Instituto Superior de Engenharia do Porto

Departamento de Engenharia Eletrotécnica
Rua Dr. António Bernardino de Almeida, 431, P-4200-072 Porto

Partilha de Pontos de Interesse, Vídeos e Experiências Sensoriais Baseada no Contexto do Utilizador

Dissertação submetida para a satisfação parcial dos requisitos do grau de
Mestre em Engenharia Eletrotécnica e Computadores
Área de Especialização em Telecomunicações

Nuno Filipe Magalhães Areias

Orientação: Professora Doutora Maria Benedita Campos Neves Malheiro

Junho de 2015

Resumo

A evolução tecnológica das últimas décadas na área das Tecnologias da Informação e Comunicação (TIC) contribuiu para a proliferação de fontes de informação e de sistemas de partilha de recursos. As diversas redes sociais são um exemplo paradigmático de sistemas de partilha tanto de informação como de recursos (*e.g.* audiovisuais). Essa abundância crescente de recursos e fontes aumenta a importância de sistemas capazes de recomendar em tempo útil recursos personalizados, tendo por base o perfil e o contexto do utilizador.

O objetivo deste projeto é partilhar e recomendar locais, artigos e vídeos em função do contexto do utilizador assim como proporcionar uma experiência mais rica de reprodução dos vídeos partilhados, simulando as condições de gravação dos vídeos. Este sistema teve como inspiração dois projetos anteriormente desenvolvidos de partilha e recomendação de locais, artigos e vídeos turísticos em função da localização do utilizador.

O sistema desenvolvido consiste numa aplicação distribuída composta por um módulo cliente Android, que inclui a interface com o utilizador e o consumo direto de serviços externos de suporte, e um módulo servidor que controla o acesso à base de dados central e inclui o serviço de recomendação baseado no contexto do utilizador. A comunicação entre os módulos cliente e servidor utiliza um protocolo do nível de aplicação dedicado. As recomendações geradas pelo sistema têm por base o perfil de utilizador, informação contextual (posição do utilizador, data e hora atual e velocidade atual do utilizador) e podem ser geradas a pedido do utilizador ou automaticamente, caso sejam encontrados pontos de interesse de grande relevância para o utilizador. Os pontos de interesse recomendados são apresentados com recurso ao Google Maps, incluindo o período de funcionamento, artigos complementares e a reprodução imersiva dos vídeos relacionados. Essa imersão tem em consideração as condições meteorológicas, temporais e espaciais aquando da gravação do vídeo.

Abstract

The development in recent decades in Information and Communication Technologies (ICT) has greatly contributed to the proliferation of data sources and resource sharing systems. The various social networks are a prime example of such sharing schemes. This abundance of information increases the importance of filtering tools such as recommendation systems which are able to make timely customised suggestions based on the user profile and current context.

The main objective of this project is to share and recommend places, articles and videos based on the user's context and to provide a richer playback experience of shared videos, simulating the recording conditions. This system was inspired by two previous projects developed for sharing and recommendation of places, articles and videos depending on user location.

The developed system consists of a distributed application comprising an Android client module, which includes user interface and the direct consumption of external support services and a server module that controls access to the central database and includes the recommendation service based on user context. Communication between the client and server modules is based on a dedicated application level protocol. The recommendations generated by the system are based on the user profile, contextual information (user position, date and current time and current speed of the user) and can be generated by user request or automatically, if new points of interest of great relevance to the user are found. The recommended points of interest are represented using Google Maps, including the period of operation, additional articles and immersive play of related videos. The immersion takes into account the weather conditions, time and space of when the video was recorded.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Equações

Lista de Excertos de Código

Glossário

Abreviatura	Descrição	Página
LOD	*Linked Open Data*	2
TIC	Tecnologias da Informação e Comunicação	5
RS	*Recommender Systems*	5
TEL	*Technology Enhanced Learning*	6
CARS	*Context-Aware Recommender System*	6
GPS	*Global Positioning System*	7
API	*Application Programming Interface*	10
CF	*Collaborative Filtering*	12
CBF	*Content-based Filtering*	12
HF	*Hybrid Filtering*	15
WSS	Windy Sight Surfers	16
POI	*Point of Interest*	17
ESN	*Erasmus Student Network*	17
LBSNS	*Location Based Social Networking Service*	22
AOSP	Android Open Source Project	27
OHA	Open Handset Alliance	27
MMS	*Multimedia Messaging Service*	29
UI	*User Interface*	29
HTML	*HyperText Markup Language*	29
VoIP	*Voice over IP*	29
HTTP	*HyperText Transfer Protocol*	29
VPN	*Virtual Private Network*	29
SMS	*Short Message Service*	29
SDK	*Software Development Kit*	31
IDE	*Integrated Development Environment*	31
XML	*eXtensible Markup Language*	37
JSON	*JavaScript Object Notation*	37
JAR	*Java ARchive*	37
CVS	*Concurrent Version System*	42
ADT	*Android Development Tools*	41
ADB	*Android Debug Bridge*	44

Abreviatura	Descrição	Página
AVDM	*Android Virtual Device Manager*	44
SGBD	Sistema de Gestão de Base de Dados	39
SQL	*Structured Query Language*	39
GPL	*General Public Licence*	40
FIFO	*First In, First Out*	73
RAM	*Random-Access Memory*	87
CPU	*Central Processing Unit*	87

Agradecimentos

A realização deste trabalho só foi possível com a contribuição e orientação da Professora Doutora Benedita Malheiro e com todo o apoio inestimável da família e amigos. A eles dedico esta dissertação.

Capítulo 1

Introdução

Este capítulo introdutório contextualiza o projeto e descreve o problema a resolver, a motivação da sua escolha e os objetivos a atingir. Finalmente, apresenta-se o plano de trabalho e a estrutura desta dissertação.

1.1 Problema

Este projeto surge na âmbito de uma linha de investigação e desenvolvimento dedicada à personalização de conteúdos. Em particular, esta dissertação descreve a realização de uma aplicação distribuída de recomendação de locais, vídeos e artigos complementares personalizados em função do contexto do utilizador. A aplicação permite, por um lado, definir e partilhar locais, realizar e partilhar vídeos georreferenciados e, por outro lado, explorar locais através da recomendação de vídeos e sugestão de artigos relacionados da Wikipédia em função do contexto do utilizador.

Este projeto reutiliza os resultados de duas aplicações móveis baseadas na localização do utilizador previamente desenvolvidas: a aplicação ErasmusApp destinada a estudantes Erasmus, que permite criar, partilhar e receber recomendações de locais da cidade do Porto [1], e a aplicação ErasmusVideoShare de realização, partilha e sugestão de vídeos georreferenciados, que inclui a recolha de informação sensorial complementar e a recomendação de artigos relacionados da Wikipédia [2].

Tendo por inspiração estes dois projetos, o objetivo desta tese foi conceber e realizar uma aplicação turística de recomendação de locais, vídeos e artigos complementares personalizados em função do contexto do utilizador e não exclusivamente da sua localização. A recomendação deve levar em conta o contexto alargado do utilizador, podendo ser disponibilizada a pedido ou automaticamente

em função da deslocação do utilizador. Adicionalmente, os vídeos deverão ser apresentados de forma a promover uma sensação de imersão, recriando, tanto quanto possível, as condições físicas aquando da sua gravação.

1.2 Motivação

A escolha desta temática no âmbito da unidade curricular de Tese/Dissertação deveu-se à oportunidade de desenvolver novas competências no domínio do desenvolvimento de aplicações distribuídas e de aplicações móveis. A oportunidade de trabalhar com uma tecnologia nova (Android) e o desafio da vídeo-imersão foram outros fatores decisivos.

1.3 Objetivos

O objetivo do projeto é conceber e realizar uma plataforma turística de partilha de locais, vídeos e artigos simples, robusta e intuitiva.

O utilizador, em termos de entradas, deverá conseguir: (*i*) criar, gerir e partilhar locais pessoais, especificando o período de funcionamento (semana/fim de semana); (*ii*) avaliar e reportar incorreções dos locais partilhados aos publicadores em questão, possibilitando a respetiva correção; e (*iii*) criar e partilhar vídeos georreferenciados dos locais que explora.

O utilizador, em termos de resultados, deverá conseguir: (*i*) reviver parcialmente a experiência da gravação durante a reprodução de um vídeo com base nos dados recolhidos durante da gravação; (*ii*) explorar os artigos Wikipédia disponibilizados sobre os locais situados na sua vizinhança; e (*iii*) solicitar e/ou obter automaticamente recomendações em função do seu perfil e contexto (posição, data/hora e nível de atividade do utilizador) relativas a locais, vídeos e artigos.

1.4 Casos de Uso e Testes Funcionais

O sistema deverá oferecer ao utilizador um conjunto de serviços contextualizados no espaço e tempo de partilha de locais, vídeos com experiências sensoriais e artigos. Em particular, os testes funcionais deverão demonstrar:

- A criação, avaliação, partilha e recomendação personalizada de locais em função do contexto espacial e temporal do utilizador;
- A recomendação de artigos provenientes da *Linked Open Data* (LOD);
- A realização, carregamento, partilha, recomendação personalizada e visualização de vídeos em função do contexto espacial e temporal do utilizador;

- O enriquecimento dos vídeos suportado pela informação sensorial e meteorológica recolhida durante a gravação.

1.5 Plano de Trabalho

O plano de trabalho está representado na Figura 1.1. O desenvolvimento e depuração do protótipo foi mais demorado que o previsto; as restantes tarefas foram realizadas no tempo previsto.

	Task	Start	End	Dur	%
	Execução do projeto	14/2/14	30/9/14	163	
1	Levantamento do estado da arte e familiarização com tecnologias envolvidas	14/2/14	23/3/14	26	
2	Escrita do capítulo "Estado da Arte"	24/3/14	31/3/14	6	
3	Especificação das características do protótipo e das tecnologias e ferramentas a utilizar	1/4/14	30/4/14	22	
4	Escrita dos capítulos correspondentes	1/5/14	14/5/14	10	
5	Desenvolvimento, teste e depuração do protótipo	15/5/14	31/8/14	77	
6	Redação dos capítulos referentes ao desenvolvimento da aplicação	1/9/14	30/9/14	22	

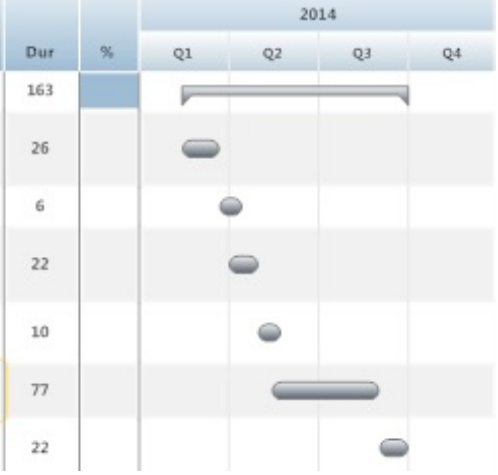

Figura 1.1: Plano de trabalho

1.6 Estrutura da Dissertação

Este capítulo – Capítulo 1 – introduz o problema, a motivação, os objetivos, os casos de uso, testes funcionais e o plano de trabalho. O Capítulo 2 apresenta o estado da arte no domínio da recomendação baseada no contexto. O Capítulo 3 contém o conjunto das tecnologias, soluções e ferramentas utilizadas na elaboração do projeto. O Capítulo 4 descreve o sistema desenvolvido, detalhando a arquitetura, módulos constituintes e funcionalidades oferecidas. O Capítulo 5 detalha o conjunto de testes funcionais realizados e os resultados obtidos. O Capítulo 6 reporta as conclusões do projeto e as sugestões de trabalho futuro. Em anexo coloca-se uma lista de hiperligações para os vídeos demonstrativos das funcionalidades da aplicação, o registo do lado do servidor e o código fonte do sistema.

Capítulo 2

Recomendação baseada no Contexto

Este capítulo apresenta o estado da arte no domínio dos sistemas de recomendação sensíveis ao contexto do utilizador. Define-se o contexto do utilizador e os diversos níveis de consciência de contexto, detalha-se a arquitetura, o processamento da informação e de geração de recomendações típica dos sistemas de recomendação sensíveis ao contexto, referindo-se os algoritmos de recomendação geralmente utilizados, e introduz-se o conceito de vídeo-imersão. Por último, são apresentados os projetos congéneres mais representativos encontrados na literatura do domínio.

2.1 Introdução Histórica

A tomada de decisão baseada em informação incompleta ou com falta de experiência é um problema frequente. Nestas circunstâncias é frequente recorrer-se à recomendação, *e.g.*, a especialistas no domínio que, mediante o nosso perfil e necessidades, sugerem serviços, produtos ou atividades de potencial interesse [3].

O desenvolvimento das Tecnologias da Informação e Comunicação (TIC) conduziu ao aumento exponencial do número de utilizadores assim como de serviços e produtos disponíveis. Esta nova realidade, que ultrapassa a escala humana, requer a adoção de sistemas capazes de processar elevadas quantidades de informação e de providenciar em tempo útil recomendações personalizadas. Os provedores de serviços e produtos passaram a recolher e manter o historial das interações de cada utilizador (perfil) para passarem a efetuar sugestões personalisadas. Assim, surgiram os sistemas de recomendação (*Recommender Systems*)

automática muito comuns em diversas áreas, *e.g.*, *e-commerce*, *e-health* e *Technology Enhanced Learning* (TEL) [4, 5].

Os sistemas de recomendação alteraram a forma como os utilizadores encontram, compram ou escolhem itens (produtos, informação, filmes, outras pessoas, *etc.*). Estes sistemas identificam e utilizam padrões comportamentais para selecionar, de entre um número elevado de itens disponíveis no sistema, aqueles que mais se adequam ao perfil de cada utilizador. Inicialmente, estes sistemas representavam dois tipos de entidades – utilizadores e itens – e recomendavam a cada utilizador o conjunto de itens mais compatível com o seu perfil estático. Mais recentemente, adotou-se a construção dinâmica do perfil dos utilizadores, permitindo levar em conta o contexto corrente e gerar recomendações mais precisas [6].

2.2 Contexto do Utilizador

O contexto engloba toda a informação que caracterize uma entidade e que seja relevante para a interação entre um utilizador e um sistema/aplicação. Assim, um sistema de recomendação consciente de contexto – *Context-Aware Recommender System* (CARS) – é um sistema que consegue utilizar informação contextual no seu processo de recomendação. Apesar da relevância da informação contextual ser sempre dependente da aplicação, há algumas componentes contextuais que são frequentemente utilizadas, tais como [7]:

- Localização;
- Tempo;
- Condições físicas (luz, som, humidade e temperatura);
- Condições de ligação (rede);
- Estado de espírito do utilizador;
- Tipo de dispositivo terminal utilizado.

Por exemplo, num sistema de recomendação de filmes, a localização do utilizador permite recomendar filmes que se encontrem em exibição em cinemas geograficamente próximos do utilizador, a informação temporal permite recomendar filmes que se encontrem atualmente em exibição. As condições da ligação pode ser usada num sistema de recomendação de vídeos (*e.g.* YouTube), adequando a qualidade do vídeo exibido ao débito da ligação. O tipo de dispositivo terminal utilizado pode ser importante para a recomendação de diferentes tipos de conteúdo, *e.g.*, em formato de vídeo/áudio num *smartphone* onde o ecrã apresenta

geralmente dimensões menores e em formato texto num *tablet* ou *laptop*. Finalmente, o estado de espírito do utilizador pode ser relevante para a recomendação de músicas/vídeos/filmes.

2.3 Classificação do Contexto

Existem duas formas de categorizar informação contextual: categorização operacional ou categorização conceptual. Enquanto a categorização operacional se baseia na forma como a informação é obtida, modelada e tratada, um esquema de categorização conceptual foca-se no significado e relações conceptuais entre componentes do contexto [8].

Abowd *et al.* [9] propõem um esquema de classificação operacional que divide a informação contextual em contexto primário e contexto secundário. Informação contextual primária corresponde à informação retirada diretamente de um sensor sem qualquer processamento ou tratamento, enquanto que informação contextual secundária é toda a informação que pode ser obtida através do processamento ou tratamento da informação contextual primária. De uma forma simples podemos considerar o número do BI de um cidadão como sendo informação primária e toda a informação passível de se obter através do número de BI (contacto, morada, data de nascimento, estado civil, *etc.*) como informação contextual secundária. Outro exemplo de informação primária e de informação secundária são os valores disponibilizados por um sensor *Global Positioning System* (GPS) e a distância relativa entre o sensor GPS e um ponto georreferenciado (*e.g.* um café/restaurante/loja). A Figura 2.1 resume as principais diferenças entre contexto primário e secundário, apresentando também alguns exemplos.

Apesar da categorização anterior ser uma das mais utilizadas, outros autores apresentam esquemas diferentes. Schilit *et al.* [11] organizaram a informação contextual em três categorias:

- *Where* – Informação contextual relacionada com a localização;
- *Who* – Informação contextual relativa às pessoas que se encontram em redor do utilizador;
- *What* – Informação contextual relativa aos recursos disponíveis na área onde o utilizador se encontra.

Um terceiro esquema de categorização, proposto por Henricksen [12], propõe a divisão da informação contextual em quatro categorias baseadas fonte da informação:

- *Sensed* – Informação contextual obtida diretamente de sensores;

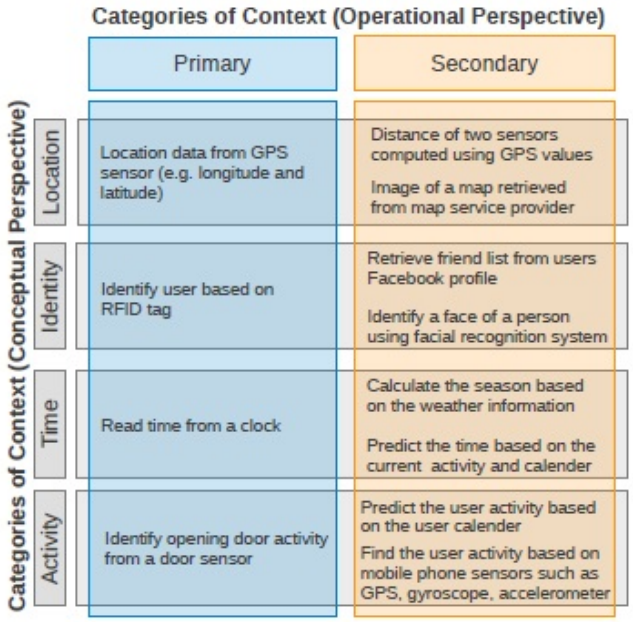

Figura 2.1: Categorias de contexto[10]

- *Static* – Informação contextual estática, ou seja que não é passível de se alterar com o tempo;
- *Profiled* – Informação contextual que se altera ao longo de tempo, mas com uma frequência baixa de alteração (*e.g.* mensalmente);
- *Derived* – Informação que é obtida através do processamento de informação contextual do tipo *sensed*.

2.4 Níveis de Consciência de Contexto

Podem-se identificar três níveis distintos de consciência de contexto baseadas no tipo de interação que a aplicação mantém com o utilizador [10]:

- Consciência passiva: O sistema monitoriza constantemente o ambiente mas precisa de confirmação do utilizador para realizar qualquer ação, *e.g.* o utilizador entra num supermercado e é-lhe apresentado uma lista de produtos que lhe possam interessar;
- Consciência ativa: O sistema monitoriza constantemente o ambiente e age de forma autónoma, não precisando de qualquer confirmação do seu utilizador, *e.g.* o sistema de deteção de fumo quando identifica fumo envia um alerta à corporação de bombeiros mais próxima;

- Consciência personalizável: O sistema permite que o utilizador defina as suas preferências, gostos e expectativas manualmente, *e.g.* a temperatura pretendida, e atua de forma a alcançar esses objetivos.

2.5 Sistemas de Recomendação Sensíveis ao Contexto

Segundo [13], um sistema de recomendação sensível ao contexto (CARS) pode obter informação contextual de três formas:

- Explicitamente: Requerendo ao utilizador ou a outras fontes a informação pretendida, *e.g.*, através do preenchimento de formulários onde o utilizador introduz os seus gostos e preferências;
- Implicitamente: Obtendo informação contextual diretamente do ambiente, sem intervenção do utilizador, *e.g.*, a localização do utilizador pode ser obtida através do sensor GPS do *smartphone*;
- Dedutivamente: Analisando a informação contextual, determinando a atividade do utilizador através da sua interação com ferramentas e recursos, e utilizando métodos estatísticos ou de *data mining*. Esta abordagem permite identificar, *e.g.*, a identidade de quem está a ver televisão numa família (pai, mãe, filho, filha) analisando quais os canais ou programas de televisão vistos.

Um sistema de recomendação geralmente utiliza mais do que um dos métodos referidos para obter a informação relevante sem necessitar de interação constante com o utilizador. No que diz respeito às abordagens arquitetónicas para aquisição de informação contextual, [14] define três abordagens:

- Acesso direto a sensores: Abordagem utilizada em dispositivos com sensores embutidos. Existe uma colheita direta de informação contextual nos sensores locais, não existindo nenhuma camada lógica ou física de aquisição ou processamento de dados. Esta arquitetura não é apropriada para sistemas distribuídos porque não permite o acesso de múltiplos clientes;
- Infraestrutura *Middleware*: Abordagem que introduz uma arquitetura de camadas, utilizando métodos de encapsulamento para separar diferentes interfaces de utilizador;
- Servidor de contexto: Abordagem distribuída que expande a arquitetura de infraestrutura *Middleware* através da introdução de um componente remoto de gestão de acessos, funcionando numa arquitetura cliente-servidor. O armazenamento dos dados contextuais dos sensores é efetuado nesse servidor,

suportando o acesso de múltiplos clientes em simultâneo. Esta abordagem tem a vantagem acrescida de aliviar carga sobre os dispositivos de aquisição do contexto, *e.g.*, equipamentos móveis com limitações a nível de processamento, memória e espaço em disco.

A Figura 2.2 representa a arquitetura do sistema *context-aware* distribuído organizada em camadas proposta por (Ailisto *et al.*) [15]:

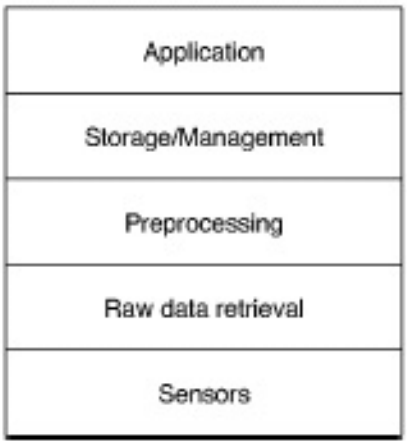

Figura 2.2: Estrutura em camadas de um CARS [14]

A primeira camada é composta pelos sensores do sistema [16]:

- Sensores físicos: Um sensor físico é um sensor de *hardware* capaz de medir grandezas físicas;
- Sensores virtuais: Elementos que adquirem informação contextual a partir de aplicações ou serviços, *e.g.* utilização de informação sobre atividade de um utilizador numa qualquer aplicação para obter informação contextual temporal;
- Sensores lógicos: Sensores que combinam informação proveniente dos sensores físicos, virtuais ou de outras fontes, *e.g.* bases de dados.

A segunda camada é responsável pela recolha de informação contextual em estado bruto, utilizando *drivers* apropriados para aceder aos sensores físicos e *Application Programming Interfaces* (API) para aceder a sensores lógicos e virtuais. A obtenção da informação dos sensores realiza-se através de um mecanismo de *query* tipicamente implementado através de componentes reutilizáveis de *software*, tornando os detalhes de baixo nível de acesso ao hardware transparentes.

A terceira camada é uma camada opcional (pré-processamento). Esta camada é responsável, em primeiro lugar, pela análise e interpretação da informação contextual recolhida pelas camadas anteriores. Esta camada também adapta a informação obtida pela camada anterior para posterior utilização. Esta adaptação consiste em operações de extração e quantificação, elevando os resultados obtidos na camada anterior para um nível superior de abstração.

A quarta camada, camada de armazenamento e gestão, é responsável pela organização da informação recolhida e processada nas camadas anteriores e pela sua disponibilização através de interfaces públicas. O acesso à informação pode ser assíncrono ou síncrono. No primeiro caso, o cliente subscreve os eventos que lhe interessam e, quando ocorre um dos eventos definidos, o utilizador recebe uma notificação ou é invocado diretamente um método do cliente (*call back*). No segundo caso, o cliente interroga o servidor à procura de alterações através da invocação de um método remoto, *i.e.*, envia uma mensagem (*polling message*) solicitando a informação, e aguarda pela resposta do servidor. O método de comunicação síncrona obtém mais rapidamente a informação, mas exige uma maior ocupação de recursos (*resource intensive*) quando comparado com o método assíncrono.

A quinta e última camada, camada de aplicação é onde é construída a aplicação cliente. A reação a eventos específicos e utilização de informação contextual recolhida são feitas nesta camada.

2.6 Representação da Informação

O modelo clássico de representação de informação em sistemas de recomendação é um modelo bidimensional baseado na representação do perfil do utilizador e do item. Todavia, quando se adiciona o contexto, este modelo transforma-se num modelo multidimensional [17] que pode ser descrito através da Equação 2.1:

Equação 2.1: Modelo multidimensional de representação de informação

$$R = Utilizador \times Item \times Contexto \tag{2.1}$$

onde $Utilizador$, $Item$ e $Contexto$ representam o utilizador, o item e o contexto atual do utilizador. O perfil do utilizador é constituído pelos dados pessoais, histórico do utilizador, *e.g.*, visualizações de itens, visualizações e classificações de itens, e preferências explícitas. O perfil do item é composto pelo tipo de conteúdo, normalmente sob a forma de meta-dados, *e.g.*, os géneros dos vídeos. Finalmente, o contexto é constituído por todos os elementos de contexto que sejam considerados relevantes para um dado sistema de recomendação, *e.g.*, informação temporal e espacial do utilizador.

2.7 Algoritmos de Recomendação

O objetivo de um sistema de recomendação é antever a relevância que um item terá para um utilizador, permitindo criar uma lista de itens de potencial interesse. A forma como esta filtragem é feita depende do algoritmo de recomendação utilizado. Existem três tipos de algoritmos de recomendação: (*i*) filtragem colaborativa – *Collaborative Filtering* (CF); (*ii*) filtragem baseada no conteúdo – *Content-based Filtering* (CBF); e (*iii*) filtragem híbrida, que combina múltiplos algoritmos de filtragem baseada no conteúdo e colaborativa.

2.7.1 Filtragem Colaborativa

A filtragem colaborativa consiste na predição da classificação que um utilizador daria a um dado item, baseando-se nas opiniões de outros utilizadores do sistema. Esta família de algoritmos recorre às classificações que os vários utilizadores atribuíram aos itens que viram e ao enquadramento do utilizador ativo num grupo de utilizadores com padrões de classificação (gostos/preferências) semelhantes para efetuar recomendações ao utilizador ativo. Esta abordagem pressupõe que utilizadores com gostos semelhantes classificam de forma semelhante um mesmo item [18]. Os algoritmos CF podem ser divididos em [19]:

- *Memory-based Collaborative Filtering*: São feitas correlações *user-to-user* e *item-to-item*, baseando-se nas classificação que os utilizadores atribuíram aos itens, para prever a classificação de novos itens. É escolhido um grupo de utilizadores semelhantes com o utilizador ativo e, através de uma combinação ponderada das classificações desse grupo de utilizadores, são feitas recomendações para o utilizador ativo;

- *Model-based Collaborative Filtering*: São estimados parâmetros de modelos estatísticos para definir a maior ou menor probabilidade de um dado item poder ser do interesse do utilizador ativo e, por último, são recomendados os itens com maior probabilidade.

A Figura 2.3 ilustra o princípio de funcionamento de um sistema de recomendação CF.

2.7.2 Filtragem Baseada no Conteúdo

Os algoritmos de filtragem baseada no conteúdo consideram que itens com características semelhantes serão classificados pelo utilizador de forma idêntica. Assim, levam em consideração o perfil do utilizador ativo (preferências e interesses) e o perfil dos itens para recomendar itens com o mesmo tipo de conteúdo

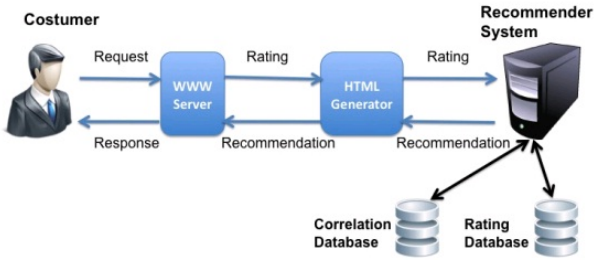

Figura 2.3: Collaborative filtering [19]

(meta-dados) que os itens previamente vistos pelo utilizador. O processo de recomendação consiste, portanto, na determinação do grau de correspondência entre os atributos do perfil de utilizador e atributos do perfil do item, *i.e.*, na eventual relevância que o utilizador atribuirá ao item [20]. A Figura 2.4 representa o funcionamento deste tipo de algoritmos.

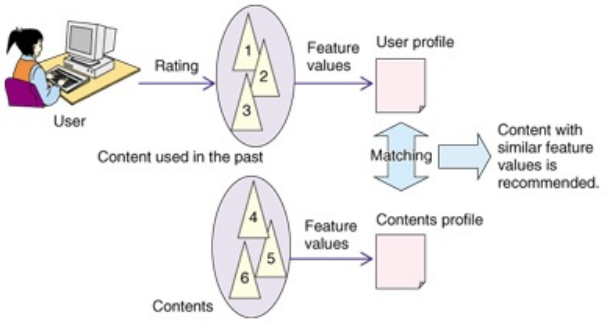

Figura 2.4: Content-based filtering[1]

2.7.3 Análise Comparativa

Os algoritmos de recomendação anteriormente descritos apresentam vantagens e desvantagens distintas. A Tabela 2.1 lista as principais vantagens e desvantagens destas duas famílias de algoritmos [21, 22, 23, 24]:

Os sistemas baseados em CBF apresentam recomendações transparentes, *i.e.*, o conjunto de características de conteúdo (*content features*) que levaram à recomendação de um dado item é conhecido, o que não acontece num sistema CF. Em relação à escalabilidade, o desempenho dos algoritmos CF decresce quando o número de itens e de utilizadores é muito elevado, sendo necessário um elevado número de recursos computacionais para gerar recomendações. Os sistemas CBF dependem dos meta-dados de caracterização do conteúdo dos itens e o seu desem-

[1] `http://findoutyourfavorite.blogspot.pt/2012/04/content-based-filtering.html`

Tabela 2.1: Comparação de algoritmos de recomendação

Características	**CF**	**CBF**
Sistema transparente	Não	Sim
Escalabilidade	Não	Sim
Independência da qualidade e quantidade de metadados	Sim	Não
Recomendações a um novo utilizador	Não	Não
Capacidade de recomendar um novo item	Não	Sim
Independência de classificações feitas por utilizadores	Não	Sim
Consideração da qualidade do item	Sim	Não
Diversificação de recomendações	Sim	Não
Qualidade de recomendação	Superior	Inferior

penho é bastante afetados quando a qualidade e/ou quantidade desta informação é reduzida. Ambas as famílias de algoritmos de recomendação são geralmente incapazes de fornecer recomendações adequadas a um novo utilizador, *i.e.*, sem perfil do utilizador não é possível estabelecer correlações com os meta-dados dos itens (CBF), nem enquadrar o utilizador num grupo de utilizadores com preferências semelhantes para recomendar itens do agrado desse grupo de utilizadores (CF). No que diz respeito à recomendação de novos itens, enquanto que o CBF é capaz de recomendar um novo item dado que apenas necessita dos meta-dados para cruzar com o perfil do utilizador ativo, o CF é incapaz de o fazer porque ainda nenhum utilizador classificou o item. A dependência das classificações dos utilizadores existente em CF dá origem a dois fenómenos: (*i*) os *shilling attacks*, que acontecem quando um utilizador classifica de forma positiva os seus itens e de forma negativa os itens de outros utilizadores (*e.g.* produtos de competidores); e (*ii*) os utilizadores *gray sheep*, que não explicitam preferências, nem classificam itens, impossibilitando a produção de recomendações. A maior desvantagem dos sistemas CBF consiste na pouca diversidade das recomendações produzidas. Ao invés, os sistemas CF são capazes de oferecer recomendações diversas e até com serendipidade (*serendipity*), *i.e.*, recomendações variadas e que constituem uma surpresa para o utilizador. Por último, uma vez que os sistemas CF levam em conta não só as características dos itens, mas também a sua classificação (o que não acontece no CBF), permitem concluir que um sistema CF providencia aos seus utilizadores geralmente recomendações de maior qualidade que um sistema CBF.

Estes dois tipos de algoritmos, dadas as suas caraterísticas distintas, complementam-se, permitindo em conjunto ultrapassar a maioria dos problemas de que enfermam individualmente. Surgiram assim os algoritmos híbridos que combinam de

alguma forma algoritmos das duas famílias. Os algoritmos híbridos sofrem apenas do problema da recomendação de itens a novos utilizadores. Segundo [25], esta questão pode ser resolvida de uma das seguintes formas:

- Requerer ao utilizador a definição dos seus gostos explicitamente durante o processo de registo;
- Apresentar recomendações não personalizadas (*population averages*);
- Obter informação demográfica do utilizador (implicitamente ou explicitamente) e utilizá-la para apresentar recomendações baseadas nas preferências de utilizadores idênticos.

2.7.4 Filtragem Híbrida

Um sistema de filtragem híbrida - *Hybrid Filtering* (HF) - é geralmente o resultado da combinação de vários algoritmos de recomendação para potenciar as vantagens e minimizar as desvantagens dos CF e CBF. Segundo [26], as arquiteturas híbridas mais comuns são:

- Ponderada (*Weighted*): Os dois tipos de algoritmos de recomendação mantêm uma classificação separada de cada item, resultando a recomendação final de uma combinação linear dos dois resultados intermédios;
- Mista (*Mixed*): Cada um dos algoritmos constituintes gera recomendações que são apresentadas conjuntamente ao utilizador;
- Comutada (*Switching*): Os vários algoritmos estão ordenados e, caso o algoritmo mais bem cotado não seja capaz de oferecer uma recomendação com a exatidão desejada, o sistema passa ao algoritmo seguinte;
- Combinação de recursos (*Feature Combination*): Sistemas que aplicam apenas um componente de recomendação, sendo este suportado por um segundo componente passivo. Em vez de processar em separado os recursos do componente passivo, estes recursos são injetados no algoritmo ativo;
- Ampliação de recursos (*Feature Augmentation*): Semelhante aos sistemas de combinação de recursos, mas não utilizam recursos em estado bruto. Estes recursos são passados para servirem de apoio ao algoritmo ativo e são usados quando existe um algoritmo eficiente que necessita de outras fontes de informação para o auxiliar no processo de recomendação;
- Cascata (*Cascade*): A escolha de itens é feita pelo algoritmo primário/ativo, sendo o algoritmo de apoio apenas utilizado para refinar a classificação (*scores*) de cada item;

- Meta-nível (*Meta Level*): Um dos algoritmos produz um modelo de recomendação que é depois utilizado pelo algoritmo primário/ativo no seu processo de recomendação (*output* de um algoritmo como *input* de outro).

2.8 Vídeo-imersão

A imersão é uma experiência subjetiva de total envolvimento num determinado ambiente e que pode ser sugerida recorrendo a som *surround*, experiências sensoriais, alta resolução e realidade aumentada. Esses recursos estão disponíveis em vários elementos multimédia, entre os quais se destaca o vídeo, onde é possível envolver o utilizador na experiência criando nele reações emocionais [27]. Um exemplo deste tipo de aplicações é o Windy Sight Surfers (WSS). Trata-se de um sistema de vídeo-imersão que permite a captura, pesquisa, publicação, visualização e sincronização de itens de TV interativa. No processo de captura vídeo são obtidos e armazenados meta-dados com informação relevante para o processo de imersão (informação espacial, velocidade, orientação e informação meteorológica). No processo de publicação do vídeo, o utilizador faz o carregamento do ficheiro vídeo e do ficheiro com os meta-dados e é automaticamente associada informação de indexação tal como a cidade/país onde o vídeo foi gravado. Aquando da visualização do vídeo, o sistema apresenta as seguintes funcionalidades de perceção [28]:

- Perceção visual: Em vídeos 360º o vídeo é representado em ecrã inteiro num *smartphone* numa tela transitória que pode ser rodada pelo toque do utilizador;
- Perceção táctil: Foi acoplado ao *smartphone* uma ventoinha externa que permite recriar as condições de vento (orientação e velocidade);
- Perceção auditiva: Através de um mapeamento em três dimensões do espaço sonoro é possível adaptar o som emitido pelo vídeo em relação à direção atual da câmara. Para além deste efeito é adicionado um efeito *doppler* cíclico, que permite recriar a noção de movimento através da manipulação sonora criando o efeito *doppler*.

A Figura 2.5 representa os fenómenos de perceção táctil e visual aquando da visualização de um vídeo 360º.

2.9 Sistemas Congéneres

Existe um número considerável de sistemas de recomendação que utilizam informação contextual para gerar recomendações personalisadas. Nesta Secção

Figura 2.5: Aplicação Windy Sights Surfers [28]

referem-se as aplicações que utilizam informação contextual do utilizador, maioritariamente do tipo espacial, para recomendar itens georreferenciados, *e.g. Point of Interest* (POI).

2.9.1 ErasmusApp

A ErasmusApp é uma aplicação móvel criada em 2012 por Karel Bruyneel no âmbito do MEEC, que pretende auxiliar os estudantes Erasmus durante a sua estadia na cidade do Porto. Consiste num sistema de recomendação consciente do contexto (*context-aware*) suportado por uma arquitetura de distribuída, incluindo um modulo cliente de interface com o utilizador e um modulo servidor que acede e gere a informação de uma base de dados central interligados através de um protocolo dedicado de nível de aplicação.

A aplicação apresenta dois modos de funcionamento distintos: *stand-alone* e partilhado. No modo de funcionamento *stand-alone* não há comunicação com o módulo servidor, podendo o utilizador armazenar locais pessoais na base de dados local do dispositivo móvel. No modo de funcionamento partilhado, que obriga a autenticação prévia, o utilizador obtém recomendações, classifica e partilha locais com outros utilizadores. A interface permite realizar as seguintes ações [1]:

- Visualizar mapa: Apresenta um mapa a localização corrente do utilizador, possibilitando a exploração da sua redondeza;
- Aceder a informação básica de Erasmus: Obtém informação sobre os hospitais mais próximos, localização do serviço *Erasmus Student Network* (ESN) da cidade do Porto, alojamento e pontos de elevado interesse (Bar Piolho, Cais da Ribeira, Praia de Matosinhos, *etc.*);
- Procurar localização: Apresenta uma lista de locais recomendados levando em conta a distância entre utilizador e local, a classificação média desse local e, caso seja especificada, a categoria de pesquisa;
- Visualizar locais pessoais: Apresenta e permite gerir (eliminar, editar, partilhar) os locais pessoais armazenados na base de dados local;

- Alterar perfil: Permite alterar a informação do perfil do utilizador armazenada na base de dados central do sistema.

A Figura 2.6 apresenta uma lista de locais recomendados pela aplicação, que pode ser ordenada por proximidade, classificação média ou pela ponderação destas duas componentes.

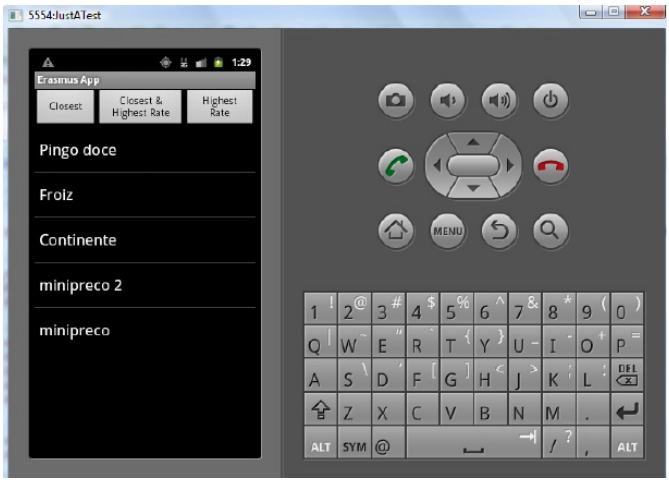

Figura 2.6: Aplicação ErasmusApp [1]

Posteriormente, em 2013, Bjorn Baecke enriqueceu a ErasmusApp com um sistema de partilha de vídeos e de pesquisa de artigos da Wikipédia relacionados com locais situados na vizinhança do utilizador. No âmbito desta segunda tese, foram adicionadas as seguintes funcionalidades [2]:

- Explorar wiki-vizinhança: Permite ao utilizador pesquisar artigos Wikipédia relacionados com locais situados na sua vizinhança através da definição do raio de pesquisa, sendo os itens representados através de marcadores vermelhos sobre um mapa;
- *Upload* de Vídeo: Permite ao utilizador gravar e partilhar um vídeo com os restantes utilizadores do sistema, sendo registada e armazenada informação complementar durante a gravação do vídeo, *e.g.*, dados meteorológicos e dos sensores físicos do dispositivo móvel;
- Explorar vídeo-vizinhança: Permite ao utilizador pesquisar e reproduzir vídeos relacionados com locais situados na sua vizinhança através da definição do raio de pesquisa, sendo os vídeos representados sobre um mapa;
- Mostrar locais próximos: Permite ao utilizador visualizar os locais próximos do local que pretende partilhar para evitar a introdução de locais duplicados na base de dados central.

A Figura 2.7 ilustra a exploração da wiki-vizinhança.

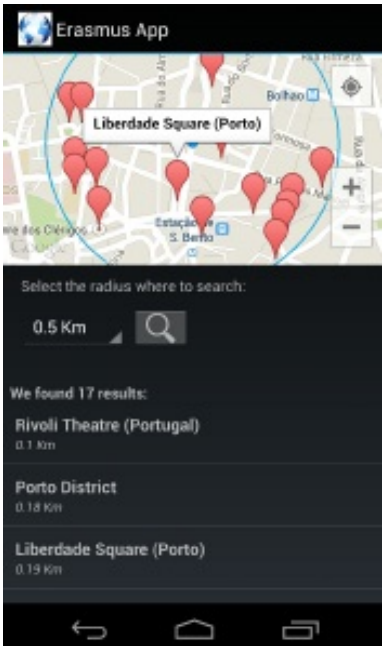

Figura 2.7: Funcionalidade de exploração da wiki-vizinhança [2]

2.9.2 Cinemappy

Cinemappy é uma aplicação móvel de recomendação de filmes sensível ao contexto. Utiliza informação contextual, *e.g.*, a localização do utilizador, para recomendar filmes em exibição em cinemas próximos. A informação relativa aos itens disponíveis no sistema é extraída da base de dados abertos DBpedia (género, atores, realizador, tema, *etc.*). Em termos da informação contextual, este sistema, para além da informação espacial, leva em conta [29]: (*i*) a companhia, *i.e.*, a(s) pessoa(s) que acompanha(m) o utilizador; (*ii*) o tempo, *e.g.*, apenas filmes com sessões a iniciarem-se num futuro próximo são recomendados; (*iii*) a co-localização, *e.g.*, recomenda cinemas localizados na vizinhança de outros POI; e (*iv*) a proximidade de pontos-âncora, *i.e.*, recomenda cinemas na vizinhança de pontos âncora previamente definidos como a casa ou o emprego.

O sistema aplica um filtro constituído por três diferentes componentes: um pré-filtro contextual, um algoritmo de recomendação CBF e um pós-filtro contextual. No pré-filtro considera a localização e informação temporal atual do utilizador para selecionar filmes que se encontrem em exibição e num raio de x km do utilizador. Caso o utilizador esteja acompanhado, é criado um micro-perfil que contempla as preferências do(s) acompanhante(s) do utilizador ativo. De seguida, o algoritmo de recomendação CBF faz a correlação do perfil do utilizador,

contemplando um eventual micro-perfil caso este exista, com a informação de conteúdo dos itens (filmes). O pós-filtro contextual apenas ordena os itens resultantes, considerando aspetos como a co-localização e a proximidade a pontos âncora. A Figura 2.8 apresenta a interface do Cinemappy.

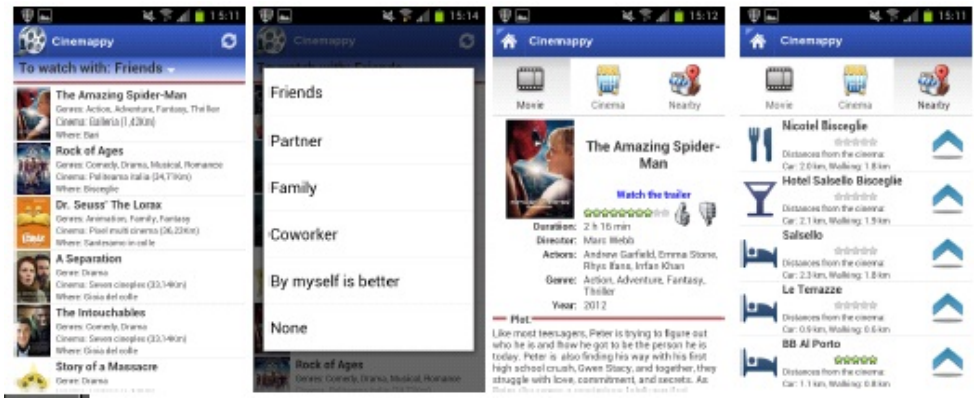

Figura 2.8: Aplicação Cinemappy[29]

2.9.3 MoreTourism

O MoreTourism é uma plataforma de recomendação de pontos de interesse turísticos. Aplica um algoritmo de recomendação híbrido composto por um algoritmo CF e um algoritmo CBF, utilizando anotações sociais (*tags*) introduzidas pelos utilizadores. O sistema de *tags* sociais é constituído pelas *user tag cloud* e *atraction tag cloud*. Enquanto a primeira é constituída pelas *tags* e classificações atribuídas pelos utilizadores ao POI em questão, a segunda é composta pelas *tags* que descrevem o POI. Quanto maior for o número de atribuições de uma *tag*, maior será o seu peso na respetiva *cloud* e, consequentemente, maior peso terá no algoritmo CBF. Correlacionando ambas *clouds* e tendo em conta possíveis relações entre *tags*, o algoritmo CF cria uma *user tag cloud* para o conjunto de utilizadores que classificaram positivamente cada POI. Neste caso, o sistema compara a *tag cloud* do utilizador ativo com a *user tag cloud* de cada POI e seleciona os POI com maior grau de semelhança. Este sistema utiliza as seguintes componentes de informação contextual [30]: (*i*) espacial, *i.e.*, a localização atual do utilizador; e (*ii*) tempo disponível para a visita, *i.e.*, para recomendar uma rota ou um único POI do agrado e em função do tempo disponível do utilizador. A Figura 2.9 ilustra o algoritmo de filtragem utilizado.

2.9.4 Rerex

O Rerex é uma aplicação turística para IPhone que permite ao utilizador um elevado nível de personalização. Cabe ao utilizador definir o contexto da sua

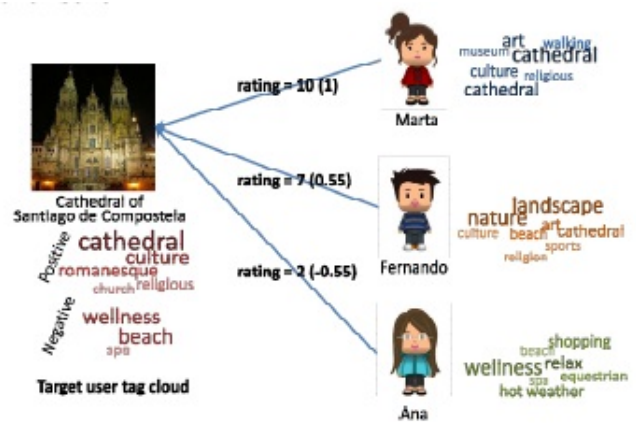

Figura 2.9: Social Collaborative Filtering [30]

visita, podendo definir manualmente os parâmetros do contexto que devem ser levados em conta durante o processo de recomendação dos pontos de interesse a visitar. O sistema contempla a seguinte informação contextual [31]:

- Distância entre utilizador e POI;
- Temperatura externa;
- Condições atmosféricas atuais;
- Companhia (acompanhado por amigo(s), familiar(es), filho(s), *etc.*);
- Tempo (dia da semana, hora atual, estação do ano);
- *Crowdedness* (se é expectável que o POI seja muito ou pouco frequentado);
- Familiaridade que o utilizador tem com a localidade;
- Estado de espírito do utilizador;
- Motivo da viagem;
- Orçamento disponível para a viagem;
- Duração da viagem;
- Meio de transporte utilizado.

Caso as recomendações obtidas não sejam do seu agrado, o utilizador pode especificar os tipos de POI do seu interesse (museus, restaurantes, hotéis, *etc.*) e requerer novas recomendações. A Figura 2.10 ilustra a introdução da informação de contexto é feita e a apresentação das recomendações.

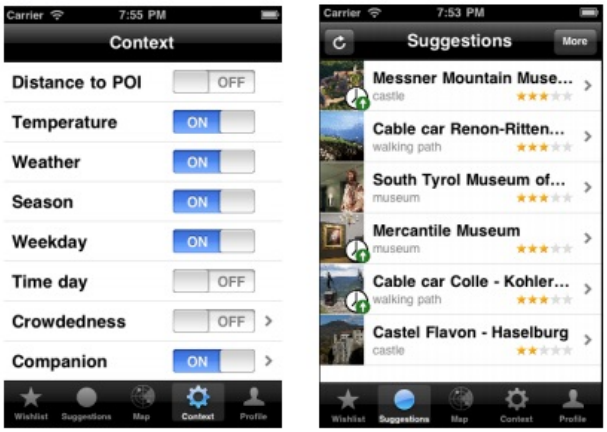

Figura 2.10: Aplicação Rerex [31]

2.9.5 Geosocial SPLIS

O Geosocial SPLIS é um serviço do tipo rede social baseado na localização – *Location Based Social Networking Service* (LBSNS) – resultante da evolução de sistemas criados anteriormente: PLIS+ [32] e SPLIS [33]. Trata-se de um sistema baseado em regras (*rule-based*) que permite a especificação de conjuntos de regras, *e.g.*, ao fim de semana entre as 19:00 e as 21:00 o utilizador pretende obter recomendações de restaurantes situados nas proximidades. Este mecanismo, em conjunto com a informação de contexto, permite produzir recomendações personalizadas. O perfil de utilizador é construído a partir da informação retirada da rede social Google+ e do formulário de registo. O utilizador define as regras a aplicar aos pontos de interesse através de um editor *Web*. Este editor permite especificar [34]: (*i*) informação meteorológica; (*ii*) propriedades definidas nos POI; (*iii*) informação temporal (dia/hora/mês); e (*iv*) informação espacial (distância máxima entre utilizador e POI). Por último, o sistema possibilita a interação entre utilizadores (ligações de amizade), combinando preferências e partilhando recomendações entre amigos próximos. A Figura 2.11 apresenta o menu de gestão de regras, que permite ao utilizador definir o conjunto de regras pessoais ou selecionar regras de entre o conjunto de regras mais populares entre todos os utilizadores ou entre os seus amigos.

A Figura 2.12 mostra o menu inicial da aplicação, a posição atual do utilizador bem como os pontos de interesse recomendados, *i.e.*, resultantes do conjunto de regras ativo e do contexto do utilizador.

2.9.6 CAMR

A *Context-Aware Media Recommendation* (CAMR) é uma plataforma de recomendação de conteúdos sensível ao contexto destinada a dispositivos móveis

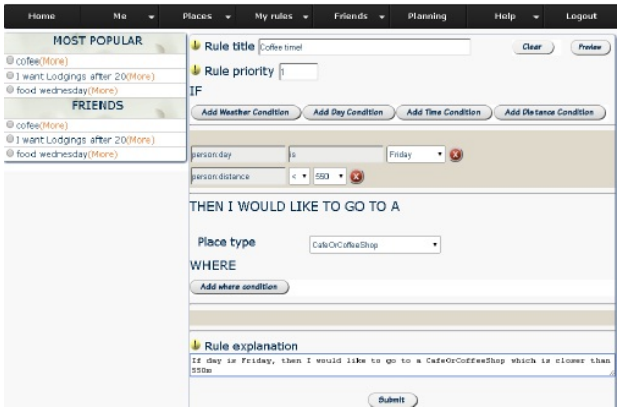

Figura 2.11: Geosocial SPLIS: Editor de regras [34]

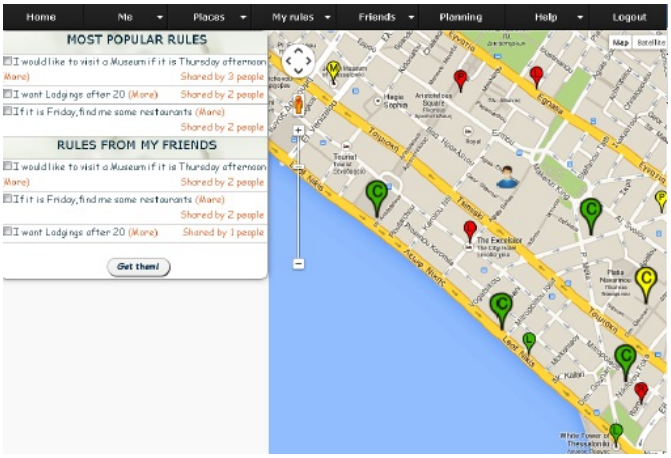

Figura 2.12: Geosocial SPLIS: Menu inicial [34]

[35]. Tem por objetivo utilizar a informação contextual recolhida no dispositivo móvel do utilizador para recomendar diferentes conteúdos (vídeos, música, texto, *etc.*). A Figura 2.13 apresenta a arquitetura do CAMR. O sistema inclui quatro serviços:

- Serviço de Reconhecimento de Contexto: Serviço que corre no dispositivo móvel do cliente responsável por monitorizar, aprender e prever o contexto do utilizador ativo. Este recolhe informação proveniente dos sensores do dispositivo, pré-processando e inferindo informação contextual de alto nível;
- Serviço de Construção do Perfil do Utilizador: Recolhe a informação contextual proveniente do serviço anterior e estima e aprende as preferências de consumo de elementos média do utilizador ativo. Estas preferências são correlacionadas com os contextos do utilizador ativo (atual e passados) e

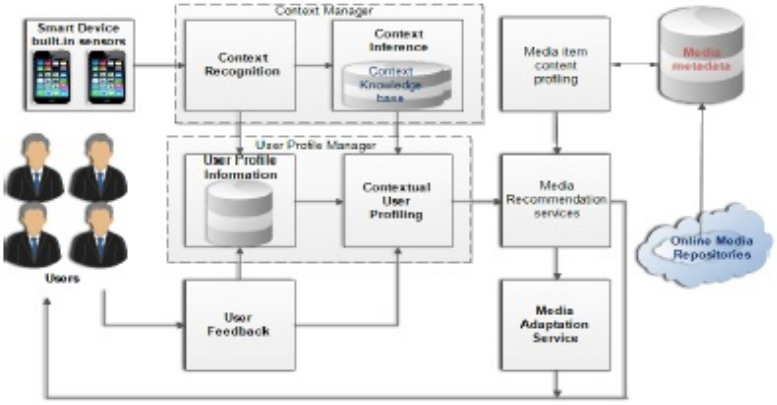

Figura 2.13: Arquitetura CAMR [35]

dos restantes utilizadores do sistema. Este serviço armazena o histórico de consumo e preferências do utilizador e é também responsável pela atualização do perfil do utilizador em conjunto com o serviço de reconhecimento de contexto;

- Serviço de Recomendação: Composto por algoritmos de recomendação conscientes do contexto e por um módulo de perfilamento de conteúdo média. Este módulo obtém meta-dados de conteúdo considerado relevante na *Web* e faz a recomendação de conteúdo média aos utilizadores tendo em conta o seu contexto e preferências. Este serviço utiliza CF, CBF e algoritmos híbridos;
- Serviço de Adaptação: Determina se o conteúdo média recomendado se encontra no formato mais apropriado às características do dispositivo móvel do cliente e às condições de ligação (rede), adaptando o formato deste conteúdo para um que seja apropriado no caso caso de ser considerado necessário.

Em termos de contexto, o sistema contempla, entre outras, as seguintes componentes:

- Local: Categoria do local onde o utilizador se encontra (casa, escritório, escola, centro comercial, *etc.*);
- Dia da Semana: Nome do dia da semana (segunda-feira, terça-feira, *etc.*);
- Fase do dia: Define a fase do dia (manhã, tarde, noite);
- Estado meteorológico: Define o estado de um determinado instante na localização do utilizador;

- Temperatura: Valor numérico da temperatura num determinado instante;
- Atividade: Estado de atividade do utilizador (sentado, a caminhar, a correr, a conduzir, a subir/descer escadas, *etc.*);
- Iluminação: Nível de iluminação do local onde o utilizador se encontra, representado num valor numérico;
- Ruído: Nível de ruído do local onde o utilizador se encontra, representado num valor numérico.

2.9.7 Análise Comparativa

A tabela 2.2 resume as principais características de cada um dos sistemas apresentados.

Tabela 2.2: Comparação de sistemas congéneres

Características	**Sistemas Congéneres**					
	ErasmusApp	Cinemappy	MoreTourism	Rerex	SPLIS	CAMR
Espaço	✓	✓	✓	✓	✓	✓
Tempo	✗	✓	✓	✓	✓	✓
Companhia	✗	✓	✗	✓	✗	✗
Meteorologia	✗	✗	✗	✓	✓	✓
Crowdedness	✗	✗	✗	✓	✗	✗
Estado de espírito	✗	✗	✗	✓	✗	✗
Temperatura	✗	✗	✗	✓	✗	✓
Atividade	✗	✗	✗	✗	✗	✓
Iluminação	✗	✗	✗	✗	✗	✓
Ruído	✗	✗	✗	✗	✗	✓
CBF	✗	✓	✗	✗	✗	✓
CF	✗	✗	✗	✗	✗	✓
HF	✗	✗	✓	✗	✗	✓
Outro	✓	✗	✗	✓	✓	✗
Vídeo-imersão	✗	✗	✗	✗	✗	✗

A Tabela 2.2 permite concluir que: (*i*) em termos da informação contextual, os sistemas Rerex e CAMR são os que utilizam uma maior diversidade; (*ii*) em termos de algoritmos de recomendação não existe nenhum padrão, destacando-se a plataforma CAMR pela aplicação dos diversos tipos de algoritmos; e (*iii*) em termos de imersão, nenhuma das plataformas estudadas oferece qualquer tipo de funcionalidade.

2.10 Sumário

Neste capítulo foram introduzidos os principais conceitos abordados nesta dissertação, nomeadamente, a definição de contexto do utilizador, de sistema

CARS e de vídeo-imersão. Foram ainda apresentados e comparados os principais algoritmos de recomendação assim como os sistemas congéneres encontrados na literatura.

No próximo capítulo serão abordadas as tecnologias e ambientes de desenvolvimento utilizados na realização do protótipo desenvolvido.

Capítulo 3

Ambiente de Desenvolvimento

Este capítulo apresenta o conjunto de tecnologias, bibliotecas e ferramentas utilizadas no desenvolvimento do protótipo, abrangendo o sistema operativo Android, as bibliotecas de interface, as soluções de armazenamento persistente de dados e o ambiente integrado de desenvolvimento adotados.

3.1 Sistema Operativo Android

O Android é um sistema operativo para dispositivos móveis. Foi inicialmente desenvolvido pela empresa Android Inc., inserido no âmbito do projeto *Android Open Source Project* (AOSP), sendo atualmente propriedade da Google Inc. O desenvolvimento e distribuição desta tecnologia foram realizados em parceria com vários membros da *Open Handset Alliance* (OHA). A Figura 3.1 resume a arquitetura do sistema operativo Android.

Um sistema Android é baseado num *kernel* Linux 2.6 modificado para a otimização de recursos. Esta otimização é necessária dada a limitação de recursos que é comum nos dispositivos móveis. Esta camada é responsável, entre outras coisas, pela gestão de processos, memória, potência e funcionalidades de rede. A camada das bibliotecas (*Libraries*) utiliza como linguagem de interface a linguagem Java e é responsável pela interface com o utilizador (através do *Surface Manager*). A camada *Android Runtime* contém as principais bibliotecas que providenciam as funcionalidades básicas do sistema Android assim como a máquina virtual usada (*Dalvik Virtual Machine*). A camada de *Application Framework* aloja a API, monitoriza o ciclo de vida das aplicações em execução e é responsável pelo acesso e partilha de informação entre aplicações.

Finalmente, a camada de aplicações é composta pelo conjunto das aplicações embutidas (*built-in apps*) e das aplicações instaladas pelo utilizador (*user apps*).

No que diz respeito às linguagens de programação, destacam-se três: Java, C e C++. As aplicações são desenvolvidas em Java, as bibliotecas em C ou C++ e o *kernel* é baseado em C. Assim, num sistema Android existe uma distância lógica entre as aplicações e o *kernel* superior à de um sistema Linux [36, 37, 38].

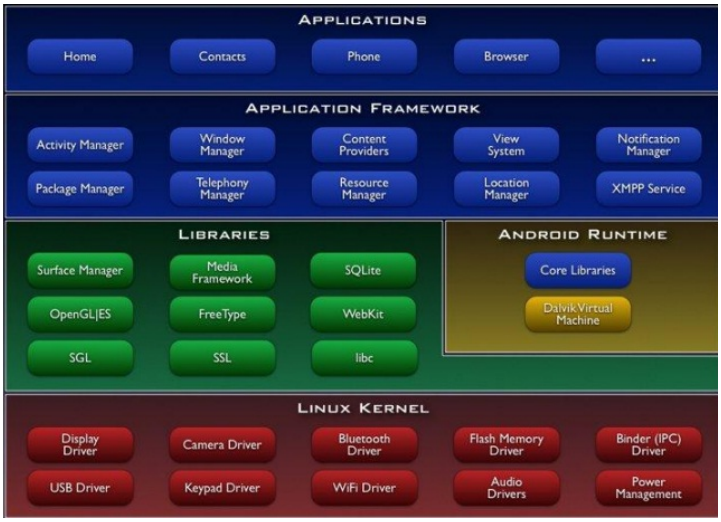

Figura 3.1: Arquitetura de um sistema Android[1]

3.2 Bibliotecas de Interface

O modelo de desenvolvimento de aplicações Android baseia-se na utilização de diversas API para aceder às diversas tecnologias e serviços.

3.2.1 Android API

Desde o seu lançamento em 2008, foram lançadas diversas distribuições Android cada uma com a sua respetiva API. A Tabela 3.1 apresenta o histórico das várias distribuições Android, desde os primórdios da tecnologia até aos dias de hoje, incluindo os aspetos mais relevantes [39].

Como um dispositivo Android só poderá executar aplicações criadas para uma API anterior ou igual à utilizada na sua versão, existe a necessidade de estabelecer um compromisso entre as funcionalidades disponíveis e quota de mercado alvo. A Figura 3.2 representa a quota de mercado correspondente a cada uma das principais versões Android, dados referentes ao mês de dezembro de 2014 [40]. Da análise da Figura 3.2 conclui-se que cerca de 80 % do mercado Android é ocupado por dispositivos com API superiores à versão 16. Assim, apesar de neste trabalho se utilizar como dispositivo de teste um Motorola Moto G com

[1] `http://en.wikipedia.org/wiki/Android_(operating_system)`

Tabela 3.1: Histórico de Android API

Versão	**Nome**	**Lançamento**	**Funcionalidades adicionadas**	**API**
0.9	Sem Nome	18 ago 2008	-	-
1.0	Apple Pie	23 set 2008	Aplicações YouTube, Google Maps e Android Market	1
1.1	Banana Bread	9 fev 2009	Capacidade de gravar anexos no serviço *Multimedia Messaging Service* (MMS)	2
1.5	Cupcake	30 abr 2009	Visualização e gravação de vídeos, suporte à tecnologia Bluetooth	3
1.6	Donut	15 set 2009	Navegação *turn-by-turn*	4
2.0-2.1	Eclair	26 out 2009	Suporte a *HyperText Markup Language* (HTML) e novo *User Interface* (UI)	5-7
2.2	Froyo	20 mai 2010	Melhoria no desempenho (velocidade), suporte a carregamento de ficheiros via *browser*	8
2.3	Gingerbread	6 dez 2010	Suporte a *Voice over IP* (VoIP) e vídeo-chamadas, aumento no número de ferramentas disponíveis para programadores	9-10
3.0-3.2	Honeycomb	22 fev 2011	*HyperText Transfer Protocol* (HTTP) *live streaming*, suporte a processamento *multi-core*, UI atualizado	11-13
4.0	Ice Cream Sandwich	19 out 2011	Reconhecimento facial (sistema de desbloqueio), suporte a clientes *Virtual Private Network* (VPN)	14-15
4.1-4.3	Jelly Bean	9 jul 2012	Melhorias em segurança e desempenho, suporte à resolução 4k	16-18
4.4	KitKat	31 out 2013	Novo UI, API pública para gestão de *Short Message Service* (SMS)	19

Version	Codename	API	Distribution
2.2	Froyo	8	0.5%
2.3.3 - 2.3.7	Gingerbread	10	9.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	7.8%
4.1.x	Jelly Bean	16	21.3%
4.2.x		17	20.4%
4.3		18	7.0%
4.4	KitKat	19	33.9%

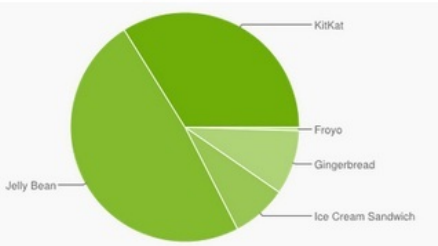

Figura 3.2: Quota de mercado de distribuições Android [40]

distribuição KitKat (API 19), pretende-se que a aplicação móvel seja compatível com a API da versão 16, assegurando um compromisso entre funcionalidades e quota de mercado abrangida.

3.2.2 Google Maps API

De modo a representar os POI na vizinhança do utilizador é necessário utilizar uma interface *map-based*. O Google Maps oferece esta funcionalidade, sendo necessário obter uma chave de acesso à respetiva API, utilizar a biblioteca Google Play Services e definir um conjunto de permissões no ficheiro Android Manifest.

3.2.2.1 API Key

Para requerer uma chave de acesso à API é necessário:

- Aceder a `https://console.developers.google.com/` e autenticar-se;
- Criar um novo projeto;
- Aceder a `API e autenticação->API` (Figura 3.3);
- Criar a sua chave acedendo a `API e autenticação->Credenciais` e escolhendo a opção `Criar nova chave->Chave do Android` (Figura 3.4);
- Obter *fingerprint* SHA1 através `Window->Preferences->Android->Build` (Figura 3.5) do Eclipse IDE;
- Inserir a chave seguida do nome do pacote da aplicação separados por ';' (Figura 3.6).

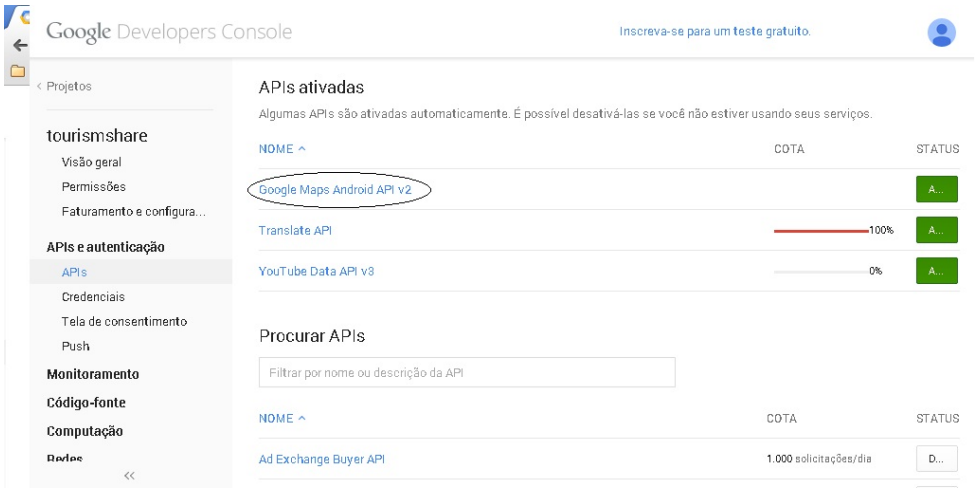

Figura 3.3: Gestão de API

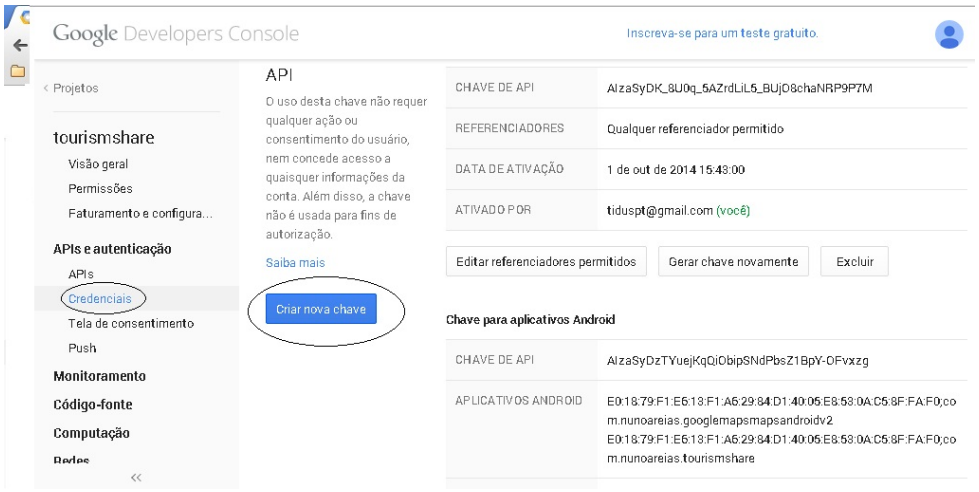

Figura 3.4: Gestão de chaves

3.2.2.2 Google Play Services

A versão 2 da API do Google Maps é parte integrante da biblioteca Google Play Services. Esta biblioteca pode ser descarregada acedendo à consola do *Software Development Kit* (SDK) *Manager* no Eclipse *Integrated Development Environment* (IDE) como ilustra a Figura 3.7. Concluída a descarga, é necessário importar a biblioteca para o *workspace* do projeto através de `File->Import->Android->Existing Android Code Into Workspace`.

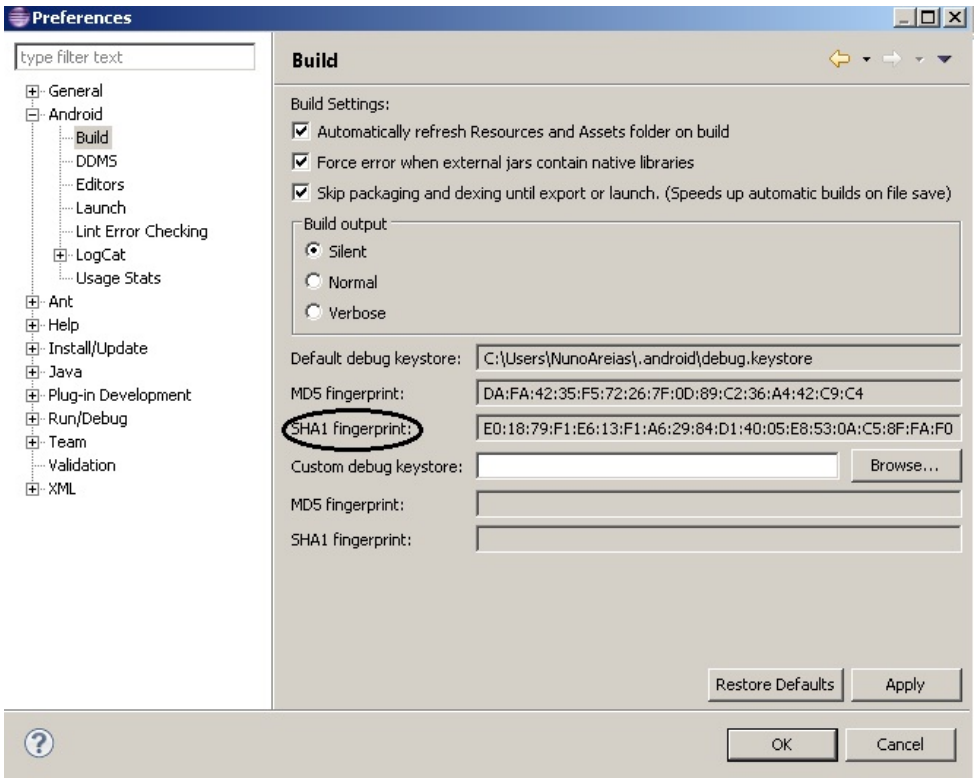

Figura 3.5: Obtenção de chave SHA1

Figura 3.6: Criação de nova chave

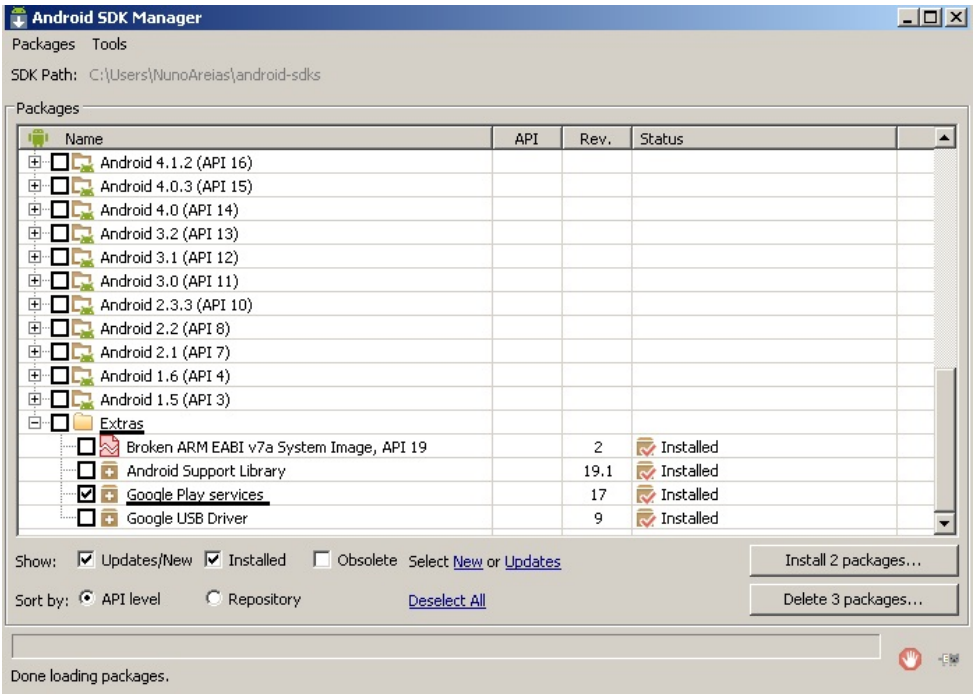

Figura 3.7: Instalação GooglePlay Services

3.2.2.3 Android Manifest

Para finalizar a configuração do Google Maps é necessário alterar o ficheiro do Android Manifest como se ilustra no Excerto de Código 3.1.

Excerto de Código 3.1: Código Android Manifest (Google Maps)

```
// Permissions
<uses-permission android:name="com.nunoareias.tourismshare.permission.MAPS_RECEIVE" />
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
}

//OpenGL ES v2.0
<uses-feature
       android:glEsVersion="0x00020000"
       android:required="true" />

//Meta-data
<application
       ...
       <meta-data
           android:name="com.google.android.gms.version"
           android:value="@integer/google_play_services_version" />
       <meta-data
           android:name="com.google.android.maps.v2.API_KEY"
           android:value="AIzaSyDzTYuejKqQiObipSNdPbsZ1BpY-OFvxzg" />

</application>
```

A utilização da API requer o uso de um número de permissões, incluindo a receção de mapas, acesso ao *Google Play Services*, acesso à Internet, dados externos e estado da ligação à rede. De seguida, é necessário indicar a utilização de uma biblioteca de suporte a gráficos 2D e 3D de alto desempenho (OpenGL ES 2.0) [41]. Finalmente, indica-se a versão do *Google Play Services* e a chave a utilizar para aceder à API.

3.2.3 YouTube API

De modo a implementar um sistema de carregamento e visualização de vídeos é necessário utilizar as API YouTube Data API, Google+ API e criar um ID OAuth.

3.2.3.1 YouTube Data API

Esta API permite o acesso a informação do YouTube, nomeadamente vídeos, canais e *playlists*. A ativação é feita de forma análoga à API do Google Maps e utiliza a mesma chave para aplicações Android. A Figura 3.8 apresenta o processo de ativação da YouTube Data API.

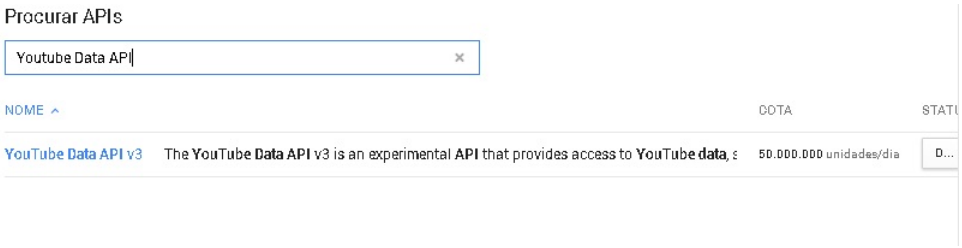

Figura 3.8: Ativação YouTube Data API

3.2.3.2 Google+ API

A Google+ API possibilita a integração da aplicação na plataforma Google+, permitindo que o utilizador se autentique através da sua conta Google e aceda ao seu perfil Google+ durante o processo de carregamento de vídeos. O processo de ativação desta API é semelhante aos anteriores (Figura 3.9).

3.2.3.3 OAuth 2.0

O OAuth é um padrão aberto de autorização de acesso a bibliotecas de interface que permite aos utilizadores de uma dada aplicação aceder e partilhar dados específicos. Esta funcionalidade pode ser ativada seguindo os seguintes passos:

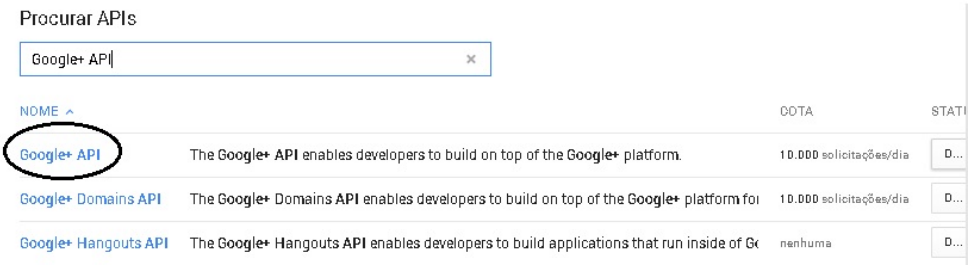

Figura 3.9: Ativação Google+ API

- Aceder a `https://console.developers.google.com/`, autenticando-se através da sua conta Google;
- Aceder API e `Autenticação->Credenciais->Criar um novo ID de cliente` (Figura 3.10);
- `Escolher a aplicação instalada-> Android`, inserir nome do pacote e chave SHA1 (Figura 3.11);
- Configurar a tela de consentimento com nome de produto, email e, opcionalmente, o logótipo da aplicação (Figura 3.12).

Figura 3.10: Menu OAuth

3.2.4 Geonames

O Geonames é um repositório aberto de dados interligados (LOD) que disponibiliza os seus dados através de múltiplos serviços *Web*. Esta base de dados é composta por mais de 10 milhões de nomes geográficos correspondendo a cerca de 7,5 milhões de recursos categorizados segundo as seguintes 9 categorias [42]:

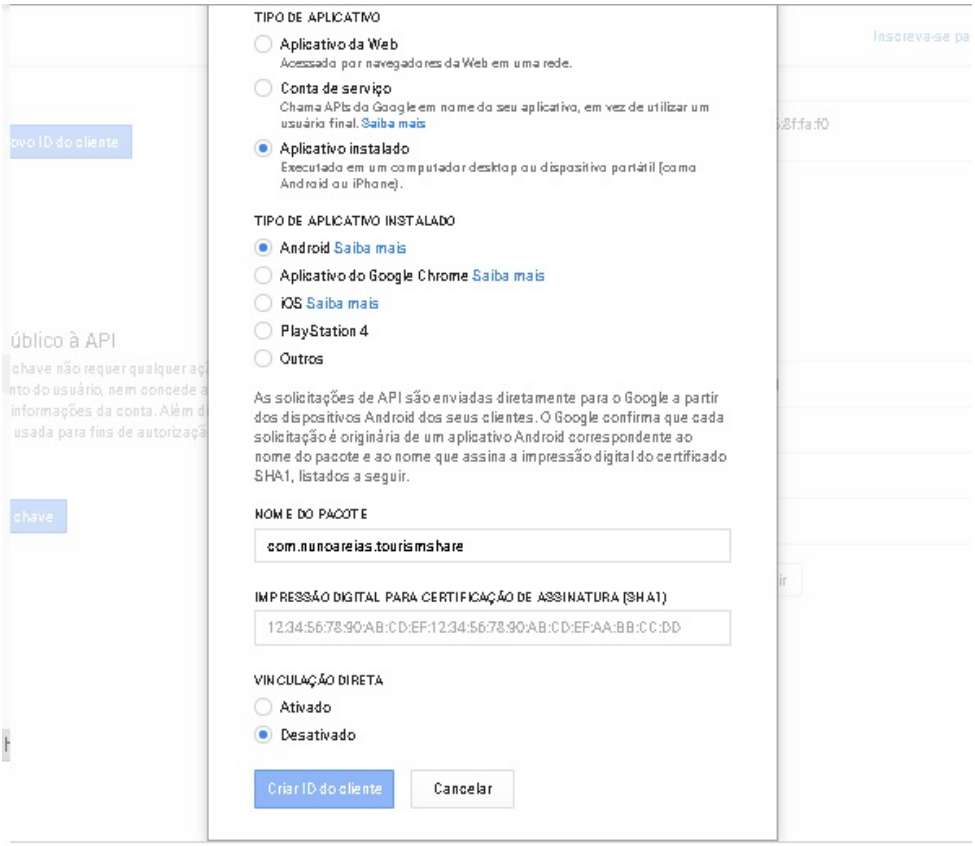

Figura 3.11: Criação de ID de cliente

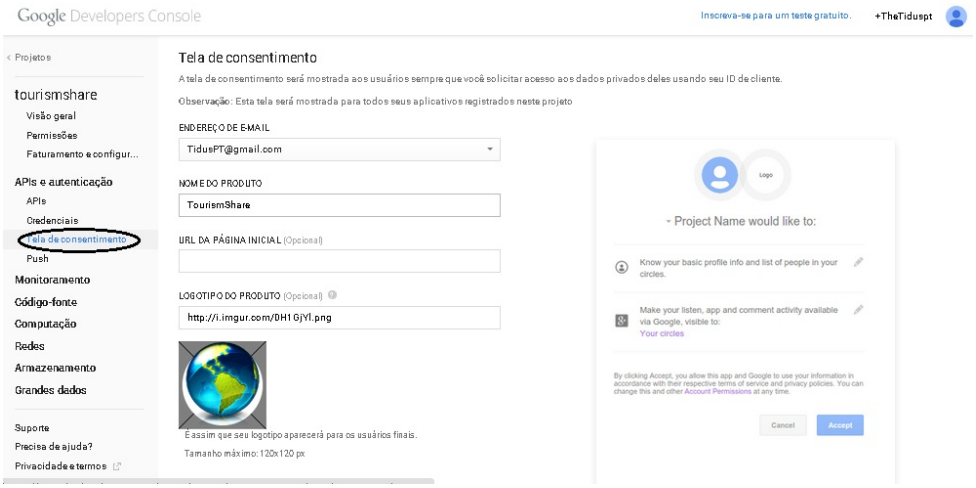

Figura 3.12: Gestão da autorização de acesso

- País, estado ou região;
- Curso de água, lago, *etc.*;
- Parques;
- Cidade, vila, *etc.*;
- Estrada, caminho de ferro, *etc.*;
- Local, edifício, *etc.*;
- Curso de água, lago, quinta;
- Montanha, colina, *etc.*;
- Subaquático;
- Florestas.

O acesso a estes dados é feito através do recurso a um serviço *Web* cuja resposta pode ser fronecida em diferentes formatos, incluindo *eXtensible Markup Language* (XML) e *JavaScript Object Notation* (JSON). Existem atualmente 38 diferentes serviços, estando eles discriminados na Figura 3.13.

Este protótipo utiliza o serviço findNearbyWikipedia que permite aceder a artigos da Wikipedia sobre locais situados num determinado raio de distância em relação a uma posição (no caso específico, a posição do utilizador).

Para aceder ao Geonames é necessário: (*i*) criar uma conta de utilizador em `http://www.geonames.org/login`; e (*ii*) instalar dois ficheiros *Java ARchive* (JAR), que contêm as classes necessárias para aceder e consumir os respetivos serviços. Os ficheiros necessários são:

- jdom-1.0.jar;
- geonames-1.1.12.jar.

A inclusão destes ficheiros no projeto é efetuada através dos seguintes passos: `Project->Properties->Resource->Java Build Path aba Libraries` e, finalmente, `add External JARs`.

[2] `http://www.geonames.org/export/ws-overview.html`

	WebService	XML	JSON	RDF	CSV	TXT	RSS	KML
1	astergdem	XML	JSON			TXT		
2	children	XML	JSON					
3	cities	XML	JSON					
4	contains	XML	JSON					
5	countryCode	XML	JSON			TXT		
6	countryInfo	XML	JSON		CSV			
7	countrySubdivision	XML	JSON					
8	earthquakes	XML	JSON					
9	extendedFindNearby	XML						
10	findNearby	XML	JSON					
11	findNearbyPlaceName	XML	JSON					
12	findNearbyPostalCodes	XML	JSON					
13	findNearbyStreets	XML	JSON					
14	findNearbyStreetsOSM	XML	JSON					
15	findNearByWeather	XML	JSON					
16	findNearbyWikipedia	XML	JSON				RSS	
17	findNearestAddress	XML	JSON					
18	findNearestIntersection	XML	JSON					
19	findNearestIntersectionOSM	XML	JSON					
20	findNearbyPOIsOSM	XML	JSON					
21	get	XML	JSON					
22	gtopo30	XML	JSON			TXT		
23	hierarchy	XML	JSON					
24	neighbourhood	XML	JSON					
25	neighbours	XML	JSON					
26	ocean	XML	JSON					
27	postalCodeCountryInfo	XML	JSON					
28	postalCodeLookup		JSON					
29	postalCodeSearch	XML	JSON					
30	rssToGeo						RSS	KML
31	search	XML	JSON	RDF				
32	siblings	XML	JSON					
33	srtm3	XML	JSON			TXT		
34	timezone	XML	JSON					
35	weather	XML	JSON					
36	weatherIcao	XML	JSON					
37	wikipediaBoundingBox	XML	JSON					
38	wikipediaSearch	XML	JSON					
	Total	36	36	1	1	4	2	1

This webservice is only available for the US.

Figura 3.13: Serviços *Web* Geonames[2]

3.2.5 OpenWeatherMap

OpenWeatherMap é um serviço que disponibiliza informação meteorológica atual, histórica ou previsões provenientes de mais de 40 000 estações meteorológicas [43]. A API permite o acesso aos dados do serviço através de:

- Coordenadas geodésicas;
- Nome da cidade;
- Identificação da cidade.

As respostas podem ser apresentadas em dois formatos: JSON/XML ou HTML. A API pode ser obtida através do seguinte processo:

- Efetuar o registo em `openweathermap.org/register`;
- Obter a API no menu `My Home->Setup` (Figura 3.14).

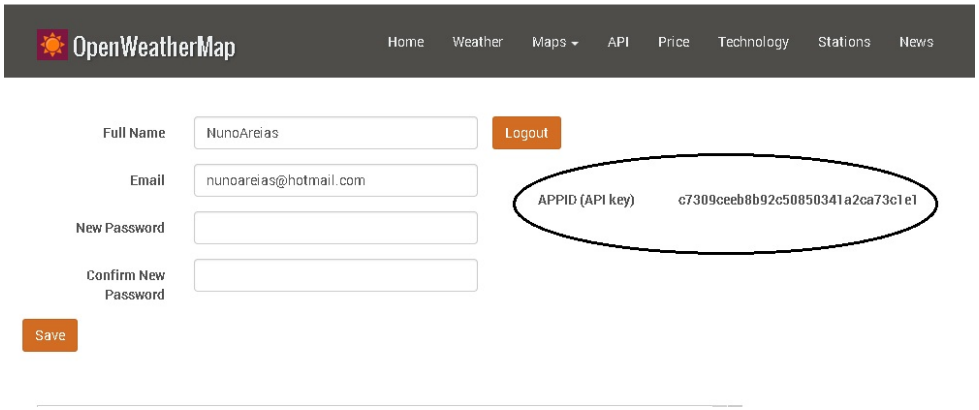

Figura 3.14: Chave da OpenWeatherMap API

3.3 Armazenamento Persistente de Dados

O sistema desenvolvido necessita de armazenar dados de forma persistente, seja para guardar informação dos utilizadores, seja dos locais/vídeos que foram carregados. Esta necessidade obriga à utilização de um Sistema de Gestão de Base de Dados (SGBD), *i.e.*, é uma ferramenta que permite criar e manter bases de dados de dimensões significativas, partilhar e aceder de forma eficiente dados entre várias aplicações e/ou utilizadores, garantindo a integridade e segurança destes [44, 45]. O sistema utiliza dois sistemas de gestão de bases de dados distintos: um sistema local para armazenar informação relevante de cada utilizador (SQLite) e um sistema central de armazenamento de informação relevante para todos os utilizadores (MySQL).

3.3.1 SQLite

O SQLite é um sistema de Gestão de Base de Dados transacional *serverless* autónomo que não tem qualquer necessidade de configuração. Este sistema é constituído por uma base de dados *Structured Query Language* (SQL) incorporada. Ao contrário de sistemas semelhantes, não inclui um módulo servidor separado e a escrita e leitura de informação é feita diretamente em ficheiros.

Cada um desses ficheiros incorpora os componentes de uma base de dados SQL, incluindo tabelas, índices, *triggers*, *etc.*

O seu código fonte é de livre utilização e o formato utilizado permite que o sistema seja multiplataforma, incluindo sistemas Android [46].

3.3.2 MySQL

O MySQL é o SGBD relacional mais popular e disseminado. Uma base de dados relacional é uma base de dados constituída por múltiplas tabelas que podem ou não estabelecer relações entre si. Uma base de dados com esta característica facilita a gestão assim como otimiza a velocidade de escrita e acesso aos seus dados. A sua arquitetura assenta num modelo cliente-servidor *multithreaded*, ou seja o servidor pode atender em simultâneo múltiplos clientes [47]. Este SGBD adota em termos de linguagem de interrogação a SQL e o seu *software* é *open source*, estando as condições de utilização definidas na GNU *General Public Licence* (GPL) [48].

3.3.3 PHPMyAdmin

A interface e gestão do SGBD MySQL pode ser efetuada através de uma consola ou através de interfaces gráficas entre as quais se destacam o PHPMyAdmin e o MySQL Workbench. Neste projeto adotou-se a aplicação PHPMyAdmin por ser uma aplicação mais leve dadas as limitações de *hardware* do computador utilizado. Esta é uma aplicação *Web* de utilização livre que permite a gestão das bases de dados MySQL graficamente e através de comandos SQL [49]. A Figura 3.15 representa a interface do PHPMyAdmin.

3.4 Ambientes Integrados de Desenvolvimento

Um ambiente de desenvolvimento integrado (IDE) é um conjunto de ferramentas de desenvolvimento de aplicações que incluem a edição, organização, compilação e depuração de código. O suporte ao desenvolvimento em linguagem Java é atualmente dominado por 3 plataformas [50]: Eclipse, Intellij IDEA e Netbeans. Na Tabela 3.2 são apresentadas as principais características de cada uma destas plataformas [51, 52, 53]:

Da análise da Tabela 3.2 conclui-se que os IDE Eclipse e Intellij IDEA são os que possuem um UI mais simples e intuitivo e o conjunto de ferramentas de depuração, filtragem, refatorização e assistência de código mais completo. O fator decisivo na escolha do IDE recaiu no facto do Eclipse ser um IDE totalmente gratuito, enquanto que o Intellij IDEA só é gratuito na versão comunidade onde muitas das funcionalidades chave não estão disponíveis.

[3] `http://sourceforge.net/projects/phpmyadmin`

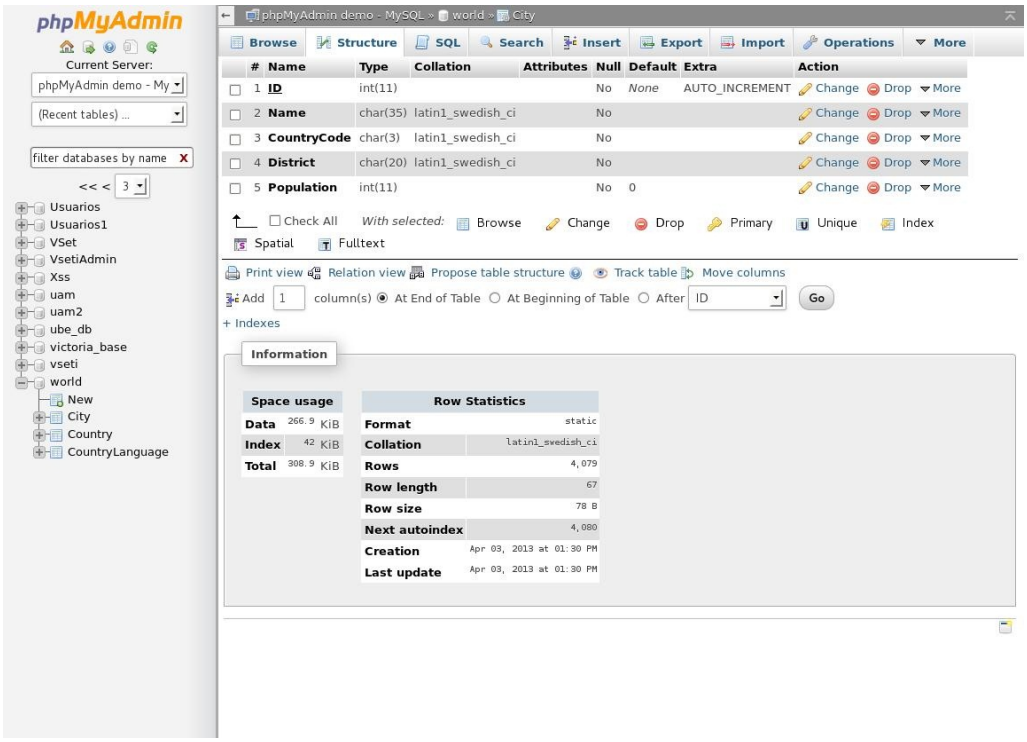

Figura 3.15: PHPMyAdmin UI[3]

3.4.1 Android Development Tools

Apesar do IDE Eclipse já incluir as ferramentas necessárias para o desenvolvimento de aplicações em Java é necessário instalar um conjunto de *plugins* (*Android Development Tools* (ADT)) de modo a acrescentar o seguinte conjunto de funcionalidades ao IDE [54]:

- Criação e gestão de projetos Android;
- Acesso a componentes do Android *framework*;
- Depuração, teste e desenvolvimento de aplicações Android.

Através do menu `Help->Install New Software` é possível iniciar o processo de instalação do ADT (Figura 3.16).

Após a instalação do ADT, o utilizador pode instalar ou utilizar um Android SDK já existente.

3.4.2 Software Development Kit

O Android SDK é composto não só pelas bibliotecas de Java necessárias ao desenvolvimento de aplicações Android, como também pelo empacotador de

Tabela 3.2: Comparação de IDE

IDE	**Funcionalidades**
Eclipse	• *Software open source*; • Suporte a múltiplos idiomas; • UI intuitivo; • Diversidade de *plugins* disponíveis; • Organização por perspetivas; • Ferramentas de filtragem, depuração e refatorização; • Integra de Ant e JUnit.
Intellij IDEA	• *Software* proprietário; • Suporte a J2EE; • Integra Ant, JUnit e *Concurrent Version System* (CVS); • UI intuitivo; • Ferramentas de depuração, assistência e automação de código; • Diversidade de *plugins* disponíveis.
Netbeans	• *Software open source*; • Multiplataforma; • Interação com bases de dados simples e intuitiva; • Comandos de execução e depuração flexíveis; • Integra o servidor de aplicações Tomcat.

aplicações, depurador, emulador e documentação [55]. Concluído o processo de instalação do SDK, resta instalar as API da plataforma que se pretenda utilizar através do Android SDK Manager (acessível em `Window->SDK Manager`). A Figura 3.17 apresenta o Android SDK Manager que inclui uma lista de ferramentas, API e *add-ons* que podem ser instalados no sistema.

3.4.3 Teste e Depuração de Aplicações

O teste das aplicações criadas pode ser efetuado quer num dispositivo físico (*smartphone*), quer num emulador. Para estabelecer a comunicação com disposi-

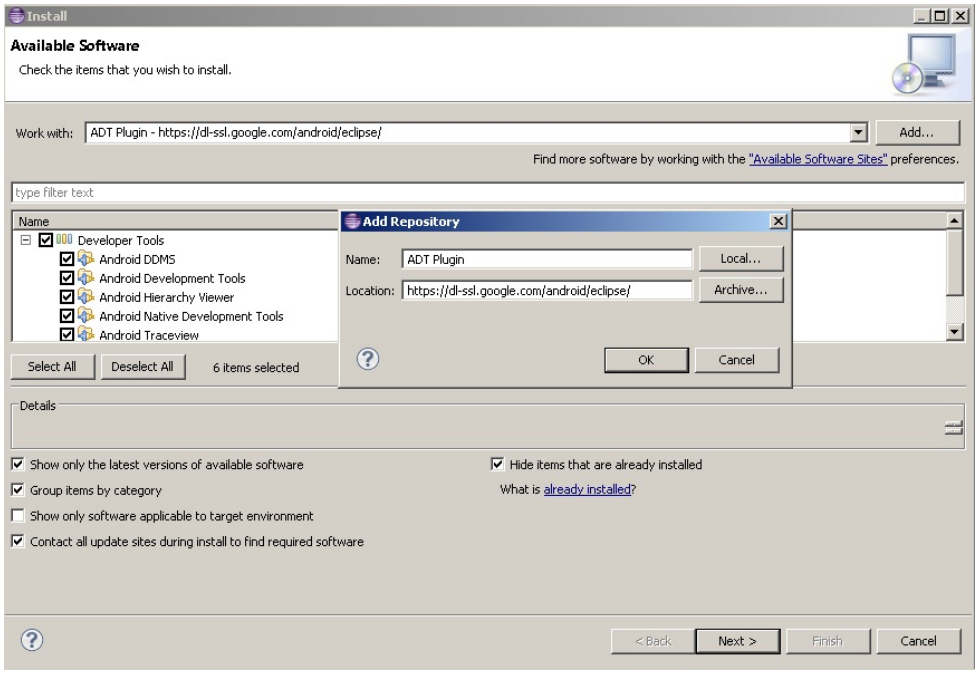

Figura 3.16: Instalação ADT

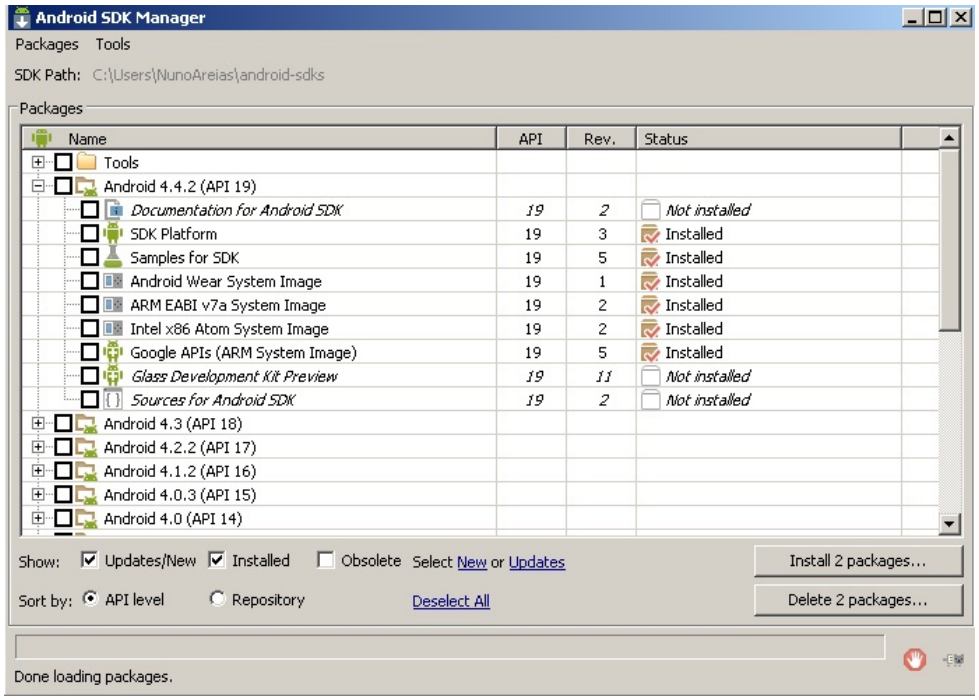

Figura 3.17: SDK Manager

tivos físicos é necessário recorrer a uma ferramenta *Android Debug Bridge* (ADB), que vem incluída no SDK. A ADB é baseada na arquitetura cliente-servidor e é composta por três componentes [56]:

- Um cliente executado no dispositivo de programação;
- Um servidor executado em segundo plano no dispositivo de programação, que estabelece a comunicação entre o cliente e o *daemon*;
- Um *daemon* executado em segundo plano no dispositivo de teste.

No caso de se optar por testar os programas num dispositivo Android físico, é necessário realizar um último passo no dispositivo. Em `Settings->Developer options` deve-se verificar se a opção `USB debugging` está ativada (Figura 3.18).

No caso do utilizador preferir utilizar um emulador para testar as suas aplicações, pode criar e gerir dispositivos virtuais através do *Android Virtual Device Manager* (AVDM) (disponível em `Window->Android Virtual Device Manager`). A Figura 3.19 mostra o UI desta ferramenta.

Para efetuar a depuração de aplicações Android, o Eclipse possui o `logcat`, que é uma ferramenta de apresentação do *output* de depuração. O `logcat` pode ser lançado através da consola do ADB. A Figura 3.20 [57] apresenta o respetivo UI.

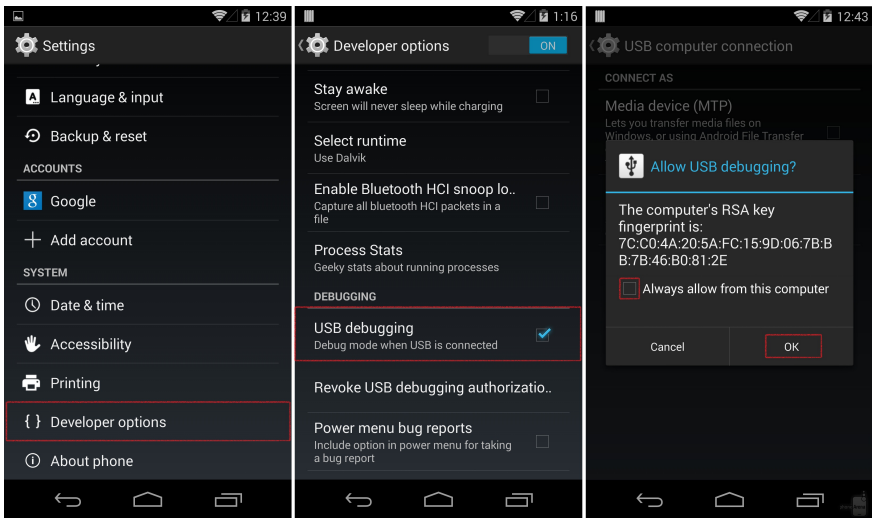

Figura 3.18: USB Debugging[4]

[4] `http://www.phonearena.com/news/How-to-enable-USB-debugging-on-Android_id53909`

[5] `http://blog.jetbrains.com/idea/2011/11/new-in-intellij-idea-11-improved-filtering-in-android-logcat/`

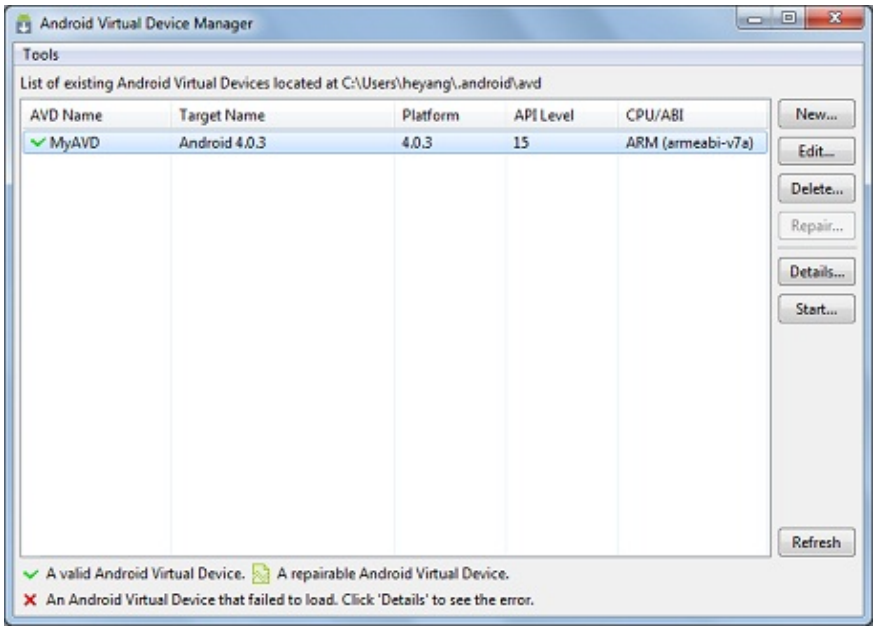

Figura 3.19: Android Virtual Device Manager

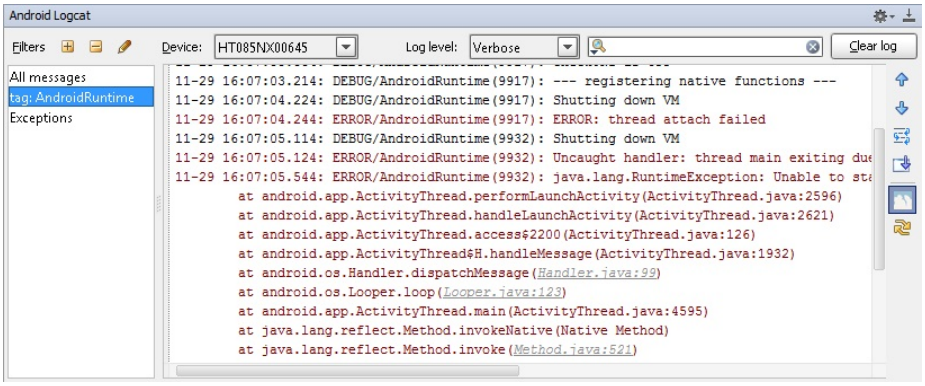

Figura 3.20: Logcat UI[5]

3.5 Sumário

A fim de permitir a replicação deste trabalho, este capítulo apresenta o conjunto das tecnologias, soluções e ferramentas utilizadas no desenvolvimento do protótipo. Inclui a descrição das características mais relevantes do sistema operativo Android, das bibliotecas de interface utilizadas (Android, Google Maps, YouTube, Geonames e OpenWeatherMap), das soluções de armazenamento persistente de dados e do ambiente de desenvolvimento selecionados.

No próximo capítulo detalha-se o sistema desenvolvido, incluindo a arquitetura, módulos constituintes e funcionalidades mais relevantes.

Capítulo 4

TourismShare

Neste capítulo descreve-se o sistema TourismShare de recomendação turística, destacando-se a arquitetura, módulos e respetivas funcionalidades. Em particular, apresentam-se os módulos de back-end, front-end, incluindo o acesso a serviços externos e serviços de armazenamento de dados.

4.1 Arquitetura

A arquitetura do sistema inclui os módulos de *front-end*, *back-end* e de interação com os serviços externos de suporte (Figura 4.1).

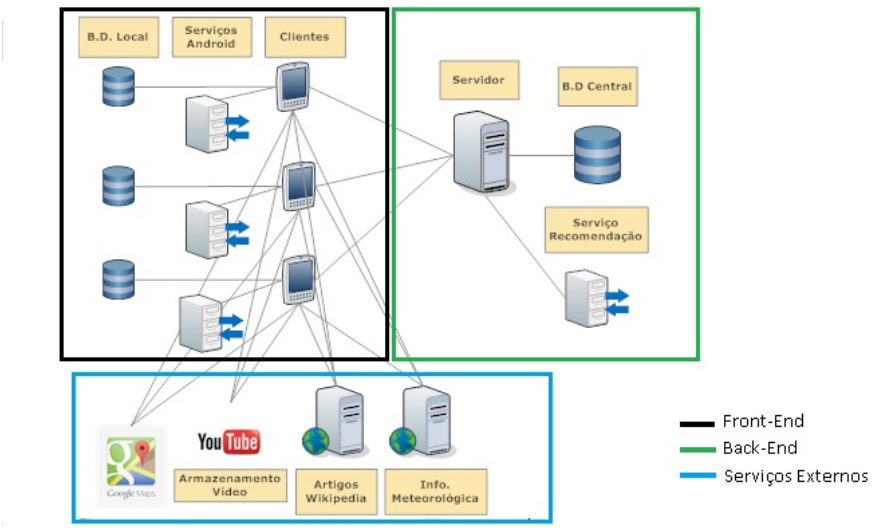

Figura 4.1: Arquitetura do sistema

O sistema é uma aplicação distribuída constituída por: (*i*) módulo cliente responsável pela interface com o utilizador (*front-end*); (*ii*) base de dados local SQLite associada ao módulo cliente, responsável por armazenar informação referente aos locais e mensagens pessoais do utilizador (*front-end*); (*iii*) serviços externos de suporte à visualização de mapas com a localização do utilizador e dos pontos de interesse ao seu redor, carregamento e visualização de vídeos, obtenção de artigos Wikipedia e de informação associada às condições meteorológicas aquando da gravação de vídeos; (*iv*) módulo servidor que gere o acesso dos clientes à informação da base de dados central e faz recomendação de itens aos clientes (*back-end*); (*v*) base de dados central MySQL que armazena toda a informação relevante para o modo de funcionamento partilhado do sistema (*back-end*).

4.2 Armazenamento Persistente de Dados

4.2.1 Base de Dados Central

A base de dados central armazena toda a informação necessária para o funcionamento da aplicação em modo partilhado e possui a estrutura apresentada na Figura 4.2.

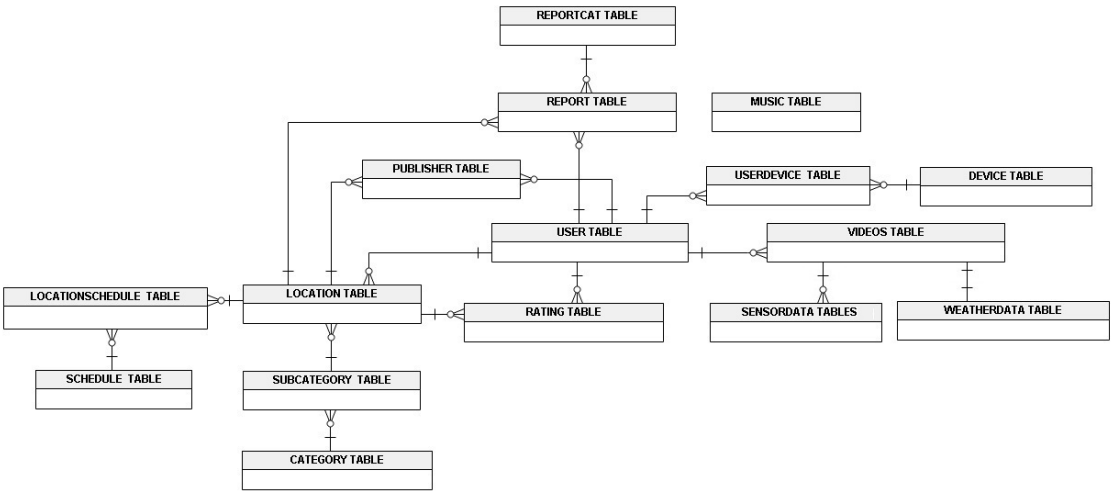

Figura 4.2: Estrutura da base de dados central

A Tabela `Location` armazena os locais partilhados de acordo com a estrutura da Tabela 4.1.

A Tabela `Subcategory` comporta as subcategorias a que podem pertencer os locais armazenados. A Tabela 4.2 apresenta a sua estrutura.

Tabela 4.1: Tabela Location

Coluna	**Função**
`location_id`	Chave primária auto-incremental do tipo inteiro
`location_name`	Nome do local
`location_desc`	Descrição do local
`location_lat`	Latitude do local (°)
`location_long`	Longitude do local (°)
`date_added`	Data de partilha do local
`location_rating`	Classificação média do local (1 a 5)
`subcategory_id`	Índice da subcategoria do local
`user_id`	Chave secundária da Tabela `Users` que identifica o publicador do local
`location_active`	Estado do local - ativo (1) ou inativo (0)

Tabela 4.2: Tabela Subcategory

Coluna	**Função**
`subcategory_id`	Chave primária auto-incremental do tipo inteiro
`subcategory_name`	Nome da subcategoria
`category_id`	Chave secundária da Tabela `Category` que relaciona a subcategoria com a respetiva categoria

A Tabela `Category`, que contém as diversas categorias a que os locais armazenados podem pertencer, apresenta a composição descrita na Tabela 4.3.

Tabela 4.3: Tabela Category

Coluna	**Função**
`category_id`	Chave primária auto-incremental do tipo inteiro
`category_name`	Nome da categoria

A Tabela `Schedule` armazena os horários de funcionamento dos diferentes locais do sistema. A Tabela 4.4 resume os seus campos.

De forma a relacionar os horários da tabela anterior com os locais, é necessário uma tabela intermediária – a Tabela `LocationSchedule`. A Tabela 4.5 detalha a sua estrutura.

Tabela 4.4: Tabela Schedule

Coluna	**Função**
`schedule_id`	Chave primária auto-incremental do tipo inteiro
`schedule_open`	Hora de abertura
`schedule_close`	Hora de fecho

Tabela 4.5: Tabela LocationSchedule

Coluna	**Função**
`locationschedule_id`	Chave primária auto-incremental do tipo inteiro
`locationschedule_day`	Horário de semana (1) ou horário de fim-de-semana (2)
`location_id`	Chave secundária da Tabela `Location` que relaciona o registo com um local
`schedule_id`	Chave secundária da Tabela `Schedule` que relaciona o registo com um horário

A Tabela `User` representa os utilizadores registados. A Tabela 4.6 especifica os seus campos.

Tabela 4.6: Tabela User

Coluna	**Função**
`user_id`	Chave primária auto-incremental do tipo inteiro
`user_name`	Nome do utilizador
`user_login`	Nome de autenticação do utilizador
`user_pw`	Senha de autenticação do utilizador
`user_email`	Endereço de *email* do utilizador
`user_gender`	Género do utilizador (M/F)
`user_country`	País de residência do utilizador
`publisher_reputation`	Reputação social do utilizador é função do número de locais que o utilizador partilhou e do número de reportes associados (0 a 1)
`av_rating_similarity`	Semelhança entre as classificações do utilizador e as classificações dos restantes utilizadores (0 a 1)
`user_regdate`	Data de registo do utilizador
`user_active`	Estado do utilizador - ativo (1) ou inativo (0)

As classificações atribuídas pelos utilizadores aos locais partilhados são armazenadas na Tabela `Rating`. A Tabela 4.7 apresenta a sua estrutura.

Tabela 4.7: Tabela Rating

Coluna	**Função**
`rating_id`	Chave primária auto-incremental do tipo inteiro
`rate`	Valor de classificação (1 a 5)
`rating_similarity`	Semelhança entre a classificação e a classificação média do local (0 a 1)
`rating_date`	Data da classificação
`location_id`	Chave secundária da Tabela `Location` que relaciona o registo com um local
`user_id`	Chave secundária da Tabela `User` que relaciona o registo com um classificador

Para evitar a múltipla definição dos locais, cada local pode ser partilhado por diversos utilizadores. Assim, é necessário associar a cada local os múltiplos utilizadores que o partilham, já que apenas o publicador inicial é representado na Tabela `Location`. No caso do publicador original deixar de partilhar o local, a gestão do local passa a ser responsabilidade do publicador mais antigo e, no caso de não existir nenhum outro publicador, o local é eliminado. A Tabela 4.8 representa esta informação.

Tabela 4.8: Tabela Publisher

Coluna	**Função**
`publisher_id`	Chave primária auto-incremental do tipo inteiro
`user_id`	Chave secundária da Tabela `User` que relaciona a publicação com um publicador
`location_id`	Chave secundária da Tabela `Location` que relaciona o registo com um local
`publisher_timestamp`	Data de modificação do registo que permite identificar qual o publicador responsável pelo local (registo mais antigo)
`publisher_active`	Estado da publicação - ativo (1) ou inativo (0)

Sempre que um utilizador encontre alguma incorreção num local publicado, pode reportar esse problema. A Tabela `Report`, que armazena estas mensagens, tem a estrutura apresentada na Tabela 4.9.

Na Tabela `ReportCat` contém as categorias dos erros da tabela anterior. A Tabela 4.10 representa a sua estrutura.

Tabela 4.9: Tabela Report

Coluna	Função
`report_id`	Chave primária auto-incremental do tipo inteiro
`report_from`	Chave secundária da Tabela `User` que relaciona o registo com o repórter do erro
`location_id`	Chave secundária da Tabela `Location` que relaciona o registo com um local
`reportcat_id`	Chave secundária da Tabela `ReportCat` que relaciona o registo com uma categoria do erro
`report_message`	Corpo da mensagem de erro
`report_date`	Data do reporte
`report_sent`	Permite identificar quais os registos que já foram apresentados ao publicador do local ao qual o erro corresponde (1) e os que ainda não foram apresentados (0)
`report_active`	Estado do reporte - ativo (1) ou inativo (0)

Tabela 4.10: Tabela ReportCat

Coluna	Função
`reportcat_id`	Chave primária auto-incremental do tipo inteiro
`reportcat_name`	Nome da categoria

A caracterização dos dispositivos utilizados pelos utilizadores é armazenada na Tabela `Device`. A Tabela 4.11 ilustra a estrutura desta tabela [58].

Tabela 4.11: Tabela Device

Coluna	Função
`device_id`	Chave primária auto-incremental do tipo inteiro
`device_brand`	Marca do dispositivo (*e.g.* motorola)
`device_model`	Modelo do dispositivo (*e.g.* XT1032)
`device_product`	Nome global do produto (*e.g.* falcon_vfgbsl)
`device_cpu`	Nome da unidade central de processamento do dispositivo (*e.g.* armeabi-v7a)
`screen_width`	Largura do ecrã (pixeis)
`screen_height`	Altura do ecrã (pixeis)
`api_version`	Versão da Android API instalada no dispositivo

De forma a associar cada dispositivos a utilizadores é necessário recorrer a uma tabela intermédia: a Tabela `UserDevice`. A Tabela 4.12 especifica a sua estrutura.

Tabela 4.12: Tabela UserDevice

Coluna	**Função**
`userdevice_id`	Chave primária auto-incremental do tipo inteiro
`user_id`	Chave secundária da Tabela `User` que associa o dispositivo a um utilizador
`device_id`	Chave secundária da Tabela `Device` que relaciona o dispositivo a um tipo de dispositivo

Na Tabela `Videos` é armazenada informação relativa aos vídeos partilhados no sistema. A Tabela 4.13 detalha a estrutura desta tabela.

Tabela 4.13: Tabela Videos

Coluna	**Função**
`video_id`	Chave primária auto-incremental do tipo inteiro
`video_name`	Nome do vídeo
`video_url`	URL de acesso ao vídeo
`video_date`	Data da partilha do vídeo
`video_lat`	Latitude final do vídeo (°)
`video_long`	Longitude final do vídeo (°)
`video_init`	Época de início de gravação do vídeo
`video_end`	Época de fim de gravação do vídeo
`video_views`	Número de reproduções do vídeo
`user_id`	Chave secundária da Tabela `User` que associa o vídeo ao seu publicador
`weatherdata_id`	Chave secundária da Tabela `WeatherData Table` que relaciona o vídeo com a sua informação meteorológica

A Tabela `SensorData` apresenta o conjunto das tabelas de armazenamento dos dados dos sensores. Estas tabelas são preenchidas automaticamente durante a gravação de cada vídeos partilhado. As várias medições, que são efetuadas de forma assíncrona durante a gravação de um vídeo, são armazenadas nas tabelas correspondentes em diferentes registos. Cada registo está associado ao vídeo correspondente e inclui a época da medição.

Estas onze tabelas possuem uma estrutura idêntica descrita na Tabela 4.14.

Tabela 4.14: Tabelas SensorData

Coluna	**Função**
`sensor_aceler`	Componentes da aceleração do dispositivo nos eixos X, Y e Z (m/s^2)
`sensor_gravidade`	Componentes da intensidade e direção da aceleração gravítica nos eixos X, Y e Z(m/s^2)
`sensor_giroscopio`	Componentes da velocidade angular do dispositivo nos eixos X, Y e Z (rad/s)
`sensor_linaceler`	Componentes da aceleração imprimida ao dispositivo, descontando a aceleração gravítica nos eixos X, Y e Z (m/s^2)
`sensor_vectorrotacao`	Representam a orientação do dispositivo como uma combinação de um ângulo e de um eixo, no qual o dispositivo tenha rodado através de um ângulo em torno de cada um dos eixos X, Y e Z
`sensor_campomagnetico`	Representa o campo magnético em torno do dispositivo nos eixos X, Y e Z (µT)
`sensor_proximidade`	Distância entre o dispositivo e um obstáculo (cm)
`sensor_luz`	Nível de luz ambiente (lx)
`sensor_pressao`	Pressão atmosférica (hPa)
`sensor_temperatura`	Temperatura ambiente (°C)
`sensor_humidade`	Humidade relativa do ambiente (%)

Tabela 4.15: Constituição de cada Tabela SensorData

Coluna	**Função**
`sensor_id`	Chave primária auto-incremental do tipo inteiro
`video_id`	Chave secundária da Tabela `Videos` que associa cada registo ao vídeo correspondente
`sensor_date`	Data do registo
`video_data`	Valor medido pelo sensor

Na Tabela `WeatherData` são guardados os dados meteorológicos obtidos durante a gravação dos vídeos. Como a informação meteorológica tem um período de atualização longo relativamente à duração de um vídeo, armazena-se apenas um registo por vídeo. A Tabela 4.16 resume o conteúdo desta Tabela [59].

Tabela 4.16: Tabela WeatherData

Coluna	**Função**
`weatherdata_id`	Chave primária auto-incremental do tipo inteiro
`weather_nuvens`	Nível de nebulosidade (%)
`weather_grausvento_url`	Direção do vento (º)
`weather_velvento`	Velocidade do vento (m/s)
`weather_pressao`	Pressão atmosférica (hPa)
`weather_humidade`	Humidade atmosférica (%)
`weather_temperatura`	Temperatura ambiente (K)
`weather_quantchuva`	Precipitação nas últimas 3 h (mm)
`weather_tempchuva`	Duração do período de precipitação (min)
`weather_quantneve`	Quantidade de neve nas últimas 3 h (mm)
`weather_tempneve`	Duração do período da queda de neve (min)
`weather_codigo`	Código do registo meteorológico
`weather_condicoesatm`	Condições atmosféricas
`weather_condicoesdesc`	Descrição das condições atmosféricas

Finalmente, na tabela `Music` são armazenados os dados relativos às músicas recomendadas ao utilizador no processo de imersão áudio. A Tabela 4.17 resume o conteúdo desta Tabela.

Tabela 4.17: Tabela Music

Coluna	**Função**
`music_id`	Chave primária auto-incremental do tipo inteiro
`music_motion`	Nível de atividade ao qual corresponde a música
`music_url`	URL de acesso à música

4.2.2 Base de Dados Local

A base de dados local é responsável pelo armazenamento de informação referente a locais e mensagens pessoais do utilizador. A Figura 4.3 apresenta a sua

estrutura.

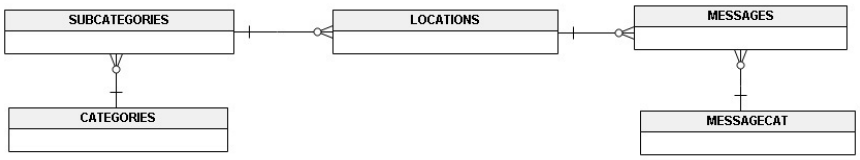

Figura 4.3: Estrutura da base de dados local

Na Tabela `locations` são armazenados os dados dos locais pessoais do utilizador. A Tabela 4.18 apresenta a estrutura de um local.

Tabela 4.18: Tabela locations

Coluna	**Função**
`location_id`	Chave primária auto-incremental do tipo inteiro
`location_name`	Nome do local
`location_desc`	Descrição do local
`location_lat`	Latitude do local (°)
`location_long`	Longitude do local (°)
`shared`	Campo booleano que indica se o local é partilhado (1) ou privado (0)
`shared_id`	Índice do local partilhado na base de dados central
`show_on_map`	Visibilidade do local - visível (1) ou invisível (0)
`active`	Estado do local - ativo (1) ou inativo (0)
`subcat_id`	Chave secundária da Tabela `subcategories` que identifica a subcategoria do local

Na Tabela `categories` são armazenadas as categorias dos locais pessoais. Esta Tabela é, em termos de composição, idêntica à Tabela 4.3 da base de dados central.

A Tabela `subcategories` é composta pelas subcategorias de cada uma das categorias da Tabela `categories`. Tem uma composição idêntica à da Tabela 4.2 da base de dados central.

Na Tabela `messages` são armazenadas as mensagens pessoais do utilizador. As mensagens são constituídas pelos *warnings* associados aos *reports* efetuados por terceiros relativos aos locais partilhados pelo utilizador. A Tabela 4.19 apresenta a estrutura desta informação.

Tabela 4.19: Tabela messages

Coluna	Função
`message_id`	Chave primária auto-incremental do tipo inteiro
`message_text`	Corpo da mensagem
`message_read`	Estado de leitura da mensagem - lida (1) ou por ler (0)
`message_date`	Data da mensagem
`message_active`	Estado da mensagem - ativa (1) ou inativa (0)
`messagecat_id`	Chave secundária da Tabela `messagecat` que identifica a categoria da mensagem
`location_id`	Chave secundária da Tabela `locations` que identifica o local partilhado reportado

Finalmente, na Tabela `messagecat` são armazenados os dados referentes às categorias das mensagens, permitindo ao utilizador identificar o tipo de *warning*. A sua composição é descrita na Tabela 4.20.

Tabela 4.20: Tabela messagecat

Coluna	Função
`messagecat_id`	Chave primária auto-incremental do tipo inteiro
`messagecat_name`	Nome da categoria

4.3 Servidor de *Back-End*

O servidor de *back-end* presta dois serviços: (*i*) atendimento dos clientes, *i.e.*, estabelecimento da comunicação e interação com a base de dados central; (*ii*) e recomendação de itens (locais, vídeos e músicas).

4.3.1 Atendimento de Clientes

A fim de estabelecer e manter em simultâneo ligações com múltiplos clientes, o servidor é composto por dois tipos de processos: o processo principal e o(s) processo(s) de atendimento. O fluxograma do processo principal é representado na Figura 4.4.

O processo principal cria, em primeiro lugar, um *logfile* onde serão armazenados todos os eventos relevantes que ocorram durante a execução do servidor. Este ficheiro terá como nome a data e hora de início de execução, facilitando a procura de qualquer evento. De seguida, o servidor estabelece uma ligação com

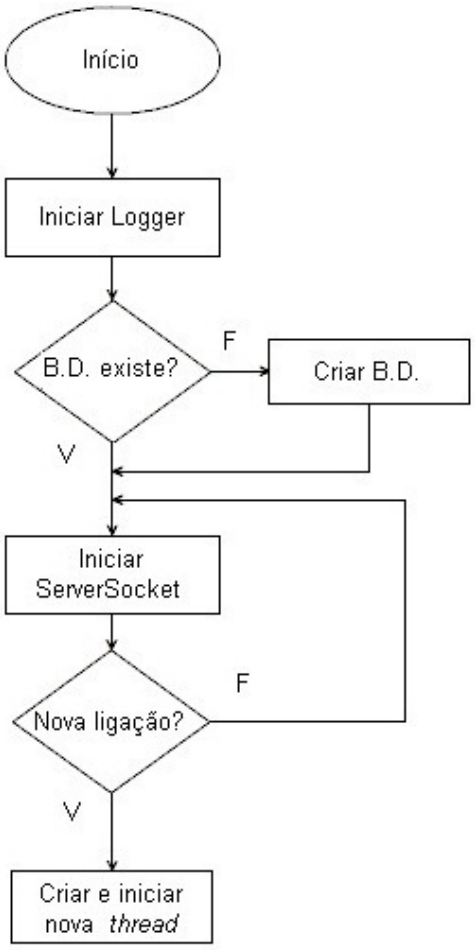

Figura 4.4: Fluxograma do processo principal do servidor

o servidor da base de dados central e verifica se a base de dados central existe (db_tourismshare). Caso não exista, cria-a de acordo com a estrutura descrita na Secção anterior. O passo seguinte consiste na criação de um `ServerSocket` para receber os pedidos de ligação dos clientes. Assim que chegar algum pedido, estabelece a ligação com o cliente, cria e lança em execução um processo dedicado de atendimento do cliente. O excerto de código 4.1 apresenta o processo de estabelecimento de novas ligações.

O processo de atendimento de cada cliente é responsável pela comunicação com o cliente, existindo tantos processos de atendimento quantos os clientes ligados. Este processo mantém-se em funcionamento até receber um pedido de terminação da ligação do cliente. O seu funcionamento é do tipo *message-driven*, *i.e.*, aguarda a receção de uma mensagem de protocolo para desencadear qualquer ação. O processamento de cada tipo de mensagem é distinto e termina com

Excerto de Código 4.1: Processo de estabelecimento de novas ligações

```
for (int i = 0; i < clients.length; i++) {
   //Waiting for clients
     try {
      clients[i] = new ClientThreadOnServerSide(server.accept(), this, i+1);
         clients[i].start();
         ++numberOfClients;
     } catch (IOException e) {
      System.exit(1);
     }
}
```

o envio da mensagem de resposta ao cliente e com a atualização do *logfile*. A Figura 4.5 apresenta o fluxograma de funcionamento deste processo. O protocolo

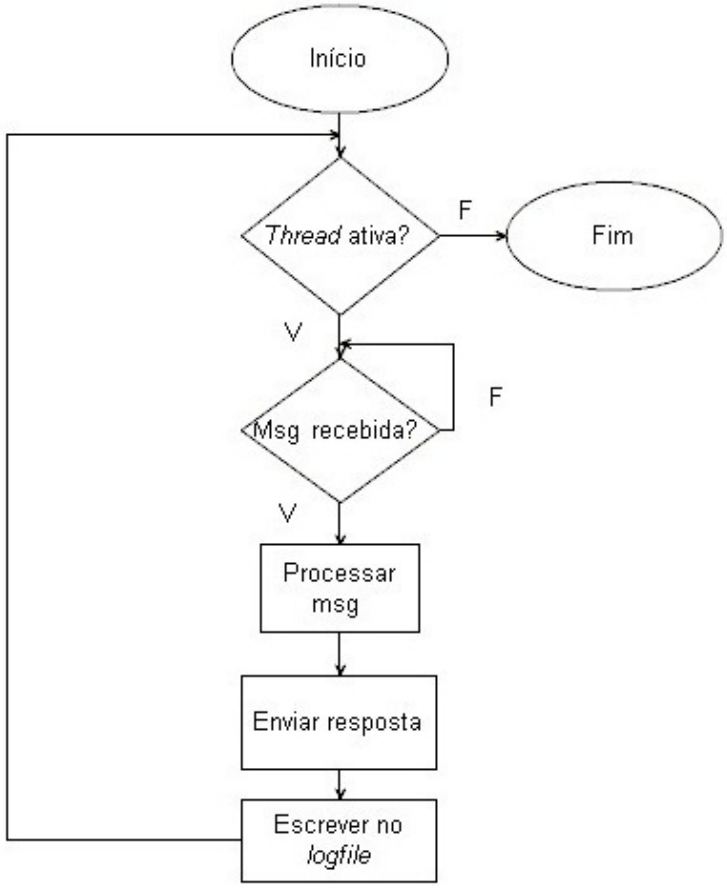

Figura 4.5: Fluxograma do processo de atendimento do servidor

é composto pelo seguinte conjunto de mensagens:

`MsgAddUser` enviada pelo cliente ao servidor quando um novo utilizador se pre-

tende registar no sistema. O processo de registo, depois de verificar que o *username* e *email* são únicos, armazena as características do dispositivo do utilizador. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgUpdateUser` enviada pelo cliente ao servidor quando um utilizador pretende atualizar os seus dados. Primeiro obtém os dados do seu perfil de utilizador e, de seguida, edita e envia o novo perfil. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgLogIn` enviada pelo cliente ao servidor quando um utilizador se pretende autenticar. O servidor verifica, para além das credenciais de acesso do utilizador, se o dispositivo do utilizador está armazenado na base de dados e, no caso deste não estar é adicionado. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgAddLocation` enviada pelo cliente ao servidor quando um utilizador pretende partilhar um local pessoal. Caso esse local pessoal seja novo, será inserido na base de dados. Caso contrário, o utilizador passa a ser um *publisher* do local. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgUpdateLocation` enviada pelo cliente ao servidor quando um utilizador pretende atualizar a informação de um dado local. Em primeiro lugar obtém a informação existente e, de seguida, edita e submete as alterações. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgDelLocation` enviada pelo cliente ao servidor quando um utilizador pretende terminar a partilha de um local. Implica a remoção do registo de partilha do utilizador e, no caso do utilizador ser o publicador original do local a responsabilidade de gestão passará a ser do publicador mais antigo. No caso de não existirem mais publicadores, o local é eliminado. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgAddRating` enviada pelo cliente ao servidor quando um utilizador classifica ou reclassifica um dado local. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgGetRating` enviada pelo cliente ao servidor quando um utilizador solicita a classificação (média e pessoal) de um dado local. O servidor responde reportando o sucesso ou insucesso da operação.

`MsgFileReport` enviada pelo cliente ao servidor quando um utilizador reporta alguma incorreção relativa a um dado local. O servidor responde reportando o sucesso ou insucesso da operação.

MsgGetSchedule enviada pelo cliente ao servidor quando um utilizador solicita o horário de funcionamento de um dado local. O servidor responde reportando o sucesso ou insucesso da operação.

MsgRecomm enviada pelo cliente ao servidor quando um utilizador ativa a recomendação automática de locais, artigos e vídeos. O servidor responde reportando o sucesso ou insucesso da operação.

MsgSearchLocations enviada pelo cliente ao servidor quando um utilizador solicita o envio de recomendações (a pedido). O servidor responde reportando o sucesso ou insucesso da operação.

MsgAddVideo enviada pelo cliente ao servidor quando um utilizador pretende submeter e partilhar um novo vídeo. O servidor responde reportando o sucesso ou insucesso da operação.

MsgAddView enviada pelo cliente ao servidor quando um utilizador visualizou um vídeo. O servidor responde reportando o sucesso ou insucesso da operação.

MsgMusic enviada pelo cliente ao servidor quando inicia o serviço de imersão áudio de modo a obter recomendação de músicas. O servidor responde reportando o sucesso ou insucesso da operação.

MsgEndConnection enviada pelo cliente ao servidor quando um utilizador pretende terminar a comunicação com o servidor. O servidor responde reportando o sucesso ou insucesso da operação.

A Figura 4.6 lista as mensagens definidas no protocolo.

4.3.2 Serviço de Recomendação

A recomendação personalizada de itens baseia-se na filtragem da informação disponível. O TourismShare disponibiliza os seguintes dados: (*i*) informação espacial (coordenadas geodésicas do utilizador); (*ii*) informação temporal (data e hora da pesquisa); (*iii*) nível de atividade do utilizador; e (*iv*) informação do dispositivo utilizado (marca, modelo, dimensões, *etc.*). O algoritmo de recomendação utilizado é constituído pelas quatro etapas ilustradas na Figura 4.7. Neste algoritmo, cada filtro é aplicado se e só se tiver à entrada cinco ou mais itens. Por outro lado, se à saída de algum filtro o número de itens for inferior a três, o serviço de recomendação utiliza os resultados do filtro anterior.

O pré-filtro é utilizado na recomendação de locais e vídeos. Colige todos os itens não publicados pelo utilizador ativo localizados num raio no máximo de 5 km de distância do local de pesquisa. Caso o utilizador tenha restringido a pesquisa a uma categoria ou subcategoria, o filtro aplica esta restrição e obtém os locais

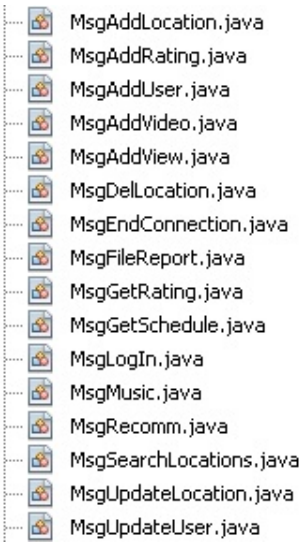

Figura 4.6: Lista de mensagens do protocolo

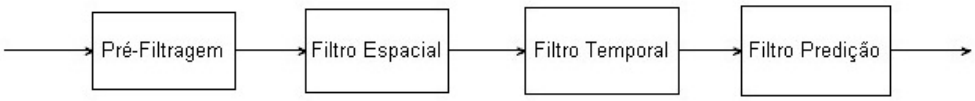

Figura 4.7: Diagrama do filtro de recomendação

da categoria ou subcategoria especificada. Finalmente, caso o utilizador tenha definido que pretende recomendações exclusivamente de locais novos, é feita uma filtragem adicional com este objetivo.

O filtro espacial é aplicado aos itens resultantes do filtro anterior, tendo por base o nível de atividade do utilizador e a distância entre cada item e o local de pesquisa do utilizador:

- Atividade nula: raio de pesquisa de 0,75 km;
- Atividade reduzida: raio de pesquisa de 1,50 km;
- Atividade média: raio de pesquisa de 2,00 km;
- Atividade alta: raio de pesquisa de 3,00 km;
- Atividade indefinida: raio de pesquisa de 0,75 km.

O terceiro filtro (filtro temporal) é aplicado para extrair todos os locais em funcionamento na data da pesquisa, *i.e.*, a data da pesquisa pertence ao período de funcionamento do local.

O último filtro consiste num algoritmo simples de predição da classificação que o utilizador atribuiria aos locais selecionados e tem por objetivo apresentar os cinco locais com maior classificação. O filtro contempla três casos: (*i*) o utilizador classificou o local; (*ii*) o utilizador não classificou o local, mas classificou locais da mesma categoria; e (*iii*) o utilizador não classificou o local nem locais da mesma categoria. Caso o utilizador tenha classificado o local, a predição da classificação do local é determinada através da Equação 4.1:

Equação 4.1: Predição baseada na classificação prévia do utilizador

$$Pred(l,c) = \beta Rating(u,l,c) + (1-\beta)Rep(p) \times \overline{Rating(l,c)} \tag{4.1}$$

onde $Rating(u,l,c)$ é a classificação que o utilizador u atribuiu ao local l da categoria c, $Rep(p)$ a reputação do *publisher* p do local l e $\overline{Rating(l,c)}$ é a classificação média do local. Como o utilizador classificou explicitamente o local, o filtro atualmente utiliza $\beta = 80$ %, *i.e.*, aplica um peso preponderante à classificação do utilizador. Caso a pesquisa não especifique uma categoria ou subcategoria, a Equação 4.1 não se restringe a locais de uma categoria ou subcategoria para evitar a sobre-especialização.

A reputação de um utilizador enquanto publicador de locais ou *publisher* é definida através da Equação 4.2:

Equação 4.2: Reputação do publicador p

$$Rep(p) = \frac{l_{pub} - l_{rep}}{l_{pub}} \tag{4.2}$$

onde l_{pub} é o número de locais publicados pelo utilizador p e l_{rep} o número de reportes ativos associados aos locais publicados pelo utilizador p. A reputação varia entre 0 % e 100 %.

A classificação média de um local é definida pela Equação 4.3:

Equação 4.3: Classificação média do local l de categoria c

$$\overline{Rating(l,c)} = \frac{1}{n}\sum_{u=1}^{n} Rating(u,l,c) \tag{4.3}$$

onde n é o número de utilizadores que classificaram o local l e $Rating(u,l,c)$ é a classificação que o utilizador u atribuiu ao local l da categoria c.

Caso o utilizador ainda não tenha classificado o local, mas já tenha classificado locais da mesma categoria, aplica-se a Equação 4.4:

Equação 4.4: Predição baseada na classificação de locais da mesma categoria

$$Pred(l,c) = \beta\overline{Rating(u,c)} + (1-\beta)Rep(p) \times \overline{Rating(l,c)} \tag{4.4}$$

onde $\overline{Rating(u,c)}$ é a classificação média que o utilizador u atribuiu aos locais da categoria c do local l, $Rep(p)$ a reputação do *publisher* p do local l e $\overline{Rating(l)}$ é a classificação média do local. Neste caso, o valor de β aplicado foi de 60 % à classificação média do utilizador aos locais da mesma categoria do local.

A classificação média de que um utilizador atribuiu a uma categoria obtém-se através da Equação 4.5:

Equação 4.5: Classificação média que o utilizador u atribuiu a locais da categoria c

$$\overline{Rating(u,c)} = \frac{1}{n}\sum_{l=1}^{n} Rating(u,l,c) \tag{4.5}$$

onde n é o número de locais da categoria c que o utilizador u classificou e $Rating(u,l,c)$ é a classificação que o utilizador u atribuiu ao local l da categoria c.

Finalmente, caso o utilizador o utilizador não tenha classificado o local nem locais da mesma categoria, aplica-se a Equação 4.6 :

Equação 4.6: Predição baseada na classificação prévia do local por terceiros

$$Pred(l,c) = \beta\overline{Rating(l,c)} + (1-\beta)Rep(p) \times \overline{Rating(l,c)} \tag{4.6}$$

onde $\overline{Rating(l,c)}$ é a classificação média do local l e $Rep(p)$ a reputação do *publisher* p do local. Neste caso, o valor de β aplicado à classificação média do local é 40 %. O peso relativo da reputação do publicador é superior (60 %) porque esta classificação não tem qualquer contribuição do utilizador ativo.

No caso da recomendação de vídeos é efetuada uma filtragem adicional com o objetivo de selecionar os vídeos populares. Este filtro determina o quociente entre número de visualizações e a idade da publicação do vídeo. Os novos vídeos partilhados, enquanto não tiverem visualizações, apresentam um valor mínimo igual a 0.3.

Finalmente, em relação à recomendação de músicas, a filtragem resume-se à seleção de músicas associadas aos diferentes níveis de atividade.

4.4 Interação com Serviços Externos

Como anteriormente foi referido, algumas das funcionalidades do módulo cliente são asseguradas através do recurso a serviços externos. Nesta Secção será apresentado o processo de interação com os diferentes serviços externos utilizados.

4.4.1 Google Maps

Este serviço disponibiliza um mapa que apresenta a posição do utilizador e dos vários pontos de interesse (sejam eles recomendações ou locais pessoais).

A renderização de um mapa em Android é efetuada através da classe `MapFragment` que, sendo uma subclasse da classe `Fragment`, permite embeber um mapa (classe `GoogleMap`) numa atividade onde são também representadas as posições do utilizador e de diversos POI [60, 61]. O Excerto de Código 4.2 apresenta o código do ficheiro XML de uma atividade, permitindo embeber um mapa na mesma.

Excerto de Código 4.2: Definição de MapFragment

```
<fragment
     android:id="@+id/map"
     android:name="com.google.android.gms.maps.MapFragment"
     android:layout_width="match_parent"
     android:layout_height="match_parent"/>
```

Os métodos de estabelecimento de ligação com o serviço Google Maps, carregamento, apresentação do mapa e interação com o mapa são assegurados pela classe `GoogleMap`. Esses métodos são descritos no Excerto de Código 4.3 [62].

Excerto de Código 4.3: Apresentação do mapa através da classe GoogleMap

```
googleMap = ((MapFragment) getFragmentManager().findFragmentById(R.id.map)).getMap();
googleMap.setMyLocationEnabled(true);
googleMap.setMapType(GoogleMap.MAP_TYPE_HYBRID);
googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 15));
```

O `fragment` com identificação `map` definido no ficheiro de *layout* é atribuído ao objeto `googleMap`. De seguida, especifica-se que se pretende apresentar a posição do utilizador num mapa do tipo híbrido centrado na posição do utilizador.

4.4.2 GeoNames

A obtenção de artigos da Wikipedia relativos à área vizinha do utilizador recorre à classe `WebService` e à classe `WikipediaArticle` contidas no ficheiro `geonames-1.1.12.jar`. Enquanto a classe `WebService` contém os métodos necessários para a autenticação e obtenção de artigos, a classe `WikipediaArticle`

define a estrutura de cada um dos artigos, permitindo que estes sejam armazenados numa lista de objetos. O Excerto de Código 4.4 resume a interação com este serviço.

Excerto de Código 4.4: Interação com o serviço GeoNames

```
WebService.setUserName("nunoareias");
try {
   wikiResults = WebService.findNearbyWikipedia(latitude, longitude, radius, "en", 40);
} catch (Exception e) {
   System.out.println("Error:" + e);
}
```

Em primeiro lugar é feita a autenticação, utilizando a conta criada para o efeito. De seguida, especifica-se o pedido de um máximo de 40 artigos em Inglês associados à região centrada na posição do utilizador e de raio máximo especificado. A Figura 4.8 apresenta a lista de atributos que compõe um artigo.

Attribute	Description
lang	ISO language code of article text
title	the article title
summary	a short summary of the article text. Around 300 chars. The text is truncated at a full stop if one is available near char 300, otherwise at the end of a word.
feature	the wikipedia feature type. A list of types is available here
countryCode	the ISO country code of the article
elevation	the elevation in metres (optional may be null), parsed from the article or reverse geocoded.
population	the population (optional may be null)
lat	latitude
lng	longitude
wikipediaUrl	URL of the article
thumbnailImg	URL of a small thumbnail image (ca 100x75 px)
rank	indication of the popularity or relevancy of an article. The rank is an integer number from 1 for the least popular articles to 100 for the most popular articles. It is calculated from the number of links pointing to an article and the article length. The articles are more or less evenly distributed over the 100 ranks.

Figura 4.8: Lista de atributos de um WikipediaArticle

4.4.3 OpenWeatherMap

Este serviço retorna a informação meteorológica do local de gravação de um vídeo e é acedido durante o processo de gravação. Para obter e processar a informação meteorológica utilizaram-se duas classes: `WeatherHTTPClient` e `JSONWeatherParser`. A primeira classe estabelece a ligação com o serviço externo através de um cliente HTTP como ilustra o excerto de código 4.5.

Em primeiro lugar constrói-se o URL da ligação HTTP, utilizando o URL definido na `String BASE_URL` concatenado com as coordenadas atuais do utili-

Excerto de Código 4.5: Interação com o serviço OpenWeatherMap

```
String BASE_URL = "http://api.openweathermap.org/data/2.5/weather?"
//Estabelecer ligacao
con = (HttpURLConnection) (new URL(BASE_URL + location)).openConnection();
con.setRequestMethod("GET");
con.setDoInput(true);
con.setDoOutput(true);
con.connect();

// Ler resposta
is = con.getInputStream();
BufferedReader br = new BufferedReader(new InputStreamReader(is));
```

zador. De seguida, configuram-se os parâmetros da ligação, incluindo o método do pedido (GET), e estabelece-se a ligação. Finalmente, cria-se e transforma-se o canal de leitura `InputStream` num objeto da classe `BufferedReader`.

A classe `JSONWeatherParser` é responsável pelo processamento da resposta JSON obtida e o Excerto de Código 4.6 resume o seu funcionamento.

Excerto de Código 4.6: Processamento da resposta do serviço OpenWeatherMap

```
//Criar objeto weather
Weather weather = new Weather();

//Criar objeto JSON
JSONObject jObj = new JSONObject(data);

//Aceder a informacao
JSONArray jArr = jObj.getJSONArray("weather");
JSONObject mainObj = getObject("main", jObj);

//Inserir informacao num objeto weather
weather.currentCondition.setHumidity(getInt("humidity", mainObj));
weather.currentCondition.setPressure(getInt("pressure", mainObj));
...
```

Cria-se um objeto JSON para comportar a resposta do serviço, escolhe-se a informação desejada (*weather*) e acede-se ao objeto `main`. Finalmente, coloca-se num objeto da classe `Weather` a informação correspondente, *e.g.*, os valores da humidade e pressão.

4.4.4 YouTube

Dado que os vídeos partilhados são armazenados no YouTube, é necessário interagir com esta plataforma. O módulo de captura e partilha de vídeos utiliza o projeto *open source* `YouTube Direct Lite` [2] e requer a instalação das seguintes bibliotecas:

- `google-api-client-1.15.0-rc.jar`;
- `google-api-client-android-1.15.0-rc.jar`;

- `google-api-services-youtube-v3-rev55-1.15.0-rc.jar`;
- `google-http-client-1.15.0-rc.jar`;
- `google-http-client-android-1.15.0-rc.jar`;
- `google-http-client-gson-1.15.0-rc.jar`;
- `youtubeandroidplayerapi.jar`;
- `guava-jdk5-13.0.jar`.

Finalmente, é necessário adicionar todas as permissões ao ficheiro `AndroidManifest.xml` como se ilustra no Excerto de Codigo 4.7.

Excerto de Código 4.7: Permissões de acesso ao YouTube Direct Lite

```
//Permissoes necessarias para funcionamento do YouTube Direct Lite
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.GET_ACCOUNTS" />
    <uses-permission android:name="android.permission.NETWORK" />
    <uses-permission android:name="android.permission.USE_CREDENTIALS" />
    <uses-permission android:name="android.permission.INTERNET" />
```

4.5 Aplicação *Front-End*

A aplicação cliente disponibiliza a interface ao utilizador. Em particular, a aplicação é constituída por um conjunto de atividades de interface com o utilizador e três serviços que obtêm informação complementar de forma autónoma. A interface suporta seis línguas (Português, Inglês, Francês, Italiano, Alemão, Espanhol) e tem traduções completas em Português e Inglês. Por omissão, a aplicação utiliza a língua definida no sistema Android (Inglês).

4.5.1 Mapa da Navegação

A Figura 4.9 apresenta o mapa de navegação da aplicação cliente. Quando se arranca a aplicação surge um *splashscreen* composto pelo nome da aplicação (TourismShare), ícone e uma barra de progresso. Esta atividade verifica se a base de dados local existe, criando-a no caso de não existir, e obtém as características do dispositivo utilizado. A informação do dispositivo será enviada para o módulo servidor durante o processo de recomendação. Após 5 s, o utilizador é encaminhado para o menu principal. O menu principal depende do tipo de utilizador. A Tabela 4.21 resume os dois casos.

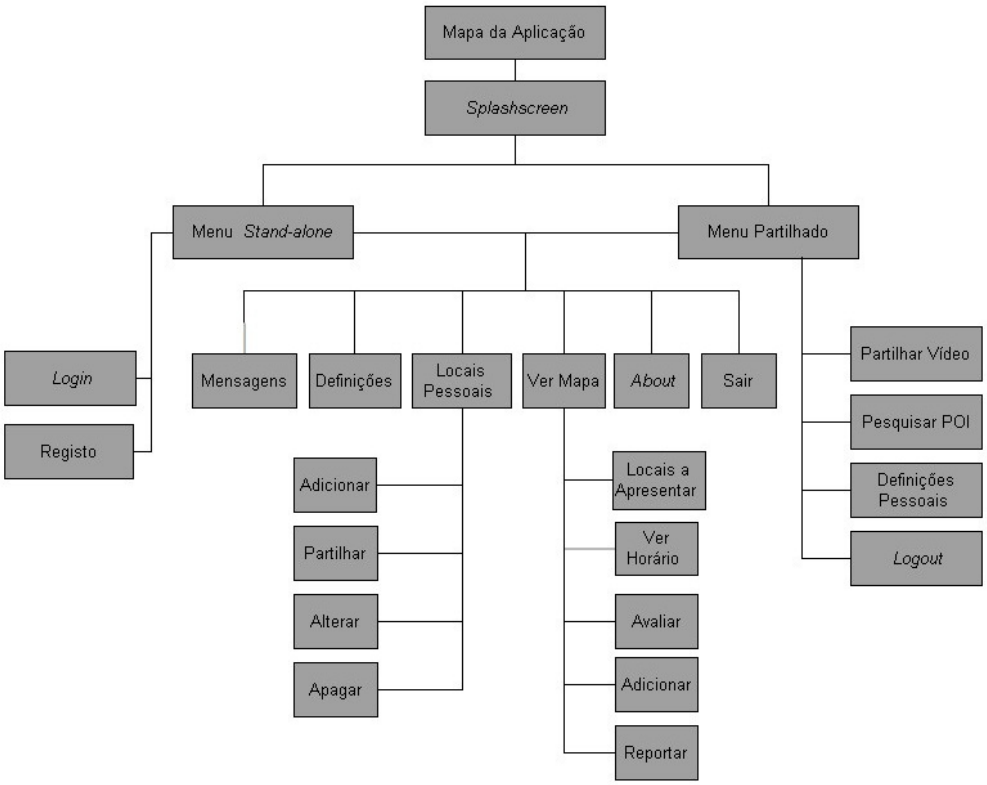

Figura 4.9: Mapa de navegação

Tabela 4.21: Tipos de utilizador

Utilizador	**Estado**	**Menu**
Autenticado	Utilizador registado e autenticado no sistema	Partilhado
Convidado	Utilizador não autenticado ou registado no sistema	*Stand-alone*

4.5.1.1 Menu *Stand-alone*

Este menu é apresentado aos utilizadores não autenticados e oferece as seguintes funcionalidades:

- *Login* - apresenta o ecrã de autenticação do utilizador composta por dois campos (nome de utilizador e palavra-chave);
- Registo - apresenta o ecrã de registo do utilizador que inclui os campos nome, nome de utilizador, palavra-chave, repetição da palavra-chave, *email*, género e nacionalidade;

- Mensagens - apresenta a caixa de entrada de mensagens do utilizador geradas pelo sistema de reportes de incorreções dos locais partilhados pelo utilizador;

- Definições - apresenta e permite alterar a configuração atual da aplicação, *e.g.*, ligar/desligar efeitos sonoros, a imersão áudio ou o modo de recomendação automática;

- *About* - reporta informação sobre a aplicação, incluindo versão e línguas disponíveis;

- Locais Pessoais - apresenta a lista de locais pessoais do utilizador (nomes, categorias e estado de partilha) e permite: (*i*) adicionar um novo local, *i.e.*, especificar o nome, descrição, categoria e subcategoria e localização (usando o sensor de posição do dispositivo ou um mapa clicável); (*ii*) alterar um local não partilhado (locais partilhados só podem ser alterados após autenticação); (*iii*) mostrar um local no mapa; e (*iv*) apagar um local não partilhado;

- Ver Mapa - apresenta o mapa da vizinhança do utilizador, incluindo a posição atual, os locais pessoais e os pontos de interesse que tenham sido recomendados recentemente, e permite: (*i*) modificar a lista de locais pessoais a apresentar no mapa; e (*ii*) adicionar um novo local à lista de locais pessoais;

- Sair - abandona a aplicação.

De referir que todos os formulários têm um sistema de validação para impedir a inserção de dados com carateres ou dimensões erradas.

4.5.1.2 Menu Partilhado

Este menu, que é apresentado aos utilizadores autenticados, oferece, para além das funcionalidades do menu *stand-alone*, as seguintes opções:

- Locais Pessoais - permite: (*i*) alterar um local originalmente partilhado pelo utilizador; (*ii*) partilhar um local pessoal[1]; e (*iii*) terminar a partilha de um local (deixar de ser um dos publicadores do local) e, caso seja o único publicador, apagar o local da base de dados central.

[1] O processo de partilha apresenta a lista dos locais partilhados situados a 1 km de distância do local a partilhar para evitar duplicados. Caso se trate de um local novo, o utilizador tem de definir o horário de funcionamento e efetuar a primeira classificação do local.

- Ver Mapa - permite: (*i*) consultar o horário de funcionamento de um local (horário de semana e de fim de semana); (*ii*) classificar ou reclassificar um local; e (*iii*) reportar problemas identificados num local partilhado por terceiros (horário errado, local inexistente, *etc.*);
- Partilhar Vídeo - permite gravar, incluindo a recolha de informação contextual via sensores físicos do dispositivo e serviço OpenWeatherMap, e partilhar um vídeo mediante à autenticação na conta Google e a atribuição de um nome ao vídeo;
- Pesquisar POI - permite solicitar recomendações em função do local (do utilizador ou definido através de um mapa clicável), categoria e tipo de item (locais, artigos e/ou vídeos);
- Definição Pessoais - permite alterar a informação de registo do utilizador mediante a introdução da palavra-chave;
- *Logout* - permite terminar a sessão de um utilizador autenticado.

4.5.2 Serviços Android

A aplicação cliente inclui três serviços Android independentes da interação do utilizador: (*i*) MyPVTService é responsável pela obtenção da posição, velocidade e tempo (PVT) do utilizador; (*ii*) o RecommendService é responsável pela recomendação automática de locais, vídeos e artigos; e (*iii*) o MusicService é responsável pela imersão áudio do utilizador. Estes serviços são iniciados no arranque da aplicação e mantém-se ativos até ao encerramento da aplicação ou até serem desativados nas definições (RecommendService e MusicService).

4.5.2.1 MyPVTService

O MyPVTService é serviço automático de determinação da PVT do utilizador essencial a todas as funcionalidades da aplicação. A Figura 4.10 contém o fluxograma de funcionamento do serviço.

O Excerto de Código 4.8 ilustra a fase de arranque do serviço, incluindo a especificação dos diferentes provedores de PVT (sensor de GPS e provedor de rede).

Trata-se de um serviço guiado por eventos, que, após a iniciação, fica a aguardar atualizações. Quando surge uma atualização, verifica qual a fonte e o respetivo período de refrescamento. O provedor de rede apresenta um período de refrescamento de 90 s, enquanto que o sensor GPS de apenas 30 s. Esta disparidade dá primazia às atualizações provenientes do sensor GPS porque apresentam maior exatidão. Independentemente da fonte (sensor GPS ou rede), caso o período de refrescamento tinha sido ultrapassado (30 s ou 90 s), o serviço armazena

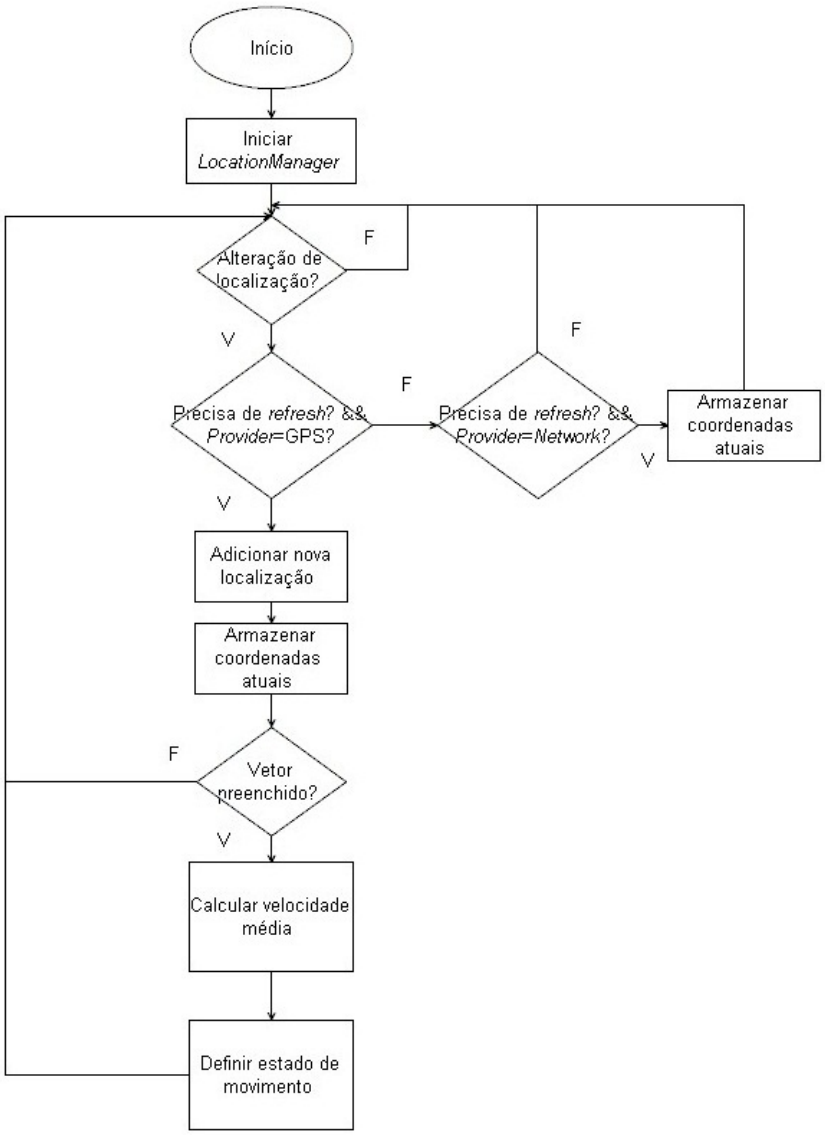

Figura 4.10: Fluxograma do MyPVTService

Excerto de Código 4.8: Iniciação de LocationManager

```
//Obtem uma referencia para o LocationManager do sistema
locationManager = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);

// Cria um listener que responde a atualizacoes de localizacao
locationListener = new LocationListener() {
   ...
}
//Atualizacoes para os dois provedores (Network e GPS)
locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, refreshTime, 0,
     locationListener);
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, refreshTime, 0,
     locationListener);
```

a nova informação. A velocidade é armazenada num *buffer* do tipo *First In, First Out* (FIFO) de dimensão 10 que permite a determinação do nível de atividade do utilizador. O nível de atividade atual do utilizador é função da sua velocidade média - ver Tabela 4.22.

Tabela 4.22: Níveis de atividade em função da velocidade do utilizador

Velocidade (m/s)	**Atividade**
[0,0;0,3[	Nula
[0,3;2,5[	Reduzida (*e.g.* a pé)
[2,5;7,0[	Média (*e.g.* de bicicleta)
[7,0;∞[	Elevada (*e.g.* de carro)

Caso o período entre atualizações seja muito elevado (> 20 min), não é possível definir o nível de atividade do utilizador (nível de atividade indefinido).

4.5.2.2 RecommendService

O RecommendService é o serviço automático de recomendação de locais. A Figura 4.11 apresenta o respetivo fluxograma.

Este serviço tem um ciclo de funcionamento com um período de 5 min. Caso o utilizador esteja autenticado, tenha a recomendação automática ativada e esteja ligado à Internet, o serviço calcula a distância entre a posição atual do utilizador e a posição registada no ciclo anterior. O algoritmo de recomendação executa apenas quando esta distância for superior a 300 m, evitando a execução redundante do algoritmo quando a localização do utilizador varia pouco. Sempre que são geradas recomendações, o sistema verifica se existem itens novos e, em caso afirmativo, notifica o utilizador.

4.5.2.3 MusicService

O MusicService é o serviço responsável pela imersão áudio. A Figura 4.12 apresenta o respetivo fluxograma.

Caso o utilizador esteja autenticado e tenha acesso à Internet, a aplicação solicita o envio das listas musicais ao servidor. Se o nível de atividade do utilizador for desconhecido, o serviço verifica periodicamente (cada 3 min) se o nível de atividade se alterou. Uma vez conhecido o nível de atividade, o serviço inicia a reprodução da lista correspondente. Existem 4 listas musicais associadas aos respetivos níveis de atividade (parado, baixo, médio, alto). Uma vez iniciada a imersão áudio, no final de cada música é verificado o nível de atividade e, em função do resultado, é selecionada uma música da lista correspondente.

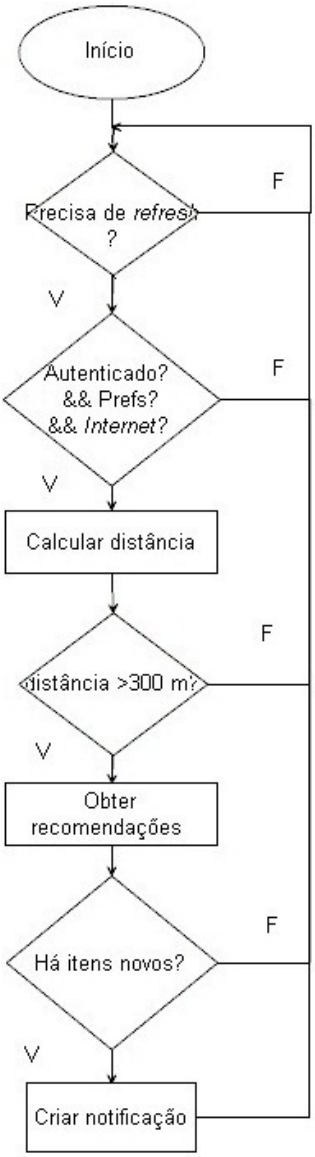

Figura 4.11: Fluxograma do RecommendService

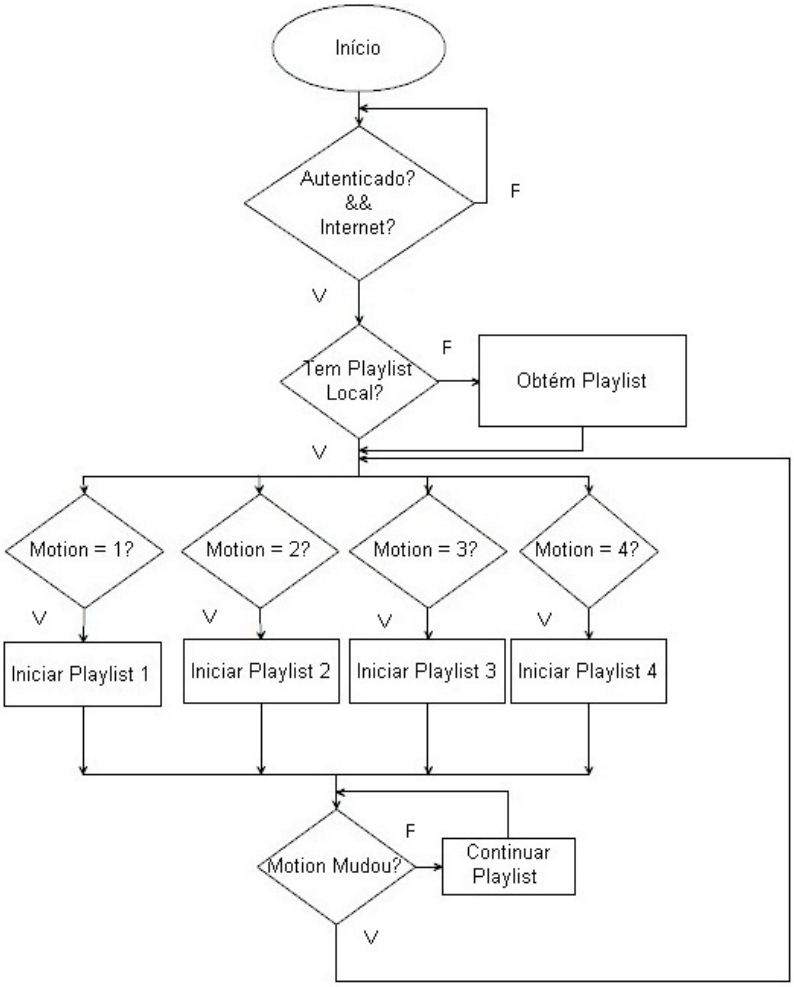

Figura 4.12: Fluxograma do MusicService

4.6 Sumário

Este capítulo descreve o sistema desenvolvido, incluindo a arquitetura, os seus componentes e funcionalidades.

No capítulo seguinte apresentam-se os testes realizados e os resultados obtidos.

Capítulo 5

Testes e Resultados

Este capítulo descreve os testes funcionais realizados e apresenta os resultados.

5.1 Testes de Funcionalidades

Os testes foram realizados com o dispositivo físico Motorola Moto G (Figura 5.1). Este dispositivo tem instalada a distribuição Android Kitkat, *i.e.*, utiliza a API Android 19. Tal como foi referido no Capítulo 3, a aplicação foi desenvolvida tendo em conta a compatibilidade com a API 16 e superiores, abrangendo um grande número de dispositivos Android.

Figura 5.1: Dispositivo utilizado[1]

[1] `http://www.phonearena.com/phones/Motorola-Moto-G_id8208`

5.1.1 Menu Principal da Aplicação

Quando a aplicação arranca, apresenta a página inicial (*splashscreen*), verifica a existência ou não da base de dados local e obtém informação relativa ao dispositivo. Passados 5 s, o utilizador é redirecionado para o menu principal: menu *stand-alone*, caso o utilizador não esteja autenticado, ou menu partilhado, no caso contrário. A Figura 5.2 apresenta o *splashscreen*, o menu *stand-alone* e, finalmente, o menu partilhado.

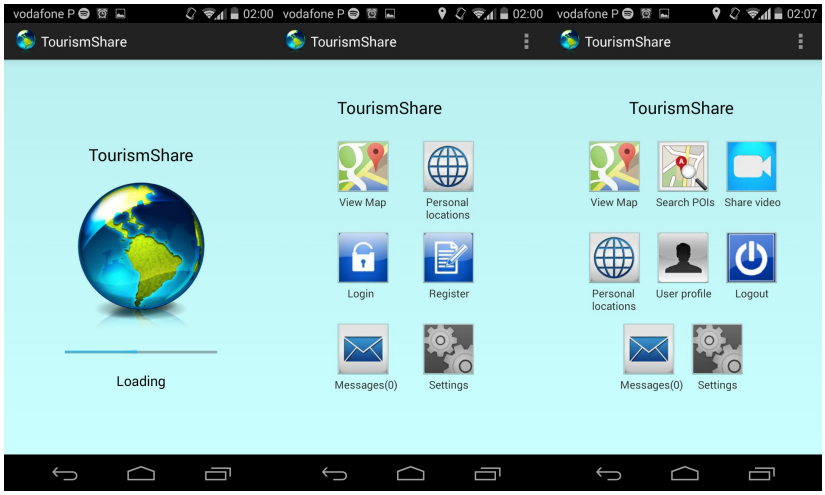

Figura 5.2: *Splashscreen* e menu principal

5.1.2 Autenticação, Registo e Gestão de Conta

De modo a poder utilizar a aplicação na sua plenitude é necessário efetuar o registo de utilizador. Este registo requer o preenchimento de um formulário que contém nome, nome de utilizador, palavra-chave, *email*, género e nacionalidade. No caso de sucesso, a conta de utilizador é criada e o utilizador é automaticamente autenticado no sistema, sendo reencaminhado para o menu partilhado. Em posteriores autenticações será apenas necessário digitar o nome de utilizador e palavra-chave. Finalmente, a gestão da sua conta pode ser feita através do menu de definições de utilizador que permite alterar qualquer dado definido aquando do registo, mediante a apresentação da palavra-chave atual do utilizador.

A Figura 5.3 apresenta, em primeiro lugar, o menu de registo seguido do menu de autenticação e, finalmente, o menu de definições de utilizador.

5.1.3 Criação de Local Pessoal

No menu de gestão de locais pessoais é apresentada a lista de locais pessoais armazenados no dispositivo, sendo estes compostos pelo nome, categoria e sím-

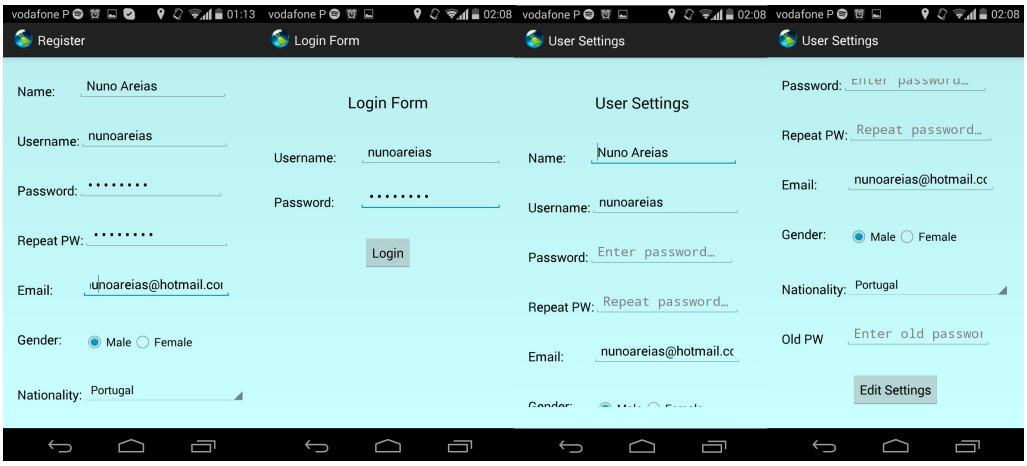

Figura 5.3: Autenticação, registo e alteração de dados pessoais

bolo de partilha, caso seja partilhado. Clicando no botão de criação de novo local, é apresentado ao utilizador um formulário para definição do nome do local, descrição do local, categoria e subcategoria e método de obtenção de georreferenciação (através da posição atual do utilizador ou utilizando um mapa clicável).

No exemplo apresentado na Figura 5.4 primeiro apresenta-se o menu de gestão de locais pessoais, de seguida, o formulário de adição de novo local, onde propositadamente se deixa por preencher a descrição do local de modo a exemplificar o funcionamento do sistema de validação de resultados desenvolvido e, finalmente, adiciona-se um novo local, utilizando o método de mapa clicável.

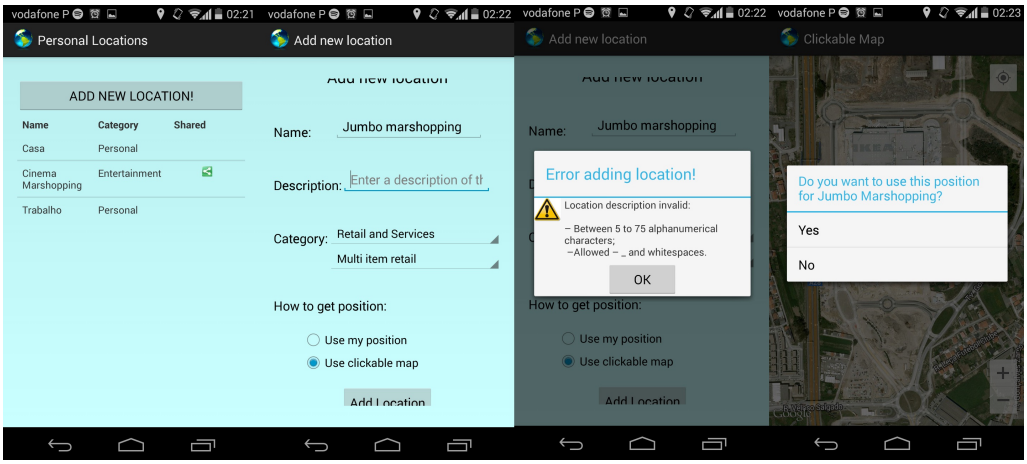

Figura 5.4: Criação de novo local pessoal

5.1.4 Partilha de Local Pessoal

No menu de gestão de locais pessoais, ao clicar no nome de um local, é apresentada a informação do local (nome, categoria/subcategoria, descrição e estado de partilha) e um menu. O menu permite ver num mapa, partilhar/deixar de partilhar, modificar e apagar local.

Na Figura 5.5 é apresentado o menu referido na imagem do lado esquerdo, de seguida, o local no mapa e, finalmente, o menu de partilha do local, onde são apresentados os locais próximos do local a partilhar de modo ao utilizador verificar se o local já existe ou se é novo. A Figura 5.6 mostra nas primeiras duas imagens o menu de partilha de novo local onde o utilizador faz uma primeira avaliação do local e define o seu horário de funcionamento. Após confirmação, o local passa a ser partilhado e é apresentado em conformidade no menu de gestão de locais pessoais como se verifica na última imagem. Finalmente, na Figura 5.7 é apresentado um exemplo de partilha de um local já existente no sistema.

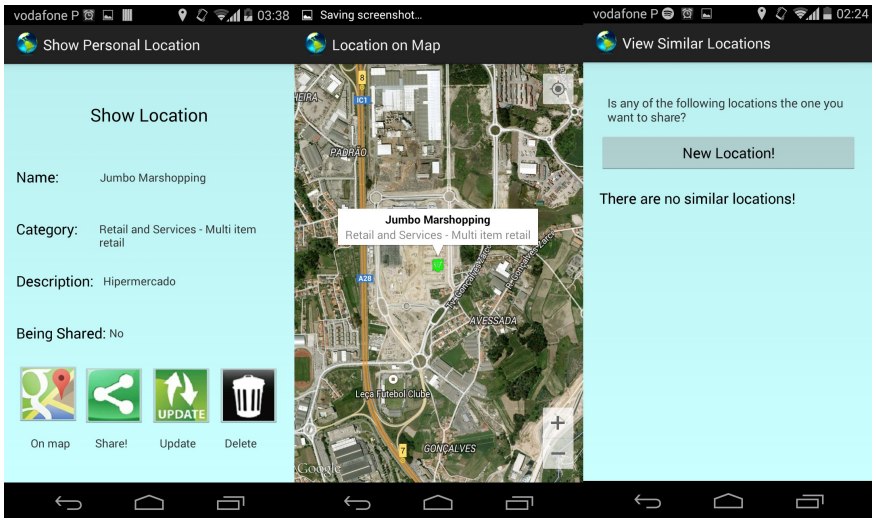

Figura 5.5: Mostrar local criado

5.1.5 Alteração de Local Pessoal

Se o utilizador pretender, é possível alterar as informações referentes a um dado local pessoal quer se trate de um local partilhado, quer de um local pessoal. No caso de locais partilhados o utilizador tem de ser o publicador original do local para o poder alterar.

Na Figura 5.8 é apresentado este processo. Nas primeiras três imagens altera-se um local partilhado e na última imagem altera-se um local pessoal.

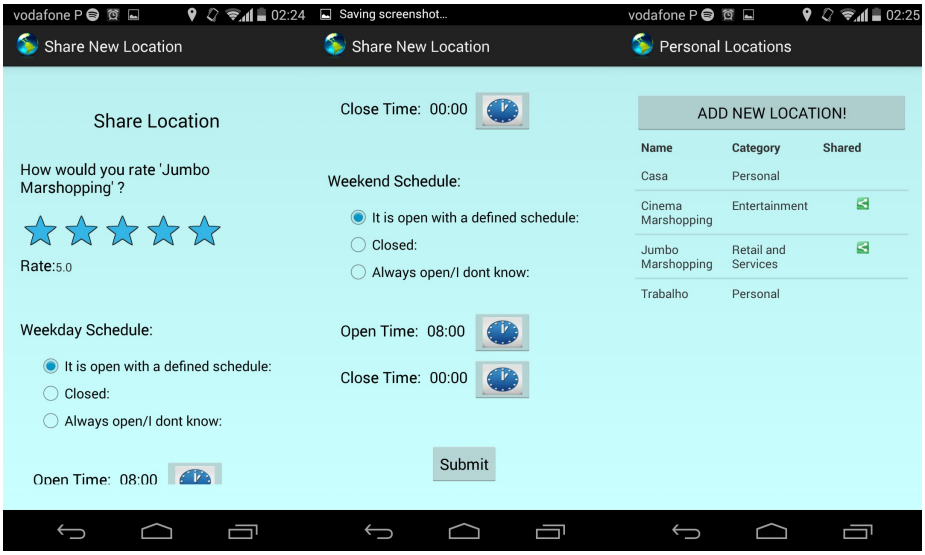

Figura 5.6: Partilhar novo local

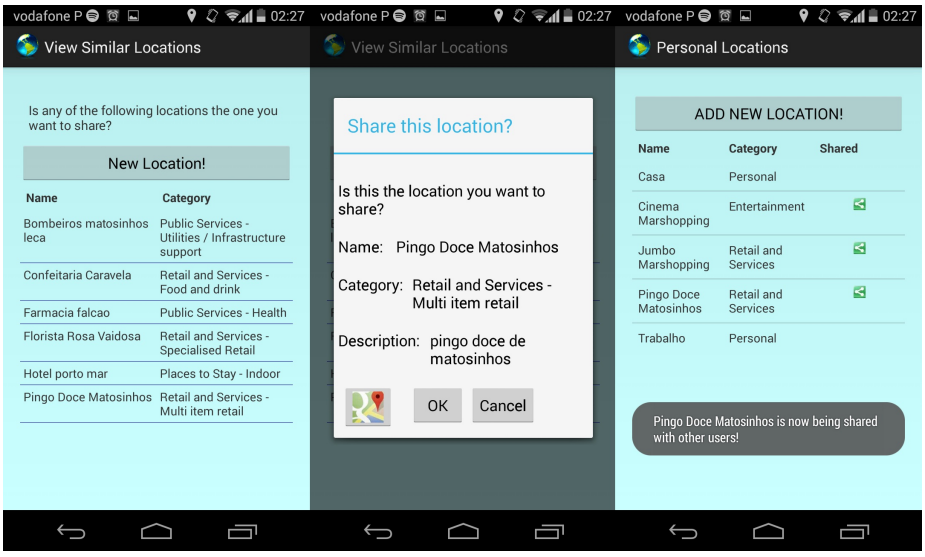

Figura 5.7: Partilhar local já existente

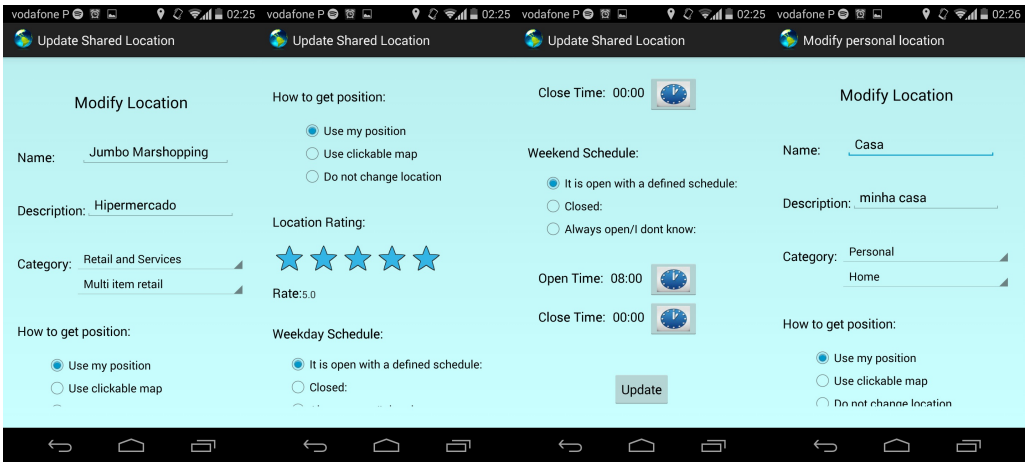

Figura 5.8: Alterar local partilhado/não partilhado

5.1.6 Partilha de Vídeo

A partilha de um vídeo apresenta um menu para o utilizador se autenticar no ecossistema Google. Em caso de sucesso é-lhe apresentada a lista de vídeos que publicou no YouTube. Clicando no botão de gravação, o utilizador é encaminhado para o processo de captura de vídeo. Após captura, o utilizador define o nome do vídeo e, de seguida, decide se pretende ou não partilhá-lo. O processo descrito está apresentado na Figura 5.9.

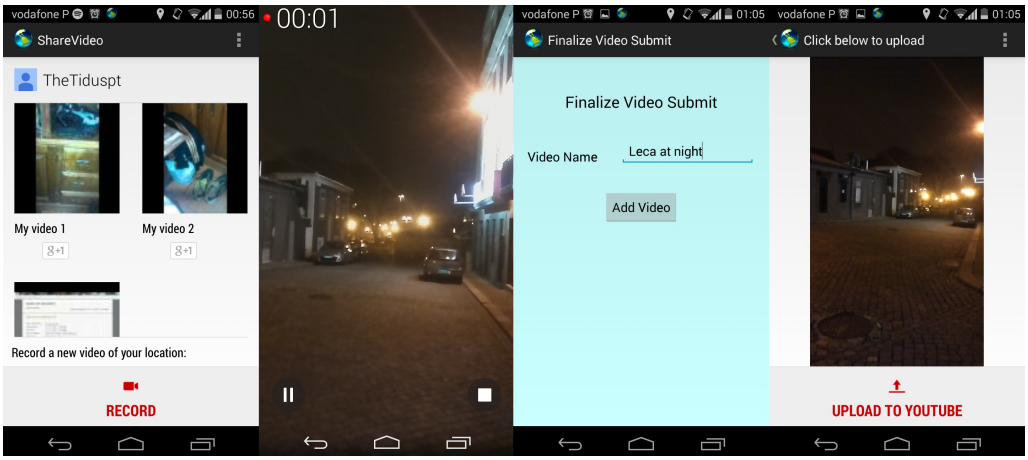

Figura 5.9: Gravar e partilhar novo vídeo

5.1.7 Pesquisa Manual de POI

O utilizador pode solicitar a recomendação de pontos de interesse na sua vizinhança ou em redor de um ponto definido, clicando no botão de pesquisa de

POI do menu principal. No menu de pesquisa o utilizador define que tipo(s) de ponto(s) pretende pesquisar (locais, artigos e vídeos), a categoria e subcategoria dos locais a pesquisar (disponível apenas se pretender pesquisar locais), o local, data e hora que pretende para esta pesquisa. Em resposta, é-lhe apresentado um mapa com a sua localização atual e cada um dos pontos de interesse encontrados representados através de marcadores, existindo marcadores identificativos de cada tipo de POI. Finalmente, através do menu de opções localizado no canto superior direito do ecrã, é possível definir-se quais os locais pessoais, para além dos POI recomendados, a apresentar no mapa.

A Figura 5.10 apresenta os menus deste processo. O marcador azul com símbolo "i" identifica os artigos, a imagem de câmara identifica os vídeos e os restantes marcadores correspondem a POI.

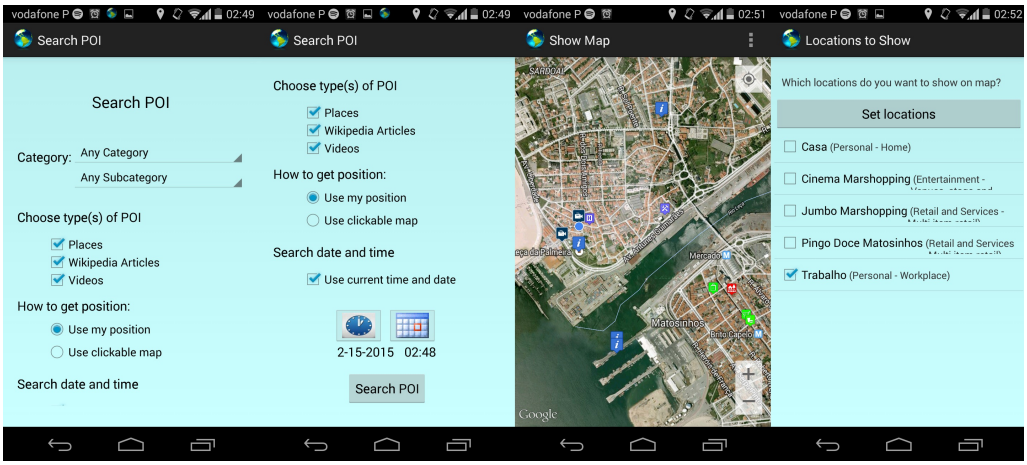

Figura 5.10: Pesquisa de POI

5.1.8 Visualização de Artigo Recomendado

O utilizador, ao clicar no marcador de um determinado artigo, obtém uma caixa de texto com o nome do artigo e a sua descrição. Voltando a clicar, o utilizador é encaminhado para o menu de apresentação do artigo onde, para além de nome e corpo do artigo, se apresentam a imagem e a hiperligação para consultar o artigo integral. Este processo é exemplificado na Figura 5.11.

5.1.9 Visualização de Local Recomendado

De forma análoga, o utilizador, caso clique no marcador de um local, é encaminhado para o menu de resumo do local onde, para além da apresentação, nome, categoria/subcategoria e descrição do local, pode verificar o horário, avaliá-lo ou reportar erros. Na Figura 5.12 apresenta-se o menu de resumo, com o horário e

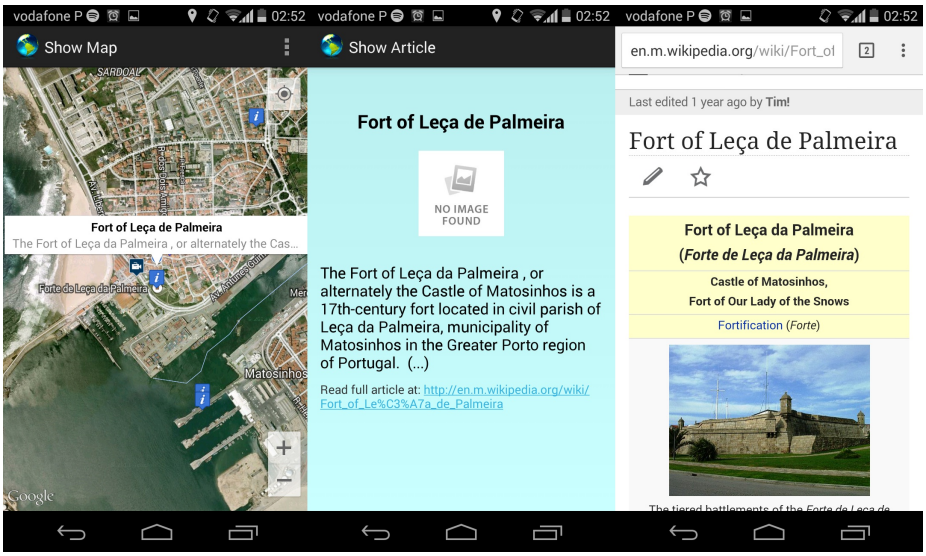

Figura 5.11: Mostrar artigo recomendado

o estado atual de funcionamento (aberto ou fechado), levando em conta o dia da semana (a pesquisa foi feita a um domingo daí ser apresentado o horário de fim de semana). Finalmente, a última imagem contém o menu de avaliação.

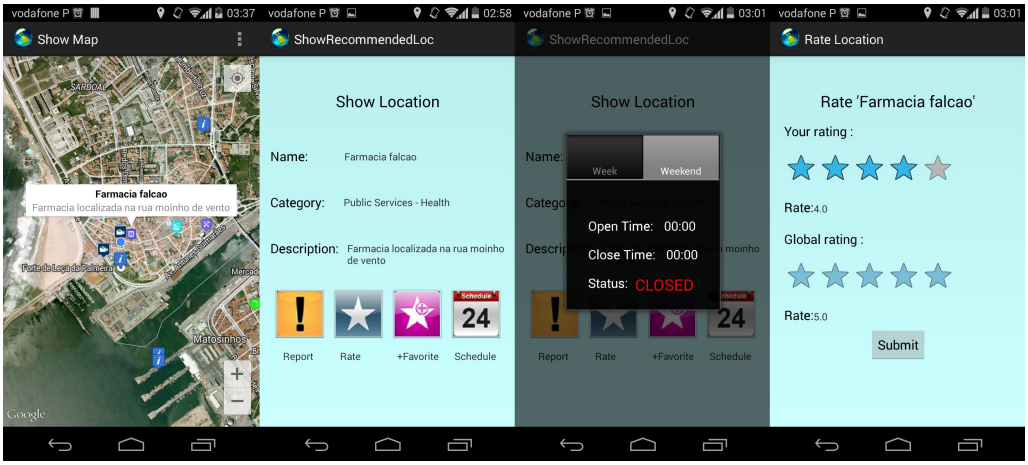

Figura 5.12: Mostrar local recomendado

5.1.10 Reporte de Local Recomendado

No caso de um local apresentar alguma incorreção (local não existente, horário de funcionamento errado, *etc.*), qualquer utilizador pode reportar o erro. No menu de reporte o utilizador define a categoria do problema e introduz uma curta descrição. O erro será enviado ao utilizador que publicou o local, permi-

tindo assim corrigir o problema identificado. Na Figura 5.13 a primeira imagem apresenta o menu de reporte. O envio de um reporte faz surgir no menu principal do publicador do local uma animação de mensagens não lidas. A imagem seguinte apresenta a caixa de mensagens do publicador e, finalmente, a mensagem recebida.

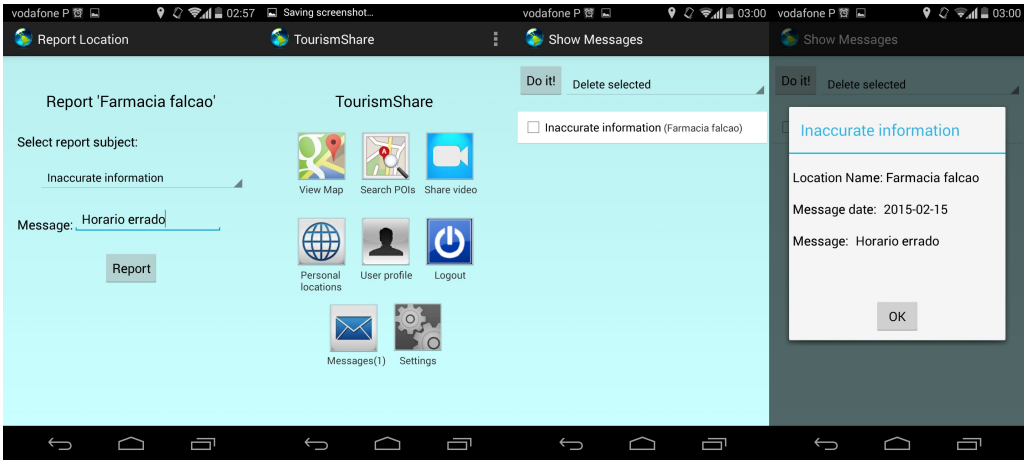

Figura 5.13: Reportar local recomendado

5.1.11 Reprodução de Vídeo Recomendado

Após o processo de recomendação de POI, ao clicar no marcador de um vídeo, arranca a reprodução do vídeo respetivo. A Figura 5.14 descreve este processo.

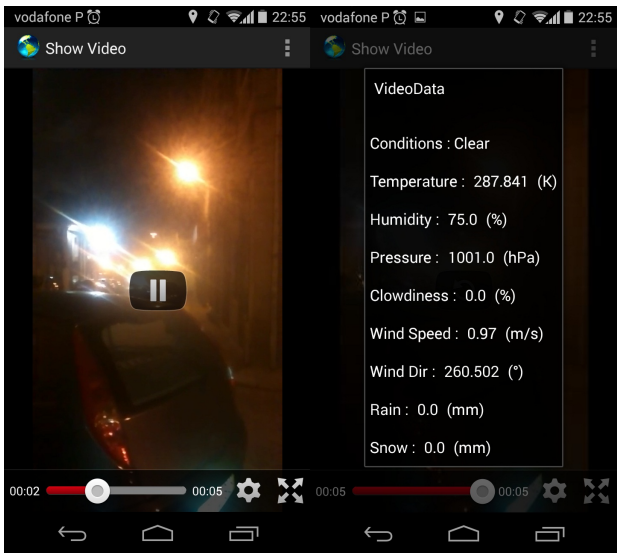

Figura 5.14: Mostrar vídeo recomendado

No lado direito da Figura 5.14 são apresentados os dados meteorológicos recolhidos aquando da gravação do vídeo. Durante a reprodução do vídeo, em função da velocidade do vento, são reproduzidos efeitos sonoros concordantes. A Tabela 5.1 sumaria os níveis de vento contemplados.

Tabela 5.1: Níveis de intensidade de vento

Velocidade vento (m/s)	**Efeito sonoro**
[2;10[	Vento de baixa intensidade
[10;25[	Vento de média intensidade
[25,∞[	Vento de alta intensidade

5.1.12 Pesquisa Automática de POI

A aplicação gera recomendações automáticas de pontos de interesse. O utilizador pode definir se pretende ou não ativar esta funcionalidade no menu de preferências. Caso esteja ativada, o utilizador irá receber uma notificação sempre que o sistema recomende um novo conjunto de POI. A Figura 5.15 ilustra o processo.

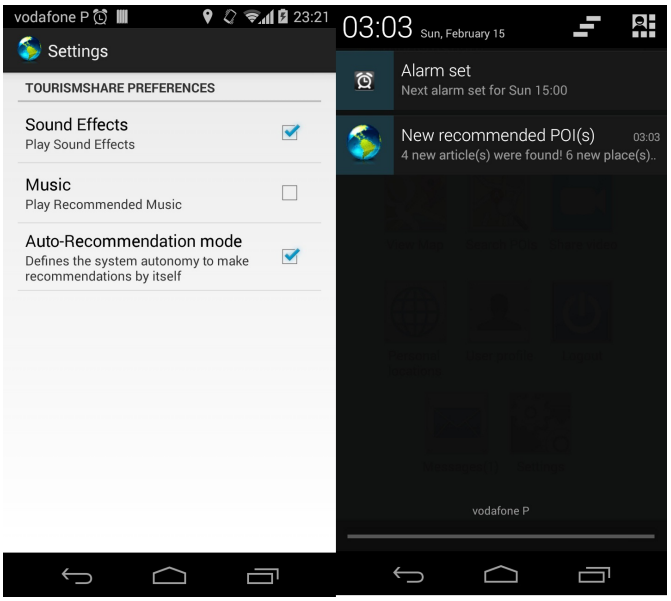

Figura 5.15: Pesquisa automática de POI

5.2 Resposta Temporal e Tráfego

O teste da aplicação foi efetuado nas seguintes condições:

- (i) Um utilizador com 5 locais pessoais e 1 mensagem armazenada na aplicação cliente;
- (ii) Quatro utilizadores com 18 locais pessoais e 6 vídeos armazenados na base de dados central;
- (iii) A plataforma de *front-end* foi um dispositivo móvel Motorola Moto G com uma *Central Processing Unit* (CPU) Quad-core 1,2 GHz Cortex-A7, memória *Random-Access Memory* (RAM) de 1 GB e com acelerómetro e sensor de proximidade como sensores embutidos [63];
- (iv) A plataforma de *back-end* foi um computador portátil Compaq Pressario com um CPU AMD Sempron M120 / 2,1 GHz e memória RAM de 4 GB [64];
- (v) Uma ligação de dados de Internet por fibra ótica que durante o dia de teste apresentou como resultado de um teste de transmissão de dados um valor médio de descarga de 19,08 Mib/s e um valor médio de carga de 1,67 Mib/s, sendo que o tempo médio de latência foi de 81 ms.

Durante o teste foram medidos o tempo médio de acesso às diferentes funcionalidades da aplicação assim como o tráfego de dados entre o dispositivo cliente e o servidor (carga) e entre o servidor e o dispositivo cliente (descarga). Os dados obtidos após 10 invocações de cada uma das funcionalidades são os apresentados na Tabela 5.2.

Tabela 5.2: Valores de tráfego das diferentes funcionalidades

Funcionalidade	**Tempo (s)**	**Carga (B)**	**Descarga (B)**
Autenticação	1,33	856	658
Registo	1,77	1144	660
Alteração de perfil	1,27	1053	663
Pesquisa manual de POI	2,01	1124	4367
Avaliar local	1,71	774	653
Reportar local	1,65	783	578
Visualização de vídeo	1,21	795	576
Partilha de local	1,91	1274	664
Modificação de local	1,88	1235	658
Partilha de vídeo	2,55	3764	576
Pesquisa automática de POI	2,32	1174	4567

A análise dos valores obtidos permitem concluir que, naturalmente, as funcionalidades que implicam maior tráfego de dados são as que apresentam maior tempo médio de resposta. A Tabela 5.3 caracteriza o sistema instalado no dispositivo móvel.

Tabela 5.3: Características gerais do sistema

Informação	**Valor**
API Android compilação	19
API Android compatíveis	16+
Dimensão da base de dados local	32,09 kiB
Espaço ocupado pela aplicação cliente	82,31 MiB

5.3 Sumário

Neste capítulo foram descritos os testes, depuração e validação de resultados do sistema desenvolvido. Caracterizou-se o dispositivo utilizado e apresentaram-se as capturas de ecrã da execução de cada uma das funcionalidades da aplicação, complementando as imagens com uma breve descrição do seu funcionamento. Os testes da resposta temporal e tráfego permitiram caracterizar o funcionamento do sistema.

Capítulo 6

Conclusões

Este capítulo discute os resultados obtidos e apresenta sugestões de trabalho futuro.

6.1 Resultados

No âmbito desta tese desenvolveu-se um sistema sensível ao contexto de partilha e recomendação de pontos de interesse (locais, vídeos e artigos da Wikipedia), incluindo vídeo e áudio-imersão. Face à aplicação original descrita no Capítulo 2, este sistema manteve a arquitetura e algumas das funcionalidades pré-existentes. Adicionalmente, criou-se uma nova interface simples e intuitiva, que assegura uma separação clara entre os modos de funcionamento *stand-alone* e partilhado. Em termos de representação dos locais, adotou-se uma organização em categorias e subcategorias e adicionou-se o contexto temporal, especificando períodos de funcionamento. Em termos das fontes de informação, desenvolveu-se um sistema de reputação dos publicadores de locais. Em termos de novos serviços, construiu-se um serviço contínuo de determinação do contexto do utilizador, um serviço de recomendação baseada no contexto do utilizador, um serviço de áudio-imersão e um serviço de reporte de incorreções identificadas em locais partilhados. O serviço de determinação do contexto do utilizador obtém a posição e velocidade do utilizador e infere o seu contexto. O serviço de recomendação automática gera e apresenta recomendações ao utilizador à medida que este se move sob a forma de notificações no caso de existirem pontos de interesse novos. O serviço de áudio-imersão reproduz músicas de acordo com o nível de atividade do utilizador. O serviço de reporte de incorreções de locais partilhados permite enviar ao publicador mensagens com os erros detetados. Caso os erros não sejam corrigidos, a reputação de publicador diminui, afetando negativamente a recomendação

dos locais que publicou. Finalmente, foi criado um sistema de vídeo-imersão básico que permite demonstrar o potencial da recolha de dados durante a gravação dos vídeos. A vídeo-imersão consiste na reprodução de sons representativos das condições meteorológicas armazenadas, *i.e.*, vento.

Enquanto a aplicação ErasmusApp permitia definir as categorias de interesse e sugeria locais, artigos e vídeos a pedido do utilizador e em função da localização do utilizador, a atual aplicação permite ao utilizador definir também subcategorias de interesse e obter recomendações automáticas em função do contexto temporal, espacial e de movimentação do utilizador assim como inclui vídeo e áudio-imersão.

6.2 Trabalho Futuro

O sistema de recomendação pode ser refinado recorrendo a informação que o sistema recolhe atualmente, nomeadamente as características do dispositivo utilizado, à integração com redes sociais e ao grau de semelhança entre avaliações.

O sistema de vídeo-imersão, que é atualmente baseado nas condições meteorológicas, pode ser mais explorado e alargado, nomeadamente, utilizando a informação já armazenada proveniente dos sensores locais. O sistema de áudio-imersão, que é atualmente baseado no nível de atividade do utilizador, pode passar a permitir a definição das respetivas listas de músicas pelo utilizador e transformar-se num sistema de recomendação de músicas em função do nível de atividade, do estado emocional e do perfil do utilizador.

A recomendação de artigos Wikipedia pode, no caso de museus, monumentos ou casas de espetáculos, fornecer a hiperligação para o *site* da instituição de modo a permitir a compra de bilhetes ou a reserva de lugares.

Bibliografia

[1] Karel Bruyneel and Benedita Malheiro. Erasmusapp: A location-based collaborative system for erasmus students. In Lieven De Strycker, editor, *ECUMICT 2014*, volume 302 of *Lecture Notes in Electrical Engineering*, pages 35–47. Springer International Publishing, 2014. [cited at p. v, 1, 17, 18]

[2] Bjorn Baecke. Context-aware video-sharing android application, 2014. [cited at p. v, 1, 18, 19, 67]

[3] Paul Resnick and Hal R. Varian. Recommender systems.(special section: Recommender systems)(cover story). *Communications of the ACM*, 1997. [cited at p. 5]

[4] Prem Melville and Vikas Sindhwani. Recommender systems. *IBM T.J. Watson Research Center*, 2010. [cited at p. 6]

[5] K. Verbert, Manouselis N., X. Wolpers Ochoa, H. M. Drachsler, I. Bosnic, and E. Duval. Context-aware recommender systems for learning: A survey and future challenges. *IEEE TRANSACTIONS ON LEARNING TECHNOLOGIES*, 2012. [cited at p. 6]

[6] Alexander Tuzhilin Gediminas Adomavicius. Context-aware recommender systems. *Carlson School of Management, University of Minnesota*, 2010. [cited at p. 6]

[7] Sung-Jin Kim Jong-yi Hong, Eui-ho Suh. Context-aware systems: A literature review and classification. *Expert Systems with Applications*, 2009. [cited at p. 6]

[8] L Feng A. van Bunningen and P. Apers. Context for ubiquitous data management. *Ubiquitous Data Management*, 2005. [cited at p. 7]

[9] G. D. Abowd and A. D. Dey. Towards a better understanding of context and context-awareness. *HUC '99 Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*, 1999. [cited at p. 7]

[10] Peter Christen Charith Perera, Arkady Zaslavsky and Dimitrios Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE COMMUNICATIONS SURVEYS AND TUTORIALS*, 2013. [cited at p. v, 8]

[11] Norman Adams Bill N. Schilit and Roy Want. Context-aware computing applications. *Mobile Computing Systems and Applications*, 1994. [cited at p. 7]

[12] K. Henricksen. A framework for context-aware pervasive computing applications. *Computer Science, School of Information Technology and Electrical Engineering, The University of Queensland*, 2003. [cited at p. 7]

[13] Ahmed A.M. Feng Xia, Asabere N.Y., Jing Li, and Xiangjie Kong. Mobile multimedia recommendation in smart communities, a survey. *School of Software, Dalian University of Technology*, 2013. [cited at p. 9]

[14] Schahram Dustdar Matthias Baldauf and Florian Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc and Ubiquitous Computing*, 2007. [cited at p. v, 9, 10]

[15] Ville Haataja Heikki Ailisto, Petteri Alahuhta, Vesa Kyllönen, and Mikko Lindholm. Structuring context aware applications: Five-layer model and example case. *Proceedings of the Workshop on Concepts and Models for Ubiquitous Computing*, 2002. [cited at p. 10]

[16] Bachir Chihani, Emmanuel Bertin, Fabrice Jeanne, and Noel Crespi. Context-aware systems: a case study. *The International Conference on Digital Information and Communication Technology and its Applications*, 2011. [cited at p. 10]

[17] Shahana Sen Gediminas Adomavicius, Ramesh Sankaranarayanan and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems (TOIS)*, 2005. [cited at p. 11]

[18] John T. Riedl Michael D. Ekstrand and Joseph A. Konstan. Collaborative filtering recommender systems. *Foundations and Trends in Human–Computer Interaction*, 2010. [cited at p. 12]

[19] Stephan Spiegel. A hybrid approach to recommender systems based on matrix factorization. *Department for Agent Technologies and Telecommunications, Technical University Berlin*, 2009. [cited at p. v, 12, 13]

[20] Marco de Gemmis Pasquale Lops and Giovanni Semeraro. *Content-based Recommender Systems: State of the Art and Trends.* Springer US, 2011. [cited at p. 13]

[21] Vineet Richariya Atisha Sachan. A survey on recommender systems based on collaborative filtering technique. *International Journal of Innovations in Engeneering and Technology*, 2013. [cited at p. 13]

[22] Xiaoyuan Su and Taghi M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009. [cited at p. 13]

[23] A. H. Nabizadeh Rafsanjani et al. Recommendation systems: a review. *International Journal of Computational Engineering Research*, 2013. [cited at p. 13]

[24] Giovanni Semeraro. Content-based recommender systems problems, challenges and research directions. *INTELLIGENT TECHNIQUES FOR WEB PERSONALIZATION and RECOMMENDER SYSTEMS*, 2010. [cited at p. 13]

[25] Jon Herlocker J. Ben Schafer, Dan Frankowski and Shilad Sen. Collaborative filtering recommender systems. *The Adaptive Web, Springer Verlag*, 2007. [cited at p. 15]

[26] Castillo Hugo. Hybrid content-based collaborative-filtering music recommendations. *Department of Computer Science, University of Twente*, 2007. [cited at p. 15]

[27] João Ramalho and Teresa Chambel. Windy sight surfers: Sensing and awareness of 360º immersive videos on the move, 2013. [cited at p. 16]

[28] João Ramalho and Teresa Chambel. Immersive 360º mobile video with an emotional perspective, 2013. [cited at p. v, 16, 17]

[29] Vito Claudio Ostuni, Tommaso Di Noia, Roberto Mirizzi, Davide Romito, and Eugenio Di Sciascio. Cinemappy: a context-aware mobile app for movie recommendations boosted by dbpedia, 2007. [cited at p. v, 19, 20]

[30] Marta Rey-López, Ana Belén Barragáns-Martínez, Ana Peleteiro, Fernando A. Mikic-Fonte, and Juan C. Burguillo. moretourism: Mobile recommendations for tourism, 2011. [cited at p. v, 20, 21]

[31] Stefan Peer Linas Baltrunas, Bernd Ludwig and Francesco Ricci. Context-aware places of interest recommendations for mobile users, 2011. [cited at p. v, 21, 22]

[32] Iosif Viktoratos, Athanasios Tsadiras, and Nick Bassiliades. Plis+: A rule-based personalized location information system. In *RuleML (2)*, 2012. [cited at p. 22]

[33] Iosif Viktoratos, Athanasios Tsadiras, and Nick Bassiliades. A rule based personalized location information system for the semantic web. In *EC-Web*, pages 27–38, 2013. [cited at p. 22]

[34] Iosif Viktoratos, Athanasios Tsadiras, and Nick Bassiliades. Geosocial splis: A rule-based service for context-aware point of interest exploration, 2014. [cited at p. v, 22, 23]

[35] Abayomi Moradeyo Otebolaku and Maria Teresa Andrade. Context-aware media recommendations for smart devices. *Journal of Ambient Intelligence and Humanized Computing*, 6(1):13–36, 2015. [cited at p. v, 23, 24]

[36] Dominique A. Heger. An introduction to the android operating environment, 2011. [cited at p. 28]

[37] Stefan Brahler. Analysis of the android architecture, 2010. [cited at p. 28]

[38] Mir Nauman Shahryar Khan, Syed Hammad, Sohail Khan, and Masoon Alam. Analysis report on android application framework and existing security architecture, 2010. [cited at p. 28]

[39] Android versions comparison. `http://socialcompare.com/en/comparison/android-versions-comparison`. Accessed: 2014-11-29. [cited at p. 28]

[40] Android distribution market share. `http://www.droid-life.com/tag/distribution/`. Accessed: 2014-12-07. [cited at p. v, 28, 30]

[41] Opengl es. `http://developer.android.com/guide/topics/graphics/opengl.html`. Accessed: 2014-12-19. [cited at p. 34]

[42] About geonames. `http://www.geonames.org/about.html`. Accessed: 2014-12-20. [cited at p. 35]

[43] Openweathermap api. `http://openweathermap.org/api`. Accessed: 2015-01-12. [cited at p. 38]

[44] Thomas J. Bergin. History of database management systems. *IEEE Annals*, 2009. [cited at p. 39]

[45] James P. Fry and Edgar H. Sibley. Evolution of data-base management systems. *ACM Comput. Surv.*, 8(1):7–42, March 1976. [cited at p. 39]

[46] About sqlite. `http://www.sqlite.org/about.html`. Accessed: 2014-12-14. [cited at p. 40]

[47] What is mysql? `http://dev.mysql.com/doc/refman/4.1/en/what-is-mysql.html`. Accessed: 2014-12-15. [cited at p. 40]

[48] Gnu licence list. `http://www.gnu.org/licenses/license-list.html`. Accessed: 2014-12-15. [cited at p. 40]

[49] About phpmyadmin. `http://www.phpmyadmin.net/home_page/index.php`. Accessed: 2014-12-17. [cited at p. 40]

[50] Java tools and technologies landscape for 2014. `http://zeroturnaround.com/rebellabs/java-tools-and-technologies-landscape-for-2014/6/`. Accessed: 2014-12-09. [cited at p. 40]

[51] Best java ide review. `http://faq.programmerworld.net/programming/best-java-ide-review.html`. Accessed: 2014-12-10. [cited at p. 40]

[52] Top 48 integrated developer environments (ides) and code editors. `http://blog.profitbricks.com/top-integrated-developer-environments-ides/`. Accessed: 2014-12-10. [cited at p. 40]

[53] Comparing java ides: Eclipse vs. netbeans vs. intellij. `http://mobiledevices.about.com/od/additionalresources/fl/Comparing-Java-IDEs-Eclipse-vs-NetBeans-vs-IntelliJ.htm`. Accessed: 2014-12-10. [cited at p. 40]

[54] Onur Cinar. *Android Apps with Eclipse*. Apress, Berkely, CA, USA, 1st edition, 2012. [cited at p. 41]

[55] Android sdk. `http://www.techopedia.com/definition/4220/android-sdk`. Accessed: 2014-12-10. [cited at p. 42]

[56] Android debug bridge. `http://developer.android.com/tools/help/adb.html`. Accessed: 2014-12-11. [cited at p. 44]

[57] logcat. `http://developer.android.com/tools/help/logcat.html`. Accessed: 2014-12-12. [cited at p. 44]

[58] Build class overview. `http://developer.android.com/reference/android/os/Build.html`. Accessed: 2015-04-20. [cited at p. 52]

[59] Openweathermap - parameters of api call for current conditions. `openweathermap.org/weather-data-current`. Accessed: 2015-01-14. [cited at p. 55]

[60] Fragments. `http://developer.android.com/guide/components/fragments.html`. Accessed: 2014-12-23. [cited at p. 65]

[61] Mapfragment class. `http://developer.android.com/reference/com/google/android/gms/maps/MapFragment.html`. Accessed: 2014-12-23. [cited at p. 65]

[62] Googlemap class. https://developer.android.com/reference/com/google/android/gms/maps/GoogleMap.html. Accessed: 2014-12-24. [cited at p. 65]

[63] Motorola moto g - full phone specifications. www.gsmarena.com/motorola_moto_g-5831.php. Accessed: 2015-02-25. [cited at p. 87]

[64] Hp compact pressario cq61 specifications. www.cnet.com/products/hp-compaq-presario-cq61. Accessed: 2015-02-25. [cited at p. 87]

Anexos

Anexo A

Vídeos Demonstrativos

Playlist: https://www.youtube.com/watch?v=UdZdjdHIY-M&index=1&list=PL0rIsJ_SO2ohKr7RhyigaPwmulIX9tiMY

Vídeo 1: https://www.youtube.com/watch?v=UdZdjdHIY-M

Vídeo 2: https://www.youtube.com/watch?v=19rZw2dcr3E

Vídeo 3: https://www.youtube.com/watch?v=_bcnVrJvEQk

Vídeo 4: https://www.youtube.com/watch?v=Z5h6jmQe9Bs

Vídeo 5: https://www.youtube.com/watch?v=SqDEpw-iIHM

Vídeo 6: https://www.youtube.com/watch?v=LvykRW6GmJk

Vídeo 7: https://www.youtube.com/watch?v=5IVXJsO-Fb8

Vídeo 8: https://www.youtube.com/watch?v=dFU16kYVPfI

Ficheiro com outputs do servidor: https://dl.dropboxusercontent.com/u/55825878/ServerOutputs.txt

Código fonte do projeto: https://dl.dropboxusercontent.com/u/55825878/Tese_1090430.rar