadores não terem custos no acesso à internet e descarregarem facilmente aplicação. Através da ligação WiFi poderão também partilhar os pontos de interesse registados, enviar os dados para o servidor, ou aceder a uma funcionalidade que permite visualizar pontos de interesse, eventos e rotas nas redondezas do utilizador.

Através de inquéritos averiguou-se a usabilidade da aplicação android. Os resultados destes inquéritos revelaram que os utilizadores ficaram satisfeitos com a usabilidade da aplicação mas que algumas interações com a interface gráfica eram dificeis de detectar sem descrição prévia da aplicação. Os utilizadores também afirmaram que recomendariam aplicação aos seus amigos e conhecidos. Os testes de performance da aplicação Android não reveleram problemas de memória ou processamento. Aplicação web foi testada em diversos browsers e verificou-se que todas as funcionalidades e elementos gráficos funcionaram correctamente.

Palavras-Chave

Aplicação móvel, Localização, Pontos de Interesse ,Turismo, Redes Sociais, Sistema Operativo Android

Acrónimos

ACID - Atomicidade, Consistência, Isolamento e Durabilidade

AJAX - Asynchronous JavaScript and XML

Apps - Aplicação Android

CSS - Cascading Style Sheets

CSV - Comma-separated values

GPS - Global Positioning System

GPX - GPS eXchange Format

HTTP - Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure

IP - Internet Protocol address

JavaEE - Java Platform, Enterprise Edition

JPA - Java Persistence API

JSON - JavaScript Object Notation

JSP - JavaServer Pages

KML - Keyhole Markup Language

OSM - OpenStreetMaps

POI - Ponto de interesse

POST - Método HTTP

REST - Representational state transfer

SDK - Software development kit

SOAP - Simple Object Access Protocol

USB - Universal Serial Bus

WiFi - Wireless Fidelity, wireless internet

WTK - Well-known text

XML - Extensible Markup Language

Índice

Figuras

Tabelas

1. Introdução

A maioria dos telemóveis utilizados actualmente, denominados *smartphones*, permitem capturar imagens e vídeos, aceder à internet e descobrir a localização do utilizador. Para explorar as suas potencialidades foi necessário desenvolver novos sistemas operativos e programas. O sistema operativo mais utilizado a nível mundial é o Android [1]. Este sistema baseado em Linux foi criado pela Google, que é também responsável pelo seu contínuo aperfeiçoamento. O Android permite o acesso aos vários recursos do *smartphone* através de programas denominados apps (aplicações). Estas podem ser descarregadas da internet através de uma plataforma denominada Google Play que pode ser acedida via navegador de internet ou através da aplicação com o mesmo nome. Esta plataforma garante que todas as suas aplicações não contêm qualquer tipo de vírus, no entanto existem inúmeras páginas da internet onde se pode descarregar as mesmas aplicações mas correndo o risco de estarem infectads por virus. As apps são executadas isoladamente de todos os recursos do telemóvel, tais como câmara fotográfica, para garantir a segurança e privacidade. Caso as apps necessitem desses recursos então o seu acesso será requisitado na sua instalação e será o utilizador a decidir se autoriza o acesso. Algumas apps permitem recolher informação dos locais em que o utilizador se encontra através dos sistemas de geolocalização e da câmara fotográfica. Através da conexão à internet as apps podem partilhar os seus dados nas redes sociais para divulgar os POIs (Pontos de interesse). Esta conecção também permite enviar os dados para um servidor, criando assim um mapa de POIs, no entanto muitos destes servidores utilizam a informação pessoal dos utilizadores para fins monetários.

Neste estágio implementou-se um sistema composto por uma aplicação móvel android, um servidor e uma aplicação desktop. O sistema permite recolher informação sobre os pontos de interesse e divulgar as zonas turísticas rurais do nosso país através dos utilizadores e das tecnologias móveis com grande foco na boa usabilidade, minimização dos custos do utilizador e longevidade da bateria dos dispositivos. O sistema final resulta de uma análise de aplicações móveis e das tecnologias existentes. Nestas aplicações analisou-se as funcionalidades, fraquezas e as respectivas interfaces gráficas seguindo-se a criação de um protótipo da interface gráfica focado na usabilidade. Posteriormente foi projectada a arquitectura de todo o sistema e a metodologia para a sua implementação. A autarquia de Amarante possui vários pontos de interesse dispersos pelo seu extenso território tornando-a candidata preferencial para a instalação do sistema. Também possui vários pontos de acesso WiFi permitindo ao utilizador não ter custos nas comunicações com a internet e descarregar aplicação. A aplicação Android foi testada por utilizadores anónimos para averiguar as suas opiniões sobre a usabilidade e averiguar a compatibilidade da aplicação com diferentes versões Android. Efectuou-se também testes de *stress* para detectar a estabilidade da aplicação. Para averiguar a longevidade da bateria foram efetuadas várias rotas com o ecrã desligado e com ecrã ligado. Estas rotas também permitiram averiguar a qualidade do sinal em meios rurais.

[1] Para mais informações consultar: http://techcrunch.com/2012/11/02/idc-android-market-share-reached-75-worldwide-in-q3-2012/

1.1. Contexto

Os *smartphones* possuem diversos mecanismos que permitem recolher informação sobre POIs muito rapidamente. Estes dados podem ser utilizados para diversos fins: divulgação de locais, criação de mapa de POIs, estatísticas sobre os POIs e rotas utilizadas.

Após análise de várias aplicações verificou-se que é possível registar a localização através do sistema de coordenadas, fotografias e descrições dos POIs. Finalmente através da internet móvel é possível partilhar estes dados nas redes sociais. No entanto estas aplicações não permitem enviar os dados para servidores que garantam a privacidade do utilizador e muitas apresentam problemas de privacidade na sua execução. Algumas aplicações também só possibilitam o acesso a todas as funções mediante um pagamento prévio mas na aplicação implementada todas as funcionalidades são gratuitas.

Os dados recolhidos pelos utilizadores são bastante úteis pois permitem criar um mapa de POIs. Assim, foi realizado um estudo para criar um servidor e de como conecta-lo à aplicação. Foi também realizado um estudo sobre possíveis métodos de comunicação com servidores. Explorou-se a possibilidade de utilizar SMS para envio de dados de texto mas estes sistemas eram bastante limitados devido a permitirem que o servidor envia-se no máximo 90 SMS mensais, um valor reduzido tendo em conta quantidade de dados que se quer trocar e de uma SMS só permitir um máximo de 160 caracteres. Em alternativa utilizou-se a conectividade à internet e por isso foi realizado um estudo sobre os tarifários disponíveis nas redes nacionais. No anexo **Erro! A origem da referência não foi encontrada.** pode consultar os detalhes dos tarifários. Após este estudo verificou-se que não há grande diferença entre as ofertas das operadoras, portanto a escolha da operadora não implica grandes diferenças de custos para os utilizadores.

O sistema será instalado na Câmara Municipal de Amarante. Esta zona possui um grande número de atracções turísticas e diversos locais para actividades ao ar livre distribuídos ao longo de várias freguesias de grandes extensões, tornando assim muito difícil construir um mapa turístico para todos esses elementos. Amarante também possui diversos pontos de acesso *wireless*, permitindo que as trocas de dados entre o *smartphone* e o servidor sejam completamente grátis. Autarquia disponibilizou um *smartphone* Android para desenvolver aplicação para o mesmo dispositivo.

1.2. Motivação

Portugal é um país bastante extenso com diversos POIs muito distribuídos. Esta dispersão dificulta a elaboração de guias e mapas e a divulgação dos POIs. Com recurso aos *smartphones* é possível recolher diversas informações sobre os POIs e partilha-las nas redes sociais contribuindo para a sua divulgação.

As aplicações existentes possuem problemas de privacidade na sua execução e não permitem envio dos dados para servidores que também garantam privacidade. Muitas das aplicações também apresentam problemas na interface gráfica sobretudo na usabilidade e poucas redes sociais para partilha de dados.

Neste estágio implementou-se um sistema composto por uma aplicação Android, um servidor e uma aplicação desktop. Espera-se que aplicação Android possua uma interface gráfica visualmente atractiva e de boa usabilidade. Contém funções de registo de localização, captura de fotografias e partilha dos dados nas redes sociais e no servidor da autarquia. A aplicação não contém problemas de privacidade. O servidor agrega os dados, permitindo à autarquia criar mapas dos POIs e guias das rotas, visualizar estatística. O servidor permite

inserção de dados sobre eventos e outras informações da autarquia, fornecendo assim aos utilizadores com informações que possam despertar o seu interesse em determinados locais. Aplicação desktop permite criar uma cópia dos dados. Após os dados estarem na aplicação desktop o utilizador pode aceder gráficos que permitem analisar as rotas.

Com este sistema espera-se aumentar a divulgação dos pontos de interesse e criar uma base de dados com rotas, pontos de interesse e eventos que possam ser utilizados pela autarquia e pelos utilizadores. Deste modo os utilizadores terão acesso a diversas informações sobre os locais enquanto a autarquia poderá identificar a origem dos seus turistas e os seus locais preferidos.

1.3. Planeamento

Para planear o trabalho ao longo do estágio foi adoptada uma metodologia de desenvolvimento de *software*. A metodologia utilizada é um aspecto importante pois permite implementar processos que possibilitam uma boa gestão do projecto. Existem várias metodologias para o desenvolvimento de *software*: modelo cascata, modelo espiral, desenvolvimento iterativo e incremental, desenvolvimento ágil, desenvolvimento rápido de aplicações.

Neste estágio utilizou-se a metodologia de desenvolvimento iterativo e incremental sendo esta abordagem acordada com os orientadores de estágio.

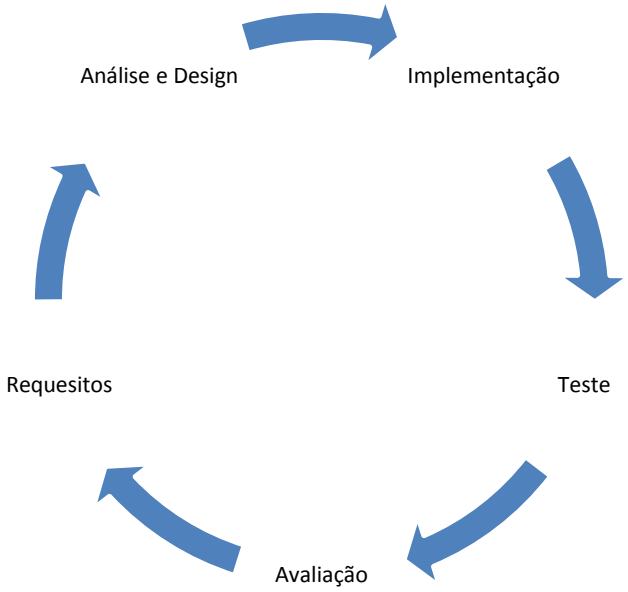

Figura 1 - Ciclo de desenvolvimento iterativo e incremental

A fase inicial do Desenvolvimento Iterativo e Incremental envolve planeamento e levantamento dos requisitos do sistema a desenvolver ao que se segue a análise dos requisitos e definição da arquitectura do sistema. Após a definição da arquitectura procede-se ao seu desenvolvimento através de iterações. Cada iteração passa por várias fases de implementação e testes, sendo que um conjunto de iterações leva à construção dos

requisitos. Concluído o desenvolvimento procede-se a testes e avaliação das funcionalidades e por último a instalação das aplicações.

Optou-se por esta abordagem com iterações de uma semana para diminuir o risco do aparecimento e propagação de erros, a necessidade de mudar os requisitos nas fases finais do projecto, que levariam a um custo elevado em termos de tempo de implementação e qualidade do sistema.

Após a definição da metodologia foi possível planear o trabalho para os dois semestres do estágio. No primeiro semestre realizou-se uma análise de requisitos para definir os objectivos e funcionalidades a implementar. Após esta fase inicial realizou-se um estudo das aplicações existentes na área do turismo e desporto que permitissem registar localização dos utilizadores. Este estudo foi realizado no emulador devido a na altura não possuir um dispositivo Android. O emulador não permite acesso ao Google Play e assim sendo procurou-se alternativas legais nomeadamente aceder à página dos criadores das aplicações. Através desta única alternativa disponível foi possível testar 30% das aplicações que se pretendia analisar. Em simultâneo foi realizado um estudo dos pacotes de dados móveis. De seguida realizou-se um estudo sobre Interacção Humano Computador para posteriormente realizar uma análise das interfaces existentes nas aplicações Android. Após esta análise foi criado um protótipo para interface gráfica da aplicação Android. Durante o primeiro semestre também foram realizados vários testes de funcionalidades em Android. Finalmente foi definida arquitectura e a metodologia da implementação do sistema. No segundo semestre iniciou-se a implementação do sistema. Inicialmente implementou-se as funcionalidades que permitiam registo da localização, gravação de fotografias e partilha nas redes sociais dos dados dos pontos de interesse. Finda esta etapa implementou-se a base de dados do servidor e a comunicação com o cliente Android. De seguida iniciou-se o desenvolvimento da aplicação desktop, implementando a comunicação com servidor e visualização das rotas e pontos de interesse. Após este desenvolvimento isolado de cada aplicação procedeu-se a um desenvolvimento simultâneo das três aplicações implementando as restantes funcionalidades. Em seguida pode-se observar a relação entre o tempo planeado para cada aplicação e o tempo realmente utilizado. O tempo é expresso em semanas:

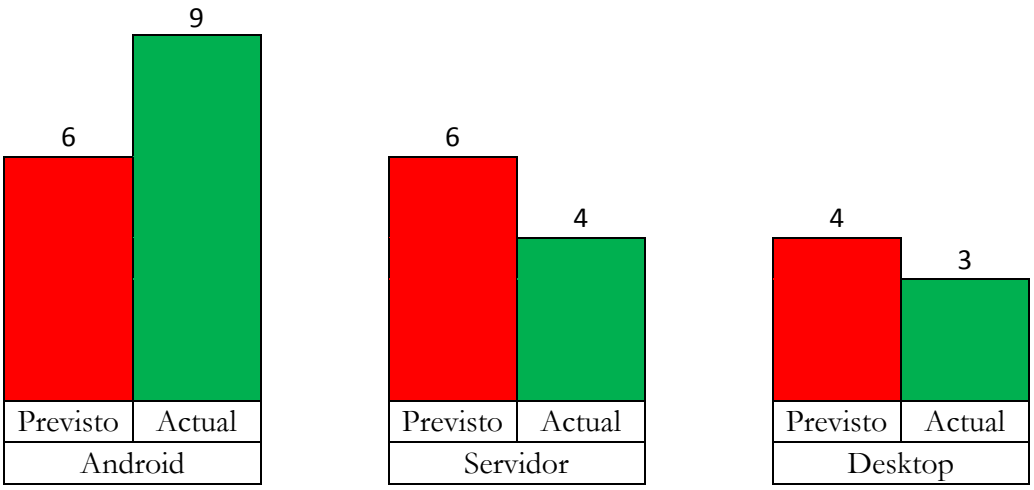

Tabela 1 – Comparação dos tempos previstos e actuais

O desenvolvimento Android demorou mais do que o esperado devido a ser uma nova tecnologia para o criador do sistema. As ferramentas de desenvolvimento de interfaces gráficas para Android são pouco maduras quando comparadas com as da tecnologia JavaEE ou web, resultando num desenvolvimento mais lento. As documentações das redes sociais contêm poucos exemplos práticos contribuindo também para esse desenvolvimento mais lento. O desenvolvimento das aplicações do servidor e desktop demoraram menos tempo do que o esperado devido a possuírem ferramentas de trabalho bastante maduras e uma documentação mais detalhada com imensos exemplos práticos. O criador do sistema também já possuía experiência com diversas tecnologias utilizadas na aplicação do servidor e desktop

1.4. Estrutura do relatório

Neste relatório poderá consultar na secção 2 um estudo de arte de aplicações com funções iguais às pretendidas, na secção 3 uma descrição de toda arquitectura utilizada na implementação da solução apresentada, na secção 4 todos os testes necessários para validação dos requisitos e análise da usabilidade, na secção 5 os problemas encontrados durante o estágio assim como as conclusões do projecto e as várias propostas para melhoramente do sistema.

2. Estado da arte

Existem várias aplicações que desempenham algumas funções semelhantes às propostas no estágio. Assim, realizou-se uma análise às suas funcionalidades, às tecnologias usadas, às permissões usadas, e às suas interfaces gráficas. A análise da interface gráfica foi realizada através da guias de (Ayob et al, 2009), (Shneiderman), (W3C, 2008), (Google, 2013).

O Android possui vários elementos[2], *action bar* representado na Figura 2 e o gesto *swipe* ilustrado na Figura 3, que permitem atingir vários objectivos definidos pelos guias referidas anteriormente. A *action bar* contém atalhos que permitem aceder às funções mais relevantes nos diferentes ecrãs e a informações sobre o ecrã em que o utilizador se encontra. Também contém um atalho que permite navegar verticalmente nas aplicações. O gesto *swipe* melhora a navegação horizontal das aplicações. Para executar este gesto basta pressionar o ecrã ligeiramente e simultaneamente mover o dedo na horizontal. Após análise das aplicações foram sugeridas melhorias para algumas interfaces gráficas e criadas duas tabelas que resumem as funcionalidades e a qualidade das interfaces gráficas.

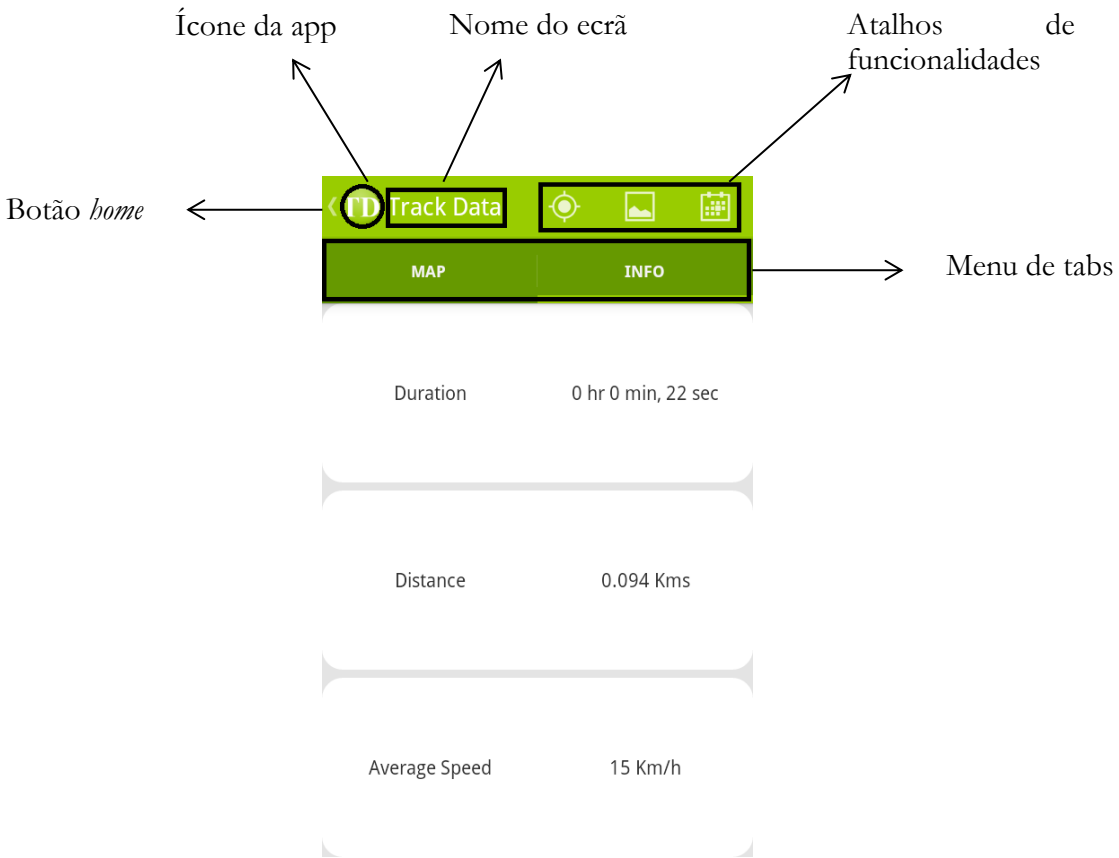

Figura 2 – Interface gráfica com diversos elementos de uma action bar

[2] Para mais informações consultar: http://developer.android.com/design/patterns/index.html

Figura 3 – Gesto Swipe

2.1. Maverick

A localização pode-se obter através do sistema GPS (Global Positioning System), WiFi (Wireless Fidelity, wireless internet) e rede móvel. A aplicação também permite armazenar as rotas assim como pontos de interesse com descrição e fotografias (figura 5) para além das coordenadas Na versão grátis a quantidade de pontos que se pode registar está muito limitado tornando-a pouco útil. O utilizador pode escolher entre os mapas do Bing ou do OpenStreetMaps para visualizar as rotas (figura 4) e envia-las para a página GPSspies.com. Também é possível exportar as rotas para os formatos KML (Keyhole Markup Language) e GPX (GPS eXchange Format). Apresenta boa usabilidade pois a sua utilização é intuitiva e a transição entre ecrãs é fluida (sem atrasos). Os elementos na interface gráfica estão bem integrados entre si permitindo aproveitar todo o espaço disponível do ecrã sem nunca o tornar muito confuso. A interface gráfica também possui uma *action bar* que disponibiliza informação relevante ao utilizador consoante o ecrã em que se encontra. Os ecrãs estão organizados hierarquicamente por dois níveis e raramente existem ecrãs irmãos (no mesmo nível). Para aceder ao topo da hierarquia (transitar do nível 2 para o nível 1) não há necessidade do botão *up* pois pode-se usar o botão *back* para o mesmo efeito. Nos ecrãs irmãos a navegação é feita através do gesto *swipe* e existe visualmente uma notificação que indica o ecrã em que o utilizador se encontra. Para aceder às opções basta pressionar o botão de opções do telemóvel podendo assim definir o intervalo e distância entre pontos a registar automaticamente assim como aceder a várias outras opções. Nas permissões apenas é requerido o acesso a recursos indispensáveis para as funções a que a aplicação se propõe desempenhar.

Figura 4 – Criação de um ponto de interesse (Maverick)

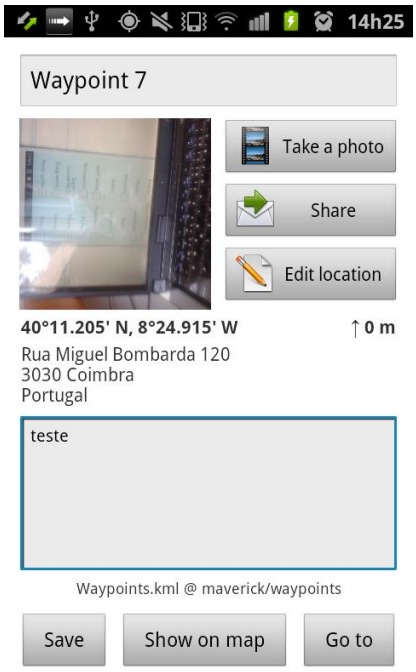

Figura 5 – Visualização da rota em tempo real (Maverick)

2.2. GPS Essentials 3.0.8

Esta aplicação permite obter localização através dos sistemas GPS, WiFi e redes móveis, armazenar rotas e visualizar rotas antigas num mapa sem necessitar de ligação à rede, tirar fotos às quais é possível adicionar descrições aos locais posteriormente. Os dados das rotas podem ser exportados tanto no formato GPX como no formato KML apenas localmente evitando assim problemas de privacidade caso se envie os dados a terceiros. As fotos podem ser posteriormente visualizadas numa galeria assim como os pontos de interesse. A interface gráfica é intuitiva (figura 6) pois permite facilmente identificar os gestos disponíveis e as funções de todos os elementos do ecrã. Também é configurável permitindo facilmente aceder às funções e ajustar os dados que se quer visualizar. A interface gráfica está hierarquicamente organizada num máximo de 3 níveis pelo que deveria ser implementada uma pequena *action bar* com o botão *up* para permitir uma navegação vertical mais rápida de acordo com os padrões de UI estabelecidos pela (Google, 2013). No ecrã de registo de localização existe um botão que permite alternar entre os valores actuais e a lista de rotas que poderia ser substituído por uma *tab* permitindo assim mais facilmente revelar a sua função ao utilizador, outra solução poderia ser a integração de dois botões numa *action bar*. A aplicação permite ao utilizador definir o tempo mínimo entre actualizações da localização mas não é dada a informação que o desgaste da bateria aumentará caso o tempo mínimo seja diminuído. Esta aplicação também contém várias funções extra que aumentam consumo da bateria se forem usadas constantemente. O rápido desgaste da bateria limita o uso desta aplicação para passeios curtos ou para passeios longos se apenas usar as funções essenciais de registo de localização e fotografias. Esta aplicação peca também pela publicidade apresentada em vários menus. Embora as permissões contenham todas as necessárias para o uso total da aplicação contém permissões extra que podem causar problemas de privacidade do utilizador: ler estado e identidade do telefone, enviar transmissão persistente através de sockets.

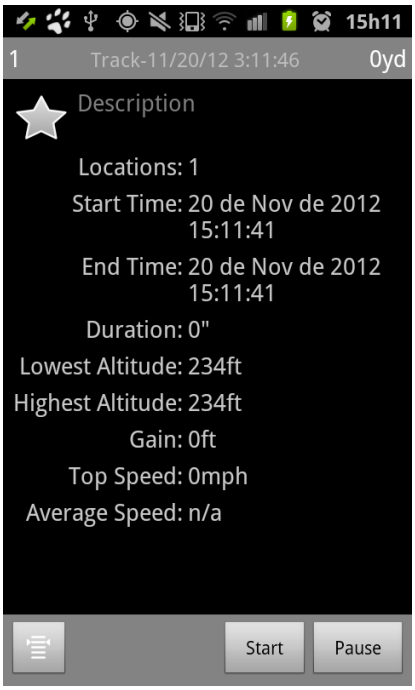

Figura 6 – Ecrã principal de registo da rota, não configurável (GPS Essentials)

2.3. GPSLogger_23

Uma aplicação bastante simples, com poucas funções extra, permite registar localização através de GPS, WiFi e redes móveis mas não permite tirar nem fotos nem notas sobre os locais visitados, nem é possível visualizar a rota embora possam ser exportadas para GPX ou KML. Esta aplicação também tem alguns extras tais como exportar os ficheiros GPX e KML para o Dropbox, OpenStreetMaps, GoogleDocs e OpenGTS cabendo ao utilizador decidir se os quer usar. Caso o utilizador não use estas opções de exportação a aplicação nunca se ligará à rede evitando consumo de tráfego. Esta aplicação contém uma interface gráfica bastante simples (figura 7) pois apresenta poucos botões e mostra apenas informação relevante. A navegação possui apenas um nível hierárquico permitindo aceder facilmente às funções a que se propõe realizar. Os menus de opções são acedidos através do botão opções do telemóvel e são bastante simples oferecendo ao utilizador as opções que realmente lhe interessa activar tais como formato no qual se grava a rota localmente, ajustar o tempo ou distância no qual é registada uma posição. A interface possui uma *action bar* que não aproveita as suas potencialidades pois é apenas utilizada para mostrar o nome da aplicação e não contém informação importante que está depois a ser mostrada no fundo do ecrã. Esta aplicação contém um problema nas permissões pois permite criar localizações fictícias, que poderão ser utilizadas por aplicações maliciosas para adulterar a localização do utilizador. A simplicidade desta aplicação assim como a possibilidade de não ligar à rede permanentemente permite preservar muita bateria no entanto é preciso ter cuidado com os valores que se inserem nas opções de intervalo de tempo da actualização da posição pois uma má configuração levará ao rápido esgotamento da bateria.

Figura 7 – Ecrã principal de registo da rota (GPS Logger)

2.4. OSMTrackerAndroid 0.6.3

A aplicação permite registar a localização, gravar rotas, tirar fotos e adicionar descrições, vocalmente ou por escrito, a pontos de interesse. No entanto não é possível registar os tipos de dados num único ponto de interesse nem definir o tipo de local onde o utilizador se encontra a cada momento. As rotas podem ser exportadas para o formato GPX e visualizadas nos mapas do OpenStreetMaps caso esteja conectado à rede ou através de *caching*. A sua interface gráfica permite aceder a todas as opções necessárias durante o registo da localização, mas a sua organização pode tornar-se confusa devido a conter demasiadas opções apresentadas num ecrã (figura 8). Esta situação é causada pelo formato em grelha que é constantemente alterado sempre que se escolhe uma opção e em que o submenu é agregado ao ecrã principal. Seria mais fácil visualizar e navegar se fosse implementada uma hierarquia com vários níveis e implementar o botão *up* para navegação vertical. As opções são acedidas através do botão de opções do telemóvel e nestes menus também existe o problema de agregação dos submenus com o menu principal. As permissões requerem acesso aos recursos necessários para o seu funcionamento mas contém permissões extra que podem violar a privacidade do utilizador: ler estado e identidade do telefone, enviar transmissão persistente através de sockets.

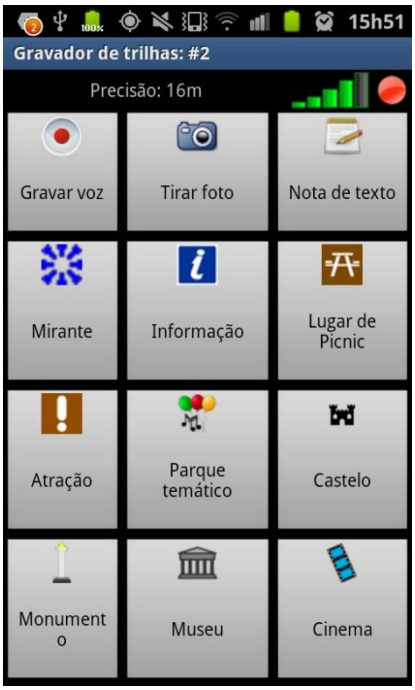

Figura 8 – Menu principal (OSMTrackerAndroid)

2.5. OruxMaps

A aplicação permite registar as localizações através de GPS e registar pontos de interesse com fotos e posteriormente adicionar descrições. As rotas podem ser visualizadas através de vários tipos de mapas *online* e os seus dados podem ser alterados. Também podem ser exportadas para formato KML e GPX e enviadas para várias páginas e até mesmo enviar para o correio electrónico do utilizador. Aplicação contém uma grande variedade de opções e permite ajustar o tempo entre as actualizações da posição podendo assim reduzir o consumo da bateria. Apresenta uma interface gráfica visualmente atractiva (figura 9) embora o menu de registo da posição contenha muitos atalhos e o mapa esteja com *zoom* ao mínimo quando é acedido na primeira vez de cada nova rota. Possui também uma *action bar* que se ajusta consoante o ecrã que o utilizador dando ao utilizador atalhos para as funções possivelmente mais usadas em cada ecrã. A interface gráfica está hierarquicamente organizada em dois níveis pelo que é dispensável a implementação do botão *up* visto que a navegação na vertical pode ser realizada pelo botão *back*. Contém ainda um pequeno tutorial que explica alguns dos botões que no entanto é insuficiente para resolver os problemas de navegação. Nas opções é possível definir o intervalo e distância entre pontos a registar automaticamente assim como aceder a várias outras opções. Nas permissões apenas é requerido o acesso a recursos indispensáveis para as funções a que a aplicação se propõe desempenhar.

Figura 9 – Visualização da rota e dados da rota (OruxMaps)

2.6. MyTracks

Esta aplicação permite registar a localização através de GPS, WiFi e rede móvel e no final é possível adicionar uma descrição à rota assim como adicionar o nome da actividade que foi realizada. Também é possível registar POIs (pontos de interesse) e pode-se adicionar o nome, tipo de POI e uma descrição. Não é possível visualizar as coordenadas sendo que apenas podemos visualizar no mapa a localização (figura 10). Os dados da rota podem ser exportados para GPX, KML e CSV (Comma-separated values) assim como podem ser enviados para o GoogleMaps e Google Docs. Aplicação também permite divulgar as rotas através do Facebook, Twitter e Google+. É também possível sincronizar o *smartphone* com outros dispositivos móveis através de *Bluetooth*, no entanto esta tecnologia desgasta bastante a bateria. A interface gráfica é bastante simples e intuitiva através de uma navegação por *tabs* entre os ecrãs principais. Tem grande variedade de opções, acedidas pelo botão de opções do telemóvel, inclusive configurar o intervalo de tempo em que se regista os locais assim como as suas distâncias. As permissões requerem acesso aos recursos necessários para o seu funcionamento mas existem permissões extra que podem violar a privacidade do utilizador: o acesso a chamadas telefónicas e todos os respectivos dados.

Figura 10 – Visualização da rota (MyTracks)

2.7. SportsTracker

Esta aplicação permite o registo dos trajectos realizados em desportos através do sistema GPS. É possível definir o tipo de actividade e tirar fotos não georreferenciadas. As rotas podem ser visualizadas através de um mapa e as respectivas fotos. A interface gráfica é composta hierarquicamente por 2 ou 3 níveis e é possível navegar rapidamente e verticalmente na hierarquia através do botão *back*. A navegação horizontal é realizada através do gesto *swipe* e de uma pequena notificação a indicar o número do ecrã que o utilizador se encontra. A interface gráfica possui ainda uma *action bar* com as duas acções mais importantes para além do registo de localização (figura 11). Os dados não podem ser exportados localmente mas podem ser partilhados socialmente para a página Sports Tracker e redes sociais Facebook e Twitter. É também possível sincronizar o *smartphone* com outros dispositivos móveis através de *Bluetooth*. Aplicação inicia logo após o *boot* do dispositivo, o que pode levar a que o dispositivo fique mais lento e aumentar o consumo de bateria. Aplicação tem algumas permissões desnecessárias para o seu correcto funcionamento e que podem violar a privacidade do utilizador: o acesso a chamadas telefónicas e dados relativos às mesmas.

Figura 11 – Visualização da rota e dados da rota (SportsTracker)

2.8. Endomondo SportsTracker

Nesta aplicação pode-se registar actividades desportivas e os trajectos realizados através dos sistemas de localização GPS, WiFi. As rotas e as respectivas fotos podem ser visualizadas através de um mapa (figura 12). Nesta aplicação não é possível ajustar o intervalo de tempo ou distância com que é realizada a actualização da posição do utilizador. Os dados não podem ser exportados localmente mas podem ser partilhados socialmente para o Facebook. É também possível sincronizar o *smartphone* com outros dispositivos móveis através de *Bluetooth*. Esta aplicação pode causar um grande desgaste de bateria pois é iniciada logo após o arranque do dispositivo. A interface gráfica é simples pois contém apenas os elementos com informações importantes para o utilizador. Esta é verticalmente composta por dois níveis e horizontalmente por dois ecrãs. A navegação vertical é realizada pelo botão *back* e não requer o botão *up* devido a ser composta apenas por dois níveis. A navegação horizontal é realizada através de um botão que permite alternar entre os dois ecrãs da aplicação. Possui ainda uma *action bar* que possibilita o acesso às restantes funções que não estão presentes nos dois ecrãs. Um problema da interface gráfica é a constante presença de publicidade. Nas permissões verificou-se alguns aspectos preocupantes que colocam em causa a privacidade do utilizador tais como o acesso à informação de chamadas telefónicas, acesso e edição da lista de contactos, possibilidade de adulterar a localização do dispositivo.

Figura 12 – Ecrã principal com dados da rota (Endomondo SportsTracker)

2.9. Runtastic

Esta aplicação permite o registo dos trajectos realizados em actividades desportivas através do sistema GPS, WiFi e rede móvel, podendo definir o tipo de actividade assim como visualizar o trajecto no GoogleMaps. Após concluir a rota é possível definir várias características tais como a satisfação com o percurso, o piso do percurso, o tempo e notas. A interface gráfica é objectiva e sobretudo configurável no ecrã de registo de localização, no qual se pode escolher no total quatro informações sobre a rota (figura 13). O ecrã fica bloqueado assim que se inicia o registo de localização e apenas se pode desbloquear através um botão específico, caso este não seja pressionado é mostrado um aviso ao utilizador. A navegação horizontal é realizada através de *swipe* nos três ecrãs da interface gráfica e a navegação horizontal é realizada através do botão *back*. Também possui uma *action bar* com o nome da aplicação e acesso a funções importantes que se ajusta consoante o ecrã em uso. Um problema da interface gráfica é a constante presença de publicidade. Não é possível ajustar o intervalo de tempo ou distância com que é realizada a actualização da posição do utilizador. Na versão paga é possível tirar fotografias associadas a cada trajecto. As actividades podem ainda ser partilhadas nas redes sociais: Google+, Facebook, Twitter. É também possível sincronizar o *smartphone* com outros dispositivos móveis através de *Bluetooth*. Nas permissões verificou-se alguns aspectos preocupantes que colocam em causa a privacidade do utilizador tais como o acesso a dados das chamadas telefónicas e acesso à lista de aplicações que o telemóvel está actualmente a executar.

Figura 13 – Dados da rota (Runtastic)

2.10. Runkeeper

Esta aplicação permite o registo dos trajectos realizados em desportos através do sistema GPS podendo definir o tipo de actividade assim como visualizar num mapa o trajecto e tirar fotografias, no entanto não parece ser possível visualiza-las posteriormente na aplicação. A interface gráfica é objectiva pois permite visualizar toda informação relevante num único ecrã durante o registo de localização (figura 14). Não existe hierarquia vertical e consequentemente não existe navegação vertical. A navegação horizontal entre os dois ecrãs é realizada através de um botão que contém uma imagem de um mapa e que permite perceber a funcionalidade do outro. Outras funções importantes estão contidas no ecrã junto a informação relevante para o utilizador no entanto os botões poderiam estar numa *action bar* para ficarem de acordo com os padrões de *design* da Google. Esta aplicação não permite ajustar o intervalo de tempo ou distância com que é realizada a actualização da posição do utilizador. As actividades podem ainda ser partilhadas nas redes sociais: Facebook, Twitter. É também possível sincronizar o *smartphone* com outros dispositivos móveis através de *Bluetooth*. A aplicação acede aos contactos do utilizador colocando em causa a privacidade dos mesmos. Nas permissões também está presente a autorização de recepção de mensagens dos servidores para a aplicação o que poderá a levar um grande aumento dos dados trocados e consequentemente o custo ao utilizador caso esteja a usar a rede móvel para troca de dados.

Figura 14 – Dados da rota (Runkeeper)

2.11. Comparação das funções

Nome	Método localização	Intervalo de registo da posição ajustável	Exportar dados	Pontos de interesse	Redes sociais	Mapa
Maverick	GPS, WiFi, Rede móvel	Sim	GPX, KML	Fotos, Descrição	Não	Offline
GPS Essentials	GPS	Sim	GPX, KML	Fotos, Descrição	Não	Offline
GPS Logger	GPS, WiFi, Rede móvel	Sim	GPX, KML	Não	Não	Não
OSM Tracker for Android	GPS	Sim	GPX	Fotos, Descrição	Não	Offline
OruxMaps	GPS	Sim	GPX, KML	Fotos, Descrição	Não	Offline
MyTracks	GPS, WiFi, Rede móvel	Sim	GPX	Descrição	Facebook, Twitter, Google+	Online
Sports Tracker	GPS	Não	Não	Fotos	Facebook, Twitter	Online
Endomondo Sports Tracker	GPS, WiFi, Rede móvel	Não	Não	Fotos	Facebook	Online
Runtastic	GPS, WiFi, Rede móvel	Não	Não	Fotos	Facebook, Twitter, Google+	Online
Runkeeper	GPS	Não	Não	Fotos	Não	Online

Tabela 2 – Resumo das funções das aplicações

Após análise da tabela 2 verificamos que há aplicações com algumas das funções pretendidas: registo de localização através de GPS e WiFi e rede móvel, criação pontos de interesse com notas dos locais e fotografias, partilha de dados em algumas sociais, mapas sem ligação à web, exportação de dados. Verifica-se que apenas 40% das aplicações analisadas permitem partilha de dados nas redes sociais e com alguma diversidade (Google+, Facebook, Twitter).

A aplicação que foi implementada neste projecto permite aceder a novas funcionalidades ainda não implementadas neste tipo de programas: detecção de eventos, rotas e POIs na proximidade do utilizador, envio de todos os dados para um servidor que garante privacidade dos dados pessoais do utilizador, partilha de rotas através da própria aplicação após validação das mesmas pela autarquia.

2.12. Comparação interface gráfica e privacidade

A tabela seguinte contém um resumo da qualidade da interface gráfica e permissões das aplicações analisadas. A Usabilidade representa análise de se aplicação contém *action bar* e é possível realizar o gesto *swipe* caso seja necessário mudar de ecrã entre ecrãs irmãos, a cotação Má reflecte ausência da *action bar* e *swipe*, Razoável representa ausência apenas de um, Boa indica que ambos estão presentes na aplicação.

Nome	Usabilidade	Permissões e privacidade
Maverick	Boa	Ok
GPS Essentials	Boa	Violação
GPS Logger	Boa	Violação
OSM Tracker for Android	Boa	Violação
OruxMaps	Razoável	Ok
MyTracks	Boa	Violação
Sports Tracker	Razoável	Violação
Endomondo Sports Tracker	Boa	Violação
Runtastic	Boa	Violação
Runkeeper	Razoável	Violação

Tabela 3 – Comparação da interface gráfica e permissões das aplicações testadas

Através da análise da tabela anterior verificamos que 80% das aplicações apresenta problemas nas permissões que colocam em risco a privacidade do utilizador. Os problemas surgem através de permissões desnecessárias para o correcto funcionamento da aplicação e que permitem acesso aos dados privados do utilizador. Em termos gráficos verifica-se que 30% das aplicações não possui *swipe* para transitar entre ecrãs irmãos ou *action bar*. Através destes resultados conclui-se que existem algumas aplicações com boas interfaces gráficas que poderão ser uma boa base para a interface gráfica da futura aplicação.

3. Arquitectura e implementação

Nas primeiras secções deste capítulo define-se a visão geral do sistema, as suas componentes e os requisitos funcionais. Nas secções restantes define-se as tecnologias utilizadas e descreve-se detalhadamente a implementação das diversas componentes do sistema

3.1. Visão geral do sistema

Na figura seguinte apresenta-se as diversas componentes do sistema e os seus respectivos fluxos de comunicação.

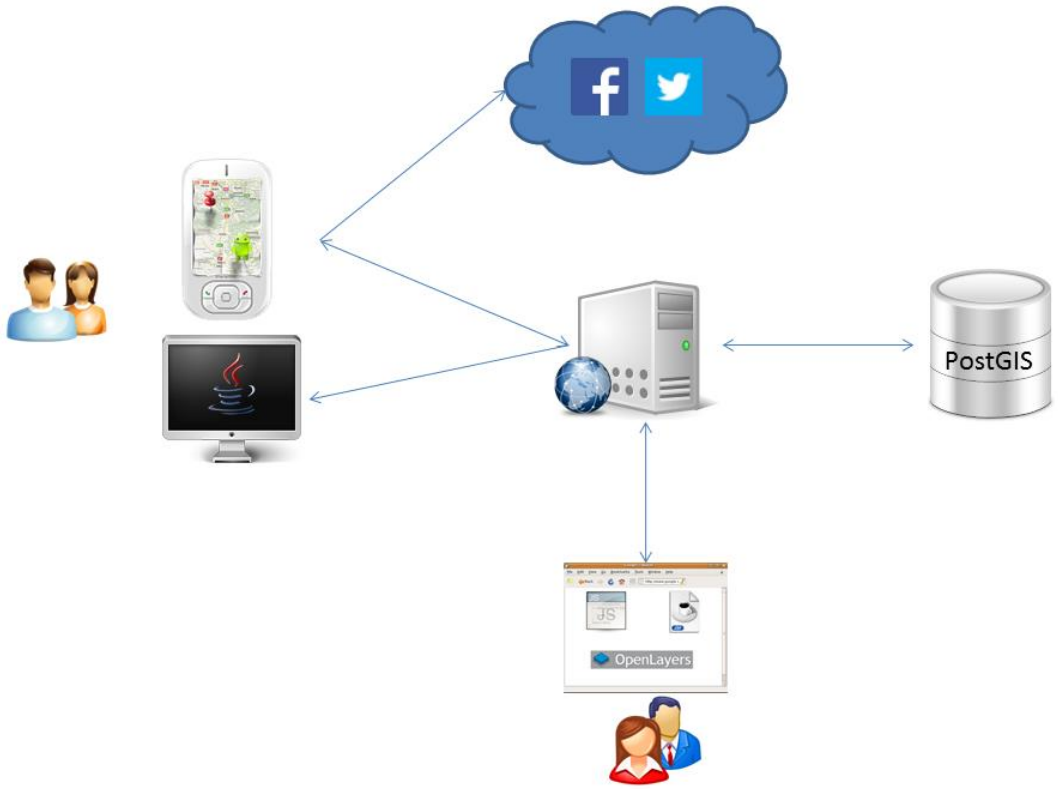

Figura 15 – Esquema alto nível do sistema

O sistema é constituído por três aplicações de diferentes tecnologias: Android, JavaEE, HTML. Aplicação Android é utilizada para registar os dados que podem ser enviados para redes sociais e para servidor da autarquia. Aplicação JavaEE sincroniza com o servidor da autarquia e permite ao utilizador criar automaticamente uma réplica dos seus dados e visualizar gráficos sobre as rotas. Aplicação Web permite à autarquia visualizar estatísticas dos acessos ao servidor assim como das freguesias e também permite inserir eventos, POIs e validar rotas inseridas pelos utilizadores. O servidor está implementado num sistema 3-Tier e aloja aplicação web da autarquia. Nas secções seguintes está detalhada a implementação destas três aplicações e do servidor.

3.2. Requisitos

A definição dos requisitos é uma etapa fulcral pois definem as necessidades dos utilizadores. Também são utilizados como guias para o projecto e permitem avaliar o resultado final do projecto. Os requisitos iniciais do projecto estão definidos em linguagem natural (humana) e definem as funcionalidades do sistema. Com base nesses requisitos criou-se várias tabelas que definem os requisitos funcionais de cada aplicação do sistema. Deste modo define-se as funcionalidades com mais detalhes a nível de implementação e de usabilidade. Adicionou-se também alguns requisitos funcionais para divulgar os pontos de interesse nas redes sociais e consequentemente ajustou-se as tabelas de requisitos. Após alguns testes foi necessário reajustar os requisitos devido a limitações tecnológicas e necessidade de elevado processamento que prejudicava a usabilidade. Alguns testes demonstraram que algumas funcionalidades necessitavam de elevado espaço no armazenamento e portanto foram removidas. Algumas funcionalidades foram melhoradas durante a sua implementação e consequentemente alterou-se as suas descrições nas tabelas de requisitos. Estes melhoramentos também removeram a necessidade de implementar outras funcionalidades e portanto foram removidas das tabelas.

3.2.1. Android

Na tabela seguinte ilustra-se as funcionalidades previamente planeadas para aplicação Android assim como as suas novas funcionalidades.

ID	Descrição	Estado
1	Registo da localização e dados da rota	Implementado
2	Criação pontos de interesses com fotos e descrição	Implementado
3	Partilha pontos de interesse nas redes sociais	Implementado
4	Visualização de rotas no mapa sem ligação à rede	Implementado
5	Comunicação com servidor	Implementado
6	Galeria de fotos	Implementado
7	Consulta de rotas de outros utilizadores	Implementado
8	Rotação automática de mapas e descoberta de POIs	Reformulado
9	Visualização de eventos nas freguesias	Reformulado
10	Pesquisa rotas de outros utilizadores	Reformulado
11	Alertas de aproximação de pontos de interesse	Reformulado
12	Criação pontos de interesses com vídeo	Descontinuado
13	Visualização de freguesias no mapa sem ligação à rede	Descontinuado
14	Notificação mensal de actualização da aplicação	Descontinuado
15	Notificação mensal de sincronização com servidor	Descontinuado
16	Exportação de dados para email e/ou Dropbox	Descontinuado
17	Actualização base de dados após download da app	Descontinuado
18	Sincronização automática com o PC	Descontinuado

Tabela 4 – Requisitos da aplicação Android

As funcionalidades 8 a 11 foram reformuladas e integradas na funcionalidade 7 para melhorar usabilidade do utilizador pois retira a necessidade de inserir dados e disponibiliza-lhe informação sobre o que o rodeia.

O requisito 12 foi descontinuado devido aos tamanhos dos seus ficheiros, que resultariam numa elevada ocupação de espaço no sistema de ficheiros. O envio para o servidor seria lento e consumiria muito tarifário do tarifário do utilizador.

A funcionalidade 13 também foi descontinuada devido a requerer elevado processamento e afectar seriamente o desempenho gráfico da aplicação.

Os requisitos 14 e 15 foram removidas devido à aplicação já conter uma notificação visual de quais as rotas que já estão sincronizadas com o servidor.

A funcionalidade 16 foi descontinuada pois poderia ser utilizada para exportar os dados para aplicações concorrentes. Deste modo poderia diminuir o envio de rotas para o servidor da autarquia e consequentemente diminuía a quantidade de dados que poderiam ser analisados assim como diminuía a partilha dos pontos de interesse no sistema.

O requisito 17 foi descontinuado visto que os mapas estão integrados na aplicação.

A funcionalidade 18 não foi implementada devido a diversas restrições técnicas para sua utilização por parte dos utilizadores: requer que o *smartphone* e o computador estejam na mesma gama de IPs (Internet Protocol address) ou que a ligação se realize por cabo USB (Universal Serial Bus). Esta funcionalidade também poderia retirar incentivo aos utilizadores de enviarem as rotas para o servidor da autarquia resultando numa diminuição dos dados que poderiam ser analisados pela autarquia.

3.2.2. Servidor

Na tabela seguinte ilustra-se as funcionalidades previamente planeadas para aplicação do servidor assim como as suas novas funcionalidades.

ID	Descrição	Estado
1	Estatísticas das freguesias	Implementado
2	Estatísticas dos IPs de acesso	Implementado
3	Inserção de POIs	Implementado
4	Inserção de eventos	Implementado
5	Visualização das rotas no mapa	Implementado
6	Validação e compressão de rotas dos utilizadores	Implementado
7	Estatísticas das acções dos utilizadores	Reformulado
8	Mapa de rotas por autarquia	Reformulado
9	Estatísticas dos downloads	Descontinuado
10	Página web da aplicação Android (publicidade)	Descontinuado

Tabela 5 – Requisitos do servidor e aplicação web da autarquia

A funcionalidade 8 foi reformulada para a 5. Esta mudança foi obrigatória visto que a visualização apenas de uma rota já requer um elevado processamento e a aplicação iria

bloquear caso se carregassem várias rotas simultaneamente. Os dados que definem as autarquias também são bastante extensos e iriam piorar o desempenho da aplicação.

Algumas das estatísticas, requisito 9, foram retiradas devido a alterações na aplicação Android que já não fornece os dados necessários. Foram implementadas estatísticas, requisito 2, nas quais se analisa a localização geográfica dos utilizadores através dos IPs de acesso ao servidor.

O requisito 10 não era um requisito definido pela autarquia e portanto tinha baixa prioridade relativamente aos definidos pela autarquia. Na fase final do desenvolvimento verificou-se que não haveria tempo disponível para o implementar e utilizou-se o tempo restante para concluir os requisitos definidos pela autarquia.

3.2.3. Desktop

Na tabela seguinte ilustra-se as funcionalidades previamente planeadas para aplicação desktop assim como as suas novas funcionalidades.

ID	Descrição	Estado
1	Sincronização com servidor	Implementado
2	Acesso a gráficos das rotas	Implementado
3	Visualização de rotas no mapa	Implementado
4	Visualização do tempo e quilómetros totais realizados	Implementado
6	Edição de descrição POIs em disco	Não implementado
7	Classificação de secções das rotas	Não implementado
8	Partilha de rotas e POIs em redes sociais	Descontinuado
9	Sincronização com Android	Descontinuado
10	Actualização de dados para aplicação	Descontinuado
11	Galeria de fotos e POIs	Descontinuado

Tabela 6 – Requisitos da aplicação desktop

O ponto 6 não foi implementado. Esta funcionalidade não era fundamental para o funcionamento do sistema, não era um requisito definido pela autarquia, e devido a restrições temporais optou-se por focar na conclusão das funcionalidades fundamentais do sistema.

O requisito 7 não foi implementado devido a problemas técnicos com API JXMapViewer, responsável pela visualização e interacção com os mapas. Esta API tem uma função que converte os pontos no ecrã em pontos geográficos do mapa mas após vários testes não foi possível utilizar esta função. Também seria necessário duplicar no servidor os dados dos pontos GPS de todas as rotas de todos os utilizadores.

A funcionalidade 8 foi descontinuada visto que esta funcionalidade está presente na aplicação Android e não é um requisito definido pela autarquia.

O requisito 9 foi descontinuado devido à aplicação Android já não suportar sincronização com PC. Após várias pesquisas e testes concluiu-se que esta funcionalidade requeria que o *smartphone* e o computador estivessem na mesma gama de IPs ou que a ligação se realize por

cabo USB. Também poderia retirar incentivo aos utilizadores de enviarem as rotas para o servidor da autarquia resultando numa diminuição dos dados que poderiam ser analisados pela autarquia.

A funcionalidade 10 foi descontinuada devido a este tipo de dados poderem ver visualizados na opção 3, através de uma galeria para cada POI quando o utilizador mantém o rato em cima de um ícone do POI.

3.3. Casos de uso

Nesta secção estão descritos os casos de uso das diferentes componentes do sistema.

3.3.1. Android

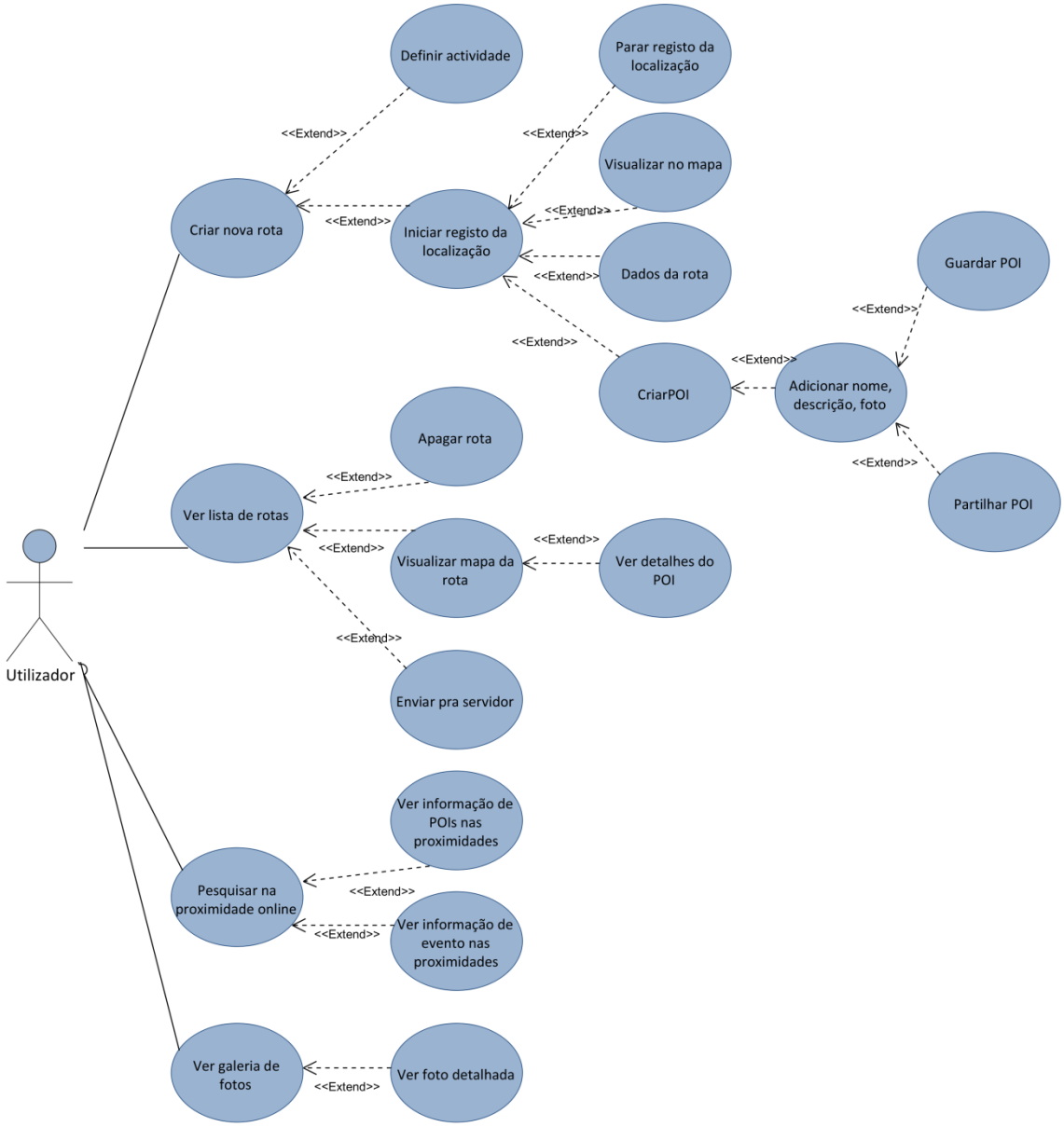

Figura 16 – Casos de uso Android

Na aplicação Android o utilizador tem acesso a quatro funcionalidades no menu principal. Caso opte por criar uma rota então poderá registar as coordenadas GPS pelos locais que visita e opcionalmente definir actividade. Enquanto o registo de localização estiver activo e exista sinal GPS o utilizador poderá visualizar os locais por onde passou no mapa, assim como visualizar alguns dados gerais da rota e também criar POIs através das coordenadas desses locais. É possível adicionar nome, descrição e fotos aos POIs. Caso tenha ligação WiFi poderá imediatamente partilhar estes dados no Facebook ou Twitter. Na opção de lista de rotas poderá visualizar a data das rotas que já gravou, identificar quais estão sincronizadas

com servidor e enviar as não sincronizadas caso tenha cobertura WiFi. Também poderá apagar as rotas ou visualizá-las num mapa com todos os dados registados na respectiva rota. Na opção da galeria acederá a uma pré-visualização das rotas e caso seleccione uma foto poderá a visualizar em ecrã completo. A opção de pesquisa *online* poderá ser acedida caso o dispositivo esteja conectado à web e tenha sinal GPS. Com esta opção pode identificar potenciais POIs, eventos e rotas nas suas imediações

3.3.2. Servidor

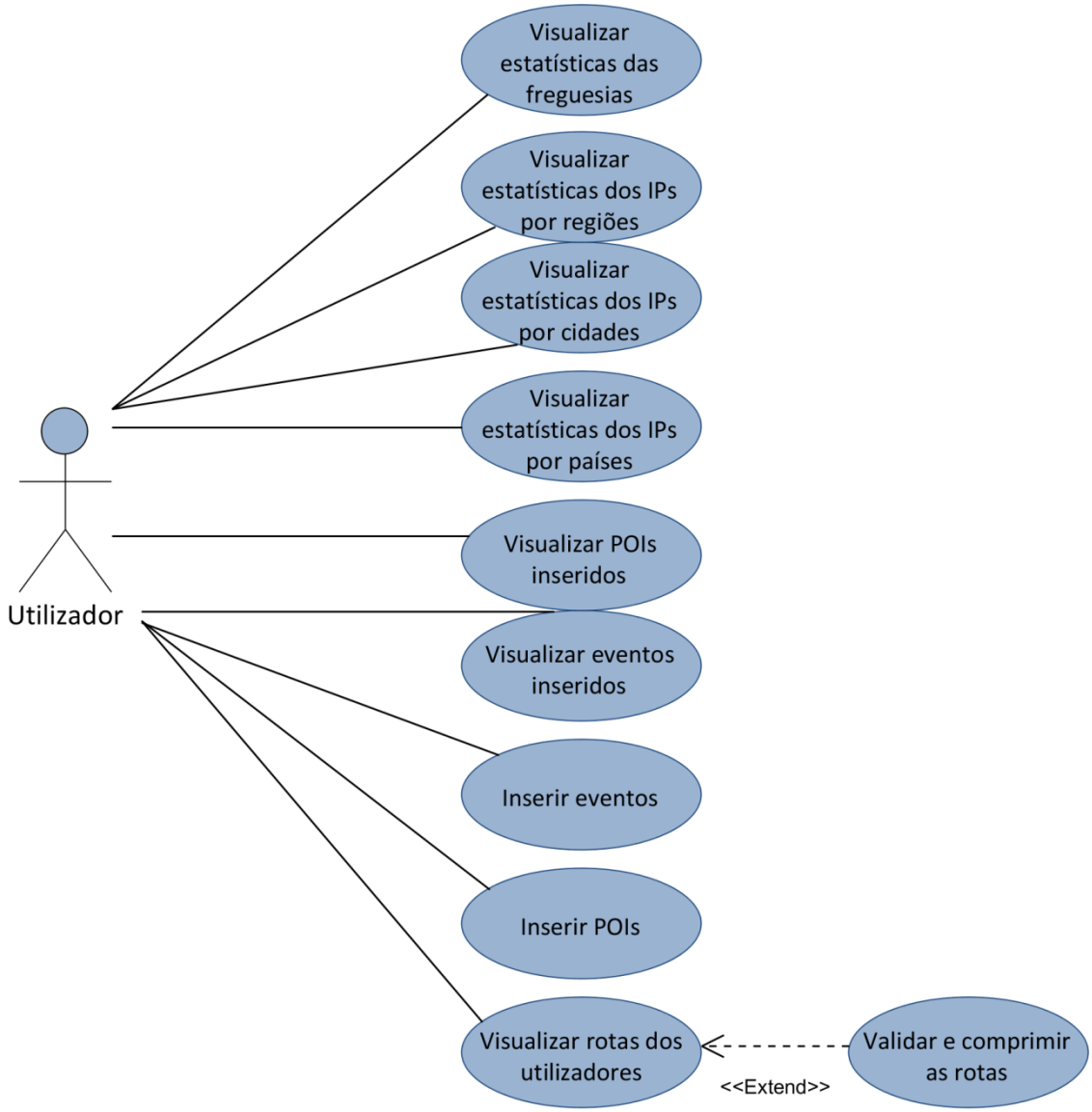

Figura 17 – Casos de uso do servidor e aplicação web da autarquia

Aplicação web da autarquia é composta por diversas páginas, ver anexo **Erro! A origem da referência não foi encontrada.**, com um menu na qual o utilizador pode aceder a todas as funcionalidades da aplicação. Aplicação só pode ser acedida caso o servidor esteja activo e *online*. Os POIs e eventos submetidos ficarão disponíveis para os utilizadores através da funcionalidade de pesquisa *online* implementada na aplicação Android. As rotas validadas também ficarão disponíveis para os utilizadores através da funcionalidade de pesquisa *online*.

3.3.3. Desktop

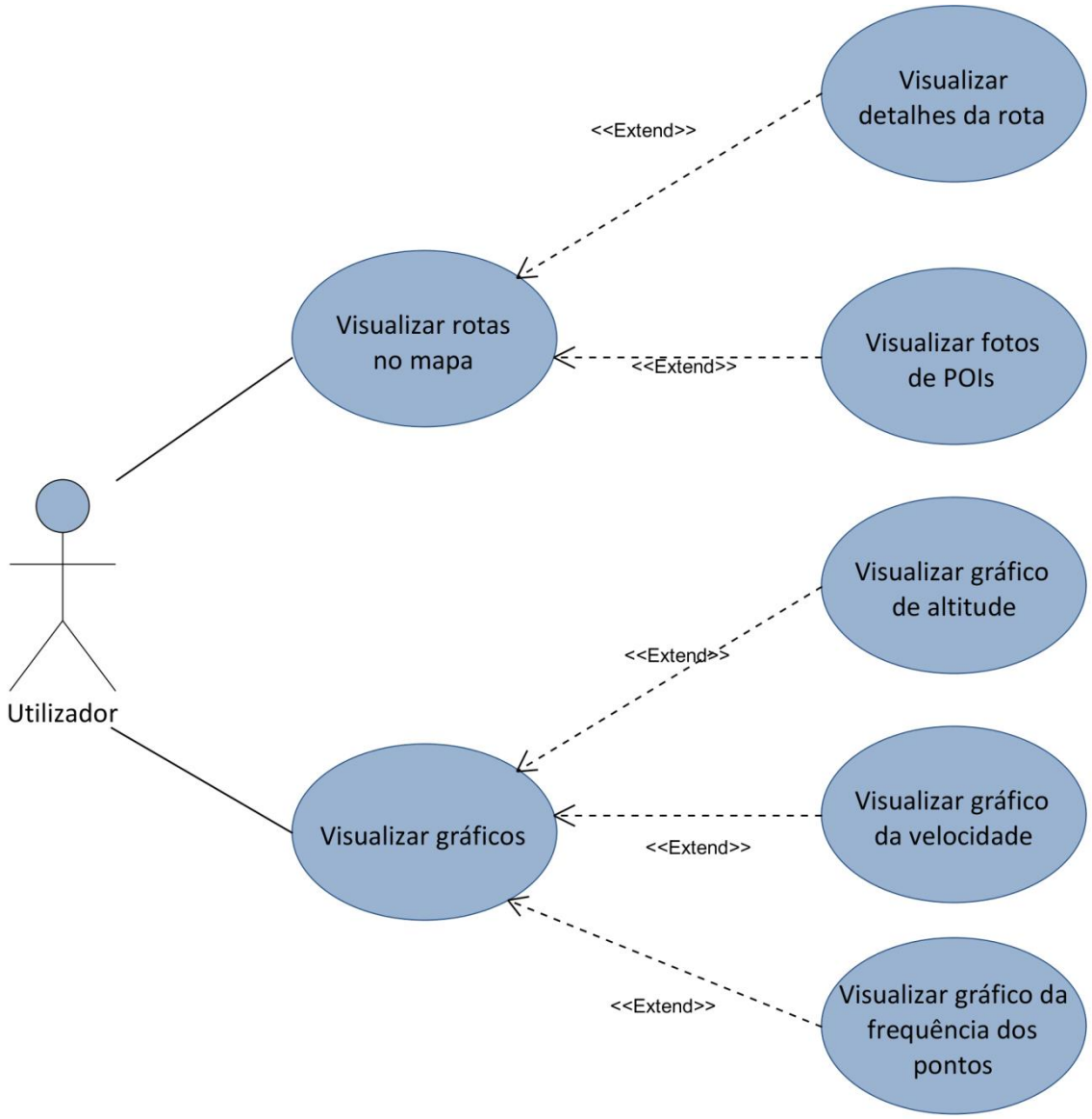

Figura 18 – Casos de uso da aplicação desktop

O total de quilómetros realizados e o tempo total das rotas podem ser visualizados no ecrã inicial. A partir deste ecrã o utilizador tem acesso a um menu que permite aceder às restantes funcionalidades. Caso seleccione "Visualizar rotas no mapa" então poderá visualizar: os dados da rota, a rota no mapa, a localização dos POIs. Ao colocar o rato por cima dos POIs poderá visualizar as fotografias e os detalhes do ponto de interesse. Se o utilizador seleccionar "Visualizar gráficos" terá acesso a um submenu no qual poderá seleccionar se quer visualizar gráfico da altitude ou velocidade ou frequência do registo da posição dos pontos.

3.4. Escolhas tecnológicas

Nesta secção ilustra-se e justifica-se as diferentes tecnologias utilizadas na implementação do sistema.

3.4.1. Android

- Facebook SDK - Conjunto de bibliotecas oficiais, que permitem autenticação e utilização da API pública do Facebook. Também possui vários elementos gráficos que permitem acesso automático a funcionalidades da API.
- Jackson – Biblioteca java utilizada para converter dados java para JSON (JavaScript Object Notation) e vice-versa. Possui extensa documentação com bastantes exemplos disponíveis e a sua utilização é bastante simples. Esta biblioteca também foi utilizada no servidor e optou-se por manter também sua utilização em Android visto que o programador já tinha conhecimentos do seu funcionamento.
- JSON – Formato de dados utilizados na troca de mensagens. Este formato permite para manter o fluxo de dados ao mínimo possível.
- OSMdroid (OpenStreetMaps) – Biblioteca grátis que permite visualizar mapas sem necessitar de ligação à internet. É a biblioteca oficial utilizada para visualização de mapas baseados em OpenStreetMaps. Optou-se por utilizar estes mapas pois seu acesso é público e estará sempre disponível. As alternativas GoogleMaps e Bing não permitem utilizar mapas *offline*. A SDK (Software Development Kit) para Android dos mapas Nokia ainda está em fase beta mas sua utilização poderá sofrer restrições devido à forte possibilidade de se tornar um produto comercial, requerendo a compra de uma licença para sua utilização.
- Sherlock Actionbar – Extensão da biblioteca de suporte da Google que facilita a utilização da *action bar*, padrão de design, em todas as versões Android através de uma única API. Após uma longa pesquisa não foi encontrada mais nenhuma biblioteca que tivesse todas estas funcionalidades.
- SQLite - Sistema de gestão de dados disponível em Android e recomendado pela Google. Os seus dados são ACID.
- Twitter4j – Biblioteca não oficial que integra API do Twitter com API do respectivo repositório de imagens TwitPic. É a única API Java e compatível com Android mencionada na página oficial do Twitter. Possui suporte para autenticação Oauth simplificando o processo de autenticação. Este processo utiliza um elemento denominado token Oauth que funciona como uma chave mestra da conta do utilizador e portanto permite identificar que o utilizador é o dono da conta. Para obter estes tokens o utilizador necessita apenas de inserir o nome e palavra-chave uma única vez e de seguida a rede social envia o token. Após esta troca de dados a comunicação pode ser realizada através de HTTPS (Hypertext Transfer Protocol Secure) e a autenticação é feita através do token Oauth.

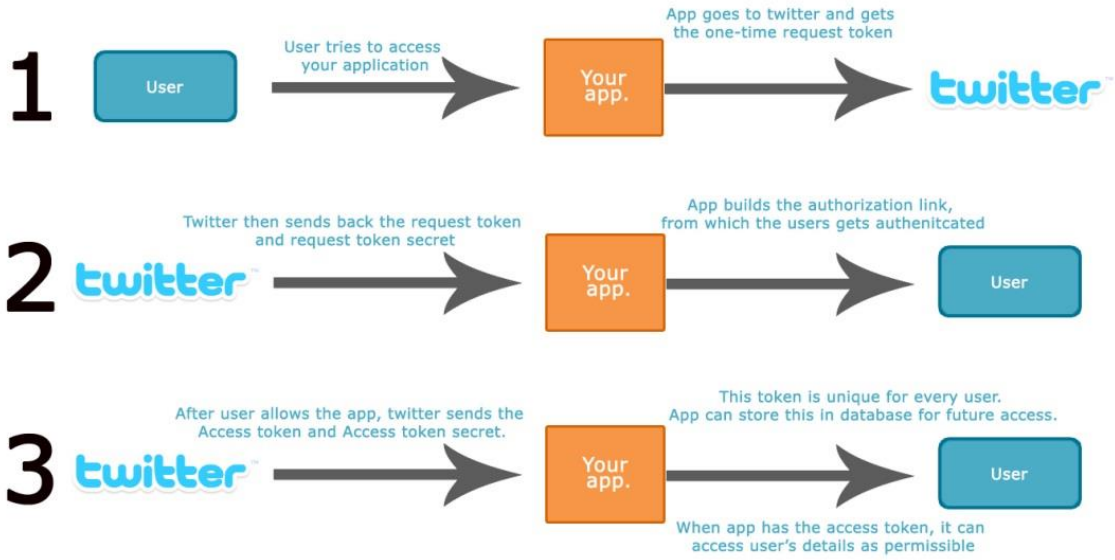

Figura 19 – Autenticação Oauth do Twitter

3.4.2. Servidor

- Ajax – Conjunto de técnicas utilizadas nos clientes JavaScript que permitem realizar chamadas assíncronas no formato JSON com servidores. Foi necessário utilizar este técnica visto que a API OpenLayers, responsável pela visualização dos mapas, está implementada em JavaScript.
- CSS – Linguagem que permite descrever o aspecto gráfico de uma página web e a formatação dos elementos das mesmas. É utilizada sobretudo nas interfaces HTML e permite separar a parte do conteúdo da parte da apresentação/formatação do documento. Também permite partilhar a formatação com outros documentos.
- Freegeoip.net – Webservice REST (Representational State Transfer) que permite geolocalizar IPs. Foram testados outros serviços grátis para o mesmo efeito mas este apresentou melhor documentação e resultados mais detalhados.
- JavaEE – Framework java para criação de Webservice.
- Java Persistence API entities – Componente da ferramenta JavaEE que permite gerir dados relacionais. Os dados são definidos em classes java, as suas relações são definidas através de notações e são automaticamente mapeados nas bases de dados. Cada objecto destas classes corresponde a uma linha duma tabela. Esta ferramenta também permite que o programador se abstraia da definição das tabelas em SQL e se concentre no modelo de dados.
- JSON – Formato de dados utilizados na troca de mensagens. Este formato permite para manter o fluxo de dados ao mínimo possível.
- GlassFish – Servidor web de código aberto para a plataforma JavaEE e suporta EJB, JPA, JSP e Hibernate. Este servidor também faz parte do IDE (aplicação que permite desenvolvimento de software) utilizado neste estágio, minimizando a necessidade de configurações. O criador deste projecto também já tinha experiência com este servidor.
- Hibernate spatial – Extensão genérica grátis utilizada na gestão de dados geográficos. Esta extensão complementa as Java Persistence API entities pois permite mapear em sql, os dados geográficos definidos nas classes java.

- HTML – Linguagem utilizada na criação de páginas web e que pode ser integrada com JavaScript e JSP. Optou-se por esta linguagem devido ao programador ter experiência com esta linguagem e de existirem bibliotecas de visualização de mapas compatíveis.
- Jackson - Biblioteca java utilizada para serializar dados java para JSON e vice-versa. Esta biblioteca é facilmente integrável com a ferramenta Jersey, conversão de dados java para JSON e vice-versa é automática.
- Jersey – API que simplifica criação de web services REST. Permite integração automática com Jackson sem necessidade de desenvolver código pois a conversão entre dados java e JSON é automática. O programador tem experiência com esta ferramenta permitindo um desenvolvimento mais rápido.
- JSP – Tecnologia de suporte à criação de páginas web dinâmicas e compatível com HTML e a Java. O programador também já tinha experiência com esta tecnologia.
- OpenLayers JavaScript – Permite adição de mapas dinâmicos numa página web. O acesso a estes mapas não têm restrições e o seu acesso será sempre público, as alternativas não garantem o acesso futuro aos seus mapas.
- Postgres + PostGIS – Sistema para gestão de dados relacionais e disponível para vários sistemas operativos. A sua extensão PostGIS suporta dados geométricos e geográficos e contêm uma grande variedade de funções para estes tipos de dados. Aparenta ser o sistema de gestão de dados grátis mais maduro devido à enorme diversidade de funções que possui e possui extensa documentação rica em exemplos de implementação.
- REST - Estilo de arquitectura de *software* para sistemas web. Utiliza protocolo HTTP (Hypertext Transfer Protocol) e HTTPS para comunicação entre os diversos elementos do sistema. Com esta arquitectura foi possível utilizar JSON reduzindo o fluxo de dados e o processamento no servidor. Alternativa SOAP não permite utilizar JSON e portanto o fluxo de dados e processamento poderia ser mais elevado.
- Vividsolutions – Biblioteca que permite definir objectos geométricos em classes. É a única biblioteca compatível hibernate-spatial e PostGIS.

3.4.3. Desktop

- Jackson – Biblioteca java utilizada para serializar dados java para JSON e desserializar dados JSON para java. Também utilizada nas restantes aplicações do projecto
- JFreechart – Ferramenta grátis para java que simplifica a criação de gráficos. Existem poucas alternativas grátis e as suas documentações eram escassas.
- JSON – Formato de dados utilizados na troca de mensagens. Este formato permite para manter o fluxo de dados ao mínimo possível.
- JXMapViewer – API java responsável pela visualização e interacção dos mapas OpenStreetMaps. Existem poucas alternativas em java para visualização destes mapas e a opção a escolhida é a que apresenta mais documentação com exemplos de implementação.
- SQLite – Sistema de gestão de dados relacionais recomendado para clientes java. Os seus dados são ACID.
- Swing – Ferramenta primária para criação de interfaces gráficas java.

3.5. Android

3.5.1. Protótipo da interface gráfica

Em seguida apresenta-se o primeiro protótipo para a interface gráfica da aplicação Android tendo em conta aspectos de usabilidade. No anexo **Erro! A origem da referência não foi encontrada.** pode visualizar toda a navegação do protótipo. Na figura seguinte ilustra-se o menu inicial da aplicação no qual o utilizador pode iniciar login, mudar palavra-chave ou registar-se no servidor da autarquia. Posteriormente o utilizador tem acesso ao menu principal da aplicação no qual se pode aceder às funcionalidades da aplicação.

Figura 20 – Login e menu principal do protótipo

Caso o utilizador seleccione opção *start tracking* será direccionado para os ecrãs da figura 21, nestes pode criar uma rota e também verificar as coordenadas dos pontos ou visualizar os pontos no mapa. A transição entre os dois ecrãs pode ser efectuada através da *tab* ou através do gesto *swipe*. O utilizador poderá durante o registo de localização activar um botão que bloqueia e desbloqueia o restante ecrã. Após concluir o registo das localizações poderá partilhar a rota nas redes sociais. Enquanto o utilizador estiver nestes ecrãs poderá também aceder à opção de criar um ponto de interesse, através do atalho câmara fotográfica no canto superior direito. Neste novo ecrã o utilizador poderá inserir informação sobre o local, visualizar as coordenadas e tirar fotografias. Se clicar no novo ícone da câmara fotográfica haverá uma transacção para um novo ecrã no qual existirá previsualização da imagem vista pela câmara. Após tirar a foto o utilizador regressará automaticamente para o menu anterior, figura 24. Os dados serão gravados quando o utilizador premir o botão guardar. Se o utilizador optar por partilhar os dados nas redes sociais surgirá uma lista com as redes sociais disponíveis. O envio será automático se o utilizador tiver inserido as credenciais anteriormente, caso contrário aplicação irá aceder à página de login da respectiva rede social e terá que efectuar o login.

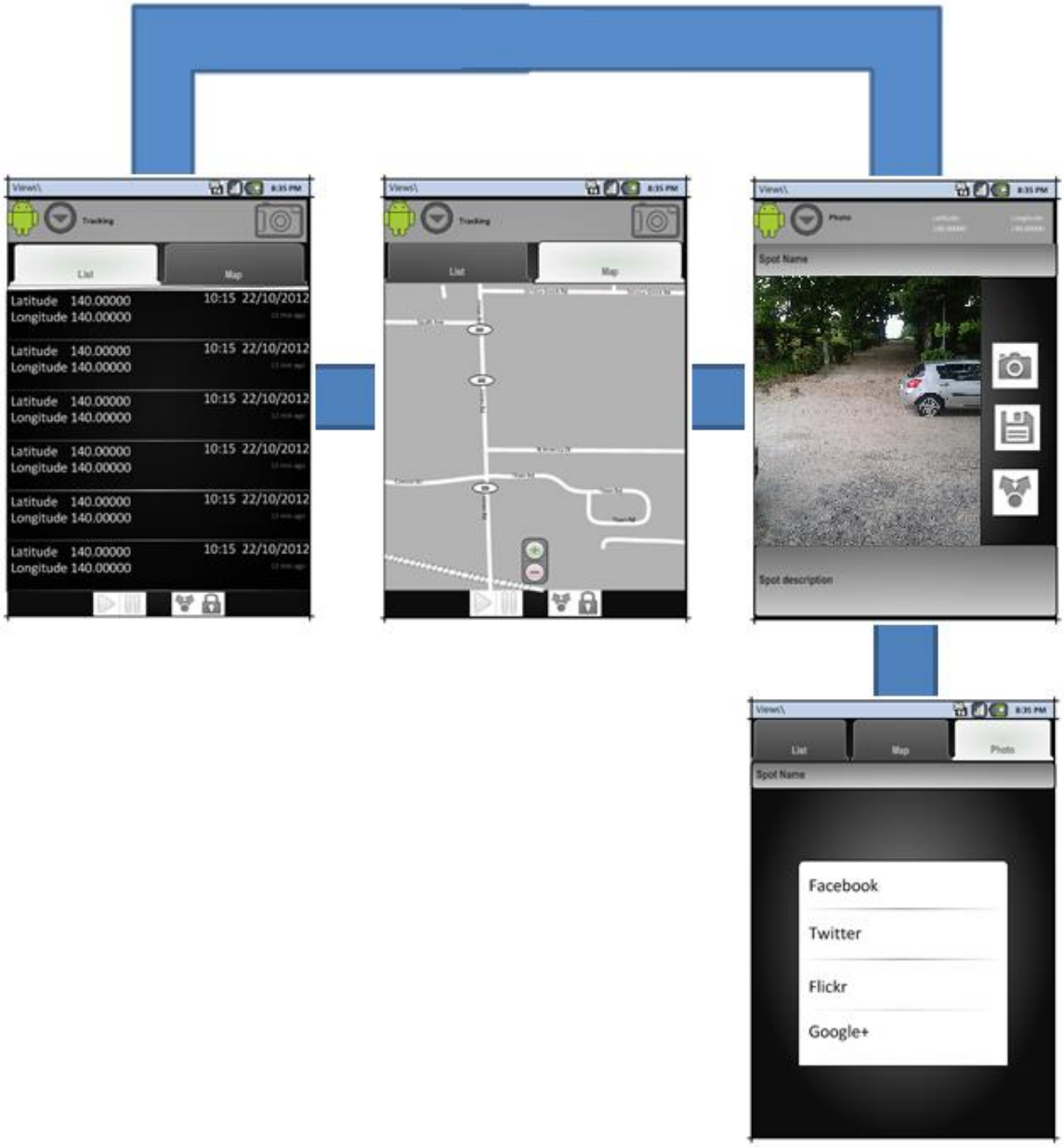

Figura 21 – Ecrãs da opção tracking do protótipo

O utilizador será direccionado para o ecrã da figura 22 caso seleccione opção *gallery*. Neste ecrã terá acesso a uma grelha que contém previsualizações de todas as fotos e se clicar numa foto irá visualiza-la em ecrã completo. A navegação para as imagens não visíveis no ecrã será feita através do gesto *scrolling* (arrastar na vertical).

Figura 22 – Ecrã da galeria de fotografias (Protótipo)

Caso escolha as opções *waypoint* ou *web search* ou *tracks* irá transitar para o ecrã da figura 23. Nas opções *waypoint* e *tracks* os dados surgem imediatamente todos listados podendo depois ser filtrados por termos na caixa de inserção de texto. No caso do *web search* os dados só surgirão após inserção de dados na caixa de texto e caso exista ligação à web. Quando o utilizador clicar numa das linhas da lista haverá transição para ecrãs que mostram os detalhes da opção escolhida.

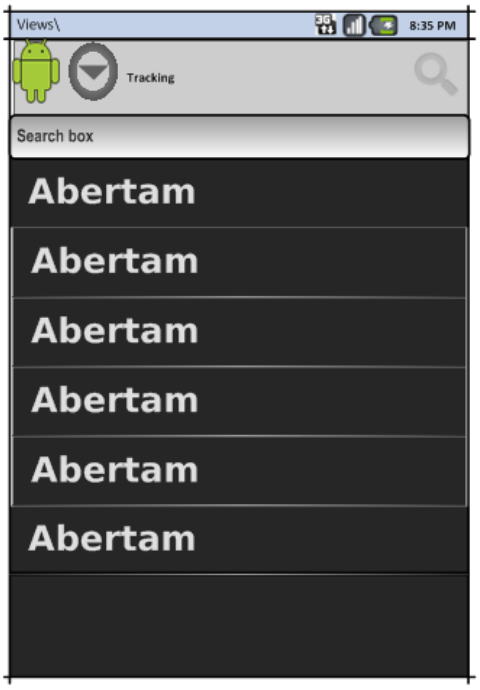

Figura 23 – Lista de POIs ou rotas

3.5.2. Navegação

Com base nos casos de uso e no protótipo implementou-se os vários ecrãs da aplicação. No anexo **Erro! A origem da referência não foi encontrada.** pode consultar um diagrama com imagens reais do *smartphone* e a navegação entre todos os ecrãs e vistas. De seguida serão descritas todas as interacções e navegações que o utilizador pode realizar.

No primeiro arranque da aplicação será verificado se os mapas estão instalados, caso não estejam então surgirá um ecrã na qual se pode visualizar o estado da instalação dos mapas. Concluída a instalação, o utilizador visualizará um ecrã de login, onde pode inserir os seus dados caso já tenha uma conta no nosso sistema ou aceder ao ecrã de registo caso não tenha conta no sistema. O login e o registo só necessitam de ser realizados uma vez mas requerem ligação WiFi para confirmação dos dados inseridos pelo utilizador. Após o sucesso do login ou registo aplicação transitará para o ecrã principal, no qual o utilizador terá acesso a 4 funcionalidades: *Track*, *My Tracks*, *Photos*, *Near me*.

Figura 24 – Ecrãs iniciais da aplicação Android

3.5.2.1. Track

Para aceder à opção *Track* necessitará de ter recepção de sinal GPS activa. Nesta opção terá acesso a três vistas, *Map* e *Info* e *POI*, e poderá alternar entre ambas através de clique no nome da vista ou através do gesto *swipe*. Na vista *Info* poderá iniciar ou concluir o registo de localização e visualizar dados gerais da rota assim como definir actividade. Na vista *Map* poderá verificar a sua localização e os pontos já registados. Caso seja necessário poderá realizar *zoom* no mapa. Na vista *POI* poderá criar um POI inserindo um nome e uma descrição e associar fotografias. Após estes três dados recolhidos poderá guardar o POI ou partilha-lo nas redes sociais. Para partilhar nas redes sociais necessitará de ligação WiFi e será enviado o nome, descrição, coordenadas GPS e a última fotografia recolhida. Na vista *POI* o utilizador poderá também pré-visualizar a última fotografia registada nesse mesmo POI.

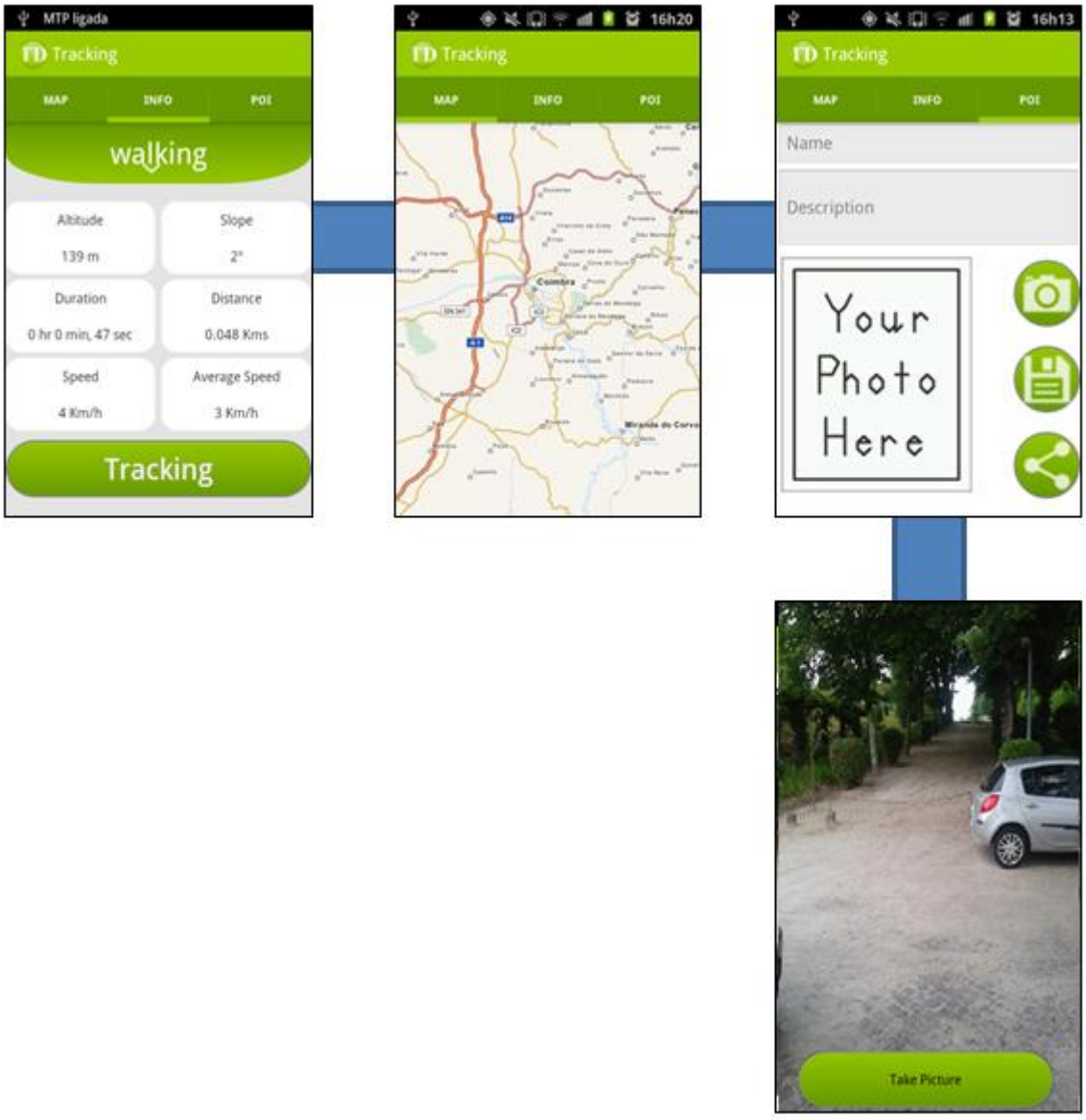

Figura 25 – Ecrãs da opção tracking

3.5.2.2. My Tracks

Na opção *My Tracks* o utilizador terá acesso a uma lista com os nomes de todas as rotas e um ícone que indicará se a rota está sincronizada com o servidor. Se o utilizador pressionar durante alguns segundos o nome de uma rota irá surgir um pop-up e poderá escolher se quer apagar a rota ou sincroniza-la com o servidor, esta última requer uma ligação WiFi.

Caso utilizador só pressione ligeiramente o nome de uma rota então aplicação transitará para um ecrã composto por duas vistas, *Map* e *Info*, e poderá alternar entre ambas através de clique no nome da vista ou através do gesto *swipe*. Na vista *Map* poderá realizar *zoom* e/ou visualizar os pontos da rota e os POIs, se clicar no ícone dos POIs poderá visualizar o nome e a descrição, se clicar durante alguns segundos no ícone terá acesso a uma galeria de pré-visualizações das fotografias do respectivo POI. Se clicar numa pré-visualização então visualizará a fotografia em ecrã completo, e um botão através do qual pode partilhar a fotografia e a informação do POI na redes sociais disponíveis. Na vista *Info* acederá ao tempo total da rota, distância total percorrida e a velocidade média.

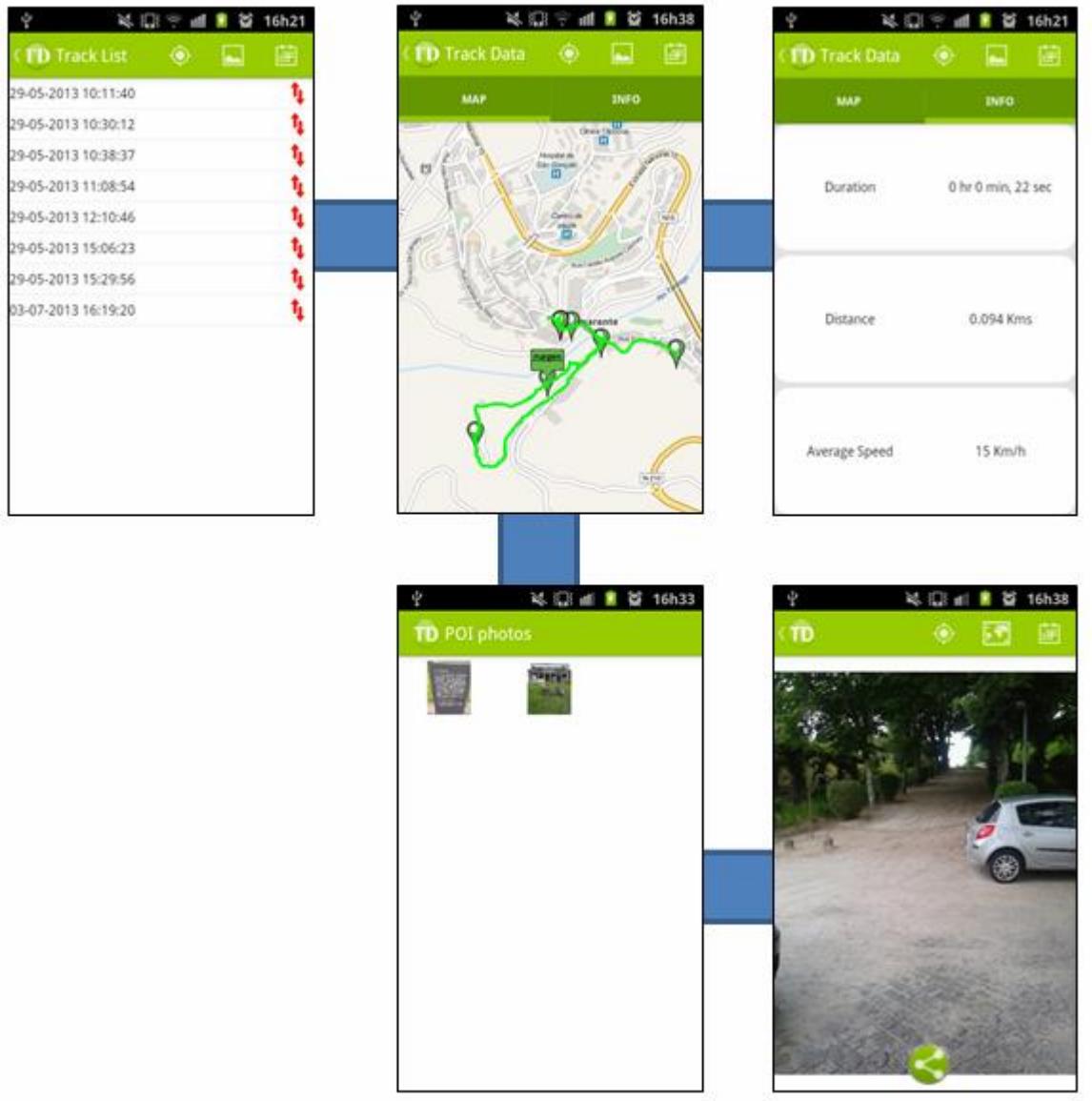

Figura 26 – Ecrã My tracks da aplicação Android

3.5.2.3. Photos

O utilizador poderá pré-visualizar todas as suas fotografias na opção *Photos*, através de uma galeria em formato grelha, quando o utilizador carregar sobre uma das pré-visualizações irá visualizar de seguida a fotografia em ecrã completo.

Figura 27 – Ecrã gallery d aplicação Android

3.5.2.4. Near me

A opção *Near me* requer acesso à localização GPS e ligação a uma rede WiFi. Nesta opção será recolhida a localização do utilizador e enviada para o servidor. Este irá responder com diversos POIs, eventos e rotas nas imediações do utilizador. Estes dados recebidos estarão representados num mapa através de ícones e figuras geométricas, o utilizador poderá aceder a detalhes destes dados carregando nos respectivos ícones. O utilizador poderá também realizar *zoom* no mapa.

Quando um utilizador está no ecrã principal das opções *My tracks*, *Photos* ou *Near me* poderá aceder rapidamente às restantes funcionalidades através dos atalhos na *action bar*.

Em todos os ecrãs ou vistas que necessitem de ligação GPS será gerada uma notificação para o utilizador caso a ligação não esteja activa. Se utilizador não activar recepção GPS então será negado o acesso à funcionalidade. Se aceitar activar então será direccionado para o ecrã da opções de localização do *smartphone* visto que é o único método disponível em todas as versões Android para activar esta funcionalidade. Após activação o utilizador deverá utilizar o botão *back* para retroceder para aplicação.

Nos ecrãs ou vistas que necessitem de ligação WiFi será gerada uma notificação caso recepção WiFi esteja desligada. Se aceitar ligar então a aplicação activará a recepção automaticamente e procurará uma ligação válida. Quando encontrar uma ligação válida o utilizador terá acesso à funcionalidade. O acesso à funcionalidade será negado caso não aceite activar a recepção WiFi.

Figura 28 – Ecrã near me da aplicação Android

3.5.3. Localização

O Android permite obter localização a partir de três sistemas diferentes: GPS, WiFi e rede móvel. Para auxiliar o desenvolvimento da aplicação Android foi criada uma tabela com uma análise comparativa dos três sistemas após um estudo dos artigos de (Abreu et al, 2008-2011). O utilizador irá passar a maior parte do seu tempo no exterior e inicialmente optou-se por utilizador o sistema de GPS e o sistema da rede celular. Após alguns testes da rede celular verificou-se que o erro da posição era bastante grande, 2 a 3 Kms, e optou-se por apenas utilizar sistema GPS. Com este sistema obtém-se a maior precisão possível.

	Precisão	**Consumo**	**Disponibilidade do sinal**
GPS	Alta	Alto	Grande cobertura no exterior, baixa em interiores
WiFi	Moderada	Moderado	Baixa cobertura: restringida a zonas com APs e interiores
Rede celular	Baixa	Baixo	Grande cobertura

Tabela 7 – Sistemas de localização no Android

3.5.4. Armazenamento

No sistema Android existem vários tipos de armazenamento disponíveis para o utilizador: preferências partilhadas, armazenamento interno, armazenamento externo, base de dados SQLite, ligação à rede.

O Facebook e o Twitter utilizam um mecanismo de autenticação denominado Oauth. Neste processo a rede social utiliza um elemento denominado token Oauth que funciona como uma chave mestra da conta do utilizador, permitindo identificar que o utilizador é o dono da conta. Este token é armazenado nas preferências partilhadas para sua utilização em comunicações futuras, deste modo associa-se o nome da rede social (chave) ao token (valor) e garante-se que os dados não podem ser acedidos por outras aplicações.

Os mapas são instalados no armazenamento externo pois a API utilizada para os visualizar *offline* requer a sua instalação numa pasta e ficheiro específicos desse sistema.

As fotos são arquivadas no armazenamento externo pois o seu espaço disponível é maior que o interno e as fotos podem ocupar muito espaço. O acesso aos dados deste sistema é global e permite ao utilizador aceder facilmente às fotos e exporta-las para outro dispositivo libertando assim espaço no *smartphone*. Quando se armazena uma foto também é gerado um ficheiro de pré-visualização da respectiva foto.

As rotas e os pontos de interesse são armazenados em base de dados SQLite, garantindo assim que estes dados privados e cruciais podem ser somente acedidos pela nossa aplicação. Também permite obter um maior desempenho para grandes quantidades de dados estruturados do que a utilização de ficheiros. Em seguida encontra-se o diagrama de classes referente aos dados que serão guardados na base de dados Android.

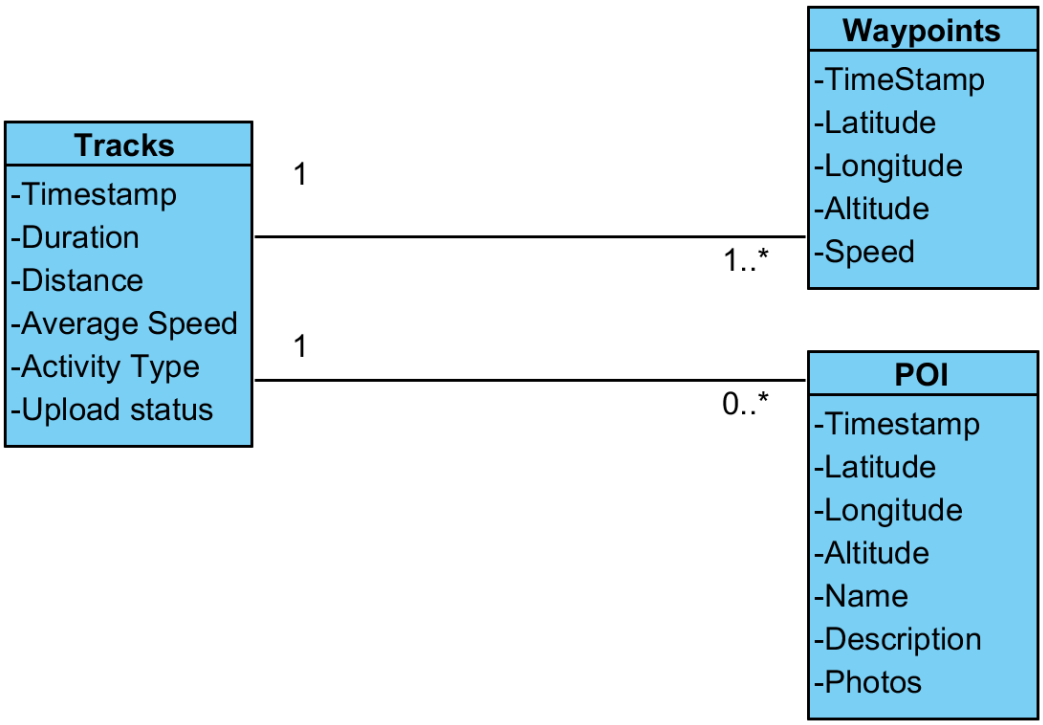

Figura 29 – Modelo lógico das rotas

3.5.5. Mapas

Para criar os mapas utilizou-se a aplicação Mobile Atlas Creator que permite descarregar manualmente os vários mapas com diferentes camadas, converte-los para imagens PNG e agrega-los num ficheiro ZIP. Assim será possível ao município actualizar facilmente os mapas. Para permitir a visualização dos mapas em Android utilizou-se a biblioteca OSMdroid que permite também mostrar as rotas e pontos de interesse. Esta biblioteca é a que apresenta melhor documentação e maturidade para disponibilização de mapas *offline* no entanto apresenta um problema: os controladores de *zoom* e movimento do mapa não são bloqueados embora as opções para esses fins estejam activas. Também se detectou que por vezes ao utilizar zoom a imagem não é ampliada e o seu espaço ficava vazio no entanto não

foi possível detectar se o problema é da biblioteca ou da capacidade de processamento do dispositivo.

3.5.6. Redes sociais

Actualmente existem várias redes sociais que permitem a divulgação de informação rapidamente através da partilha de texto e imagens. Na tabela seguinte encontra-se uma análise das APIs das redes sociais mais conhecidas que permitem partilhar texto e fotos através de dispositivos Android. Inicialmente pretendia-se suportar na aplicação todas as redes sociais da tabela seguinte que permitissem o envio de imagens e texto, mas devido à fraca qualidade de documentação e de exemplos das APIs, optou-se por suportar as duas redes sociais mais utilizadas: Facebook e Twitter.

Nome	APIs
Twitter	REST API (xml/json) + oauth1
Flickr	REST/XML-RPC/SOAP
Facebook	Android graph api + oauth2
Tumblr	REST API + Oauth 1.0
Myspace	Não foi possível aceder API
Pinterest	Sem API
Instagram	API não permite envio imagens
Google+	REST API (xml/json) + oauth2

Tabela 8 – Redes sociais e respectivas APIs

Para implementar o envio dos dados dos POIs recorreu-se à ferramenta Facebook SDK. Esta ferramenta está em constante aperfeiçoamento mas algumas funcionalidades têm sido retiradas. Seguiu-se vários exemplos fornecidos pelo Facebook para implementar autenticação com Oauth mas tal não foi possível porque está a ser descontinuada. Para resolver este contratempo seguiu-se a nova proposta de autenticação no Facebook: a utilização da aplicação oficial do Facebook. Nesta solução a autenticação é toda realizada pela aplicação do Facebook, o que implica que o utilizador necessitará de a ter aplicação instalada caso pretenda partilhar os POIs pela nossa aplicação. Este factor não deverá afectar muito o uso da aplicação desenvolvida visto que aplicação do Facebook é das mais instaladas do GooglePlay e provavelmente os nossos utilizadores já terão esta aplicação instalada.

Para enviar dados dos POIs para o Twitter utilizou-se a biblioteca Twitter4J. Esta biblioteca não oficial integra API do Twitter com API do respectivo repositório de imagens TwitPic. Possui também suporte para autenticação Oauth, simplificando o processo de autenticação, assim como suporta compressão gzip reduzindo o fluxo de dados.

3.5.7. Comunicação

A comunicação com servidor da autarquia é realizada assincronamente através de HTTP por arquitectura REST. As mensagens são enviadas no formato JSON para manter o fluxo de dados ao mínimo possível. Para converter os dados java para json e vice-versa recorreu-se à biblioteca Jackson.

```
{
    "name": "myname",
    "password": "mypass",
    "email": "myemail@gmail.com"
}
```

Figura 30 – Exemplo JSON

O mecanismo assíncrono é implementado através de classes internas que estendem a classe *AsyncTask*, deste modo a interface gráfica não bloqueia à espera da resposta do servidor. Esta classe actualiza a interface gráfica consoante a resposta do servidor, em caso de erro é criado um *pop-up* através de um objecto *Toast*. Esta classe conecta-se ao servidor através de um *HttpClient*. Para mais detalhes sobre as classes de mensagens enviadas ao servidor consulte ao ponto 3.9.9.9.

3.5.8. Action bar e retro compatibilidade

Aplicação foi compilada na versão 4.1 do Android para assegurar uma melhor experiência de utilizador e aproveitar toda as potencialidades do sistema Android, tendo sido testada na versão 2.3.6. Para assegurar que aplicação corre em versões mais antigas foi implementado um sistema de retro compatibilidade através da biblioteca SherlockActionBar que contém biblioteca de suporte da Google e ainda suporte para *action bar*.

Figura 31 – Sherlock Action Bar implementada na aplicação Android

3.5.9. Implementação de baixo nível

Aplicação Android é composta por duas packages: *core* e *support*. A packages *support* contém várias funcionalidades e dados partilhados pelas diversas *activities* e *fragments,* enquanto a *core* contém as funcionalidades e dados destinados apenas a cada módulo da aplicação. No anexo 7.5 pode averiguar todas as packages e respectivas classes.

3.5.9.1. Módulo core.start

Quando se inicia aplicação é iniciada a classe *Launching* que possui a sua interface gráfica com ecrã preto. Esta classe verifica se os directórios e os ficheiros do mapa existem no armazenamento externo através do método *folderExists()*. A implementação deste armazenamento depende do fabricante do *smartphone* e tanto pode ser um cartão SD ou uma drive *flash*. Para ultrapassar esta ambiguidade utiliza-se a instrução *Environment.getExternalStorageState()* que permite aceder a todos os tipos de armazenamento externo. Se os directórios não existirem então serão criados. Se os directórios existirem mas o ficheiro dos mapas não for encontrado então aplicação procederá notificará o utilizador e procederá à instalação através do método *copyMapsToSD()*. A notificação é implementada através de um *ProgressDialog* que permite ao utilizador verificar a percentagem da instalação já realizada. Para actualizar o *ProgressDialog* e simultaneamente instalar os ficheiros utilizou-se uma *MyAsyncTask* que estende uma *AsyncTask* e executa o método *doInBackground()* quando é criada. Neste método os mapas contidos no ficheiro da aplicação são localizados através de

um *AssetManager*. De seguida são abertos com um *InputStream* e escritos para o disco com *OutputStream*. Após concluída a instalação dos mapas ou caso os mapas já estejam instalados é verificado se o utilizador já está registado no sistema. Esta verificação é feita através das *SharedPreferences* e do método *getString()* que verifica se a chave *userName* possui valor associado. Se devolver um valor então aplicação transitará para actividade *MainMenu*, caso contrário transitará para actividade *Login*.

Na actividade *Login* o utilizador poderá inserir as suas credenciais nas *EditText* do ecrã ou então aceder à funcionalidade *Register* através do atalho na *action bar*. Se inserir as credenciais e pressionar o botão então será enviado um pedido de login para o servidor. Se forem aceites então serão gravadas nas *SharedPreferences* através de um *SharedPreferences.editor* e a aplicação transitará para actividade *MainMenu*, se forem recusadas então utilizador será notificado que as credenciais estão erradas e poderá inserir novamente os dados.

Caso o utilizador tenha optado por registar então será criada a actividade *Register* na qual o utilizador poderá inserir as novas credenciais e as enviar para o servidor. Estas serão armazenadas nas *SharedPreferences* através de um *SharedPreferences.editor* após o sucesso do registo. Finalizado o registo a aplicação transitará para actividade *MainMenu*.

Actividade *MainMenu* é a principal da aplicação, após as credenciais do utilizador estarem armazenadas será sempre a segunda actividade a ser executada, sendo precedida da *Launching*. Esta actividade possui quatro *ImageViews* que detectam eventos sobre si mesmas e cria um objecto *AppSharedVars* (consulte a secção 3.5.9.8 para mais detalhes sobre *AppSharedVars*).

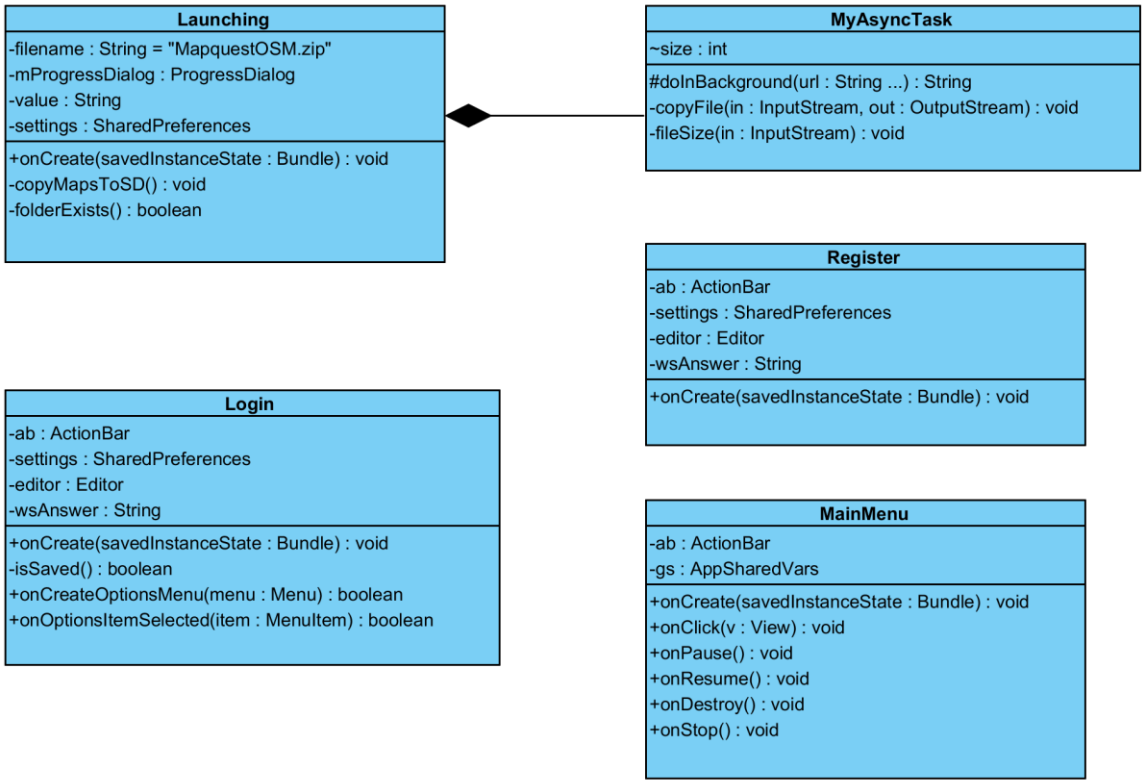

Figura 32 – Diagrama de classes do módulo core.start

3.5.9.2. Módulo core.findnearme

Esta actividade será criada caso o utilizador tenha seleccionado a opção Near me. O *layout* não está definido num ficheiro XML e portanto é gerado dinamicamente na classe. A visualização do mapa é realizada através do objecto *MapView* da biblioteca OSMdroid. Através deste objecto define-se a origem dos dados remotos e locais. A ligação à internet foi cancelada através do método *setUseDataConnection()* para evitar consumo de tráfego, visto que os mapas estão armazenados localmente. Após esta configuração é gerado uma mensagem no ecrã do utilizador com um objecto progressDialog notificando que aplicação está à procura de sinal GPS. Após a construção da notificação inicia-se a procura da localização. Utilizou-se o método *requestSingleUpdate*() do objecto *LocationManager* para obter uma única posição, desactivando de seguida o sistema GPS e poupando assim a bateria. Após a recepção da posição é enviado um pedido para o servidor, através do método *findEventsNearMe*(), com a localização do utilizador. O servidor responde com uma mensagem que contém três listas: dados dos eventos, dados dos POIs, dados das rotas. As rotas são visualizadas através de um *PathOverlay* que permite configurar a cor da linha que representa o caminho e é adicionado à *Mapview*. Cada evento e POI são representados através de um *OverlayItem* que permite definir o texto a ser visualizado quando os ícones dos eventos ou POIs são pressionados. Todos os *OverlayItem* são adicionados a um *ItemizedOverlayWithFocus* que permite a detecção de toques nos ícones e visualizar os dados contidos no respectivo *OverlayItem*.

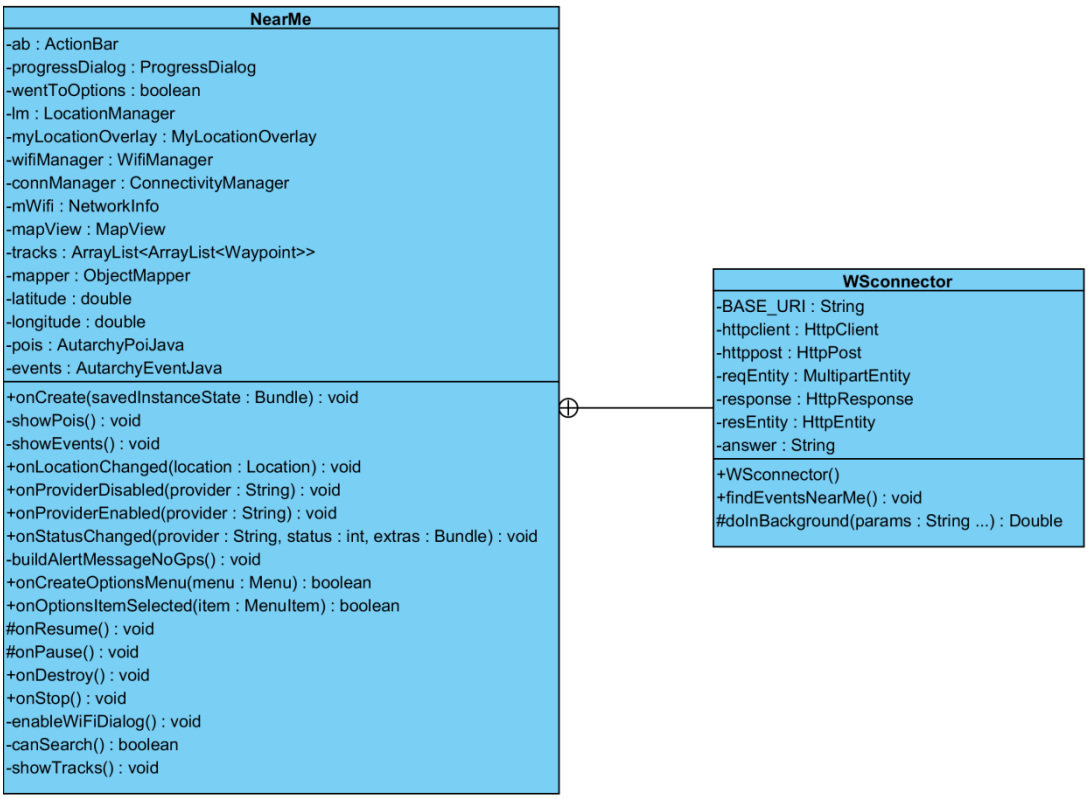

Figura 33 – Diagrama de classes do módulo core.findnearme

3.5.9.3. Módulo core.allphotogallery

Actividade principal, *FullGalleryActivity* utiliza um *GridView* e obtém uma lista das fotografias contidas no directório das fotografias da aplicação. De seguida pesquisa no *MediaStore* as pré-visualizações das fotografias e todas são carregadas para memória. Após finalizar o carregamento são passadas para o ImageAdapter, que as associa à *view* para que possam ser visualizadas. Também as associa a um método que se activa caso uma pré-visualização seja pressionada. Quando uma imagem é pressionada o método dispara, criando um *Intent* para iniciar a actividade *FullImageActivity*, e associa ao *Intent* o nome da fotografia. Esta nova actividade recebe o nome da fotografia, carrega-a para memória e associa à *view*.

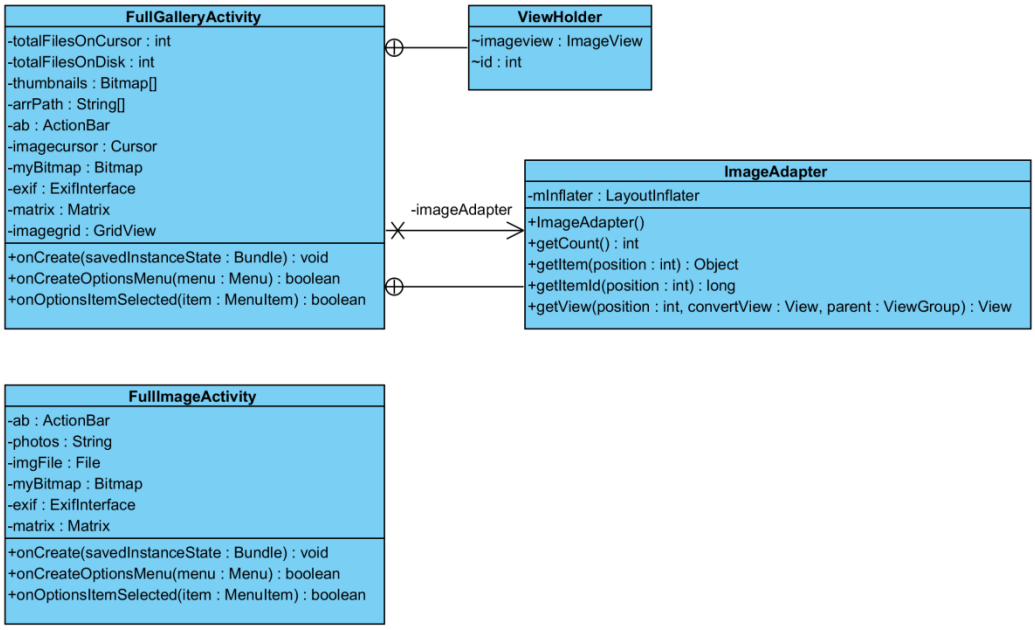

Figura 34 – Diagrama de classes do módulo core.allphotogallery

3.5.9.4. Módulo core.mytracks

O diagrama de classes deste módulo está disponível no anexo 7.6

O utilizador iniciará este módulo quando escolher opção My tracks. A actividade *TrackList* é iniciada mas o seu conteúdo é dinâmicos e como tal foi necessário implementar o seu layout em código java. A classe *TrackList* estende a *SherlockListActivity* permitindo criar uma actividade com formato de lista dinâmica e lidar com eventos. Esta classe acede à base de dados através do *DBConnector* (consulte seção 3.5.9.10 para mais detalhes sobre *DBConnector*), e recebe uma lista com os nomes das rotas e o respectivo estado *uploaded/not uploaded*. Estes dados são enviados para um *BaseAdapter* que gere o layout de cada objecto da lista. Para definir o formato de cada elemento da lista utilizou-se um layout no qual se definiu uma *TextView* e uma *ImageView*, para visualizar o nome da rota e a imagem de estado respectivamente. Se o estado for *uploaded* então a imagem não será visível, caso seja *not uploaded* então a imagem será visível. Finalmente o *BaseAdapter* é adicionado a uma *ListView* e a lista é visualizada pelo utilizador.

Os eventos da *TrackList* são geridos pela *ListView* através da adição de um *OnItemLongClickListener* para lidar com os eventos pressionar durante alguns segundos e um *OnItemClickListener* para lidar com os eventos de pressionar uma vez rapidamente. Caso o

OnItemLongClickListener seja activado então será gerada um menu pop-up com a opção de apagar rota ou sincronizar com servidor. Caso opte por sincronizar então os dados serão enviados para o servidor através do protocolo HTTP e do respectivo método POST. A mensagem é construída através do objecto *MultipartEntity* que permite associar *String* e imagens. Se o *OnItemClickListener* for activado então será criado um *Intent* que iniciará uma actividade *SavedTabsWithSwipe*. Esta actividade estende uma *SherlockFragmentActivity*, que permite utilização de fragmentos. Com os fragmentos divide-se a interface gráfica em várias componentes, sendo cada uma gerida pelo respectivo fragmento. Esta classe possui um *ViewPager* e um *TabsAdapter*. O *ViewPager* detecta os gestos swipe enquanto o *TabsAdapter* é responsável por construir as tabs, associar um fragmento a cada uma, implementar a visualização do menu de tabs e detectar eventos nesse menu. O *ViewPager* também é integrado no TabsAdapter, deste modo é possível transitar para outra tab através do gesto swipe. O menu de tabs disponibiliza acesso a dois fragmentos: *SavedTrackMap* e *SavedTrackStats*. O fragmento *SavedTrackStats* mostra algumas estatísticas da rota através de *TextViews*. O *SavedTrackMap* mostra os pontos geográficos da rota e a localização dos POIs. As rotas são visualizadas através de um *PathOverlay* que é adicionado à Mapview. Os POI são representados através de um *MyOverlayItem*. Esta classe estende *OverlayItem* e implementa *Serializable*. Através do OverlayItem obtém-se acesso a construtores através dos quais se define o texto a ser visualizado quando os ícones dos POIs são pressionados. Todos os *MyOverlayItem* são adicionados a um *ItemizedOverlayWithFocus* que permite a visualização dos dados. É também adicionado um *OnItemGestureListener* para detecção de toques nos ícones e visualizar os dados contidos no respectivo *MyOverlayItem*. Se o gesto for um simples toque rápido então surgirá um balão com o nome e descrição, caso seja um toque longo então será iniciada uma nova actividade, *Gallery*, através de um *Intent*. Também se associa a lista dos nomes das fotografias do POI ao *Intent*. A nova classe recebe lista de fotografias, de seguida pesquisa no *MediaStore* as pré-visualizações das fotografias e todas serão carregadas para memória. Após finalizar o carregamento são passadas para o *ImageAdapter* que as associa à view para finalmente serem visualizadas. Também as associa a um método que se activa caso uma pré-visualização seja pressionada. Quando uma imagem é pressionada o método dispara, criando um *Intent* para iniciar a actividade *FullImageActivity*, e associa o nome da fotografia ao respectivo *Intent*. Esta nova actividade recebe o nome da fotografia, carrega-a para memória e associa à *view*.

3.5.9.5. Módulo core.tracking

O diagrama de classes deste módulo está disponível no anexo 7.7 A actividade principal desta opção, *RealTimeTabsWithSwipe*, estende uma *SherlockFragmentActivity*, que permite utilização de fragmentos. Com os fragmentos divide-se a interface gráfica em várias componentes, sendo cada uma gerida pelo respectivo fragmento. Esta classe possui um *ViewPager* e um *TabsAdapter*. O *ViewPager* detecta os gestos swipe enquanto o *TabsAdapter* é responsável por construir as tabs, associar um fragmento a cada uma, implementar a visualização do menu de tabs e detectar eventos nesse menu. O *ViewPager* também é integrado no *TabsAdapter*, deste modo é possível transitar para outra tab através do gesto swipe. O menu de tabs disponibiliza acesso a três fragmentos: *RealTimeTrackingMap*, *RealTimeTrackingStats* e *POIcreation*. O fragmento *RealTimeTrackingStats* mostra algumas estatísticas da rota. O *RealTimeTrackingMap* mostra os pontos geográficos da rota e a localização dos POIs. O fragmento *POIcreation* permite criar pontos de interesse.

A visualização dos dados no *RealTimeTrackingStats* é realizada através de *TextViews*. Neste ecrã também é possível definir a rota através de um *Spinner* que permite apresentar o valor actualmente escolhido e se o utilizador o pressionar então surgirá automaticamente um pop-up com as opções disponíveis e após a selecção de um valor guarda a opção escolhida e

actualiza o valor ao utilizador. Este ecrã possui um *ToggleButton* que permite activar e desactivar o registo de rota e automaticamente mudar o aspecto gráfico do botão quando é pressionado. No final deste subcapítulo pode verificar como é realizado o registo de localização.

As rotas são visualizadas *RealTimeTrackingMap* através de um *PathOverlay* que é adicionado à sua Mapview. Os POI são representados através de um *MyOverlayItem*. Esta classe estende OverlayItem e implementa *Serializable*. Através do OverlayItem obtém-se acesso a construtores através dos quais se define o texto a ser visualizado quando os ícones dos POIs são pressionados. Todos os *MyOverlayItem* são adicionados a um *ItemizedOverlayWithFocus* que permite a visualização dos dados.

No fragmento *POIcreation* é possível inserir os dados do POI nos campos *EditText*. O acesso à camara fotográfica é realizado através de um botão. De seguida cria-se um *Intent* e é criada uma nova actividade através do método *startActivityForResult()* Este método possibilita que a nova actividade envie dados para a anterior quando a actual terminar. Na nova actividade pode-se visualizar a imagem da câmara em tempo real através de uma *SurfaceView*. Quando o utilizador carregar no botão de registar fotografia então a imagem é registada, o nome da foto é adicionada ao *MediaStore* ficando imediatamente disponível no sistema de ficheiros. Adição ao *MediaStore* permite também gerar automaticamente um ficheiro de pré-visualização. Quando o utilizador executar o comando back a actividade actual envia o nome da fotografia da anterior. A actividade actual é então terminada, e a anterior passa a ser novamente a actividade actual e irá mostrar a ultima fotografia numa *ImageView*. Caso o utilizador pressione o botão de salvar então os dados são armazenados e todos os campos serão limpos. Não é possível retroceder este passo. Caso o utilizador opte por partilhar as rotas, será criado um menu através de um *AlertDialog*. Se o utilizador seleccionar Twitter, os dados de login serão pesquisados nas preferências partilhadas. O login será automático caso os seus dados sejam encontrados. Caso não sejam encontrados, será aberta uma página web dentro de uma *WebView*, na qual o utilizador necessita de inserir as credenciais. Os dados são armazenados nas preferências partilhadas para utilização futura assim que o servidor responder com o token *Oauth*. De seguida a aplicação envia os dados num pedido HTTPS e o ecrã do menu será removido assim que o envio dos dados terminar. Caso o utilizador seleccione partilha no Facebook, o SDK realiza a autenticação através da aplicação do Facebook. Após o sucesso os dados são enviados automaticamente para a rede social.

Nesta opção o registo da posição é realizado através de um *Service* pois os dados necessitam de ser partilhados entre os vários fragmentos. O serviço arranca quando o botão de início de registo de rota é activado e é destruído quando o botão de início de rota é novamente pressionado. Este serviço utiliza um *LocationManager* e o seu método *requestLocationUpdates()*. Nesta função configura-se a frequência mínima de actualização do sinal através do tempo ou distância, no entanto a frequência real dependerá da qualidade do sinal e da capacidade de processamento do dispositivo móvel. Definiu-se que a frequência mínima da actualização da posição será de 1 segundo ou 1 metro para detectar todos os pontos da rota. Para lidar com os eventos do LocationManager utilizou-se um implementou-se um *LocationListener*, definindo as acções no método *onLocationChanged()*. Neste método cria-se um objecto *Intent* no qual se armazena os dados GPS e invoca-se o método *sendBroadcast* da componente *Service* para enviar os dados para diferentes componentes da interface gráfica que estão a executar concorrentemente e que não podem ficar pendentes entre si. Os fragmentos recebem as mensagens do serviço através de um *BroadcastReceiver*.

3.5.9.6. Módulo core.socialnetworks

Através deste módulo os POIs são partilhados nas redes sociais. A classe *FacebookUpload* é responsável por os partilhar no Facebook enquanto a classe *MainActivity* é responsável pela partilha no Twitter.

A actividade *Twitter* lê as preferências partilhadas através to método e verifica se o utilizador autenticou alguma vez na rede social através da aplicação Android. Se não forem encontrados esses dados então aplicação transitará para um *WebView* que permite acesso à página de autenticação do Twitter. Os tokens Oauth são armazenados nas preferências partilhadas para uso futuro caso autenticação obtenha sucesso, caso contrario o utilizador necessitará de realizar o login no Twitter novamente. Após a autenticação no Twitter ou caso sejam detectados os tokens Oauth nas preferências partilhadas, a aplicação enviará o pedido com o POI. Para autenticar aplicação é necessário implementar um ConfigurationBuilder que contém dados de autenticação da aplicação no Twitter e TwitPic. A autenticação do utilizador é implementada através dum objecto *AccessToken*, que contém os dados do token Oauth. O *ConfigurationBuilder* e o *AccessToken* são depois adicionados a um objecto *Twitter* concluindo assim a configuração da autenticação. Para enviar os dados do POI utiliza-se um objecto *StatusUpdate* que no seu construtor recebe todos os dados de texto do POI, a imagem é associada através do método *setMedia*(). A imagem tem de ser previamente carregada para memória através de um File. Para realizar o pedido HTTPS ao Twitter utiliza-se o método *updateStatus*(), do objecto *Twitter*, e que recebe por parâmetro o objecto *StatusUpdate*. Este método devolve uma resposta no formato Response da API Twitter4j. Após concluído o envio e obtida a resposta aplicação retorna para a vista POI do ecrã *Track*.

Actividade *FacebookUpload* utiliza aplicação do Facebook para autenticar os utilizadores. Utiliza um Session e o seu método *getActiveSession*() para obter os dados de autenticação no Facebook. A imagem a enviar é carregada para um *Bitmap* através do método *BitmapFactory.decodeFile()* que recebe por parâmetro a localização da fotografia. Após carregada a imagem cria-se um *Bundle* no qual se armazena a respectiva imagem e os dados de texto do POI. Este *Bundle* é posteriormente associado a um objecto *Request*. Para finalmente enviar os dados executa-se o método assíncrono *executeAsync*() do *Request*. Este método trata do enviar os dados e retornar o sucesso ou insucesso da operação. Após a chamada deste método a aplicação retorna para a vista POI do ecrã *Track*

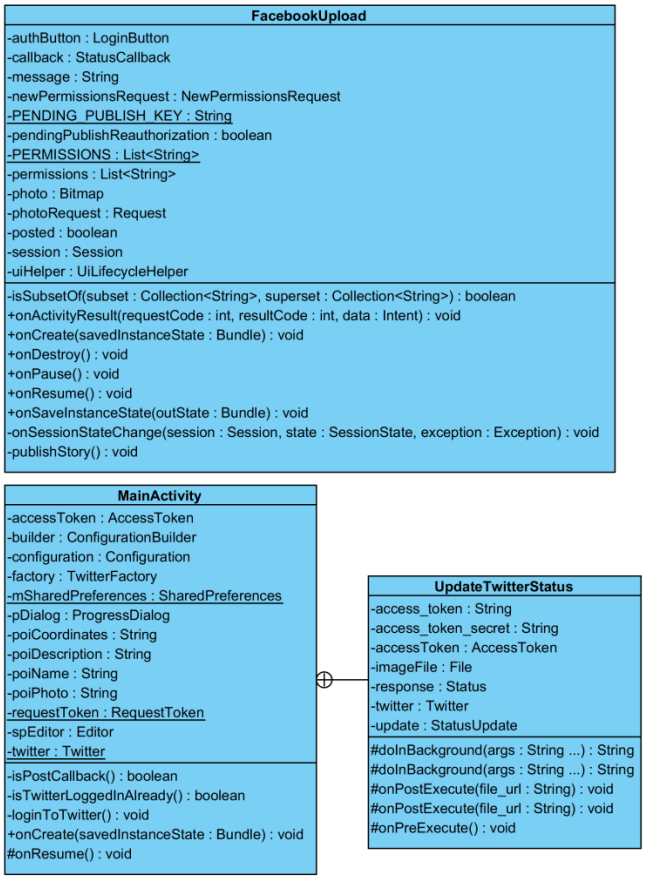

Figura 35 – Diagrama de classes do módulo core.socialnetworks

3.5.9.7. Módulo support.utilities

Na classe *Constants* define-se diversas constantes utilizadas na aplicação. A classe *Functions* possui funções de conversão de tempo, partilhadas por diversas actividades. No anexo 7.9 pode consultar o respectivo diagrama de classes

3.5.9.8. Módulo support.shared

Este módulo possui diversos objectos utilizados por diversas componentes da aplicação Android. A classe *AppSharedVars* permite a partilha de dados entre as diferentes actividades. Através desta classe é possível partilhar a ligação à base de dados para todas as actividades. No anexo 7.10 pode consultar o respectivo diagrama de classes

3.5.9.9. Módulo support.webservice

As classes *OutMessage* e *NearMeMessage* são utilizadas em pedidos que necessitam de enviar listas, sendo depois convertidas para JSON através do método *writeValueAsString*() do objecto ObjectMapper. Apos a construção da mensagem, esta é associada a um objecto HttpPost através do método *setEntity*(). O pedido é executado através do método *execute*() do HttpClient e contendo o pedido HttpPost. Este método devolve um objecto HttpResponse do qual se pode extrair a resposta através do seu método *getEntity*() que devolve um HttpEntity. A resposta pode ser convertida para String através do método *EntityUtils.toString()*.

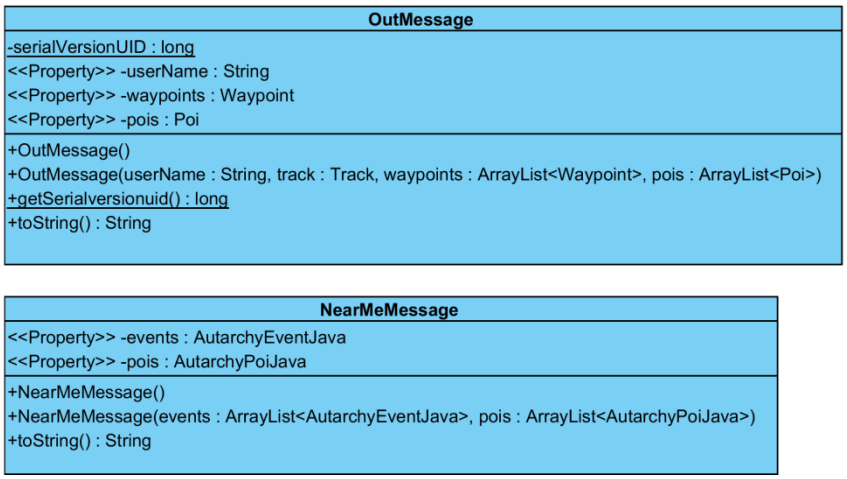

Figura 36 – Diagrama de classes do módulo support.webservice

3.5.9.10. Módulo support.database

Para visualizar o diagrama de classes consulte o anexo 7.11. As tabelas foram definidas na classe *GPSdatabase* que estende um *SQLiteOpenHelper*. Através do construtor define-se o nome do ficheiro que contém todos os dados. No método *onCreate()* acede-se a um objecto *SQLiteDatabase* que se liga directamente à base de dados. Através do método *execSQL()* do *SQLiteDatabase* executa-se uma *querie* para cada tabela, definindo-as e armazenando as alterações no ficheiro. Para realizar *queries* de pesquisa, inserção ou remoção implementou-se uma classe auxiliar, *DBconnector*. No seu construtor acede-se ao objecto *GPSdatabase* e às suas configurações através do método *getWritableDatabase(),* que retorna um objecto *SQLiteDatabase*. Através deste objecto procede-se então à consulta e alterações na base de dados sem necessidade de escrever código SQL. Para inserir novos dados utiliza-se um *ContentValues* e o seu método *put()*, que permite associar os nomes das colunas e os respectivos valores. Posteriormente executa-se o método *insert()* do objecto *SQLiteDatabase*, passando por parâmetro os *ContentValues* e o nome da tabela. Para alterar valores na base de dados a abordagem é semelhante à inserção no entanto utiliza-se o método *update()*, com possibilidade de definir uma condição para que linhas deve ser realizada a actualização. Para remover dados utiliza-se o método *delete()* no qual se indica o nome da tabela e a condição de remoção. As pesquisas são realizadas através do método *query()* , no qual se define a tabela em questão, os campos que se pretende devolver e a condição de pesquisa. A execução deste método é devolve um Cursor que contém todos os dados representados em *String*.

3.6. Servidor

Na imagem seguinte ilustra-se abordagem 3-tier do servidor assim como algumas tecnologias utilizadas na sua implementação.

Figura 37 – Arquitectura 3-tier do servidor

3.6.1. Camada de dados

Para auxiliar o desenvolvimento do sistema foi realizado um estudo sobre diferentes modelos de base de dados. Na tabela seguinte encontra-se uma análise comparativa de SQL e NoSQL realizada através dos artigos de (Hoff, 2010) e (Leavitt, 2010).

	NoSQL	**SQL**
Modelo de dados	Pares chave-valor, grafos, documentos, colunas	Relacional
ACID	Não	Sim
Tempo Inserção	Mais rápido	Mais lento
Tempo de acesso	Mais rápido	Mais lento
Escalável	Facilmente distribuída	Dificilmente distribuída
Maturidade	Menor	Maior
Conhecimento do público geral	Menor	Maior
Tempo de aprendizagem	Menor	Maior

Tabela 9 – Comparação de sql com nosql

Após análise da tabela anterior optou-se por utilizar SQL pois permite obter consistência dos dados e possuem dados relacionais. Também apresenta maior maturidade e são de maior conhecimento público tornando a manutenção mais fácil. Após esta decisão realizou-se uma pesquisa de sistemas de gestão de base de dados que possuíssem suporte para dados geográficos. Nesta pesquisa foi tida em conta os tipos de dados geográficos, funções aplicáveis a esse tipo de dados e a documentação. Decidiu-se utilizar Postgres com a extensão PostGIS pois contém os dados geográficos necessários para armazenar as rotas e POIs, uma boa documentação e várias funções úteis para o projecto tais como: intersecção de pontos e linhas, compressão de linhas (redução do número de pontos). Todos os dados excepto as fotografias serão armazenados na base de dados, as fotografias são armazenadas directamente no disco num directório para cada utilizador.

De seguida pode-se consultar um diagrama que elabora as packages e classes utilizadas na criação das base de dados:

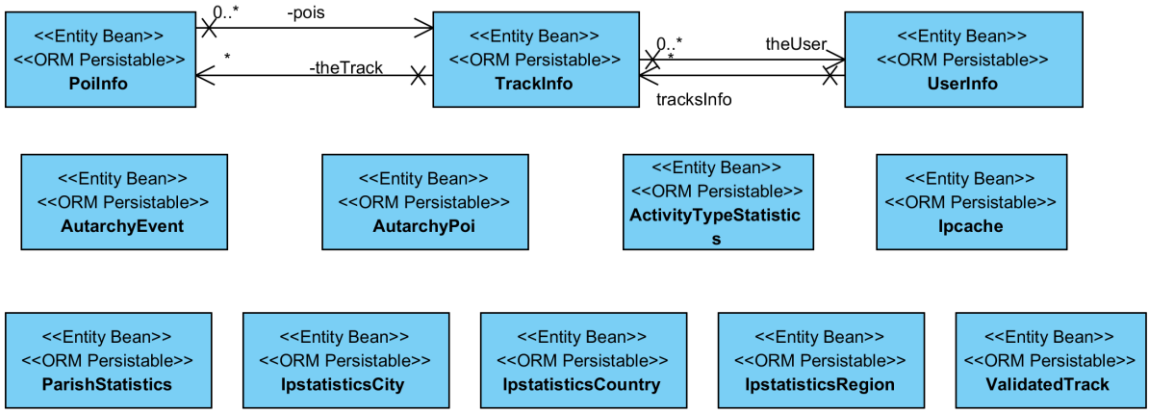

Figura 38 – Diagrama de classes que definem as tabelas base de dados

No anexo 9.8 pode consultar este diagrama mais detalhado. As classes *PoiInfo*, *TrackInfo* e *UserInfo* são responsáveis pela persistência dos dados recebidos da aplicação Android. As classes *IpstatisticsRegion*, *IpstatisticsCountry*, *IpstatisticsCity* armazenam as estatísticas obtidas através de geolocalização dos IPs. A classe *ParishStatistics* possui as estatísticas das autarquias. As classes *AutarchyPoi* e *AutarchyEvent* contêm os dados inseridos pela autarquia e que podem ser enviados para os clientes Android. A classe *ValidatedTrack* possui as rotas validadas e comprimidas pela autarquia.

3.6.2. Camada lógica

Nesta camada estão definidos os diversos métodos que permitem realizar todas as operações com os clientes Android e com a página web da autarquia assim como persistir os dados recebidos de todos os clientes.

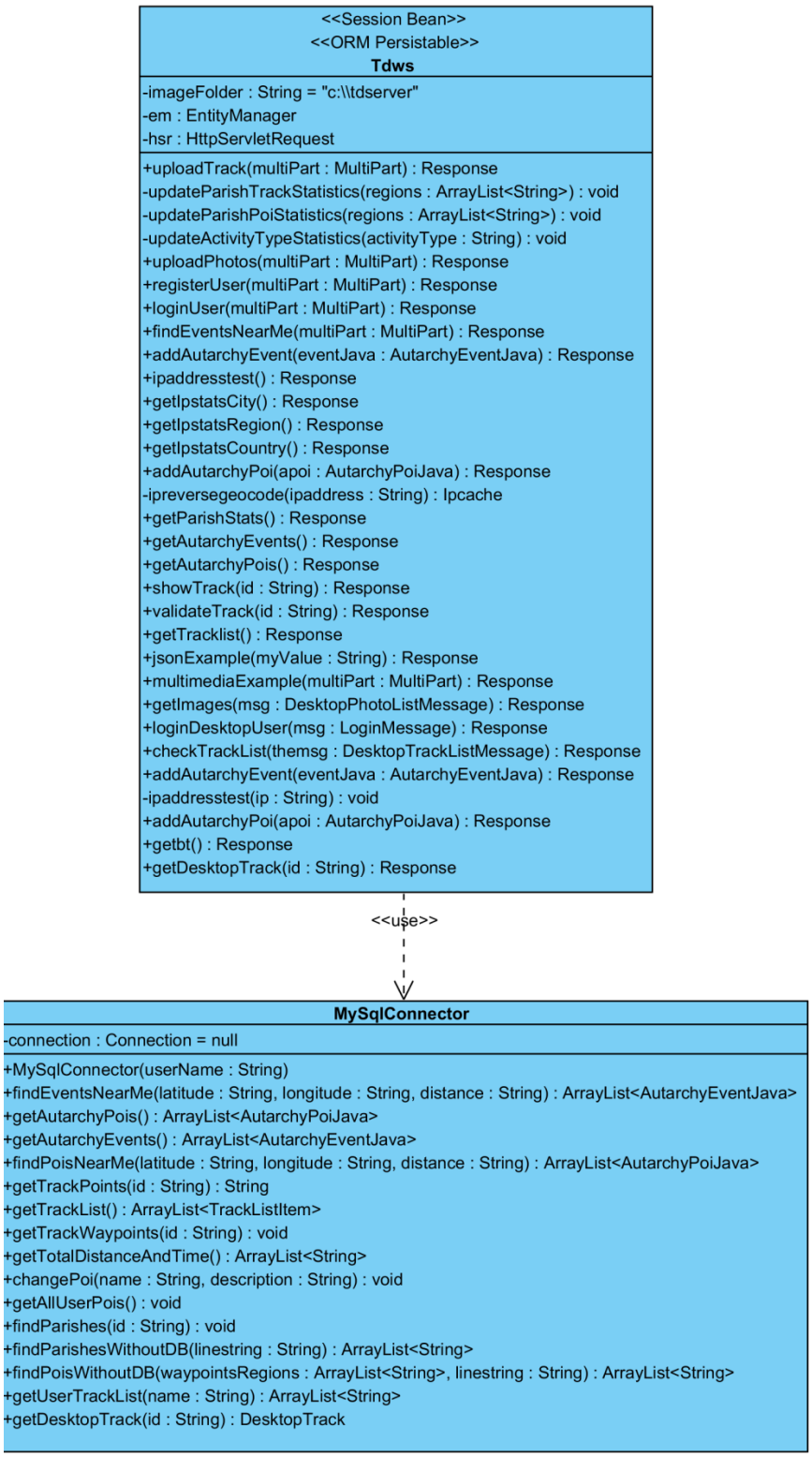

Figura 39 – Diagrama de classes da camada lógica

3.6.2.1. Persistência e pesquisa de dados

O servidor tem acesso a dois tipos de armazenamento: base de dados e sistema de ficheiros. O servidor contém as imagens em memória num objecto *BufferedImage* e para as armazenar necessita de reservar o espaço através de um File, em seguida é persistida através do método *write()* da classe *ImageIO*.

As tabelas da base de dados são definidas por entidades, os seus dados não geográficos são definidos em objectos nativos java com anotações de JPA, os dados geográficos são definidos com dados geométricos da ferramenta Vividsolutions, com anotações do hibernate-spatial. São utilizados dois tipos de dados geométricos da ferramenta Vividsolutions, *Point* para armazenar localização de um POI e dos diversos pontos da rota, *LineString* para armazenar as rotas validadas pela autarquia.

A ferramenta Vividsolutions não está disponível em Android portanto foi necessário criar objectos java para armazenar os pontos das rotas e dos POIs. Consequentemente é necessário converter as posições registadas no Android para os dados da Vividsolutions. Esta conversão é efectuada através de um *WKTReader* e do seu método *read()* que recebe os pontos no formato WKT (Well-Known Text).

```
POINT(1 2 3)
LINESTRING(1 2 3, 4 5 6, 7 8 9)
```

Figura 40 – Exemplo de um Point e Linestring no formato WKT

A interacção entre a base de dados e o servidor é realizada através de um *EntityManager* e da classe *MySQLconnector*. O método *persist()* do *EntityManager* permite notifica-lo que existem dados para serem escritos no entanto só são realmente escritos após a chamada do método *flush()*. As pesquisas por chave primária são executadas através do método *find()* do *EntityManager*. As pesquisas restantes foram implementadas na classe *MySQLconnector* devido a problemas no acesso a dados geométricos do PostGIS, o servidor dava erro indicando que não reconhecia os tipos de dados embora os controladores estivessem instalados. Esta nova solução devolve os dados geométricos no formato WKT evitando assim o erro previamente descrito. Alguns métodos necessitam de persistir grandes quantidades de dados geométricos, para evitar diminuir o tempo de execução do método i os dados são armazenados várias vezes durante a execução do método. Inicialmente utiliza-se o método *persist()* seguida do i para guardar alguns dados, à medida que se vão processando novos dados utiliza-se o método *merge()* do *EntityManager* seguido do *flush()* que permite sincronizar o objecto contido em memória com o da base de dados.

As pesquisas implementadas na classe *MySQLconnector* utilizam um *Connection* para se ligarem à base de dados. As *queries* são definidas através do objecto *Statement* que é inicializado com método *createStatement(),* do objecto *Connection*. A pesquisa é realizada através do método *executeQuery()* do objecto *Statement* e recebe por parâmetro a respectiva *querie* SQL. A execução deste método devolve os resultados para um *ResultSet*. Este objecto é iterado e os dados de cada coluna são acedidos através do método *getString()* que recebe por parâmetro o nome da coluna que se pretende extrair.

3.6.2.2. Estatísticas

Para implementar as estatísticas das freguesias foi necessário recolher dados que as definissem num mapa. A Carta Administrativa Oficial de Portugal contém estes dados actualizados no entanto a sua importação não foi simples pois os dados estão no formato European Terrestrial Reference System 1989 enquanto os dados GPS estão no formato World Geodetic System 84. Através do programa QuantumGIS procedeu-se à conversão do sistema referencial dos dados e de seguida importou-se todos os dados através da aplicação Postgis Shapefile and DBF loader exporter. Após o carregamento retirou-se algumas colunas das tabelas que não eram úteis para o projecto.

A detecção é realizada quando um cliente Android sincroniza uma rota com o servidor. É realizada através do método *findParishesWithoutDB(),* que recebe por parâmetro a rota num objecto LINESTRING (WKT). Este método utiliza a função *st_intersects()* do PostGIS para detectar a quais freguesias pertencem os pontos da rota. A função recebe por parâmetro a rota (LINESTRING) e a região (LINESTRING), detecta se alguns dos pontos da rota estão contidos numa região e devolve o nome da região em caso afirmativo. Esta função é invocada para cada região, o contador é actualizado na tabela *ParishStatistics* e as regiões são devolvidas para a classe que invocou o método *findParishesWithoutDB()*. Estas regiões são posteriormente utilizadas no método *findPoisWithoutDB()*. Este método é bastante semelhantes ao *findParishesWithoutDB()* no entanto já se sabe que os POIs só poderão estar situados nas regiões devolvidas pelo método anterior. Deste modo reduz-se o espaço de pesquisa e a função *st_intersects()* só será executada para essas regiões, tornando o sistema mais rápido.

Para implementar estatísticas de IPs recorreu-se ao serviço externo freegeoip.net que permite localizar geograficamente os IPs, identificando a cidade, região, país e outros dados não relevantes para o projecto. Este serviço está acessível por uma interface REST, com troca de dados no formato JSON. O serviço está no entanto limitado a 10.000 pedidos por hora e para reduzir o número de acessos implementou-se uma tabela na qual se guarda um histórico dos IPs localizados geograficamente. Deste modo não será necessário realizar pedidos ao serviço externo visto que os dados do respectivo IP já estão na tabela. Antes de iniciar o pedido ao webservice é verificado se o IP está contido na tabela de *caching* de IP, *IPcache*, através do método *find()* do *EntityManager*. Caso esteja então as estatísticas de IPs são imediatamente actualizadas, caso contrário é necessário aceder ao serviço externo. Para aceder ao serviço utiliza-se um *DefaultHttpClient* e executa-se o seu método *execute()* que recebe um *HttpGet*. No *HttpGet* define-se o caminho do serviço A execução deste método devolve um *HttpResponse*, do qual se pode extrair os dados relevantes através do seu método *getEntity()*. Os dados são posteriormente convertidos para uma *String* JSON através do método *EntityUtils.toString()*. O objecto JSON é então convertido para *GeoIp* (java) através de um *ObjectMapper* e do seu método *readValue()* que recebe por parâmetro o objecto JSON e o tipo de classe. As estatísticas IP são então actualizadas assim como a tabela de *caching*.

3.6.2.3. Compressão de rotas

A compressão das rotas é realizada através da função st_*simplify* do PostGIS. Esta função recebe os dados geográficos de uma rota e a distância máxima entre dois pontos. Com estes dados irá tentar reduzir a quantidade de pontos utilizados para definir a rota sem alterar muito a sua representação geométrica. Esta função utiliza o algoritmo de Douglas-Peucker que normalmente tem complexidade O(n log n) mas no pior caso terá O(n2). Optou-se por não implementar outros algoritmos de compressão pois as alternativas teriam complexidade O(n2) em todos os casos possíveis.

Esta funcionalidade foi implementada no método *validateTrack()* da classe *Tdws*, classe do webservice. Este método pesquisa na base de dados a rota através da sua chave primária e retorna os pontos da rota no formato *Linestring* (WKT), de seguida estes dados são comprimidos através da função. Os pontos são depois convertidos para objecto *Geometry*, da Vividsolutions, através do método *read()* do objecto *WKTReader*. O *Geometry* é depois adicionado a um *ValidatedTrack* que é persistido na base de dados através do método *persist()* e *flush()* do *EntityManager*, deste modo a tabela *ValidatedTracks* é actualizada.

3.6.2.4. Identificação de POIs por proximidade

Para detectar POIs, eventos e rotas na proximidade do utilizador recorreu-se à função *st_dwithin()*. Esta função recebe por parâmetro as coordenadas GPS do utilizador, a localização de um POI, evento ou rota validada e a distância máxima entre dois pontos. Com estes verifica se a distância das duas localizações é igual ao inferior à inserida.

Esta funcionalidade foi implementada no método *findEventsNearMe()* da classe *Tdws* que recebe as coordenadas do cliente Android e a distância. De seguida invoca os métodos *findPoisNearMe()* e *findEventsNearMe(),* da classe *MySqlConnector* , para identificar respectivamente os POIs e eventos nas proximidades do utilizador. Estes métodos utilizam a *st_dwithin()* para detectar os eventos e POIs e devolvem a informação desses mesmo elementos, sendo os dados geográficos devolvidos no formato Linestring (WKT). As rotas validadas são obtidas com metodologia semelhante aos eventos e *POIs* no entanto o acesso à base de dados é efectuado através do *EntityManger* que retorna os ids das rotas nas proximidades e posteriormente é efectuado um acesso para retirar as coordenadas das diversas rotas. Todos os dados são adicionados a uma mensagem *NearMeMessage* que será enviada para o cliente Android, através da instrução *Response.status(Response.Status.ACCEPTED).entity(nearMeMessage).type(MediaType.APPLICATION_JSON).build()*.

3.6.3. Camada de apresentação

A camada de apresentação é constituída por duas componentes distintas, uma página web para a autarquia e um webservice REST. Este webservice é responsável por comunicar com os clientes Android e desktop enquanto aplicação web permite aceder às funcionalidades para os funcionários da autarquia.

3.6.3.1. REST

A comunicação com os clientes Android e página web é realizada através de um webservice seguindo arquitectura REST. Este é implementado através de um *stateless javabean*, que possui vários métodos POST. Estes métodos recebem maioritariamente dados JSON excepto no caso do envio das rotas da aplicação Android que requer recepção de dados multimédia e portanto utiliza-se um *MultiPart*. A conversão de dados JSON para java e vice-versa é realizada automaticamente pela ferramenta Jersey quando são recebidos directamente pelo método. Todos os métodos retornam um Response que será enviado no formato JSON. Nos métodos que se recebe um *MultiPart* o processo é mais complicado, é necessário extrair cada parte individualmente através dos métodos *getBodyParts().get(n)* e os armazenar num *BodyPartEntity*. Se o objecto esperado for nativo do java ou implementado pelo sistema então será convertido para *String* através do método *getEntityAs(String.class)* do *BodyPartEntity*. Esta *String* contém os objectos no formato JSON mas é necessário utilizar um ObjectMapper e o seu método *readValue()* para os converter para java pois não foram recebidos directamente pelo método. Se o objecto contido no *BodyPartEntity* for uma imagem então é associado a

um *InputStream* que permite que a imagem seja lida através do método *read()* da classe *ImageIO* e armazenado num *BufferedImage*.

```
@POST
@Path("/jsonExample")
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
public Response jsonExample(String myValue) {
    return Response.status(200).entity(myValue).build();
}
```

Figura 41 – Método que recebe dados JSON

```
@POST
@Path("/multimediaExample")
@Consumes(MediaType.MULTIPART_FORM_DATA)
public Response multimediaExample(MultiPart multiPart) throws IOException {
    int n = 10;
    BodyPartEntity imageEntity = null;
    String someValue = multiPart.getBodyParts().get(0).getEntityAs(String.class);
    for (int i = 1; i < n; i++)
        imageEntity = (BodyPartEntity) multiPart.getBodyParts().get(n).getEntity();
    return Response.status(200).entity(someValue).build();
}
```

Figura 42 – Método que recebe dados multimédia

3.6.3.2. Página web

A página web é responsável pela interacção entre servidor e os utilizadores da autarquia. A componente visual foi implementada com recurso a HTML, JSP, CSS e JavaScript em todas as páginas web.

A página com mapas utiliza também a ferramenta OpenLayers, que possibilita utilização de mapas em tempo real do servidor OpenStreetMaps mas só está disponível em JavaScript.

A página de mapas inicia-os assim que é carregada. As rotas são representadas através de um *layer* iniciado com *OpenLayers.Layer.Vector* e que o modo de actualização é forçado, deste modo as rotas no mapa são actualizadas em todos os níveis de *zoom*. Os dados da rota são obtidos através de uma chamada Ajax que contém: o endereço do método a invocar, o nome da rota, o formato dos dados e a codificação da mensagem.

```
var request = $.ajax({
    type: 'post',
    url: 'http://localhost:8080/tdserver/webresources/tdserver/showTrack',
    data: {id: d},
    datatype: 'json',
    contentType: "application/json; charset=utf-8"
});
```

Figura 43 – Invocação Ajax do webservice REST

O servidor responderá com uma rota no formato WKT, nomeadamente uma *Linestring*, que contém todos os pontos da rota. As rotas utilizam o sistema de referência WGS1984 enquanto API utiliza Spherical Mercator Projection, portanto é necessário converter as coordenadas sempre que se pretende inserir um novo objecto no mapa. Após a conversão adiciona-se os pontos ao *layer* através do método *addFeatures()* e finalmente invoca-se o método *refresh()* com parâmetro *force:true* para garantir que o mapa é actualizado em todos os níveis de *zoom*.

As restantes páginas utilizam um módulo que permite a comunicação com servidor:

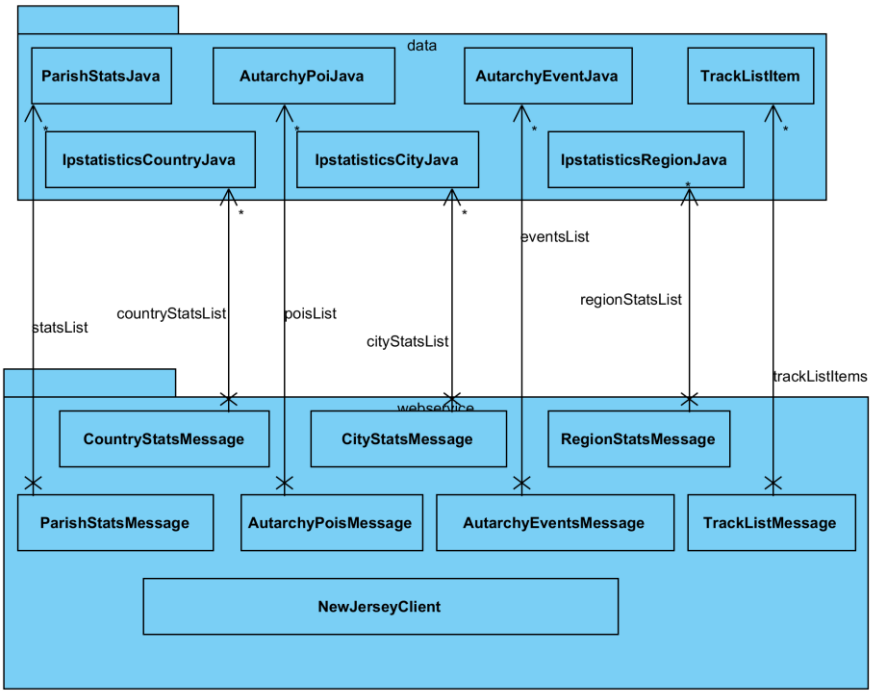

Figura 44 – Diagrama de classes da aplicação web

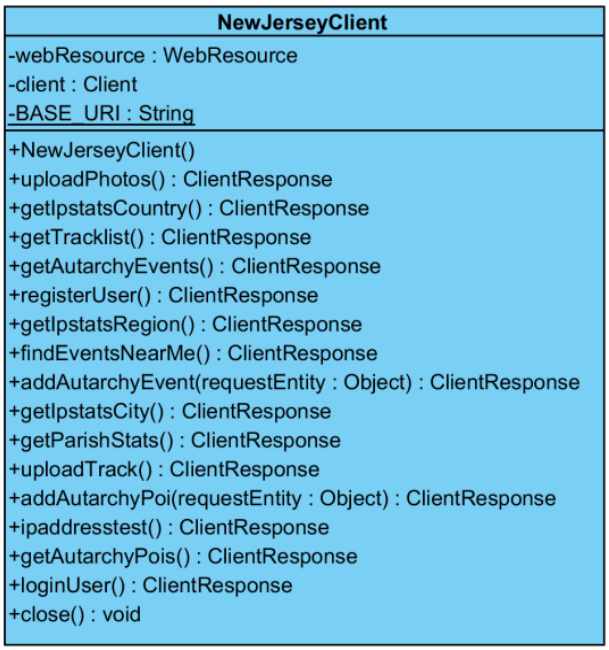

Figura 45 – Classe NewJerseyClient da aplicação web

A package data contém várias classes que definem objectos utilizados na troca de mensagens com o servidor. A package webservice contém diversas classes que implementam listas dos dados definidos na package data e contém também uma classe *NewJerseyClient* responsável pela comunicação com o servidor. Esta classe foi implementada com a ferramenta Jersey e um objecto desta classe é iniciado através do método *NewJerseyClient()*. Este método começa por criar um *ClientConfig* da ferramenta Jersey, de seguida constrói um *Client* através do método *create()* e passando a *ClientConfig* por parâmetro, finalmente a ligação é iniciada através de um *WebResource* obtido através da instrução *client.resource(BASE_URI).path("tdserver")*. Os métodos desta classe são invocados dos JSPs da aplicação web através da construção de um objecto NewJerseyClient, de seguida executa-se o método pretendido e passando por parâmetro a mensagem. O método retorna um *ClientResponse*. Para extrair o contudo relevante utiliza-se o método *getEntity()* do *ClientResponse* que devolverá uma *String*. As respostas estão em JSON portanto utiliza-se um *ObjectMapper* e o seu método *readValue()* que recebe por parâmetro a *String* JSON e tipo da classe para qual se pretende converter.

3.7. Desktop

A package com.mycompany.desktopguimap é responsável pela componente visual de toda aplicação e por ler os dados da base de dados.

A package estagio.webservice é responsável pela comunicação com servidor e contém as mensagens trocadas com o servidor.

A package estagio.data contém dados utilizados na package com.mycompany.desktopguimap e nas mensagens trocadas com o servidor

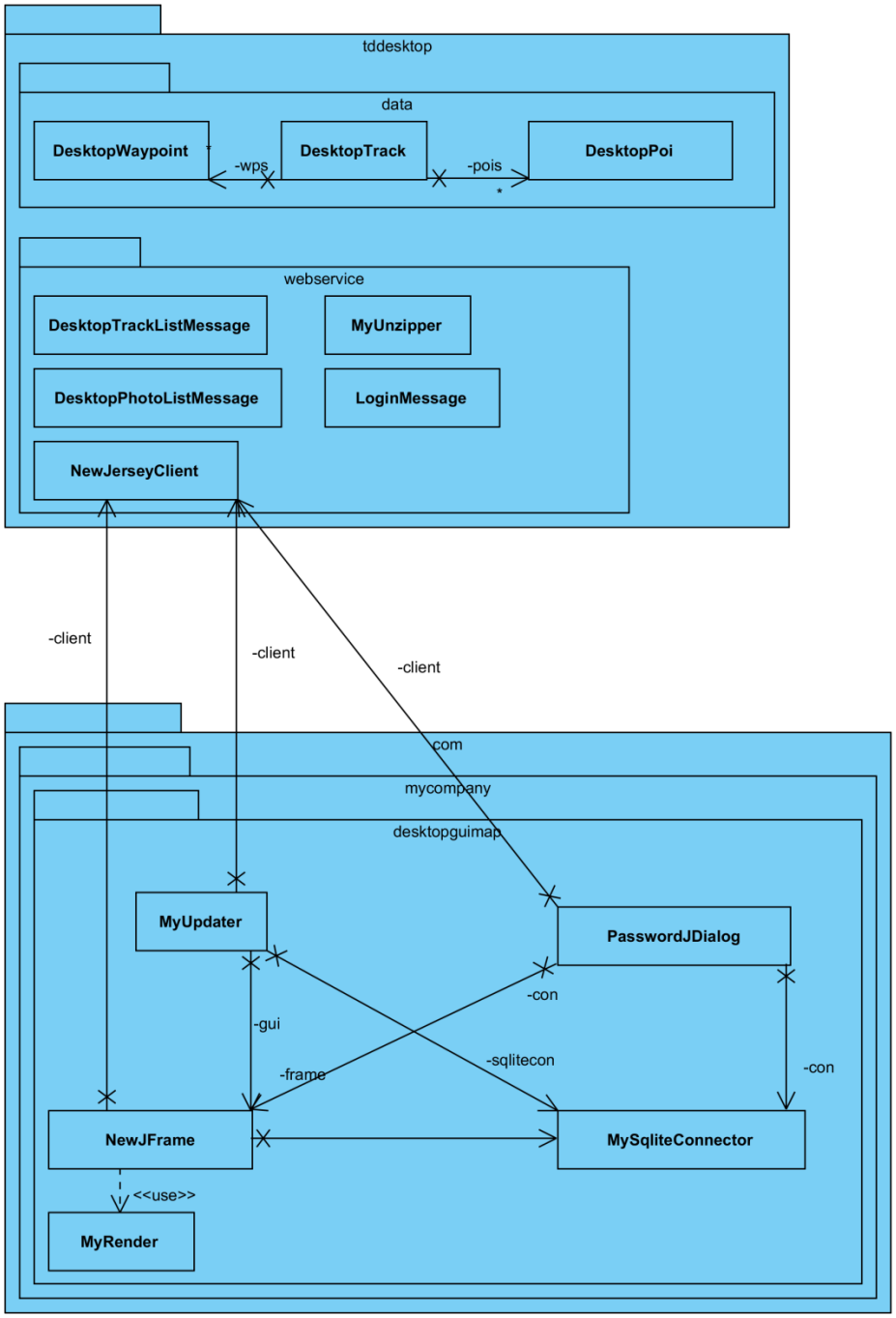

Figura 46 – Diagrama de classes da aplicação desktop

3.7.1. Interface gráfica

A interface gráfica da aplicação foi implementada com recurso a Java Swing, a ferramenta primária para criação de interfaces em JavaEE. Para melhorar a componente de gráficos utilizou-se a biblioteca JFreechart que permite simplificar o desenvolvimento desta componente. Antes de criar os mapas é necessário definir o seu tipo e neste caso utilizou-se um *XYSeriesCollection*, que permite criar um gráfico com eixos X e Y. Após este passo constrói-se as séries de dados através do método *add()* objecto *XYSeries*, que recebe por parâmetro um valor X e um valor Y. O objecto *XYSeries* é finalmente adicionado ao *XYSeriesCollection* através do método *addSeries()*. Para visualizar o mapa adiciona-se o

XYSeriesCollection a um objecto *JFreeChart*. Este novo objecto também permite definir legendas, nome do gráfico, título dos eixos e a orientação do gráfico. Para actualizar um gráfico utiliza-se o método *clear()* do objecto *XYSeries* e adiciona-se os novos valores.

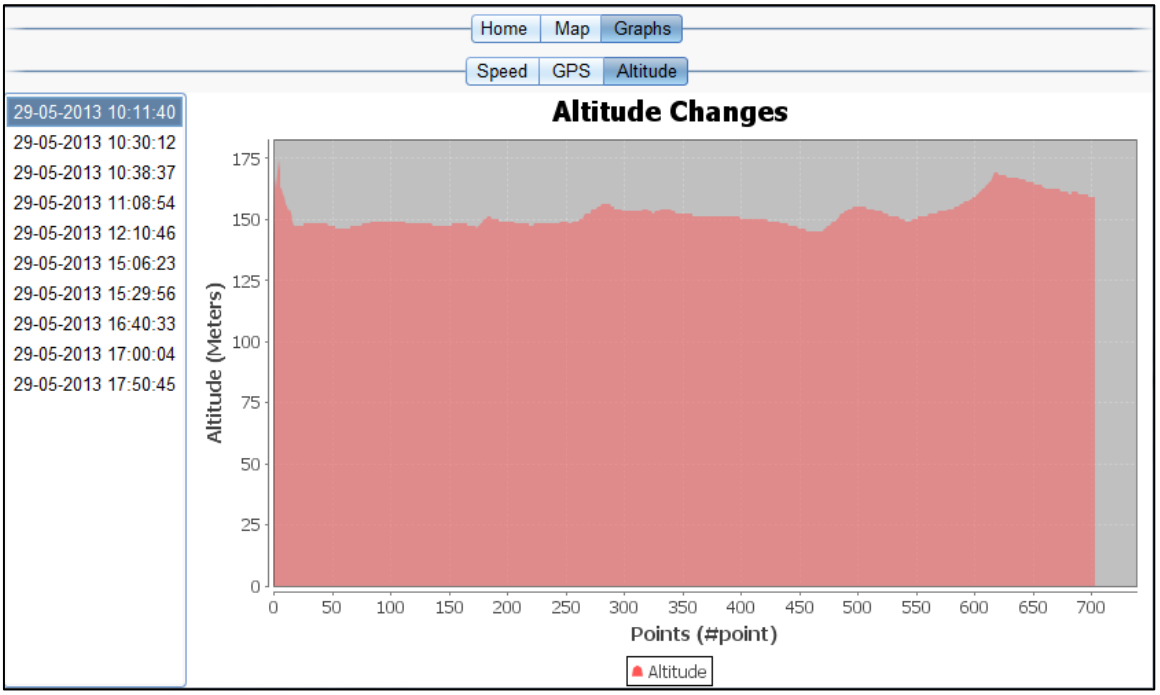

Figura 47 – Gráfico de altitude

Os mapas estão implementados com a biblioteca JXMapViewer. O objecto *JXMapKit* permite criar um mapa e configurar várias opções tais como o fornecedor dos mapas, centrar mapa, definir nível de *zoom*, mostrar mini mapa, mostra controlos de zoom e mostrar *slider* do mapa.

As rotas são criadas através do objecto *Painter<JXMapViewer>* e os POIs são criados através de um *WaypointPainter*. Durante implementação descobriu-se que se forem adicionados directamente ao mapa então só o último adicionado é que estaria visível. Para ultrapassar esta dificuldade adicionou-se o *Painter<JXMapViewer>* a um *CompoundPainter* e o *WaypointPainter* a outro *CompoundPainter*. Finalmente estes dois objectos são adicionados a outro *CompoundPainter*, que é adicionado ao mapa e finalmente todos os dados são visíveis. Foi necessário também desactivar o sistema de *caching* dos *CompoundPainter* para garantir que os dados no mapa eram redimensionados de acordo com o nível de *zoom*. Para actualizar o mapa é necessário repetir todo o procedimento descrito neste parágrafo. Esta biblioteca não possui elementos gráficos para disponibilizar o nome e a descrição dos POIs. Para contornar este problema utilizou-se uma *JLabel* com conteúdo dinâmico. Os dados da *JLabel* são actualizados quando a biblioteca detecta que o rato está sobre um POI, e torna-a visível. Assim que o rato sair do ícone de POI a *JLabel* fica invisível. Para visualizar as fotografias detecta-se quando o rato está sobre um ícone de um POI e as pré-visualizações das respectivas fotos são carregadas e visualizadas numa lista vertical. As imagens são carregadas através da classe *MyRender* que estende a classe *JButton*, deste modo cada imagem detecta os seus eventos. A classe *MyRender* lê a imagem para um *Image* através do método *ImageIO.read()* e redimensiona-as.

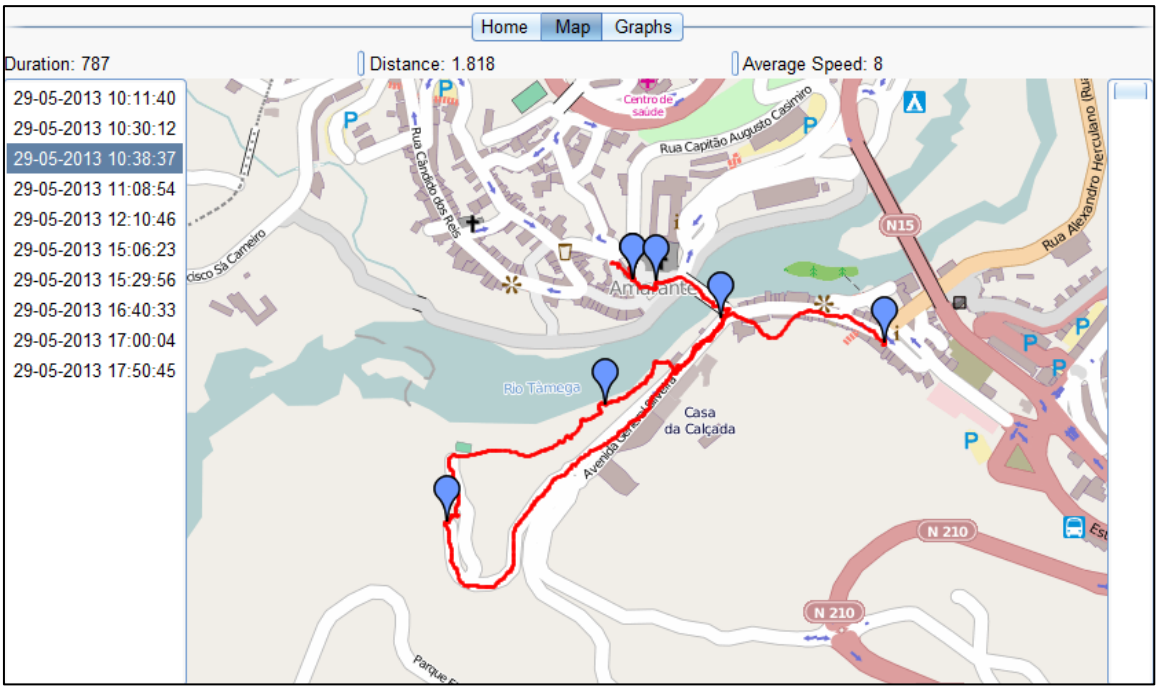

Figura 48 – Informação de uma rota

3.7.2. Persistência de dados

Os dados recebidos do servidor são armazenados no sistema SQLite. Este sistema de gestão de dados relacionais possui características ACID devido a utilizar transacções serializadas, atómicas, consistentes, isoladas e duráveis mesmo que a transacção seja interrompida por um *crash* do programa ou sistema operativo ou falha de energia.

Esta componente está implementada na classe *MySqliteConnector*. As *queries* são definidas através do objecto *Statement* que é inicializado com método *createStatement(),* do objecto *Connection*. Cada pesquisa é realizada através do método *executeQuery()* do objecto *Statement* e recebe por parâmetro a respectiva *querie* SQL. Este método devolve os resultados para um *ResultSet*. Este objecto é iterado e os dados de cada coluna são acedidos através do método *getString*() que recebe por parâmetro o nome da coluna que se pretende extrair.

3.7.3. Comunicação

A comunicação é realizada através de um cliente REST desenvolvido com a ferramenta Jersey. As mensagens são codificadas no formato JSON através da ferramenta Jackson. Esta biblioteca permite converter objecto java em JSON e vice-versa e os dados são enviados através de um pedido POST.

No início da execução da aplicação é apresentado um ecrã de *login* para o utilizador autenticar aplicação no sistema. Após autenticação a aplicação irá sincronizar com o cliente, aplicação pede ao servidor uma lista das rotas, aplicação verifica quais estão em disco e em seguida realiza um pedido para cada rota nova que deseja descarregar. Aplicação não sincronizará com servidor caso utilizador cancele o login ou insira dados errados. As rotas e as imagens são recebidas através do protocolo HTTP. O cliente pede a lista de rotas do utilizador contidas no servidor e depois compara com as que já tem armazenadas. De seguida o cliente efectua dois pedidos para cada nova rota, o primeiro para as fotos e o segundo para todos os dados da rota armazenados na base de dados do servidor. A rota é

armazenada no final destes dois pedidos. Com esta abordagem garante-se que tanto as fotos como os restantes dados das rotas são sempre armazenados mesmo que haja falhas na comunicação com o servidor durante os dois pedidos. As fotos de cada rota são recebidas num zip criado pelo o servidor, e descompactadas para a pasta "c:/tddesktop".

3.8. Instalação do sistema

Inicialmente acordou-se com a autarquia instalar o servidor numa máquina virtual através de acesso remoto, no entanto na fase final do projecto a autarquia não teve os recursos necessários para permitir a instalação do sistema. Para ultrapassar este contratempo acordou-se criar um manual de instalação (ver anexo 7.15) para as diferentes componentes que serão enviadas em suporte digital e também por correio electrónico. Os executáveis e códigos das aplicações também serão enviados por suporte digital e correio electrónico. O criador do sistema também dará apoio por correio electrónico ou poderá instalar o sistema mesmo depois da conclusão do estágio.

4. Testes

Os testes de *software* são uma componente importante do desenvolvimento pois permitem analisar o sistema de modo a detectar os seus limites. Também permitem detectar os defeitos, diminuindo a probabilidade de o utilizador os detectar. Através dos testes também se pode provar que o sistema é confiável e garantir que os requisitos foram atingidos. Os testes podem ser divididos em duas grandes categorias: funcionais e não funcionais.

4.1. Funcionais

Na fase inicial do desenvolvimento do projecto utilizou-se diversos testes unitários para testar funcionalidades das APIs e SDKs. Estes testes permitiram analisar a execução das funcionalidades singulares evitando que problemas nestas componentes se propagassem para funcionalidades mais complexas. Após o sucesso dos testes unitários realizou-se testes de integração. Nesta fase testou-se o funcionamento conjunto das diversas componentes unitárias que permitiram construir funcionalidades mais complexas. Posteriormente testou-se todo o sistema através da análise do funcionamento das diversas funcionalidades de cada aplicação e da interacção entre as diversas aplicações. Durante o desenvolvimento foi necessário alterar algumas componentes já implementadas e procedeu-se a testes de regressão que permitiram analisar o impacto das alterações nas funcionalidades dependentes das alteradas. Finalmente realizou-se testes de aceitação para verificar que os requisitos foram atingidos na óptica do criador do sistema. Esta análise denomina-se *white-box testing* e caracteriza-se por ser realizada por individuo que possui extenso conhecimento do código fonte. Deste modo é possível analisar o sistema em toda a sua extensão e optimizar o código com base nos resultados, no entanto esta abordagem consome bastante tempo e requer ferramentas externas. Estes testes foram realizados manualmente e utilizou-se a ferramenta Netbeans e Eclipse. De seguida ilustra-se o modelo utilizado para os testes *white-box* e os resultados. Nos anexos 9.12, **Erro! A origem da referência não foi encontrada.** e **Erro! A origem da referência não foi encontrada.** pode consultar em detalhe os parâmetros de cada teste de acordo com o modelo utilizado.

ID
Nome
Descrição
Pré-condições
Pós-condições
Dados necessários

Tabela 10 – Modelo dos casos de teste

4.1.1. Testes dos requisitos

Através das tabelas 11,12 e 13 averigua-se que cerca de 90% dos requisitos definidos na arquitectura foram implementados. O ponto 3 da tabela 11 foi implementado e testado em

Junho, no entanto em Agosto testou-se novamente e verificou-se que o Twitter bloqueou a API utilizada neste projecto. Não foi possível reimplementar esta funcionalidade para o Twitter com nova API devido à fase de implementação ter sido finalizada e estar a decorrer a fase dos diversos testes. O ponto 6 da tabela 13 não foi implementado. Esta funcionalidade não era fundamental para o funcionamento do sistema, não era um requisito definido pela autarquia, e devido a restrições temporais optou-se por focar na conclusão das funcionalidades fundamentais do sistema. O requisito 7 da tabela não foi implementado devido a problemas técnicos com API JXMapViewer, responsável pela visualização e interacção com os mapas. Esta API tem uma função que converte os pontos no ecrã em pontos geográficos do mapa mas após vários testes não foi possível utilizar esta função. Também seria necessário duplicar no servidor os dados dos pontos GPS de todas as rotas de todos os utilizadores.

ID	Descrição	Estado
1	Registo da localização e dados da rota	Implementado
2	Criação pontos de interesses com fotos e descrição	Implementado
3	Partilha pontos de interesse nas redes sociais	Implementado
4	Visualização de rotas no mapa sem ligação à rede	Implementado
5	Comunicação com servidor	Implementado
6	Galeria de fotos	Implementado
7	Consulta de rotas de outros utilizadores	Implementado

Tabela 11 – Resultados dos testes da aplicação Android

ID	Descrição	Estado
1	Estatísticas das freguesias	Implementado
2	Estatísticas dos IPs de acesso	Implementado
3	Inserção de POIs	Implementado
4	Inserção de eventos	Implementado
5	Validação e compressão de rotas dos utilizadores	Implementado

Tabela 12 – Resultados dos testes do servidor e aplicação web

ID	Descrição	Estado
1	Sincronização com servidor	Implementado
2	Gráficos das rotas	Implementado
3	Visualização de rotas no mapa	Implementado
4	Tempo e Quilómetros totais realizados	Implementado
6	Edição de descrição POIs em disco	Não implementado
7	Classificação de secções das rotas	Não implementado

Tabela 13 – Resultado dos testes da aplicação desktop

4.1.2. Validação da estatísticas das freguesias

Realizou-se uma visita Amarante, percorrendo a zona a pé e com aplicação Android activa. Registou-se vários locais e criou-se vários pontos de interesse durante cerca de duas horas. A visita foi realizada sob orientação de funcionário da secção de Turismo da autarquia que indicou os pontos de interesse mais relevantes e a uma rota utilizada pelos turistas. Após a conclusão da rota procedeu-se a uma rápida viagem para captar pontos de interesse fora da freguesia de Amarante. Esta rota foi posteriormente enviada para o servidor. Finalmente averiguou-se que as rotas estavam a ser detectadas nas freguesias corretas através da aplicação web.

4.1.3. Validação da detecção de POIs nas proximidades

A autarquia disponibilizou dados geográficos de pontos de interesse assim como seus nomes e descrições. Estes dados foram posteriormente utilizados para popular as tabelas utilizadas na detecção de POIs nas proximidades. Na visita a Amarante activou-se a funcionalidade *Near me* do *smartphone* em vários locais e verificou-se que alguns dos pontos de interesse eram visualizados nesta funcionalidade. Os dados visualizados variavam consoante a posição permitindo concluir que a detecção estava a trabalhar como esperado. No local de trabalho realizou-se a mesma experiencia e verificou-se que os locais não eram visualizados.

4.2. Não Funcionais

Durante o projecto analisou-se também as capacidades de diversas componentes e o impacto da sua utilização. Analisou-se os efeitos do GPS na bateria, a qualidade do sinal da rede móvel em espaços rurais, e o espaço ocupado por um vídeo com qualidade máxima. A qualidade do sinal em meios rurais revelou-se péssima com erros de 2 a 3 quilómetros em grande parte da rota e portanto esta funcionalidade foi descartada. A funcionalidade de criação de POIs com vídeo também foi descartada pois os testes do vídeo a máxima qualidade permitiram concluir que ocupam bastante espaço, cerca de 1 Mb por segundo. Caso se tivesse implementado a funcionalidade de vídeo então o utilizador iria demorar bastante tempo a sincronizar as rotas com o servidor e se estivesse a utilizar o tarifário do seu telemóvel poderia facilmente consumir grande parte do seu tráfego. Os vídeos também desgastam bastante a bateria e iria diminuir a longevidade da aplicação.

4.2.1. Qualidade do sinal em Amarante

Durante o desenvolvimento do projecto realizou-se uma viagem pedonal em Amarante, recolhendo vários dados GPS e fotografias para criação de rotas e POIs. As condições atmosféricas eram ligeiramente adversas, céu nublado com ligeiros chuviscos. De seguida apresenta-se os resultados obtidos duma amostra de 2560 pontos GPS recolhidos:

	Distância (m)
Média	4,01
Máximo	8,00
Mínimo	4,00
Desvio	1,17

Tabela 14 – Análise da qualidade do sinal em Amarante

O erro mínimo em dispositivos especializados na recepção de sinal GPS é de 3 metros [3], com base neste facto e com os dados da tabela anterior afirmar que os resultados são bastantes positivos e que a qualidade do sinal é bastante boa tendo em conta as condições atmosféricas.

4.2.2. Usabilidade da aplicação Android

Para avaliar a usabilidade da aplicação Android procedeu-se à distribuição da aplicação por um conjunto de utilizadores e um documento com uma breve explicação dos requisitos de *hardware*, objectivos e funcionamento da aplicação. Estes ficheiros foram distribuídos através de um directório Dropbox. Infelizmente não foi possível testar aplicação com número elevado de utilizares devido às funcionalidades necessitarem que o utilizador realize passeios ao ar livre durante alguns minutos. Foi também dado apoio presencial ou via telemóvel e web a todos os utilizadores. Cedeu-se temporariamente o telemóvel de testes a utilizadores que não possuíam *smartphones* Android, tendo sido acompanhados presencialmente para esclarecimento de dúvidas e verificação do correcto uso do equipamento. No final de cada teste realizou-se um inquérito para recolher a opinião relativamente à experiencia de utilizador. O questionário foi criado com base no modelo de (Lund, 2001), pode ser visualizado em http://tinyurl.com/qj36tn6. Com este modelo avalia-se a facilidade de utilização, facilidade de aprendizagem e a satisfação geral através de várias perguntas e cotação de 1 (mínimo) a 7 (máximo). Adicionou-se ao modelo base algumas perguntas sobre o perfil do utilizador, a versão Android dos seus *smartphones* e um campo reservado a críticas e sugestões.

Na tabela seguinte ilustra-se o perfil dos utilizadores:

Utilizador	Sexo	Idade	Habilitações Literárias	Actividade Laboral	Área em que trabalha/estuda
1	Masculino	30 - 33	Licenciatura pré-Bolonha	Trabalhador	Gestão e Contabilidade
2	Masculino	22 - 25	Mestrado	Estudante	Informática
3	Masculino	26 - 29	Mestrado	Estudante	Electrotécnica
4	Masculino	26 - 30	Mestrado	Estudante	Electrotécnica
5	Feminino	26 - 30	Licenciatura pós-Bolonha	Trabalhador	Assessoria
6	Masculino	26 - 29	Mestrado	Trabalhador	Informática
7	Masculino	22 - 25	Mestrado	Trabalhador	Informática
8	Masculino	26 - 29	Mestrado	Estudante	Informática
9	Masculino	30 - 33	Mestrado	Trabalhador	Informática
10	Masculino	26 - 29	Mestrado	Trabalhador	Informática
11	Masculino	26 - 29	Mestrado	Estudante	Multimédia

[3] Valor revelado pela FFA em http://www.gps.gov/systems/gps/performance/accuracy/

Tabela 15 – Perfil dos utilizadores

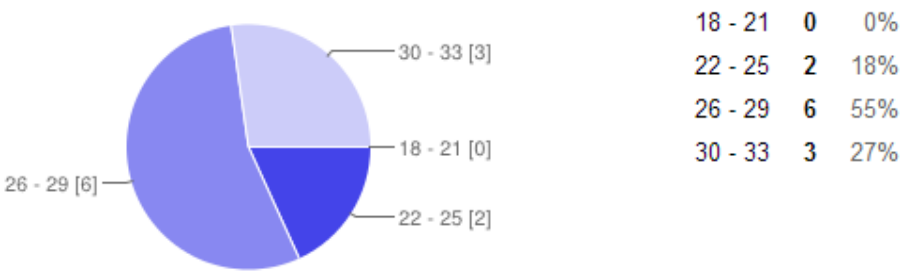

Figura 49 – Distribuição das idades

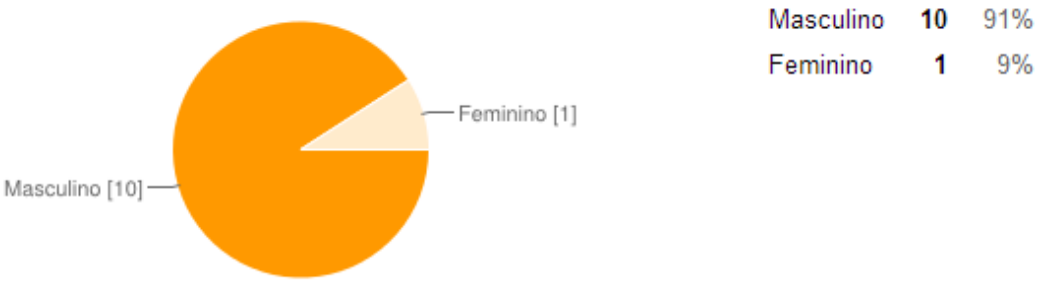

Figura 50 – Distribuição do género

Através da tabela 15 e figuras49 e 51 verifica-se que utilizadores são maioritariamente do sexo masculino, que todos utilizadores têm mais de 22 anos e formação de diversas áreas.

Para analisar os resultados do inquérito implementou-se o seguinte procedimento:

- Agrupou-se as perguntas pelas três categorias do inquérito (facilidade de utilização, facilidade de aprendizagem e a satisfação geral),
- Calculou-se a média e desvio padrão de cada categoria
- Comparou-se a média com os valores da cotação

Obtiveram-se os seguintes resultados:

	Facilidade de utilização	Facilidade de aprendizagem	Satisfação geral
Média	5,75	6,52	5,69
Desvio padrão	1,03	0,59	1,19

Tabela 16 – Resultados dos inquéritos

Através da tabela anterior verificamos que cada categoria obteve bons resultados pois as suas médias possuem valores positivos (maiores ou igual a 4)

Através dos inquéritos também se recolheu críticas e sugestões dos utilizadores, que são de seguida enumeradas e analisadas:

- "A utilização da aplicação com o telemóvel com o ecrã com brilho no máximo é proibitiva, sujeita a bateria a uma descarga rápida". Esta situação também se verificou nos testes realizados pelo autor do sistema e existe possibilidade de programaticamente reduzir o brilho do ecrã, no entanto a visibilidade das interfaces gráficas poderia ser afectada negativamente visto que não é possível analisar o ambiente em que o utilizador se encontra nem as suas capacidades visuais. Esta situação depende exclusivamente dos utilizadores que terão de reduzir o brilho do seu telemóvel caso queiram visualizar sempre aplicação.
- "Talvez alguns utilizadores menos experientes necessitem de balões de informação de primeira utilização". Após análise desta crítica (e sugestão) definiu-se que seria interessante para trabalho futuro criar um botão com ponto de interrogação na *action bar* em todos os ecrãs. Este o ser pressionado criaria uma nova componente na interface gráfica que explicasse as funcionalidades dos vários elementos da respectiva interface e os gestos disponíveis no respectivo ecrã.
- "Na vista de mapas não é possível ocultar o balão do pop-up com a descrição do Point of Interest, sendo necessário carregar em outro para o primeiro se ocultar". Este aspecto foi detectado durante implementação da aplicação, no entanto é uma limitação da biblioteca utilizada na visualização e interacção dos mapas. Como trabalho futuro terá que se verificar se as novas versões da biblioteca resolvem este problema ou então alterar a biblioteca.

De seguida enumera-se e analisa-se as sugestões dos utilizadores:

- "Exportar percursos directamente para o telemóvel". Esta opção está disponível em várias aplicações do mercado mas tal como foi descrito anteriormente neste documento, esta funcionalidade poderia retirar incentivo a sincronizar as rotas com os utilizadores.
- "Possibilitar *pinch to zoom*". Esta funcionalidade está disponível em ecrãs multitoque, para a realizar basta pressionar e arrastar dois dedos na mesma direcção para diminuir o *zoom* ou na direcção oposta para aumentar o *zoom*. Não está disponível na aplicação implementada visto que cria conflitos com o gesto *swipe* que pode ser utilizado na aplicação implementada. Esta explicação foi dada ao utilizador e o mesmo revelou que preferia realizar *pinch to zoom* em detrimento do gesto *swipe*. Neste projecto seguiu-se as recomendações da Google e portanto optou-se pelo gesto *swipe*, no entanto caso mais utilizadores tenham a mesma opinião então como trabalho futuro aplicação deveria ser alterada para reflectir essas opiniões relativamente a este aspecto

4.2.3. Compatibilidade da aplicação Android

Através dos inquéritos descritos no ponto 4.2.2 constatou-se que aplicação funciona correctamente nas seguintes versões Android: 2.3.7 (oficial), 2.3.8 (não oficial), 4.1 (oficial), 4.1 (oficial). Com base nestes resultados e nos dados oficiais da Google[4] pode-se afirmar que aplicação é compatível com 89,5% dos dispositivos Android.

[4] Distribuição de utilizadores por plataforma: http://developer.android.com/about/dashboards/index.html

4.2.4. Desempenho da bateria e sinal GPS

Durante o mês de Agosto realizou-se vários percursos pedestres em zonas rurais do concelho de Soure, percorrendo um total de 16.3Kms durante 4h31m49s (4.5 Km/h) com céu limpo. Não foi possível testar com outras condições atmosféricas visto que o tempo não se alterou nesta região durante o mês dos testes. A aplicação foi sempre iniciada com o telemóvel ligado à fonte de alimentação, sendo apenas removido após obtenção de sinal GPS inicial. Deste modo garante-se que o nível de bateria é igual no início dos testes. O sistema WiFi esteve sempre desligado e não existiam outras aplicações a correr em background. Os testes foram compostos em duas partes, na 1ª fase registou-se os percursos com o ecrã desligado e na 2ª fase registou-se os mesmos percursos com o ecrã sempre ligado. Armazenou-se na base de dados também o estado da bateria (percentagem) e o erro GPS para futura análise. Para analisar a bateria mediu-se os tempos que a bateria demorava a diminuir 1% em todos os trajectos. A última percentagem obtida em cada trajecto foi descartada visto que apenas havia registo de quando foi atingida

Ecrã	Desligado	Ligado
Média (s)	590	152
Desvio Padrão (s)	132	27

Tabela 17 – Média e desvio padrão dos testes à bateria

Os testes com ecrã sempre desligado permitiram verificar que aplicação consumia 1% da bateria em média 9m50s com desvio padrão de 2m12s. Através dos testes com ecrã sempre ligado verificou-se que a bateria diminuía 1% em média 2m32s com desvio padrão de 25s. Com estes resultados verifica-se que aplicação consome 1% de bateria entre 2m5s (152-27=125) e 12m02s (590+132=722). Caso aplicação seja iniciada com a bateria a 100% no telemóvel dos testes então poder-se-á registar posições entre 3h:28m20s a 20h3m20s.

Durante os percursos registaram-se 14845 pontos GPS durante 4h31m49s (16309s). Com estes dados verifica-se que o telemóvel demorou aproximadamente 1 segundo para receber uma posição GPS.

Com os valores do erro do sinal GPS verificou-se que a primeira posição tem um erro bastante elevado, desde os 48 a 128 metros, mas nas restantes posições o erro variava pouco, entre os 4 e 16 metros. Esta anomalia na primeira posição é derivada de o Android não ter uma posição recentemente armazenada para a comparar com actual. Para as restantes posições o erro médio do sinal foi de 7 metros com desvio padrão de 2 metros. A qualidade do sinal é bastante boa considerando que o erro mínimo nos equipamentos especializados de recepção de sinal GPS é de 3 metros[5].

4.2.5. Análise da memória da aplicação Android

Para analisar a memória utilizada pela aplicação, *heap,* conectou-se o *smartphone* de testes ao computador através do cabo USB. Deste modo foi possível analisar a memória da aplicação através da ferramenta DDMS do IDE Eclipse. Para detectar a tamanho máximo da *heap* utilizou-se as instruções da figura seguinte:

[5] Valor revelado pela FFA em http://www.gps.gov/systems/gps/performance/accuracy/

```
Runtime rt = Runtime.getRuntime();
long maxMemory = rt.maxMemory();
Log.e("onCreate", "maxMemory:" + Long.toString(maxMemory));

ActivityManager am = (ActivityManager) getSystemService(ACTIVITY_SERVICE);
int memoryClass = am.getMemoryClass();
Log.e("onCreate", "memoryClass:" + Integer.toString(memoryClass));
```

Figura 51 – Código para detectar heap máxima de um dispositivo Android

Através destas instruções verificou-se que o tamanho máximo era de 64 MBs, no entanto as aplicações Android normalmente não podem utilizar toda esta *heap*. O valor normal e máximo da *heap* é definido pelos fabricantes mas não se encontrou nenhuma fonte oficial que contenha estes dados. Nas versões Android 3.0 há possibilidade de permitir que uma aplicação utilize toda a *heap* através de uma configuração, no entanto não há garantia que o sistema operativo realmente forneça toda *heap* à aplicação. Devido a não haver esta garantia optou-se por não utilizar esta funcionalidade e garantir que aplicação funciona correctamente com a tamanho normal da *heap*. Durante o desenvolvimento da aplicação carregou-se diversas imagens simultaneamente para ocupar bastante memória e detectar o tamanho normal da *heap* do telemóvel de testes. Verificou-se que este tamanho no *smartphone* Samsung Ace 2 está definido para os 24MBs, após este limite a execução era cancelada resultando num *crash* de memória. Durante a validação das funcionalidades averiguou-se o estado do *heap* e verificou-se que raramente excedia os 10 MBs, no entanto quando se registava uma fotografia aumentava significativamente. Este aspecto também se verificou num outro telemóvel com câmara fotográfica de maior resolução e bastava apenas uma fotografia para aplicação exceder o limite máximo da *heap*. Face a este problema optou-se por captar as fotografias com a resolução mínima, reduzindo assim o seu espaço em disco e memória, evitando os *crashes*. Também permitiu sincronizar as rotas mais rapidamente com o servidor. Testou-se novamente nos dois telemóveis e não se verificou *crashes* devido a problemas de memória.

4.2.6. Testes de stress da aplicação Android

Utilizou-se a ferramenta Monkey do SDK Android para realizar testes automatizados. Esta ferramenta permite definir o número de testes a realizar e de seguida cria eventos aleatórios na GUI, activando assim as diferentes funcionalidades da aplicação. Com esta ferramenta pode-se detectar *bugs*, e averiguar se a rápida e constante utilização de funcionalidades e mudanças de ecrã causam problemas de processamento, memória e na base de dados.

Utilizou-se o comando "adb shell monkey -p tdclient.core -v 10000" para efectuar estes testes, definindo assim 10000 eventos aleatórios na aplicação. Testou-se todas as funcionalidades conjuntas e as funcionalidades *tracking*, *my tracks*, *gallery*, *near me* isoladamente. No teste de todas funcionalidades conjuntas não se verificou problemas. Nos testes das funcionalidades isoladas *tracking*, *near me* e *gallery* também não se verificou problemas. No teste isolado à funcionalidade *my tracks* verificou-se que a ligação à base de dados encerrava após alguns minutos intensivos de sua utilização mas será difícil esta situação acontecer com um utilizador real visto que dificilmente conseguirá efectuar tantos eventos constantes e rápidos quanto esta ferramenta. Os testes dos utilizadores também não revelaram este problema.

4.2.7. Compatibilidade de aplicação web em diferentes navegadores

As tecnologias utilizadas na implementação da aplicação web do servidor permitem que seja executada nos três navegadores mais utilizados: Chrome, Firefox e Internet Explorer[6]. No entanto estas tecnologias e os navegadores podem conter *bugs* e portanto testou-se aplicação web nesses três navegadores para confirmar o seu correcto funcionamento. Através destes testes verificou-se que aplicação corre sem problemas nas versões "Chrome 29.0.1547.57 m", "Firefox 23.0.1", "Internet Explorer 10.0.9200.16635".

[6] Para mais informações consulte: http://www.w3schools.com/browsers/browsers_stats.asp

5. Conclusões e trabalho futuro

Com a utilização da aplicação Android para recolher a localização e fotografias de locais e a respectiva partilha nas redes sociais divulga-se rapidamente um local numa página web com milhares de potenciais interessados. O envio destes dados para o servidor de Amarante permite à autarquia verificar qual freguesia mais visitada pelos utilizadores e visualizar a origem dos seus turistas que utilizem aplicação. O envio de dados para o servidor e aplicação desktop permitem ao utilizador criar uma cópia de todos os seus dados sem necessitar de configurações, bastando apenas ter ligação WiFi no Android e enviar para o servidor, posteriormente a aplicação desktop sincroniza automaticamente com servidor. Esta cópia permite ao utilizador poupar espaço no seu telemóvel visto que é bastante reduzido comparado com um computador e permite visualizar as rotas em ecrãs maiores, com mais detalhes e em máquinas com maior processamento. Com as rotas enviadas pelos utilizadores e validadas pela autarquia é possível aos utilizadores obterem informação fiável sobre os locais que os rodeiam.

Os inquéritos de usabilidade realizados permitiram averiguar que os utilizadores ficaram satisfeitos com a usabilidade da aplicação mas que algumas interacções com a interface gráfica eram difíceis de detectar sem descrição prévia da aplicação. Os utilizadores também afirmaram que recomendariam aplicação aos seus amigos e conhecidos. A aplicação é estável visto que seus os testes de performance não revelaram problemas de memória ou processamento, e funciona correctamente nas versões 2.3.7 (oficial), 2.3.8 (não oficial), 4.0 (oficial) e 4.1 (oficial). Com base nestes resultados e nos dados oficiais da Google pode-se afirmar que aplicação é compatível com 89,5% dos dispositivos Android. A aplicação web foi testada em diversos *browsers* e verificou-se que todas as funcionalidades e elementos gráficos funcionaram correctamente e portanto poderá ser utilizada em qualquer desses três *browsers*.

Durante o projecto encontrou-se vários desafios, principalmente na implementação de mapas em java e Android. Existem bastantes ferramentas para utilização de mapas em páginas web mas para aplicações desktop existem poucas soluções grátis e viáveis. Nesta componente não foi possível utilizar Google Maps apesar dos mapas serem *online* pois a licença grátis não permite distribuição de aplicações que utilizem a sua API. Estes contratempos implicaram maior consumo de recursos para implementação da aplicação Android. Durante o projecto descobriu-se que apesar da famosa API Google Maps estar bastante mais evoluída que as restantes, a sua utilização é bastante restritiva caso não se queira obter uma licença paga. Destas restrições destaca-se a de não ser permitido utilizar mapas *offline* com licença grátis, o que levou a procurar soluções alternativas, optando-se pela API OSMdroid que utiliza mapas OpenStreetMaps, permitindo utilização de mapas *offline* com licença grátis na aplicação Android. Esta API contém também alguns problemas, a documentação oficial é bastante pobre e grande parte resume-se ao código aberto das classes. Felizmente há bastantes comunidades que disponibilizam exemplos e com boas explicações. O OpenStreetMaps contém uma API muito bem documentada, cheia de exemplos funcionais que permitiram um rápido desenvolvimento da componente de mapas em páginas HTML. A documentação das APIs das redes sociais também dificultou bastante a sua integração na aplicação Android. As APIs do Facebook e Twitter possuem documentação extensa e com vários exemplos. Existem também várias comunidades que disponibilizam diversos exemplos e bibliotecas de suporte não oficiais. As restantes redes sociais analisadas neste trabalho possuem uma documentação fraca que se resume à indicação da tecnologia de comunicação, e uma breve descrição dos métodos suportados. Recentemente o Twitter alterou a sua API e desactivou a versão anterior que tinha sido

utilizada neste projecto, resultando na desactivação da partilha de POIs no Twitter através da aplicação Android. A captura de fotografias também foi desafiante, alguns modelos de telemóvel captam uma fotografia vertical mas armazenam-na rodada 90º. Para evitar este problema rodou-se a imagem 90º para o lado oposto antes de a gravar, assim a imagem ficará correcta quando o Android a rodar. Esta solução no entanto implica uma maior utilização de memória pois é necessário conter duas imagens em memória. Os testes de utilizadores anónimos detectou-se problemas de memória ao captar fotografias em dispositivos com câmaras de grandes resoluções. Para evitar que aplicação final tivesse este problema optou-se por captar as fotografias com resolução mínima visto que apresentam boa qualidade na sua visualização em *smartphones*.

Todas as componentes do sistema implementado podem ser melhoradas. As bibliotecas OSMdroid estão em constante actualização, é por isso recomendável que aplicação seja actualizada para utilizar sempre a sua última versão, caso sejam corrigidos os problemas ocasionais de imagens não carregadas ou a sua performance seja melhorada. Os dados dos mapas OpenStreetMaps são públicos mas o seu acesso directo aos servidores oficiais do OpenStreetMaps poderá sofrer restrições caso existam sobrecargas. Seria bastante interessante criar um servidor com os dados OpenStreetMaps e configurar aplicação desktop e página da autarquia para utilizar esse servidor de modo a evitar sobrecarga dos oficiais. Deste modo também se poderia actualizar os mapas Android mais rapidamente e com mais níveis de *zoom* através da ferramenta Mobile Atlas Creator. Com base nas datas das rotas pode-se implementar um módulo que permita averiguar as estações do ano em que mais turistas se deslocam Amarante. Com servidores mais poderosos em processamento pode-se adaptar o sistema para realizar estatísticas de freguesias para todo o país.

6. Bibliografia

Abreu, P. H., Vinhas, V., & Mendes, P. (2008). Business intelligence through real-time tracking using a location system towards behaviour pattern extraction. *Proceedings of the international conference on e-business*, (pp. 51-57).

Abreu, P. H., Vinhas, V., & Mendes, P. (2009). Wi-Fi Real-Time Location BI applied to a Retail Company. *International Conference on Enterprise Information Systems - Proceedings of Databases and Information Systems Integration*, (pp. 11-16).

Abreu, P. H., Vinhas, V., Mendes, P., Luís, R. P., & Garganta, J. (2010). Real-Time Wireless Location and Tracking System with Motion Pattern Detection, Robot Localization and Map Building. *In-Tech*, 467-492.

Abreu, P. H., Xavier, J., Reis, L. P., & Petry, M. (2011). Location and automatic trajectory calculation of mobile objects using radio frequency identification. *6th Iberian Conference on Information Systems and Technologies (CISTI)*, (pp. 1-3).

Amarante, C. M. (s.d.). *Amarante*. Obtido em 3 de 12 de 2012, de Município de Amarante: http://www.cm-amarante.pt/index.php?info=YTo0OntzOjQ6Im1lbnUiO3M6MzoiY2FtIjtzOjM6Im1pZCI7YToyOntpOjA7czoyOiIxNSI7aToxO3M6MjoiMjkiO31zOjM6ImNpZCI7czoyOiIyOSI7czoxMzoidmVyc2FvX3RhYmVsYSI7czo2OiJvbmxpbmUiO30=

Amarante, C. (s.d.). *Rede Wireless*. Obtido em 3 de 12 de 2012, de Município de Amarante: http://www.cm-amarante.pt/index.php?info=YTo0OntzOjQ6Im1lbnUiO3M6MzoiY2FtIjtzOjM6Im1pZCI7YToyOntpOjA7czoyOiIxNiI7aToxO3M6MzoiMTI4Ijt9czozOiJjaWQiO3M6MzoiMTI4IjtzOjEzOiJ2ZXJzYW9fdGFiZWxhIjtzOjY6Im9ubGluZSI7fQ==

Ayob, N. Z., Hussin, A. R., & Dahlan, H. M. (2009). Three Layers Design Guideline for Mobile Application. *Information Management and Engineering, 2009. ICIME '09. International Conference*, (pp. 427-430).

Bartholomew, D. (2010, Julho). SQL vs. NoSQL. *Linux Journal #195*.

Bonnet, L., Laurent, A., Sala, M., Laurent, B., & Sicard, N. (2011). REDUCE, YOU SAY: What NoSQL can do for. *22nd International Workshop on Database and Expert Systems Applications*, (pp. 483-485).

Google. (2013). *Design*. Obtido de Android Developers: http://developer.android.com/design/index.html

Graf, T. (2011/2012). Power-efficient Positioning Technologies for Mobile. *SNET2 Seminar 2011/12*, (pp. 2-4). Berlin University of Technology.

Hoff, T. (6 de 12 de 2010). *What The Heck Are You Actually Using NoSQL For?* Obtido em 30 de 11 de 2012, de Highscalability: http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html

Leavitt, N. (Febreiro de 2010). *Will NoSQL Databases Live Up to Their Promise?* Obtido em 30 de 11 de 2012, de Ieeexplore: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5410700

Lin, K., Kansal, A., Lymberopoulos, D., & Zhao, F. (s.d.). *Energy-Accuracy Aware Localization for Mobile Devices*. Obtido em 2012 de 12 de 11, de Microsoft Research: http://research.microsoft.com/pubs/120831/aloc_kansal.pdf

Lund, A. M. (10 de 2001). *Measuring Usability with the USE Questionnaire*. Obtido em 17 de 5 de 2013, de Stcsig: http://www.stcsig.org/usability/newsletter/0110_measuring_with_use.html

Nokia maps for mobile web. (s.d.). Obtido em 25 de 8 de 2012, de Betalabs nokia: http://betalabs.nokia.com/trials/nokia-maps-for-mobile-web

Perez, S. (2 de 11 de 2012). *IDC: Android Market Share Reached 75% Worldwide In Q3 2012*. Obtido em 15 de 11 de 2012, de IDC: Android Market Share Reached 75% Worldwide In Q3 2012: http://techcrunch.com/2012/11/02/idc-android-market-share-reached-75-worldwide-in-q3-2012/

Shneiderman. (s.d.). *Schneiderman Golden Rules*. Obtido em 10 de 10 de 2012, de UW Faculty Web Server: http://faculty.washington.edu/jtenenbg/courses/360/f04/sessions/schneidermanGoldenRules.html

Smith, C. (7 de 10 de 2011). *NoSQL or SQL? Do you have to choose?* Obtido em 30 de 11 de 2012, de RackSpace: http://www.rackspace.com/blog/nosql-or-sql-do-you-have-to-choose/

Song, M., Song, H., & Fu, X. (2011). Methodology of User Interfaces Design Based On Android. *ICMT - International Conference of Multimedia Technology 2011*, (pp. 408-411).

Twitter. (s.d.). *Twitter Libraries*. Obtido em 25 de 8 de 2012, de dev.twitter.com: https://dev.twitter.com/docs/twitter-libraries

W3C. (2008). *Mobile Web Best Practices 1.0*. Obtido em 10 de 10 de 2012, de W3: http://www.w3.org/TR/mobile-bp/

Wang, C., Duan, W., Ma , J., & Wang, C. (2011). The research of Android System Architecture and application programming. *International Conference on Computer Science and Network Technology*, (pp. 786-788).