Instituto Superior de Engenharia do Porto

# Módulo de Realidade Aumentada para uma aplicação LBS

**Raul Duarte Moreira Valério Pinto**

Dissertação para obtenção do Grau de Mestre em

**Engenharia Informática,**

Área de Especialização em

**Sistemas Gráficos e Multimédia**

**Orientador: Prof. Doutor Paulo Baltarejo**

**Co-orientador: Prof. Doutor Luís Lino Ferreira**

**Júri**:

Presidente:

Doutora Maria de Fátima Coutinho Rodrigues

Vogais:

Doutor Filipe de Faria Pacheco Paulo

Doutor Paulo Manuel Baltarejo de Sousa

Doutor Luís Miguel Moreira Lino Ferreira

Porto, Outubro de 2013

# Resumo

As aplicações móveis de serviços baseados na localização, denominados LBS (*Location-based services*), disponibilizam serviços ao utilizador baseadas na sua localização geográfica. Este tipo de serviços começou a surgir ainda na década de 90 e, à medida que o número de dispositivos móveis cresceu de forma exponencial, a sua oferta disparou consideravelmente. Existem várias áreas com aplicabilidade prática, mas o foco desta tese é a pesquisa e localização de pontos de interesse (POI's).

Através dos sensores que os dispositivos móveis atualmente disponibilizam, torna-se possível localizar a posição do utilizador e apresentar-lhe os pontos de interesse que estão situados em seu redor. No entanto essa informação isolada revela-se por vezes insuficiente, uma vez que esses pontos de interesse são à partida desconhecidos para o utilizador. Através do serviço *coolplaces*, um projeto que pretende dedicar-se à pesquisa e partilha de POI's, podemos criar a nossa rede de amigos e de locais, beneficiando assim da respetiva informação de contexto de um determinado POI.

As inovações tecnológicas permitiram também o aparecimento de aplicações de Realidade Aumentada nos dispositivos móveis, isto é, aplicações capazes de sobrepor imagens virtuais a visualizações do mundo real. Considerando a visualização de POI's num dado ambiente, se encararmos a Realidade Aumentada como um potenciador da interação do utilizador com o mundo real, rapidamente identificamos as potencialidades da junção destes conceitos numa só aplicação.

Sendo assim, o trabalho desenvolvido nesta tese pretende constituir um estudo sobre a implementação e desenvolvimento de um módulo de Realidade Aumentada para a aplicação móvel do serviço *coolplaces*, fazendo uso da tecnologia disponível no mercado de forma a proporcionar uma experiência inovadora e acrescentar valor à referida aplicação.

**Palavras-chave**: Realidade aumentada, Pontos de interesse, Dispositivos móveis

# Abstract

The mobile applications based upon the user's localization, named LBS (*Location-based services*), present services to the user based on the geographical position of his device. This type of services first appeared in the nineties (1990s) and, by the time mobile devices offer was increasing rapidly, they also thrived. There are plenty of areas with practical applicability of these services, but the main focus of this thesis is the search and localization of Points of Interest.

With the use of the sensors that are a part of the mobile devices nowadays, it has become possible to locate the user's position and display to him the POI's in their surroundings. However this sole information may reveal itself insufficient since these POI's may be unknown to the user. With the use of the coolplaces service, a project that intends to be a reference of search and share of POI data, we can make our own network of friends and POI's, taking advantage of the context information of certain POI.

The technological developments also allowed the uprise of Augmented Reality applications for the mobile devices, in other words, applications capable of overlap virtual images on top of real world representations. If we take under consideration the visualization of a POI in its environment, thinking of Augmented Reality as an enhancer of the user interaction with the real world, we quickly realize the potential of mixing these two concepts in one application.

With that being said, the work developed in this thesis seeks to provide a study of the implementation and development of a Augmented Reality resource to be included and to enhance the mobile application of the coolplaces project, with the help of the technology that the market has to offer, in a way that can add value to the application while providing an exhilarating experience to the users.

**Keywords**: Points of Interest, Augmented Reality, Mobile devices

# Agradecimentos

Em primeiro lugar quero agradecer à minha família e à minha namorada pelo apoio e pelas horas despendidas a trabalhar.

Queria deixar um agradecimento especial ao Helder Tato e ao Jorge Martins, meus grandes amigos e parceiros, sem os quais não teria sido possível desenvolver o projeto *coolplaces*.

Por fim, não podia deixar de agradecer ao Prof. Paulo Baltarejo pela disponibilidade demonstrada e pela preciosa ajuda na compilação deste documento.

# Índice

**1 Introdução ..... 1**

1.1 Contexto e Motivação ..... 1

1.2 Objetivos e Contribuições ..... 2

1.3 Estrutura ..... 2

1.4 Síntese ..... 3

**2 Serviços baseados na localização e Realidade Aumentada em dispositivos móveis ..... 5**

2.1 Dispositivos móveis ..... 6
- 2.1.1 Introdução ..... 6
- 2.1.2 *Android* ..... 7
- 2.1.3 *iPhone OS* ..... 8
- 2.1.4 *Windows Phone* ..... 8
- 2.1.5 *Symbian* ..... 9
- 2.1.6 *BlackBerry OS* ..... 9

2.2 Serviços baseados na localização ..... 9
- 2.2.1 Introdução ..... 9
- 2.2.2 Desafios ..... 11
- 2.2.3 Soluções propostas ..... 11

2.3 Realidade aumentada ..... 12
- 2.3.1 Introdução ..... 12
- 2.3.2 Desafios ..... 13
- 2.3.3 Soluções propostas ..... 14
- 2.3.4 Considerações gerais ..... 15

2.4 Aplicações Ar Browser ..... 16
- 2.4.1 Introdução ..... 16
- 2.4.2 Estado da arte ..... 17
- 2.4.3 Desafios ..... 18
- 2.4.4 Soluções propostas ..... 20

2.5 Técnicas de visualização de Realidade Aumentada ..... 21

2.6 Conclusões ..... 24

**3 Análise de aplicações ..... 27**

3.1 Layar ..... 27

3.2 LibreGeoSocial ArViewer ..... 29

3.3 Wikitude ..... 30

3.4 Conclusões ..... 31

**4 Análise de Soluções ..... 33**

4.1 Layar API ..... 33
4.2 Metaio ..... 36
4.2.1 SDK ..... 37
4.2.2 junaio Plugin ..... 37
4.3 Droidar ..... 38
4.4 Mixare ..... 38
4.5 Wikitude SDK ..... 39
4.6 ARViewer SDK ..... 40
4.7 ARLab SDK ..... 41
4.8 Análise dos resultados ..... 41
**5 Desenvolvimento do Módulo AR do coolplaces mobile ..... 45**
5.1 Estrutura coolplaces mobile ..... 45
5.2 Estruturas de dados ..... 47
5.2.1 Objeto Place ..... 47
5.2.2 Objeto GeoNode ..... 48
5.2.3 Objeto Photo ..... 48
5.3 Novo paradigma ..... 49
5.4 Métodos de acesso aos dados ..... 50
5.5 Parsing dos dados ..... 51
5.6 Estrutura lógica da aplicação ..... 52
5.7 Funcionamento da aplicação ..... 54
**6 Conclusões ..... 59**
6.1 Testes de usabilidade ..... 59
6.2 Discussão ..... 64
6.3 Limitações ..... 64
6.4 Perspetivas futuras ..... 65
6.5 Considerações finais ..... 66
**7 Referências ..... 69**
**Anexo 1 – Diagramas UML ..... 73**
**Anexo 2 – Extratos de código ..... 76**

# Lista de Figuras

Figura 1 – Diagrama de arquitetura do sistema operativo *Android* ..... 7
Figura 2 – Modelo de comunicação LBS (Chin, 2012) ..... 10
Figura 3 – Modelo lógico de uma aplicação *Ar Browser* (MacWilliams, 2005) ..... 16
Figura 4 – Arquitetura típica de uma aplicação *AR Browser* (Vinny Reynolds, n. d.) ..... 17
Figura 5 – Algoritmo *Compass Filtering* (J. B. Gotow, n. d.) ..... 20
Figura 6 – Exemplo prático de distorções espaciais (Denis Kalkofen, 2011) ..... 24
Figura 7 – Arquitetura *Gateway* (Ben Butchart, 2011) ..... 28
Figura 8 – Arquitetura *Web* (Ben Butchart, 2011) ..... 29
Figura 9 – Arquitetura *Standalone* (Ben Butchart, 2011) ..... 30
Figura 10 – Configuração de uma *Layer* ..... 34
Figura 11 – Opções de gestão de uma *Layer* ..... 35
Figura 12 – Ferramentas de teste de uma *Layer* ..... 35
Figura 13 – Exemplo de utilização do *Metaio* ..... 36
Figura 14 – Fluxo lógico da solução *Mixare* (mixare, n. d.) ..... 38
Figura 15 – Exemplo de utilização do *Wikitude* ..... 40
Figura 16 – Exemplo de utilização do protótipo desenvolvido com a solução *Layar* ..... 43
Figura 17 – Fluxo lógico da utilização do módulo de Realidade Aumentada desenvolvido para a aplicação *coolplaces mobile* ..... 46
Figura 18 – Arquitetura dos módulos implementados ..... 53
Figura 19 – Diagrama de arquitetura de blocos da aplicação *coolplaces mobile* ..... 53
Figura 20 – Ecrã de início de sessão da aplicação ..... 54
Figura 21 – Ecrã principal da aplicação ..... 55
Figura 22 – Módulo de exploração de POI's *AR Places* ..... 55
Figura 23 – Módulo de exploração *AR Friends* ..... 56
Figura 24 – Detalhe do módulo de exploração *AR Places* ..... 57
Figura 25 – Detalhe do módulo de exploração *AR Friends* ..... 57
Figura 26 – Gráfico correspondente aos valores médios de avaliação de cada parâmetro ..... 60
Figura 27 – Gráfico correspondente à avaliação da eficácia ..... 61
Figura 28 – Gráfico correspondente à avaliação da satisfação do utilizador ..... 62
Figura 29 – Gráfico correspondente à avaliação da facilidade de aprendizagem ..... 62
Figura 30 – Gráfico correspondente à avaliação da facilidade de memorização ..... 63
Figura 31 – Diagrama de classes da aplicação coolplaces mobile ..... 73
Figura 32 – Diagrama de caso de uso do módulo de exploração de POI's *AR Places* ..... 74
Figura 33 – Diagrama de caso de uso do módulo de localização de amigos *AR Friends* ..... 74
Figura 34 – Diagrama de classes que representa a relação das classes *NearFriendInfo*, *UserCheckInInfo* e *UserEssentialInfo* ..... 75

# Lista de Tabelas

Tabela 1 — Análise comparativa dos desafios dos três tipos de aplicações ..... 25
Tabela 2 — Análise comparativa das soluções ..... 42
Tabela 3 — Avaliação dos parâmetros de usabilidade ..... 60

# Acrónimos e Símbolos

## Lista de Acrónimos

| | |
|---|---|
| **API** | *Application programming interface* |
| **AR** | *Augmented Reality* |
| **BTS** | *Base Transceiver Station* |
| **CSS** | *Cascading Style Sheets* |
| **GIS** | *Geographic information system* |
| **GPS** | *Global Positioning System* |
| **GSM** | *Global System for Mobile Communications* |
| **HMD** | *Head-mounted display* |
| **HTML** | *HyperText Markup Language* |
| **HTTP** | *Hypertext Transfer Protocol* |
| **JSON** | *JavaScript Object Notation* |
| **LBS** | *Location-based service* |
| **POI** | *Point of interest* |
| **SDK** | *Software development kit* |
| **SURF** | *Speeded Up Robust Features* |
| **WCF** | *Windows Communication Foundation* |
| **XML** | *Extensible Markup Language* |

# 1 Introdução

## 1.1 Contexto e Motivação

No ano de 2011 desenvolvi em conjunto com dois amigos, também alunos do ISEP, um site dedicado à pesquisa e partilha de pontos de interesse (POI's) denominado *coolplaces*. Embora esteja atualmente a atravessar uma fase de reestruturação, entre algumas das funcionalidades disponíveis estão a possibilidade de criar listas de POI's favoritos, bem como de gerir listas de amigos, classificar um determinado POI e inserir comentários sobre o mesmo. Torna-se assim possível combinar esta informação de forma a obter referências sobre um determinado POI, quer através de testemunhos em forma de texto submetidos pelo nosso círculo de amigos, quer pelo simples fato de sabermos que determinado POI consta na lista de favoritos de amigos cuja opinião valorizamos.

Posteriormente, no decorrer do Mestrado e no âmbito da disciplina de Sistemas Móveis, desenvolvi novamente em conjunto com os mesmos dois amigos uma aplicação LBS, fazendo uso da base de dados do *coolplaces* à qual chamamos de *coolplaces mobile*. Esta aplicação permite ao utilizador aceder em qualquer local a informação relativa aos POI's e sua posição geográfica, sem esquecer a respetiva informação referente à opinião do grupo de amigos do utilizador.

É então que no decorrer de uma outra disciplina do Mestrado surge a temática da Realidade Aumentada, uma tecnologia capaz de integrar imagens virtuais em visualizações do mundo real. Com esta tecnologia torna-se possível conceber aplicações móveis que permitam ao utilizador navegar no mundo real, ao mesmo tempo que obtêm informações sobre os POI's que está a visualizar através do dispositivo móvel. No entanto, a informação disponibilizada pelas aplicações de Realidade Aumentada existentes no mercado, que permitem a visualização de POI's, por vezes carece de determinada informação de contexto no que diz respeito à opinião dos seus utentes e, nos casos em que esta informação existe, tem origem em opiniões anónimas cuja autenticidade e validade não podemos aferir.

Considerando, que num contexto de pesquisa de POI's, estes são tipicamente desconhecidos do utilizador, rapidamente nos apercebemos da mais-valia que essa informação acima mencionada pode representar. Numa era cada vez mais social, partilhamos quase tudo com a nossa rede de amigos, no entanto quanto mais informação nos é disponibilizada mais difícil se torna filtrá-la de forma a tomarmos decisões. Porém, temos tendência a valorizar a informação quando ela provém de determinados amigos em quem temos mais confiança. Essa informação, quando associada a POI's, pode oferecer o referido contexto.

Tendo em conta o potencial da Realidade Aumentada enquanto visualizador de conteúdos, surgiu o interesse de adicionar valor acrescentado à aplicação *coolplaces mobile*. A motivação na base desta tese está assim na possibilidade de contribuir com uma nova perspetiva, capaz de um acréscimo de qualidade à típica aplicação de pesquisa e visualização de POI's, dotando-os de informação de contexto obtida através da rede *coolplaces*.

## 1.2 Objetivos e Contribuições

O objetivo principal desta tese é o desenvolvimento de um módulo de Realidade Aumentada para acrescentar à já referida aplicação *coolplaces mobile*. Este módulo de navegação e pesquisa deverá ser capaz de apresentar os POI's com a respetiva informação, em sobreposição a uma vista do mundo real, fazendo ainda uso da informação de contexto (amigos, comentários sobre os POI's, *checkins* [1], etc.) que o *coolplaces* contém, proporcionando ao utilizador uma experiência mais imersiva de pesquisa de POI's.

Para o desenvolvimento de tal módulo torna-se necessário efetuar um levantamento dos desenvolvimentos, das tecnologias utilizadas e das aplicações existentes no mercado, não só na área da Realidade Aumentada mas também na área das aplicações LBS, servindo assim também esta tese como um documento do atual estado de desenvolvimento destas duas tecnologias.

## 1.3 Estrutura

Este documento divide-se em três grandes partes: Introdução, Corpo da Tese e Conclusão.

Na Introdução, como o próprio nome indica, é indicado um breve levantamento do estado da arte nos domínios a que a tese se dedica, assim como uma contextualização do âmbito da tese e um esclarecimento de em que medida o que foi feito contribui para o seu progresso.

O Corpo da Tese é composto pelos capítulos 2, 3, 4 e 5 e inclui o levantamento completo do estado da arte, a análise lógica e funcional de algumas aplicações existentes no mercado, o

[1] Plural de checkin, o ato de um utilizador sinalizar a sua presença num local físico através de ação sobre uma representação virtual do mesmo (página web, elemento multimédia, etc…). A informação de um checkin permite identificar o local onde foi efetuado e a data em que tal ocorreu.

levantamento das soluções identificadas e respetiva análise, a justificação e o processo utilizado na escolha da solução implementada e por fim detalhes técnicos e estrutura lógica da implementação da solução escolhida, problemas encontrados e respetivas resoluções.

Por último, na conclusão é efetuada uma análise e discussão dos resultados obtidos, bem como uma reflexão sobre os objetivos que foram propostos e o resultado final alcançado, assim como eventuais aspetos a melhorar, bem como perspetivas de desenvolvimentos futuros das tecnologias envolvidas.

## 1.4 Síntese

Existe atualmente uma grande oferta de aplicações LBS, mas tendo em conta apenas as que se especializam na pesquisa e visualização de POI's através de Realidade Aumentada, denominadas aplicações *AR (Augmented Reality) Browser,* as mais populares são: *Wikitude* (www.wikitude.com), *Layar* (www.layar.com) e *Junaio* (www.junaio.com) (J. Grubert, 2011). Tipicamente estas aplicações possuem um módulo de exploração, permitindo visualizar e aceder a informação referente aos POI's (contactos, horários, preços, etc.) bem como elementos multimédia (imagens e vídeos, entre outros).

No entanto, o *feedback* dos seus clientes, quando tal é aplicável, nem sempre se encontra disponível e quando é possível de obter é proveniente de utilizadores anónimos, o que poderá levar a uma eventual desvalorização desse mesmo *feedback*, uma vez que existe tendência a valorizar mais as informações quando sabemos qual a sua proveniência e conhecemos os gostos de quem forneceu essa mesma informação.

O trabalho desenvolvido no decorrer desta tese foi no sentido de dotar de significado esse mesmo *feedback*, utilizando para tal uma rede de informação baseada no serviço *coolplaces*. Deste modo o utilizador terá ao seu dispor toda uma informação de contexto, cuja proveniência poderá julgar de acordo com a sua rede de amigos, de modo a ter disponível mais e melhor informação aquando da pesquisa por determinado POI.

Procurou-se realizar um levantamento dos problemas mais comuns deste tipo de aplicações em particular, assim como de aplicações LBS em geral, tendo-se verificado que as próprias aplicações existentes no mercado disponibilizam API's (*Application programming interface*) e SDK's (*Software development kit*) que já implementam algumas das soluções possíveis para esses mesmos problemas, podendo ser utilizadas no módulo a desenvolver. Sendo assim, além do levantamento do estado da arte das tecnologias abordadas nesta tese, foi ainda efetuada uma primeira fase de análise das soluções disponibilizadas com respetiva criação de protótipos do novo módulo a desenvolver, para posteriormente ser escolhido um com vista ao desenvolvimento final.

# 2 Serviços baseados na localização e Realidade Aumentada em dispositivos móveis

Neste capítulo vamos abordar a temática da geolocalização e da Realidade Aumentada em dispositivos móveis, introduzindo os conceitos-chave destas tecnologias, algumas das suas aplicações práticas, os desafios que ambas enfrentam, assim como possíveis soluções, tendo sempre por foco desenvolvimentos e aplicações relacionados com a identificação e visualização de POI's.

Um conceito muito abordado neste capítulo é o conceito de *tracking*, pelo que importa definilo neste contexto. Assim sendo, por *tracking* entendesse o ato de um dispositivo móvel responder a mudanças de localização e orientação, perpetradas pelo utilizador, de modo a que os indicadores visuais referentes aos POI's sejam corretamente sobrepostos numa vista real. O objetivo deste processo é assim, o de garantir que os indicadores visuais virtuais sejam integrados no ambiente envolvente de modo a que pareçam fazer parte deste. Em alguma literatura este conceito é acompanhado do conceito de *registration*, sendo que o *tracking* diz respeito à identificação dos objetos do mundo real e à resposta às mudanças de localização do utilizador, enquanto a *registration* é responsável pelo correto alinhamento entre a informação real e virtual (Henrysson, 2007).

De seguida vamos aprofundar ligeiramente o conceito dispositivo móvel, apresentando as suas características e dando exemplos concretos desses dispositivos.

## 2.1 Dispositivos móveis

### 2.1.1 Introdução

Os dispositivos móveis são equipamentos computacionais, tipicamente pequenos, chegando a caber num bolso e que por norma podem ser manuseados com o recurso de apenas uma mão, que normalmente possuem um ecrã de reduzidas dimensões. Estes dispositivos possuem um sistema operativo podendo executar vários tipos de aplicações, sendo que a maioria pode estar equipada com sensores GPS (G*lobal Positioning System*), capazes de determinar a posição do dispositivo em qualquer lugar da Terra, e capacidade para se conectarem à internet. São ainda normalmente capazes de reproduzir áudio e vídeo bem como de permitir a visualização de imagens. Exemplos destes dispositivos são os *smartphones*, os *tablets*, as consolas portáteis de jogos e o relógio-calculadora, entre outros.

Os *tablets* são dispositivos móveis que possuem um ecrã tátil relativamente maior em comparação com os *smartphones* e que estão equipados com câmara, microfone e acelerómetro, capaz de medir a aceleração do dispositivo, isto é, capaz de detetar movimentos do mesmo. A sua imagem de marca é a interação do utilizador com o dispositivo que é feito por norma através da utilização de gestos e movimentos com os dedos sobre o ecrã tátil (Magazine, n. d.), tendo-se tornado populares a partir do ano 2010.

Relativamente aos *smartphones*, são um tipo de dispositivo móvel que reúne características além do típico telefone (Scoop, n. d.) que basicamente só servia para efetuar chamadas. Alvo de um grande desenvolvimento nos últimos anos estão atualmente dotados de acesso à internet, ecrãs tácteis de alta resolução, câmaras, além de vários sensores tais como: os já mencionados GPS e acelerómetro, giroscópio, capaz de identificar a rotação do dispositivo, entre outros. Em suma, deixaram de ser telemóveis que apenas permitiam efetuar chamadas para se tornarem plataformas ideais para serviços orientados ao consumidor. Isto porque estão disponíveis em larga escala e são mais baratos e mais práticos do que outros dispositivos com algumas dessas características, como é o caso dos HMD (*Head-mounted Display*), semelhantes a óculos, mas com lentes especiais sob a forma de ecrã capazes de permitir a visualização de conteúdos digitais. (S. Gammeter, 2010).

A aplicação *coolplaces mobile* foi desenvolvida para *smartphones* que utilizem o sistema operativo *Android*, por isso toda a pesquisa e todo o trabalho apresentado tem por base essa mesma plataforma. De seguida vamos efetuar uma breve apresentação de alguns sistemas operativos com particular destaque para o *Android* pelos motivos já mencionados.

### 2.1.2 *Android*

O *Android* é um sistema operativo *Open Source*[2], desenvolvido pela *Google* (www.google.com), especificamente para ser utilizado em dispositivos móveis como *smartphones* e *tablets*, para que os programadores pudessem tirar total partido das capacidades destes dispositivos móveis. O sistema é facilmente adaptável às características do dispositivo onde se encontra instalado, suportando várias resoluções de ecrã e compatibilidade com gráficos bidimensionais, tridimensionais, para dar apenas um exemplo. As aplicações desenvolvidas neste sistema são capazes de utilizar as funções base do dispositivo como efetuar chamadas, enviar mensagens de texto ou utilizar a câmara por exemplo, o que permite aos programadores desenvolverem aplicações mais ricas e atrativas (Alliance, n. d.).

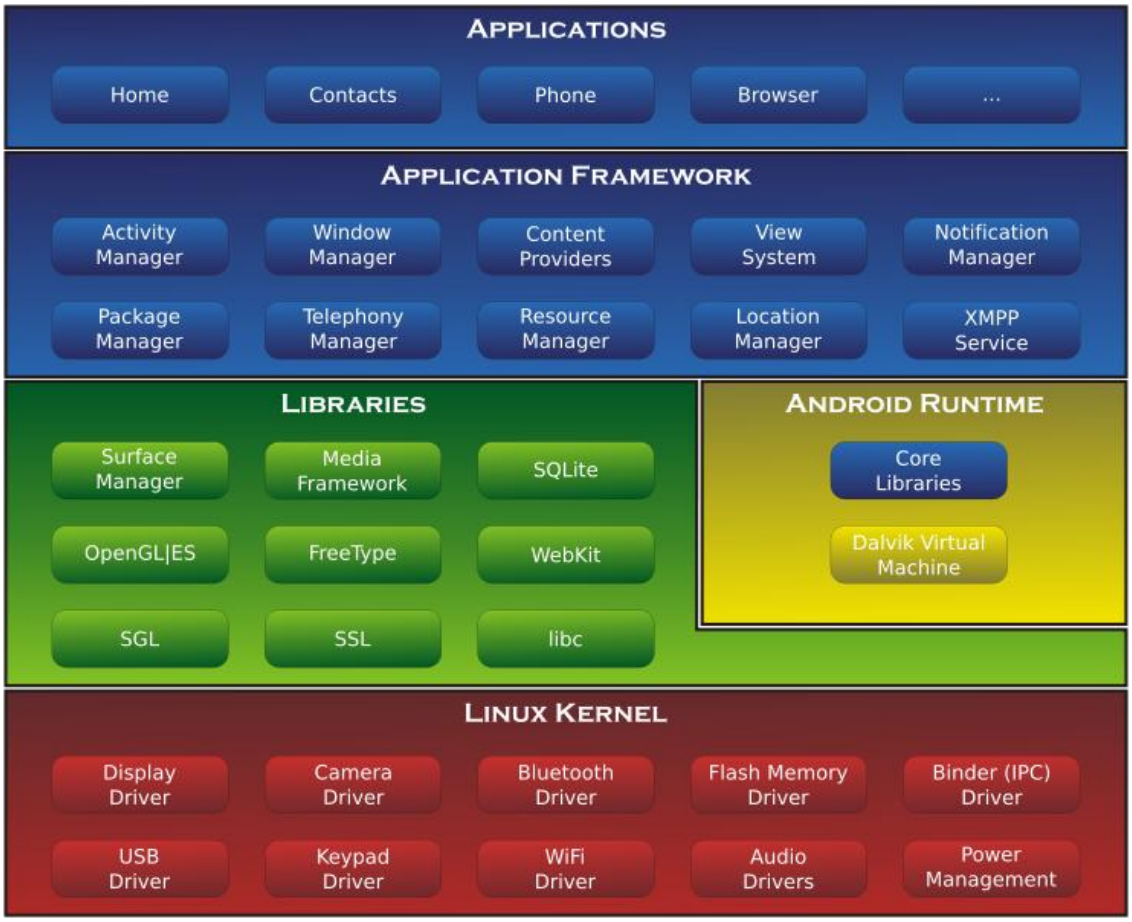


Figura 1 – Diagrama de arquitetura do sistema operativo *Android*

O *Android* é baseado num *Kernel* de *Linux*, responsável pela ligação com o *hardware* e pela gestão de memória do dispositivo. Assente no *Kernel* do *Linux*, está uma máquina virtual (*Dalvik Virtual Machine*) e um conjunto de bibliotecas com as funcionalidades base do *Android* (Bray, 2010). A máquina virtual converte o código *Java* em que são escritas as aplicações em

[2]*Software* cujo código é disponibilizado gratuitamente para quem o quiser estudar, alterar e distribuir.

código máquina, o que permite que qualquer aplicação compilada por esta máquina execute em qualquer dispositivo com o sistema operativo *Android* instalado, independentemente do processador que este utilize. Por fim, conforme esquematizado na Figura 1, existe uma camada denominada A*pplication Framework*, que disponibiliza uma série de recursos à camada de aplicações (Bray, 2010).

Uma das particularidades dos dispositivos *Android* é a gestão de memória, que foi concebida para garantir baixos consumos de energia, uma vez que este é um ponto bastante importante num dispositivo móvel. Uma aplicação que não esteja a funcionar é suspensa pelo sistema e, embora tecnicamente aberta, deixa de consumir recursos de bateria e de processamento até que tal seja de novo necessário. Ainda ao nível da gestão de memória, se os recursos de processamento disponíveis estiverem num nível baixo, o sistema começa a descartar aplicações e processos numa ordem inversa à última data de utilização.

A *Google* continua a contribuir para o desenvolvimento e aperfeiçoamento desde sistema operativo, lançando atualizações regulares. Inclusivamente disponibiliza um ambiente de desenvolvimento para os sistemas operativos *Windows*, *Linux* e *OS X*.

### 2.1.3 *iPhone OS*

O *iPhone OS*, ou *iOS*, é um sistema operativo desenvolvido pela *Apple Inc*. (http://apple.com/) para a sua linha de smartphones denominada *iPhone*. Lançado pela primeira vez em 2007 foi concebido apenas para utilização em dispositivos de *hardware Apple*. Estima-se que tenha cerca de vinte e um por cento da quota do mercado de *smartphones*, estando apenas atrás do *Android* (IDC, 2013). Baseado no sistema operativo *OS X*, o sistema operativo dos computadores desenvolvidos e distribuídos pela *Apple*, este sistema está organizado em quatro camadas: Core OS, o núcleo do sistema operativo; *Core Services*, conjunto de API's de serviços; *Media*, camada que disponibiliza uma série de recursos, nomeadamente de multimédia; *Cocoa Touch*, camada que contém a lógica da interface com o utilizador.

### 2.1.4 *Windows Phone*

Sistema operativo para *smartphones* desenvolvido pela *Microsoft* (www.microsoft.com/) Lançado em 2010, foi o primeiro produto da Microsoft a utilizar uma nova interface gráfica de nome Metro. Esta interface apresenta um *design* inovador em que o ecrã é composto por uma série de mosaicos que o utilizador pode rearranjar livremente e representam atalhos para as aplicações e recursos do dispositivo. Uma das características deste sistema é a capacidade de reconhecer mais do que um ponto de contacto em simultâneo com o ecrã. Está ainda configurado para combinar conteúdos locais com conteúdos *online*, apresentando dados armazenados localmente com dados armazenados nas contas das redes sociais do utilizador e é capaz de reproduzir vários formatos de áudio e vídeo.

### 2.1.5 *Symbian*

*Symbian* é um sistema operativo desenvolvido para *smartphones* desenvolvido originalmente pela *Symbian Ltd*., empresa que foi posteriormente adquirida pela *Nokia* (www.nokia.com/), estando atualmente a cargo da Accenture (www.accenture.com/). Lançado em 2002 chegou a ser o sistema operativo de mais utilizado no mercado até que em finais de 2010 foi destronado pelo *Android*, tendo sido o sistema de eleição da maioria dos dispositivos *Nokia* durante esse período.

### 2.1.6 *BlackBerry OS*

Sistema operativo para dispositivos móveis desenvolvido pela *BlackBerry* (http://blackberry.com/) para a sua linha de dispositivos de igual nome. Além de ser capaz de efetuar várias tarefas em simultâneo (*multitasking*), oferece uma forte componente de integração de contas de email empresariais, permitindo sincronização com uma série de programas de gestão de emails, tarefas e contactos.

## 2.2 Serviços baseados na localização

### 2.2.1 Introdução

Os desenvolvimentos nas comunicações móveis e sem fios, aliados a dispositivos cada vez mais baratos, permitem atualmente aos utilizadores interagir e comunicar praticamente em qualquer lugar e em qualquer altura. Como consequência, registou-se um aumento da informação sobre a localização das pessoas à medida que utilizam estes dispositivos, fomentando o desenvolvimento de inúmeras aplicações que fazem uso da localização do utilizador e do contexto que o rodeia.

Segundo Varshney (2011) os LBS são serviços que disponibilizam ao utilizador informação com base na sua localização e preferências. Graças a este serviço personalizado, as aplicações LBS foram denominadas de aplicações rainhas do negócio das aplicações móveis (Watson, 2008). Ainda segundo Kaplan (2011) estes serviços permitem ao utilizador receber informação acerca do meio envolvente, economizando tempo e dinheiro e permitindo efetuar escolhas melhores e mais bem informadas. Segundo H. Xu (2010), existem dois mecanismos de entrega de informação utilizados nas aplicações LBS, *pull* ou *push*. As aplicações do tipo *pull* são as que são iniciadas por iniciativa do utilizador, ou seja, o utilizador efetua um pedido específico dando a conhecer a sua localização por iniciativa própria. As do tipo *push* são ativadas por iniciativa dos fornecedores de conteúdo, enviando informação para o utilizador com o seu consentimento (subscrição prévia) ou mesmo sem consentimento algum (sem subscrição).

Existe uma grande variedade de aplicações LBS, mas recentemente tem-se registado um aumento do interesse em aplicações comerciais como o *Google Maps* (http://maps.google.com/) e o *Yelp* (www.yelp.com/), e em serviços de partilha como o *Google Latitude* (recentemente descontinuado), *Gowalla* (http://blog.gowalla.com/) e o *Foursquare* (https://foursquare.com/). Como consequência desse interesse, alguns destes serviços têm registado um rápido aumento da sua base de utilizadores (*Foursquare* mais de um ponto sete milhões de utilizadores, *Gowalla* mais de duzentos e cinquenta mil utilizadores) (Mattias Rost, n. d.). Estas aplicações fornecem dois tipos de informação: informação geral sobre POI's e informação especializada sobre uma determinada área. Mais recentemente tem surgido um modelo de negócio baseado em anúncios baseados na localização que consiste na apresentação de anúncios em determinadas aplicações do dispositivo de acordo com a localização do mesmo. Estes anúncios promovem assim serviços ou POI's comerciais nas imediações da localização do utilizador.

No contexto desta tese, o foco da análise recai sobre informação geral de POI's, uma vez que é esse o tipo de informação que a aplicação *coolplaces mobile* disponibiliza.

As aplicações LBS estão estruturadas de acordo com o modelo de comunicação apresentado na Figura 2, que se encontra dividido em três camadas: *positioning layer*, *middleware layer* e *application layer* (Voisard, 2004).

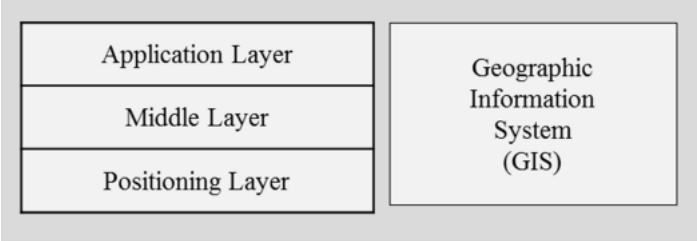


Figura 2 – Modelo de comunicação LBS (Chin, 2012)

A primeira camada é responsável pelo cálculo da localização do dispositivo em conjunto com dados fornecidos por um sistema de informação geográfica (GIS[3]), a segunda é responsável pela integração e comunicação com a terceira camada, que representa a aplicação propriamente dita, ou seja, a camada com a qual o utilizador interage, sendo responsável pela apresentação dos conteúdos.

A localização do dispositivo pode ser representada de duas formas: localização no espaço e descrição textual. A localização espacial é normalmente representada num sistema de coordenadas latitude-longitude-altitude, em que a latitude varia entre zero e noventa graus a Norte ou Sul do equador, a longitude pode variar entre zero e cento e oitenta graus a Este ou Oeste do meridiano de *Greenwich*, e por fim a altitude que representa os metros acima do nível do mar. Quanto à descrição textual, esta é normalmente representada por um endereço que pode conter o nome da rua, cidade e código postal (Shukla, 2012).

[3] Sistema de hardware e software capaz de captar e analisar dados geográficos.

No que diz respeito ao modo como a localização pode ser obtida os dois processos possíveis são a localização através da rede móvel de comunicações, em que o dispositivo é capaz de detetar qual a estação de transmissão (BTS - *Base Transceiver Station*) que está a utilizar e a sua respetiva localização e a localização através de satélite, em que o sistema GPS utiliza vinte satélites em órbita que calculam o tempo que os sinais enviados demoram a chegar ao dispositivo podendo assim triangular a sua posição, sendo que normalmente basta que o dispositivo esteja ao alcance de quatro satélites para que seja obtido um resultado (Shukla, 2012). A localização através da rede é o método mais básico e barato, no entanto, como as células GSM (*Global System for Mobile Communications*) podem ter entre dois a vinte quilómetros de diâmetro, a sua precisão é inferior à da localização por GPS (Shukla, 2012).

Uma vez que estes serviços se baseiam no utilizador, para que estes se tornem um negócio rentável, sustentável e gerador de lucro é necessário aumentar o número de utilizadores com uma participação ativa constante. Para tal acontecer há no entanto um conjunto de desafios que necessitam de ser endereçados.

### 2.2.2 Desafios

Um primeiro grande desafio que estas aplicações enfrentam está relacionado com os custos das transferências de dados. A comunicação entre os dispositivos e os fornecedores de serviços através de serviços de dados sem fios pode acarretar grandes custos que o utilizador comum não está disposto a pagar. Tal fato pode impedir um utilizador casual de se tornar num utilizador mais assíduo deste tipo de serviços.

Outro grande desafio que estas aplicações encontram está diretamente relacionado com a sua grande mais-valia. De forma a poderem providenciar informação de acordo com as suas preferências de localização, precisam de ter acesso a essa mesma informação. Tal pode levar os utilizadores a sentirem-se permanentemente vigiados e a sentirem a sua privacidade ameaçada, o que pode afetar negativamente a utilização destes serviços (Zhou, 2012). Ainda segundo Zhou (2012), existem ainda alguns receios em torno das questões de privacidade que podem impedir uma maior adesão do público em geral a este tipo de aplicações.

Atualmente existem já diversos algoritmos de geolocalização que funcionam consoante as particularidades da rede móvel. No entanto, outro dos principais problemas passa precisamente pela dificuldade em encontrar uma solução geral que funcione independentemente do meio onde o utilizador se encontra, ou seja, uma solução que funcione quer este se encontre dentro de um edifício, no exterior, em meios rurais ou meios urbanos.

### 2.2.3 Soluções propostas

O problema dos custos das transferências de dados está neste momento nas mãos dos operadores de rede, enquanto não baixarem os custos para o utilizador destas transferências

de dados ou enquanto não apresentarem planos de dados mais em conta, esta questão será sempre um potencial entrave a uma maior implementação destes serviços.

Em relação às questões de privacidade, os provedores de serviços LBS precisam de adotar medidas que dissipem esses mesmos receios por parte dos utilizadores (Zhou, 2012). Algumas das medidas sugeridas são a criação e divulgação de políticas de privacidade, que informem aos utilizadores em que medida a informação recolhida pode ser utilizada e armazenada e a utilização de técnicas de encriptação de forma a assegurar que os dados não caiam em mãos de terceiros.

Quanto ao problema da determinação da localização do dispositivo, a solução parece passar pela utilização dos vários sensores disponíveis nos dispositivos atuais de forma a melhorar a capacidade de geolocalização dos mesmos (Marco Anisetti, 2011). Existem várias soluções propostas, desde a utilização de bases de dados de POI's fazendo uso de atributos como as respetivas coordenadas GPS e fotos dos mesmos, à utilização do acelerómetro para detetar pequenas movimentações dentro de uma determinada área (Marco Anisetti, 2011). A utilização de fotos de determinados edifícios como base de comparação é uma solução que tem sido testada. Consiste em utilizar a câmara do dispositivo móvel para obter imagens do meio em redor do utilizador, utilizando técnicas de comparações visuais para as comparar com imagens obtidas de uma base de dados de edifícios diferenciadores (públicos ou históricos, entre outros) para assim determinar a localização do dispositivo. Esta solução, no entanto, apresenta algumas desvantagens como os custos das transferências de dados (envio das imagens para um servidor externo) e gastos de energia provocados pela utilização da câmara de forma recorrente (Marco Anisetti, 2011).

## 2.3 Realidade aumentada

### 2.3.1 Introdução

Realidade aumentada é uma tecnologia que permite sobrepor imagens virtuais numa visualização do mundo real (Azuma, 1997), complementando a realidade e proporcionando ao utilizador uma experiencia visual enriquecida. É uma variação do conceito de Realidade Virtual, com a diferença de que nas tecnologias de Realidade Virtual o utilizador é inserido totalmente num ambiente sintético, enquanto que nas tecnologias de Realidade Aumentada este é capaz de visualizar o mundo real sendo os objetos virtuais colocados sobre o mesmo, ou seja, a Realidade Aumentada permite complementar o mundo real ao invés de o substituir (Azuma, 1997). Podemos considerar que a Realidade Aumentada potencia a perceção e a interação do utilizador com o mundo real, apresentando informação capaz de o ajudar a executar determinadas tarefas. Segundo Brooks (1996) a Realidade Aumentada é um exemplo de *Intelligence Amplification*: utilizar um dispositivo computacional como uma ferramenta que torne uma tarefa mais fácil de executar para um humano.

A temática da Realidade Aumentada tem sido objeto de pesquisa e de desenvolvimentos nas últimas duas décadas (J. Grubert, 2011), tendo as primeiras aplicações desta tecnologia surgido associadas ao dispositivo HMD, que apesar de funcionar numa lógica de mãos livres, bastando o utilizador olhar para o objeto desejado, não alcançou sucesso comercial. Mais recentemente, com a rápida evolução dos *tablets* e dos *smartphones*, estes tornaram-se uma alternativa bastante popular, sendo que os ecrãs de alta resolução e as câmaras de alta definição que os compõem se revelam aliciantes para o desenvolvimento de aplicações de Realidade Aumentada (Schmalstieg, 2012). Apesar de algumas limitações de *hardware* e o facto de estes dispositivos estarem otimizados para baixos consumos de energia em detrimento de alta performance levantarem alguns desafios, existem atualmente importantes aplicações desta tecnologia em videojogos, marketing interativo e publicidade, entre outras áreas (Schmalstieg, 2012).

Segundo J. Grubert (2011), o utilizador principal de aplicações de Realidade Aumentada, não é ainda o utilizador comum, mas faz parte de um grupo de pessoas que revela grande interesse por tecnologia e normalmente adere cedo às novas tecnologias que surgem no mercado. Apesar destes mesmos utilizadores compreenderem o potencial que estas aplicações aparentam, a opinião geral é de que a performance atual ainda não vai ao encontro das suas espectativas, sendo que imprecisões no processo de *tracking* são um dos fatores mais apontados como responsáveis pela não concretização desse potencial.

### 2.3.2 Desafios

O processo típico de uma aplicação de Realidade Aumentada é composto por cinco etapas: a construção de um mundo virtual com um sistema de coordenadas idêntico ao mundo real, a determinação da posição e da orientação do utilizador, o ajuste da câmara no mundo virtual de acordo com essa posição e orientação, a visualização do mundo real através do ecrã do dispositivo e, por fim, a combinação dos objetos virtuais com a vista do mundo real (Thomas, n. d.).

O primeiro desafio é o de determinar a posição da câmara em relação aos objetos do meio envolvente, incluindo mudanças de localização e orientação do dispositivo. Os três principais métodos para determinar esta posição são: através dos sensores do dispositivo, através da utilização de marcadores ou através da análise de características identificadoras do objeto (utilizando métodos computacionais de análise de imagens). No âmbito desta tese a utilização de marcadores não é passível de ser utilizada, uma vez que tal requer uma marcação prévia dos objetos, através de códigos QR[4] por exemplo, ou através da utilização de padrões visuais, sendo tal processo normalmente aplicado em imagens, cartazes, etc. Por outro lado, os sensores disponíveis nos *smartphones* atuais podem ser utilizados para efetuar os vários cálculos necessários, mas a sua precisão nem sempre é a ideal, uma vez que estes dispositivos

[4] Código de barras bidimensional capaz de armazenar informação, cuja leitura pode ser efetuada por um dispositivo móvel.

não possuem antenas dedicadas, podendo existir interferências de outros sinais principalmente em meios urbanos. Finalmente, os métodos de análise de características identificadoras do objeto são computacionalmente intensivos e podem estar além das capacidades dos dispositivos atuais e qualquer solução centralizada que retire ao dispositivo o peso desse processamento envolve grandes transferências de dados, o que não é desejável tendo em conta o custo dessas mesmas transferências (Z. Zhou, 2009).

O segundo desafio tem a ver com a visualização e passa por garantir uma integração transparente e natural entre as imagens reais e as imagens virtuais. Envolve colocar e orientar corretamente os objetos na cena, o que requer um modelo geométrico da mesma. A precisão deste processo é extremamente importante, na medida em que a mais-valia de uma experiência de Realidade Aumentada está na qualidade da integração e da visualização da informação apresentada ao utilizador, sendo que se tal precisão não for alcançada, a experiencia deteriora-se rapidamente (J. B. Gotow, n. d.).

Para além de criar uma cena realista, é também importante determinar o que mostrar num dado momento. Numa aplicação que forneça informação relativa a objetos da cena, o ecrã pode ficar sobrecarregado com demasiada informação, e daí a importância de os dados serem posicionados de maneira a não ocultarem objetos reais, nem se sobreporem uns aos outros.

Segundo J. Grubert (2011), os dois primeiros desafios são um fator que compromete significativamente uma utilização continuada deste tipo de aplicações por parte dos utilizadores, o que mais reforça a importância da sua resolução.

### 2.3.3 Soluções propostas

Uma possível solução para o primeiro desafio passa por utilizar métodos computacionais de análise de imagens. Um exemplo desses métodos é o *nature feature tracking*, que consiste na recolha e comparação de determinadas características de uma imagem, com uma base de dados dessas mesmas características e a respetiva localização no meio envolvente. Este processo divide-se em duas fases, em primeiro lugar é construído um modelo tridimensional da cena com representação das respetivas características identificadores, tendo posteriormente lugar uma segunda fase de identificação das mesmas no vídeo captado pela câmara do dispositivo. Para tal é necessário que a imagem utilizada seja adequada (com alto contraste) e que capte características únicas sem no entanto ser demasiado complexa, pois como vimos no ponto anterior estes métodos tendem a ser computacionalmente pesados, o que pode inviabilizar a sua utilização. Alguns algoritmos, um dos quais o algoritmo SURF (*Speeded Up Robust Features*), podem ser utilizados para comparação e reconhecimento de imagens, sendo a rapidez a sua principal característica (Herbert Bay, n. d.). Uma segunda opção passa por utilizar uma abordagem mista, utilizando os sensores do dispositivo e os métodos anteriormente referidos para afinar a pesquisa (Schmalstieg, 2012). Ou seja, a solução passaria por utilizar os sensores do dispositivo para obter uma localização, ainda que

parcial e com um eventual desvio, que no entanto seria utilizada para limitar os registos de imagens que seriam utilizados na comparação, tornando o processo mais rápido e eficaz.

Relativamente à visualização, tendo a posição calculada e a câmara em funcionamento (está a ser utilizada para apresentar o conteúdo ao utilizador) cada *pixel*[5] da mesma pode ser mapeado num ponto pertencente a uma esfera com centro no próprio dispositivo e com um raio igual ao alcance visual, sendo essa informação utilizada para a colocação dos elementos virtuais sobre a cena. Algumas técnicas de visualização especificamente para Realidade Aumentada serão abordadas em detalhe posteriormente.

Para finalizar, de forma a prevenir um eventual sobrecarregamento do ecrã com informação, uma solução bastante prática passa por oferecer ao utilizador a hipótese de definir um determinado alcance, deixando assim de fora informação que não esteja suficientemente próxima mas que no entanto poderia sobrepor-se a outra mais relevante, ou ainda a implementação de filtros por temas e/ou categorias que igualmente iriam contribuir para a redução de informação a apresentar num dado momento.

### 2.3.4 Considerações gerais

A temática da Realidade Aumentada não é recente, no entanto tem conhecido grandes desenvolvimentos nos últimos anos em parte devido à nova geração de dispositivos móveis e respetivo *hardware*. Assim sendo, é ainda uma tecnologia em evolução, longe de uma fase de maturação, com muitos conceitos por definir e com alguma falta de padrões no que diz respeito à estrutura das aplicações, não existindo um consenso quanto à melhor abordagem para inserir corretamente o utilizador e os elementos virtuais numa cena do mundo real. Inclusivamente, segundo Olivier Hugues (2011) enquanto tecnologia emergente existem cinco barreiras que têm ainda que ser ultrapassadas: desenvolvimentos técnicos, sua interoperabilidade e capacidade de integração; falta de metodologias na análise dos sistemas existentes e das soluções disponíveis; avaliação e demonstração do valor acrescentado das aplicações desenvolvidas; questões de segurança relativas ao comportamento do utilizador enquanto está a utilizar as aplicações, uma vez que o conteúdo virtual pode alheá-lo de determinadas ocorrências; questões de usabilidade relacionadas com a natureza dos interfaces das aplicações.

[5] Elemento mais pequeno de uma imagem. Uma imagem é composta tipicamente por milhares de *pixels*.

## 2.4 Aplicações Ar Browser

### 2.4.1 Introdução

Conforme mencionado no ponto 1.4, as aplicações *Ar Browser* são aplicações LBS especializadas na pesquisa e visualização de POI's, que apresentam o conteúdo com recurso à Realidade Aumentada, pelo que alguns dos desafios que enfrentam são comuns tanto às aplicações LBS, como às aplicações de Realidade Aumentada.

Estas aplicações assemelham-se de certa forma a um *browser* na medida em que também permitem visualizar conteúdos multimédia, com a diferença já assinalada do modo como esse conteúdo é apresentado. Normalmente acedem a recursos externos ao dispositivo através de protocolos e *webservices*[6] e suportam uma grande variedade de formatos multimédia como imagens, áudio, vídeo e modelos tridimensionais, entre outros (J. Grubert, 2011).

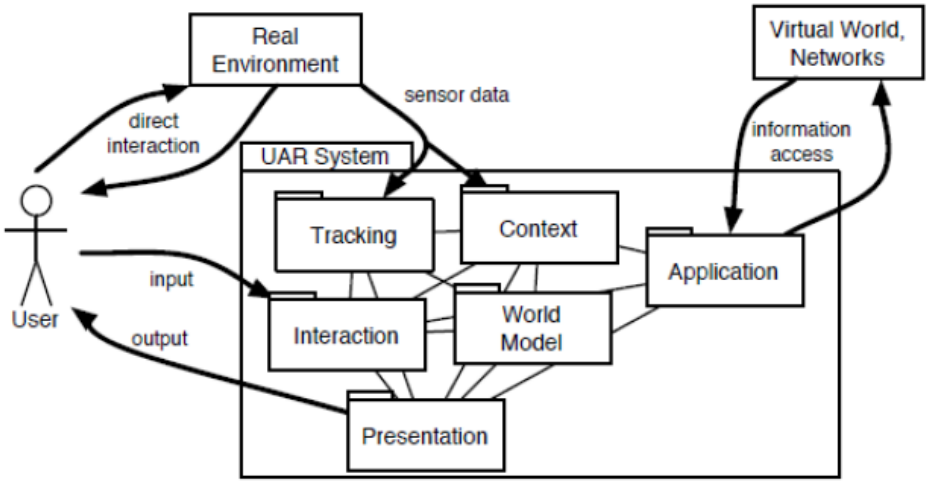


Figura 3 – Modelo lógico de uma aplicação *Ar Browser* (MacWilliams, 2005)

Segundo Ben Butchart (2011), com base no modelo proposto na Figura 3 uma aplicação *Ar Browser* é composta por vários subsistemas: o subsistema *Tracking*, responsável por detetar e responder a alterações na localização e orientação do utilizador, de modo a que os objetos virtuais possam ser sobrepostos de forma convincente na vista do mundo real; o subsistema *Application*, responsável pelo fluxo lógico da mesma, bem como pela coordenação entre os outros subsistemas; o subsistema *World Model*, responsável pelo armazenamento e acesso à representação digital do meio envolvente, incluindo POI's; o subsistema *Presentation*, responsável pela apresentação dos conteúdos, incluindo *streams* de vídeo do meio envolvente; o subsistema *Context*, responsável por disponibilizar informação de contexto referente ao utilizador.

[6] Plural de webservice, um método de comunicação capaz de comunicar entre dois dispositivos computacionais através da *World Wide Web*.

Numa aplicação típica, o utilizador aponta o seu dispositivo para determinados objetos e visualiza a versão "aumentada" no ecrã do dispositivo, ou seja, o meio envolvente do utilizador é enriquecido com informação virtual que coexiste com o mundo real (Azuma, 1997). O grande desafio destas aplicações passa então por combinar o conteúdo virtual com o ambiente real envolvente, o que requer métodos para determinar a localização do utilizador, assim como o seu campo de visão. Uma vez que estas aplicações são utilizadas em ambientes abertos, que dificilmente podem ser preparados para o efeito, utilizam principalmente os dados dos sensores do dispositivo em detrimento da utilização de imagens (marcadores), fato que reduz a precisão do cálculo da posição do dispositivo.

### 2.4.2 Estado da arte

Os *smartphones* atuais, equipados com acesso à Internet, câmaras, sensores de GPS e sensores geomagnéticos, tornaram possível o desenvolvimento de *Ar Browsers* comerciais acessíveis ao público em geral (J. Grubert, 2011). Algumas das principais aplicações existentes no mercado são: *Wikitude*, *Layar, Junaio, Localscope* (www.cynapse.com/localscope), Cyclopedia (www.chemicalwedding.tv/cyclopedia.html), *Tripwolf* (www.tripwolf.com), entre outras (Z. Yovcheva, 2012).

No âmbito destas aplicações, existem vários conteúdos que podem ser apresentados ao utilizador, com destaque para a localização e informação dos POI's nas imediações do utilizador. No entanto ainda não se registou uma adesão do público em geral (Väänänen-Vainio-Mattila, 2011), pelo que as potencialidades deste tipo de aplicações permanecem por explorar na sua plenitude. A arquitetura típica, conforme podemos ver na Figura 4 é composta por 3 partes: um *Ar browser*, ou seja, a aplicação propriamente dita capaz de apresentar o conteúdo virtual integrado numa vista do mundo real, um servidor externo e por fim um repositório de POI's, sendo que o servidor e o repositório podem ou não ser um só.

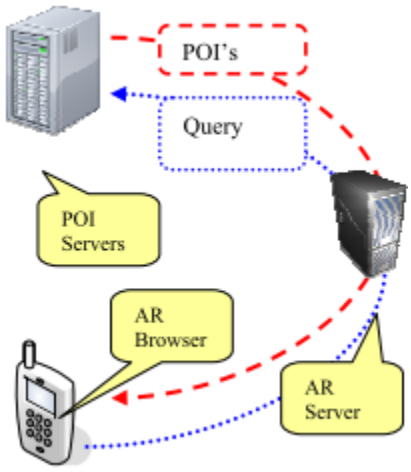


Figura 4 – Arquitetura típica de uma aplicação *AR Browser* (Vinny Reynolds, n. d.)

O *Ar Browser* é o que permite ao utilizador aceder e interagir com os conteúdos virtuais, pois é o componente visível da mesma, pode-se mesmo afirmar que para o utilizador o *browser* é a aplicação, sendo inclusivamente o componente que dá o nome a este tipo de aplicações. O servidor é responsável pela comunicação entre o *browser* e o repositório de POI's, ou seja, é responsável por receber os pedidos do *Ar Browser* sempre que é efetuada uma pesquisa por POI's, redirecionando esses pedidos para o repositório e devolvendo a respetiva resposta deste de volta ao *Ar Browser*. O servidor atua assim como ponte, ou elo de ligação do sistema.

Todas as aplicações mencionadas possuem um módulo de exploração e permitem aceder a conteúdos multimédia através de anotações virtuais interativas (Z. Yovcheva, 2012). No entanto nenhuma das aplicações oferece informação de contexto baseada em feedback de outros utilizadores num contexto social, isto é, opiniões e um sistema de classificação de POI's devidamente enquadradas num contexto social.

Apesar do seu potencial os *AR Browsers* apenas conseguiram obter um por cento do mercado de aplicações móveis, existindo uma grande diferença entre o número de *downloads* destas aplicações e o respetivo número de utilizadores ativos, o que demonstra que a maioria dos utilizadores não usa regularmente este tipo de aplicações. Uma possível explicação é a de que na ansia de dominar o mercado e a respetiva concorrência, as aplicações estão a ser lançadas fazendo uso das tecnologias mais recentes sem passarem por um processo cuidadoso de testes e sem oferecerem o devido suporte aos utilizadores (Tamarjan, 2012). Tal como indicado no ponto 2.3.4 como aplicações de Realidade Aumentada que são, os *AR Browsers* estão numa fase de desenvolvimento e evolução com muito por definir e padronizar. Pode-se mesmo afirmar que existe uma falta de heurísticas e de metáforas bem definidas para este tipo de aplicações o que implica uma série de decisões a nível de *design*. É necessário investigar, desenvolver e proporcionar uma melhor experiência ao utilizador, através de uma gestão mais fluída dos componentes visuais e da informação que lhe é apresentada (Z. Yovcheva, 2012).

### 2.4.3 Desafios

Como referido em 2.3.2, um dos principais problemas das aplicações de Realidade Aumentada é a dificuldade em determinar a posição da câmara em relação aos objetos do meio envolvente que serão apresentados pelo dispositivo. Essa dificuldade reflete-se no ato de determinar o campo de visão/posição do utilizador, o que pode desvirtuar a experiência, uma vez que se o conteúdo virtual não for integrado corretamente na vista do mundo real, isso pode confundir e desorientar o utilizador. Em ambientes com conteúdos previamente marcados (por ex. códigos QR) a sua identificação pode ser utilizada para determinar a posição do utilizador mais facilmente, sendo por isso uma boa solução para ambientes controlados/fechados. Esta marcação pode conter a informação relativa aos objetos, facilitando assim a sua identificação e o respetivo cálculo da posição relativamente ao dispositivo. No entanto como já foi descrito no ponto 2.4.1 esta solução não é passível de ser utilizada neste contexto. Quanto à utilização dos sensores do dispositivo, apesar de ser uma

alternativa válida levanta um conjunto de desafios, nomeadamente a precisão do sensor GPS, do sensor geomagnético e a pouca fiabilidade dos dados, uma vez que segundo D. Marimon (n. d.) o sensor GPS presente nos dispositivos atuais calcula a posição com um desvio de sensivelmente vinte metros, enquanto o sensor geomagnético só é preciso até cerca de vinte graus, o que para um ponto localizado a cem metros do dispositivo pode representar um desvio aproximadamente de até trinta e sete metros. Atualmente determinar a orientação/posição do utilizador num ambiente livre e não previamente preparado continua a ser um problema complexo sem uma solução ideal (Poelman, 2010).

O segundo desafio, a visualização já abordada no ponto 2.3.2, passa por renderizar os elementos virtuais num ambiente real, isto é, desenhar os elementos sobre a vista do mundo real. Independentemente do método utilizado para identificar a localização do utilizador é necessário ter em conta a precisão dessa mesma renderização, uma vez que a usabilidade da aplicação está diretamente relacionada com tal precisão. Se a apresentação da informação não for a mais correta tal poderá induzir o utilizador em erro, uma vez que o contexto assumido pelo mesmo poderá não ser o correto. Algumas aplicações amenizaram este problema através da utilização de *OpenGl* [7] para uma renderização mais rápida (D. Perritaz, 2009). Também conforme referido no ponto 2.3.2 serão apresentadas técnicas de visualização mas em detalhe posteriormente.

Um terceiro desafio já abordado nos pontos 2.3.2 e 2.3.3 é a quantidade de informação presente num dado momento no ecrã, o que pode limitar a legibilidade da mesma. As aplicações *Junaio* e *Localscope* resolveram este problema permitindo ao utilizador definir os temas que desejam visualizar (Z. Yovcheva, 2012).

Devido à natureza destas aplicações, estas herdam alguns dos problemas das aplicações LBS referidas no ponto 2.2.2. Um desses problemas, o elevado custos das transferências de dados já foi abordado nesse mesmo ponto. Outro desafio também aí abordado diz respeito a eventuais receios de privacidade por parte do utilizador provocados pela necessidade do dispositivo obter a sua localização GPS.

Por fim, numa aplicação de geolocalização, principalmente em zonas com grande densidade de POI's, é necessário ir identificando os que estão próximos do utilizador à medida que este se vai movimentando. O desafio passa então por encontrar um equilíbrio entre o armazenamento dos POI's e a frequência de pedidos à base de dados, assim como os parâmetros a utilizar nesses mesmos pedidos, uma vez que os resultados têm que ser obtidos quase imediatamente à medida que o utilizador se vai deslocando. Tudo isto tem ainda que ter em conta o consumo de energia e o custo para o utilizador destes pedidos à base de dados.

É ainda importante referir que subsistem alguns problemas de alocação de memória e gestão de recursos aquando do manuseamento de *frames*[8] da câmara, assim como problemas de

[7] *OpenGl* é uma API de computação gráfica capaz de efetuar processamento digital de imagem e vídeo.
[8] Plural de *frame*, uma imagem correspondente a um determinado momento de um vídeo. É a junção de todas as *frames* que produz o movimento do vídeo.

sincronização entre a colocação da informação e o cálculo da posição dos objetos (S. Gammeter, 2010).

### 2.4.4 Soluções propostas

Em relação ao primeiro desafio, uma possível solução passa por tentar reduzir o ruído das medições através de algoritmos próprios para o efeito. Uma solução sugerida por (J. B. Gotow, n. d.) e denominada *Compass Filtering*, combina uma aplicação de filtros *Finite Impulse Response* (Rabiner, 1975) com análise estatística dos dados. De seguida apresentamos na Figura 5 o algoritmo mencionado.

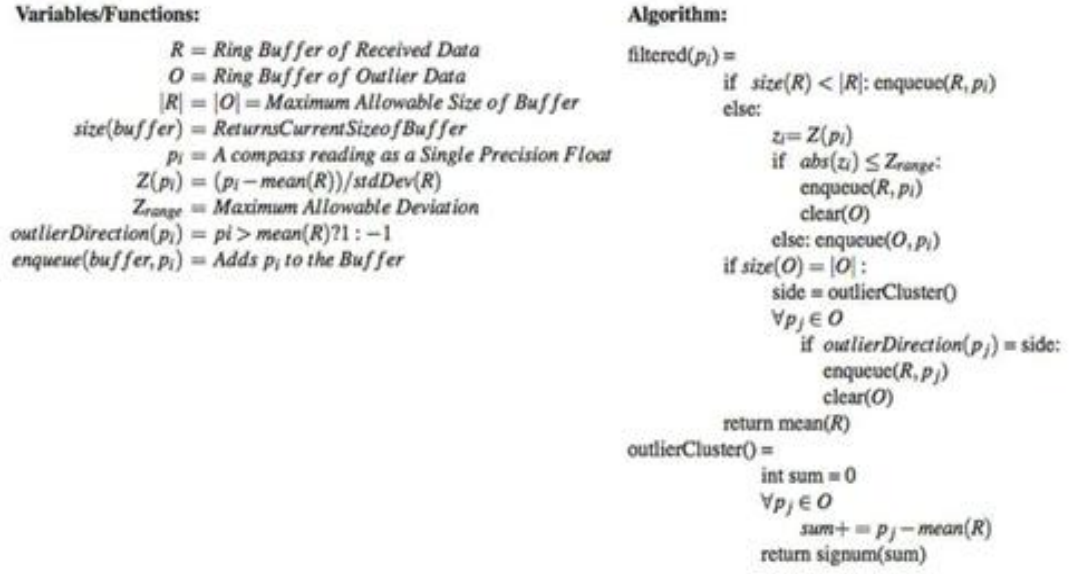


Figura 5 – Algoritmo *Compass Filtering* (J. B. Gotow, n. d.)

Este algoritmo contém dois *buffers*[9] de dados, sendo que um deles (R) aceita todos os pontos numa fase inicial. Assim que a sua capacidade de armazenamento for atingida, por cada novo ponto é calculado o seu desvio padrão (Z), e se este for superior ao limite máximo ($Z_{range}$) os dados são armazenados no segundo *buffer* (O), caso contrário são adicionados ao *buffer* R sendo os dados do *buffer* O removidos. Quando o limite máximo do *buffer* O for atingido é calculada a média e os dados são divididos em dois conjuntos, valores superiores à média e valores inferiores à média. O conjunto com mais valores é adicionado ao *buffer* R e os dados do *buffer* O são removidos. Quando requisitado o valor devolvido corresponde à média do *buffer* R (J. B. Gotow, n. d.). O cálculo da média e do desvio padrão são operações computacionalmente pesadas, no entanto, após os primeiros cálculos é possível minimizar esse custo realizando apenas cálculos parciais e tendo em conta a variação dos valores (J. B. Gotow, n. d.).

[9] *Buffer* de dados é uma região física de armazenamento de memória.

Em relação ao segundo desafio, uma possível solução, passa por dividir cada ponto de acordo com a sua longitude/latitude numa grelha bidimensional. Cada bloco da grelha contém todos os pontos de uma área específica e são carregados da base de dados de acordo com o índice correspondente da grelha. Esta solução dispensa comparações numéricas e permite consultas mais simples à base de dados. Outra opção passa por utilizar uma solução capaz de calcular a posição do utilizador do lado do cliente e ao mesmo tempo efetuar o reconhecimento do lado do servidor baseado em coleções de fotos de monumentos disponíveis. Esta proposta inclui ainda a utilização de um método de comparação de características visuais (baseado no algoritmo SURF), apoiado pela informação recolhida dos sensores do dispositivo. Esta abordagem híbrida permite obter rapidamente os objetos da base de dados, sem no entanto esta ter que estar do lado do cliente, ao mesmo tempo que limita o número de comunicações entre ambos, o que se revela importante devido aos custos de transmissão de dados. A usabilidade deste sistema no entanto depende da quantidade de imagens de POI's presente na base de dados (S. Gammeter, 2010). Uma solução semelhante é apresentada em (D. Marimon, n. d.), em que um sistema híbrido utiliza os dados GPS e uma imagem capturada pelo cliente para determinar a localização do utilizado. Posteriormente essa localização é utilizada de forma a realizar um despiste inicial de modo a descartar imagens de POI's que não estejam localizados dentro de um determinado alcance, contribuindo para um cálculo mais célere.

Para o terceiro desafio, uma vez que já foi abordado anteriormente e o problema se revela semelhante, as soluções que foram apresentadas em 2.2.3, a aplicação de filtros temáticos e a definição de um determinado alcance a partir do qual os POI's não são apresentados, são também possibilidades a ter em conta neste contexto.

## 2.5 Técnicas de visualização de Realidade Aumentada

Conforme referido no ponto 2.3.2 um dos desafios da Realidade Aumentada, e por consequência dos *Ar Browsers*, é a visualização. Esse processo passa por garantir uma integração transparente e natural entre as imagens reais e as imagens virtuais. Segundo Denis Kalkofen (2011) é importante que estas imagens sejam cuidadosamente combinadas ao invés se serem simplesmente sobrepostas uma sobre a outra. Se os elementos virtuais forem gerados independentemente do ambiente real a integração entre os dois pode não ser alcançada com sucesso, uma vez que não forem tidos em conta os objetos reais que serão sobrepostos pelos elementos reais isso pode causar problemas de perceção de profundidade ao utilizador. Para evitar estas imprecisões e gerar visualizações coerentes é necessário ter em conta as próprias limitações dos sistemas de Realidade Aumentada, o fato de não raras vezes estas visualizações terem de ser efetuadas com dados insuficientes, informações incompletas sobre o ambiente real e até dados erróneos ou dessincronizados provenientes de processos de *tracking* incompletos (Denis Kalkofen, 2011). Para melhor perceber as técnicas de visualização propostas para mitigar estes problemas importa explicar como o nosso cérebro perceciona alguns conceitos.

Um desses conceitos é a profundidade. O nosso cérebro é capaz de percecionar uma série de níveis de profundidade diferentes, quando visualizamos múltiplos objetos, os mais afastados de nós aparentam um movimento mais lento que os objetos que estão localizados mais perto, sendo essa diferença de velocidade transformada numa distância aproximada entre os objetos, ou seja, só se os objetos ou o observador de mexer é que os diferentes níveis de profundidade se tornam visíveis. Outro tipo de níveis de profundidade são percecionados quando focamos um único objeto, os músculos dos olhos contraem e relaxam consoante a distancia a que estamos desse objeto. Quanto relaxamos mais esses músculos significa que o objeto está mais longo, pelo contrário, quanto mais os contraímos significa que este estão mais perto. Um terceiro tipo de níveis de profundidade só se manifesta quando os seguintes fenómenos ocorrem: oclusão, quando dois objetos se sobrepõem o que está mais perto do observador "esconde" o que estiver mais afastado; tamanho relativo, objetos mais distantes aparentam ser mais pequenos que os mais próximos; altura relativa, objetos cuja base esteja localizada numa altura superior da imagem aparentam estar mais longe; detalhe, objetos mais próximos oferecem mais detalhes; perceção atmosférica, devido a poerias na atmosfera objetos mais afastados aparentam estar ofuscados; sombras, dependendo da localização da fonte de luz determinados objetos podem lançar sombras sobre outros (Denis Kalkofen, 2011).

Uma das técnicas que podem ser utilizadas passa então por simular estes níveis de profundidade alinhando esse parâmetros da câmara virtual aos da câmara real. Isso pode ser alcançado através da criação de uma estrutura virtual que represente o que está a ser captado pela câmara real, permitindo assim um correto alinhamento dos níveis de profundidade do utilizador e a correta visualização das relações de profundidade entre objetos reais e virtuais. Esta técnica pode ser aplicada a outros parâmetros, além da profundidade, de forma aumentar ainda mais a perceção da visualização. No entanto, esta técnica não resolve os problemas de oclusões, sobreposição de dois objetos, uma vez que se um objeto virtual se sobrepuser a um objeto, o objeto virtual pode ser percecionado como estando a flutuar na visualização do mundo real.

Segundo Denis Kalkofen (2011) uma integração credível de objetos virtuais num ambiente real só é possível se as oclusões entre objetos reais e virtuais forem resolvidas. Uma das técnicas utilizadas para resolver este problema é a utilização de objetos *Phantom* (fantasmas) e consiste na geração de objetos *Phantom*, objetos virtuais invisíveis correspondentes aos objetos reais da cena, antes mesmo da geração dos objetos virtuais definitivos serem adicionados. Deste modo é possível efetuar testes de comparação de profundidade recorrendo ao *OpenGl* detetando assim eventuais oclusões. Esta abordagem permite também uma geração eficiente de sombras dos objetos virtuais sobre o ambiente real (Michael Haller, 2003). Uma técnica ligeiramente diferente consiste na utilização de um tipo de objetos semelhantes, os *Video Phantom*, que ao invés de serem invisíveis são antes uma representação opaca dos objetos reais. Como resultado, o objeto real vai ficar obscurecido sendo assim identificado em função dessa cor.

Outro dos problemas que pode afetar as cenas de Realidade Aumentada é o fato de que por vezes a informação disponibilizada ao sistema não é suficiente, na medida em que não

existem dados completos sobre o ambiente real de forma a gerar uma representação virtual do mesmo. Em aplicações capazes de utilizar técnicas de visualização de raio x, isto é, técnicas que permitam ver através de objetos do mundo real, assume-se que os dados virtuais estão completamente sobrepostos pelos objetos reais, sendo assim possível calcular a profundidade dos mesmos dando a informação necessária ao sistema para preservar as perceções de profundidade. No entanto as técnicas de raio x mencionadas dependem por sua vez de outras técnicas de deteção de características de imagem. São essas técnicas que vamos apresentar de seguida.

Uma dessas técnicas é a de deteção de arestas, que permite identificar arestas de objetos reais através da sobreposição de imagens previamente construídas para o efeito sobre a vista do ambiente real. Outra técnica utilizada é a de deteção de características salientes, como a matiz das cores, a luminosidade e o movimento. Através da aplicação de uma série de filtros de imagens a estas características é possível obter o efeito raio x numa cena (Denis Kalkofen, 2011).

Enquanto as técnicas apresentadas até agora tentam resolver alguns problemas de perceção, é preciso ter em conta as limitações de *hardware* dos próprios sistemas que podem influenciar a visualização. Restrições como o reduzido tamanho do ecrã, que limitam o campo de visão do utilizador também têm que ser levadas em conta. Algumas técnicas de manipulação de cena permitem ultrapassar alguns destes problemas. Por exemplo, distorções não lineares permitem aumentar o campo de visão e manipulações de objetos reais e virtuais conseguem melhorar a visibilidade dos mesmos (Denis Kalkofen, 2011).

A manipulação de objetos reais e virtuais no entanto exige que sejam cumpridos três pressupostos de forma a permitir uma composição correta da cena. Os objetos reais têm que ser rearranjados de forma convincente, de seguida conteúdo virtual tem que ser gerado para ser colocado na localização original dos objetos movidos e, por fim, tem que ser resolvidos todos os problemas de oclusão originados pela manipulação dos objetos. De forma a conseguir resolver esses problemas de oclusão podem ser utilizadas as técnicas de objetos *Video Phantom,* e *Phantom* anteriormente descritas. Embora estas técnicas garantam a deslocação de objetos reais para outro local da imagem, por vezes os objetos virtuais gerados para cobrir a área original apenas o conseguem fazer parcialmente. É assim necessário projetar todos os *pixels* dos objetos *Phantom* uma segunda vez de forma a identificar os *pixels* em duplicado de forma a poder remove-los. Este processo é denominado de renderização *dual phantom*, e apesar de poder ser acelerado é um processo que exige bastante a nível de performance do dispositivo devendo apenas ser utilizado e casos pontuais. As técnicas de manipulação de cena apresentadas podem em alguns casos deixar espaços “vazios” sem qualquer conteúdo real ou virtual. São assim necessárias técnicas capazes de preencher estes espaços. Uma dessas técnicas de restauração mais rápidas e simples consiste no preenchimento do espaço “vazio” com uma média da informação visual do fundo da restante cena.

Relativamente às técnicas de distorções mencionadas anteriormente e que permitem aumentar o campo de visão, são de particular utilidade de forma a dar destaque a POI's que embora perto não estejam no campo de visão imediato do utilizador. Aplicando o conceito dos objetos *Video Phantom* já mencionados é possível distorcer a organização espacial da cena removendo partes da mesma que não contenham nenhum POI para ganhar espaço ou inclusivamente removendo objetos físicos que estejam a provocar oclusão conforme podemos verificar na Figura 6.

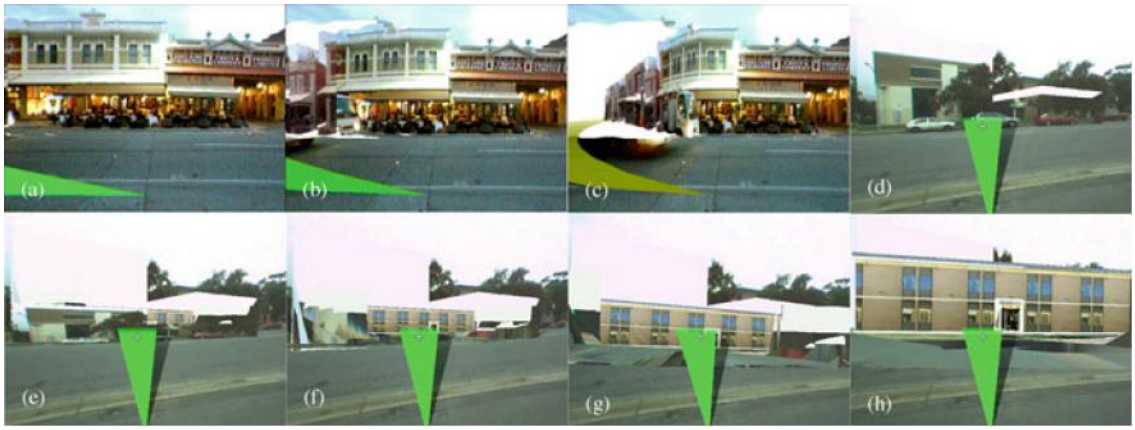


Figura 6 – Exemplo prático de distorções espaciais (Denis Kalkofen, 2011)

As técnicas de visualização aqui apresentadas abordam diversos desafios que o processo de visualização de uma aplicação de Realidade Aumentada por ter que resolver. A interação entre objetos virtuais e reais é uma das grandes mais-valias desta tecnologia, no entanto se o processo de visualização não for bem executado a qualidade da experiencia pode deteriorar-se significativamente com impacto a nível da usabilidade das aplicações e da satisfação do utilizador.

## 2.6 Conclusões

Como indicado no ponto 2.4.1, as aplicações *Ar Browser*, são aplicações LBS com um foco mais específico e que apresentam o conteúdo recorrendo à Realidade Aumentada, sendo por isso normal que partilhem alguns dos desafios, e soluções. No entanto, podemos verificar ao longo deste capítulo que uma parte desses mesmos desafios e soluções são também transversais a determinadas aplicações de Realidade Aumentada.

Tabela 1 — Análise comparativa dos desafios dos três tipos de aplicações

| | LBS | Realidade Aumentada | Ar Browser |
|---|---|---|---|
| **Receios Privacidade** | X | — | X |
| **Cálculo da localização GPS** | X | —X | X |
| **Cálculo da posição da câmara** | — | X | X |
| **Tracking** | — | X | X |
| **Gestão da informação no ecrã** | — | X | X |
| **Processamento e Bateria** | X | X | X |
| **Custo transferências de dados** | X | — | X |

Conforme podemos visualizar na Tabela 1, existem de fato desafios transversais em aplicações dos três tipos, assinalados com um "X" em cada tipo de aplicação, sendo que um "—" significa que o desafio em questão não se coloca. Há ainda um caso que apresenta tanto um "X" como um "—" uma vez que para as aplicações de Realidade Aumentada o desafio do cálculo da localização GPS tanto se pode aplicar, o caso dos *Ar Browsers*, como não se aplicar de todo.

Exemplo de desafios transversais aos vários tipos de aplicações são então os desafios de localização, mas a diferença é que enquanto nas aplicações *Ar Browser*, o objeto deste desafio é o POI e a câmara, tal não é necessariamente verdade nas aplicações de Realidade Aumentada, uma vez que pode não precisar de calcular a posição GPS para apresentar um modelo tridimensional baseado numa imagem por exemplo. Apesar desses desafios, segundo J. B. Gotow (n. d.) a utilização de sensores é uma alternativa viável em ambientes onde não é possível uma preparação prévia. Outro desafio comum são as limitações de processamento e de gasto de energia que impossibilitam algumas das soluções apresentadas de serem implementadas na sua totalidade, apesar de aqui essa transversalidade ser explicada pelo fato de ser explicitamente imposta por limitações dos dispositivos. Independentemente disso, a natureza dos problemas é suficientemente próxima o que permite concluir que eventuais desenvolvimentos numa destas três áreas poderão ter reflexo nas restantes, contribuindo assim para o aparecimento de soluções mas atrativas para o utilizador comum. Por fim um desafio que não depende diretamente de evoluções tecnologias nas áreas em questão são os custos de transferências de dados. Infelizmente neste momento apenas dependem dos operadores móveis de telecomunicações, a oferta de pacotes de dados com custos mais atrativos para o utilizador.

No que diz respeito ao desenvolvimento do módulo de Realidade Aumentada, que é o objetivo principal desta tese, a maioria das soluções propostas saem do âmbito desse desenvolvimento. No entanto, apesar dos desafios, os *smartphones* disponíveis atualmente são capazes de suportar aplicações de Realidade Aumentada, pois possuem a capacidade de processamento suficiente e os sensores necessários (J. B. Gotow, n. d.). Como prova disso, existem várias aplicações disponíveis no mercado, que estão em constante desenvolvimento e disponibilizam o trabalho desenvolvido sob a forma de API's e SDK's, pelo que o foco dos próximos capítulos será a análise da estrutura dessas mesmas aplicações e as vantagens e desvantagens das API's e SDK's disponibilizados. Deste modo será possível verificar as

diferentes estruturas das aplicações existentes no mercado, assim como escolher a solução que proporcione melhores condições para o desenvolvimento do referido módulo.

# 3 Análise de aplicações

Neste capítulo vamos analisar mais detalhadamente alguns dos *Ar Browsers* existentes no mercado, que funcionalidades oferecem, o modo como estão estruturados e em que maneiras diferem do modelo lógico proposto na Figura 3. Será apresentada uma aplicação por cada um dos modelos propostos por (Ben Butchart, 2011), com enfase nas diferenças entre os respetivos modelos.

## 3.1 Layar

A aplicação *Layar* pertence a uma companhia com o mesmo nome (www.layar.com) sediada na Holanda. É uma aplicação de Realidade Aumentada baseada em geolocalização que suporta modelos tridimensionais, *nature feature tracking* e ações programáveis, isto é, é possível configurar uma determinada ação relativa a um POI que seja despoletada mal o utilizador entre no raio de ação do respetivo POI. Lançada em 2009, está disponível para *Android*, *iPhone OS*, *Symbian* e *Blackberry OS* fazendo inclusivamente parte do lote de aplicações pré-instaladas em milhões de dispositivos de alguns dos maiores distribuidores de *smartphones* (Layar, 2011), tendo ainda anunciado em Janeiro de 2011 a marca de um milhão de utilizadores ativos num total de dez milhões de *downloads* da aplicação a juntar às já referidas pré-instalações em diversos dispositivos.

A aplicação obtém os dados GPS e, em conjunto com os sensores do dispositivo, determina a localização do utilizador para depois efetuar um pedido ao servidor que, por sua vez, responde retornando a informação no formato JSON[10], informação essa que contém os POI's cuja localização esteja dentro de um determinado alcance ajustável com base na posição do utilizador. Esta aplicação utiliza um conceito denominado *Layer*, que consiste num conjunto de definições que depois são utilizadas para a obtenção e visualização dos POI's. Cada *Layer* aponta para um recurso externo responsável pela devolução dos POI's com base num conjunto de parâmetros configuráveis. As *Layers* disponibilizadas pela aplicação podem ser desenvolvidas e submetidas pelos próprios utilizadores mediante um processo de validação, podendo assim os mesmos contribuir para o desenvolvimento da comunidade.

[10] Formato de intercâmbio de dados leve e rápido de transferir e interpretar.

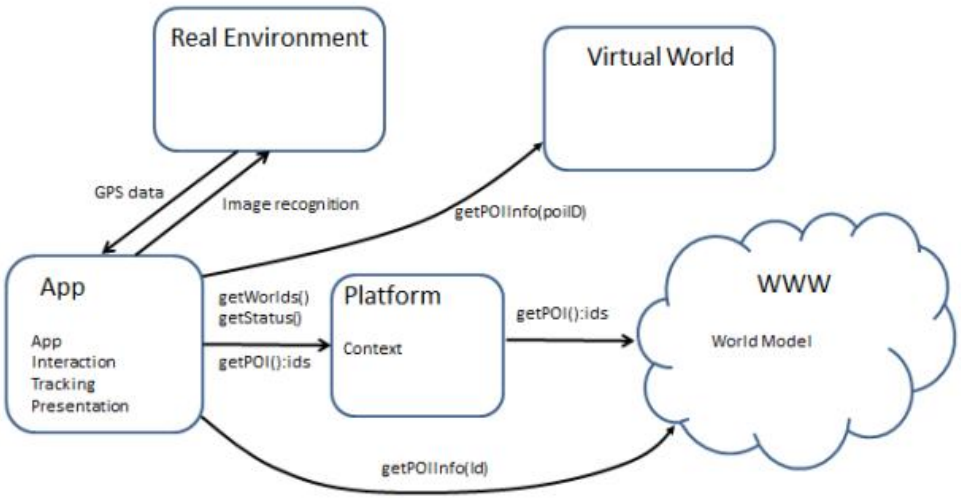


Figura 7 – Arquitetura *Gateway* (Ben Butchart, 2011)

Na Figura 7 podemos visualizar o modelo lógico da aplicação. Esta arquitetura é composta por dois principais subsistemas, sendo que a sua principal característica é o facto de a maior parte dos subsistemas estarem maioritariamente integrados na aplicação, com exceção dos dados dos POI's e das configurações das *Layers* (Ben Butchart, 2011).

O primeiro grande subsistema é, então, o subsistema *App*, que corresponde à aplicação instalada no dispositivo móvel, sendo ainda responsável pelo *tracking*, bem como pela interação e apresentação dos conteúdos virtuais. Por outro lado o subsistema *Platform* é responsável por responder aos pedidos da aplicação e fornecer dados de contexto, como informação relacionada com os POI's, através de *webservices* alojados em servidores externos (Ben Butchart, 2011). Enquanto que a manutenção do subsistema *Platform* é da responsabilidade do *Layar*, os *webservices* estão a cargo do utilizador responsável pela respetiva *Layer*, servindo o subsistema como ponte ou elemento de ligação entre a aplicação e a W*orld Wide Web*. Também na Figura 7 podemos visualizar os subsistemas *Virtual World* e *Real Environment*. Estes subsistemas são transversais a todas as arquiteturas que vamos apresentar e correspondem respetivamente ao mundo virtual composto pelos indicadores visuais de Realidade Aumentada apresentados ao utilizador e com os quais ele pode interagir, e ao mundo real, ou seja o mundo físico em redor do utilizador e cujos de onde os sensores do dispositivo obtêm informação (GPS por exemplo) para compor o mundo virtual.

Esta arquitetura oferece algumas vantagens sob o ponto de vista dos utilizadores e programadores, uma vez que o subsistema *Platform* é acessível via *web* proporcionando uma ferramenta de testes e agilizando assim a criação e divulgação de *Layers* pelos próprios utilizadores.

## 3.2 LibreGeoSocial ArViewer

O *ArViewer* (www.libregeosocial.org/node/24) é uma aplicação móvel *Open Source* de Realidade Aumentada, que suporta diversos conteúdos multimédia, tais como: imagens, áudio, vídeo e texto. Além dos referidos conteúdos multimédia a aplicação suporta ainda um modo de visualização em forma de listagem e uma ferramenta de pesquisa. Contrariamente à generalidade das aplicações que efetuam apenas um cálculo das coordenadas GPS, esta aplicação calcula também a altitude a que se encontra o dispositivo, de forma a apresentar a representação visual dos POI's de uma forma o mais fidedigna possível. Por fim, a aplicação é capaz de funcionar tanto em ambientes exteriores (ruas por exemplo) como interiores (dentro de edifícios) desde que para este último seja possível efetuar uma pré-marcação do mesmo através de códigos QR (LibreGeoSocial, n. d.).

Em comparação com a estrutura indicada no ponto 3.1, esta aplicação utiliza um modelo lógico ligeiramente diferente.

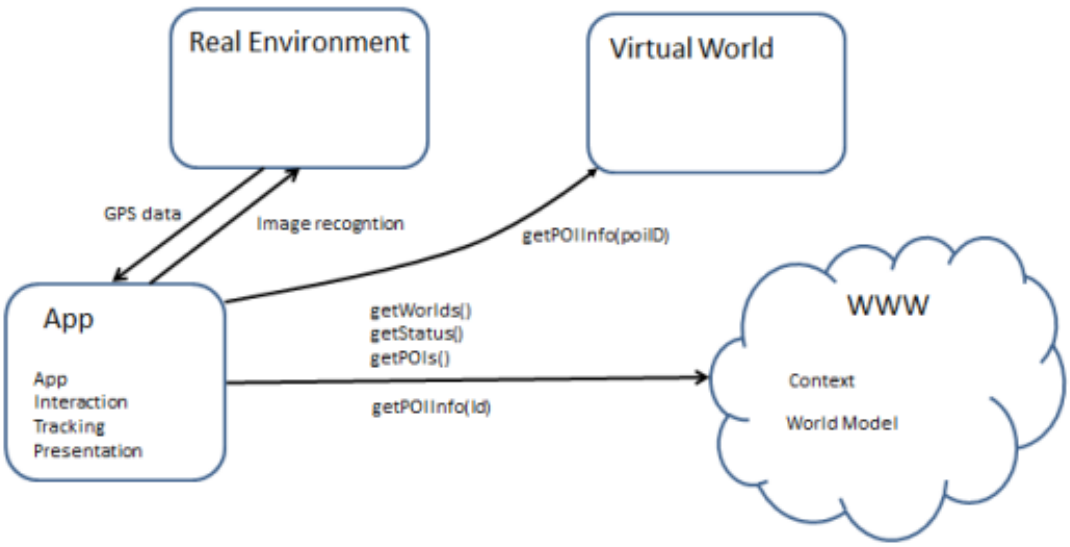


Figura 8 – Arquitetura *Web* (Ben Butchart, 2011)

Podemos visualizar na Figura 8 algo semelhante ao subsistema *App* já descrito no ponto 3.1, sendo a principal diferença a ausência do subsistema *Platform*, responsável pelo armazenamento de informação de contexto na arquitetura *Gateway*. Os dados de contexto, nomeadamente os detalhes dos POI's são obtidos diretamente da W*orld Wide Web*, sem ser utilizada nenhuma estrutura previamente definida pela plataforma, o que liberta os programadores da estrutura que o subsistema *Platform* implica, tornando as aplicações menos dependentes. (Ben Butchart, 2011).

## 3.3 Wikitude

O *Wikitude* é uma aplicação móvel de Realidade Aumentada que foi publicada pela primeira vez em 2008 por uma empresa Austríaca, Wikitude GmbH (www.wikitude.com) e está disponível para os sistemas *Android*, *iPhone OS*, *Symbian*, *Windows Phone* e *Blackberry OS*, tendo sido eleita como o melhor AR Browser dos últimos quatro anos (lester, 2012). Foi a primeira aplicação que utilizou uma abordagem LBS aplicada à Realidade Aumentada, suporta *tracking* através de GPS e através de marcadores e obtém a informação de contexto que apresenta através de serviços como o *Wikipedia* (http://en.wikipedia.org) e o *Qype* (www.qype.com), um serviço europeu de conteúdos gerados pelo utilizador. A aplicação permite alterar o provedor de dados utilizado, uma boa solução caso os resultados obtidos não estejam a ser os desejados, além de permitir também ao utilizador efetuar pesquisas sob o conteúdo apresentado. Segundo a descrição da aplicação na plataforma de distribuição de aplicações móveis *Google Play* (https://play.google.com/store), oferece conteúdo interativo de mais de cem milhões de POI's.

Inicialmente fechada a desenvolvimentos e contribuições da sua comunidade de utilizadores, esta aplicação inclui, a partir da sua terceira versão, uma outra aplicação denominada de *Wikitude.me* (www.wikitude.me) que permite aos utilizadores criarem o seu próprio conteúdo (D. Marimon, n. d.).

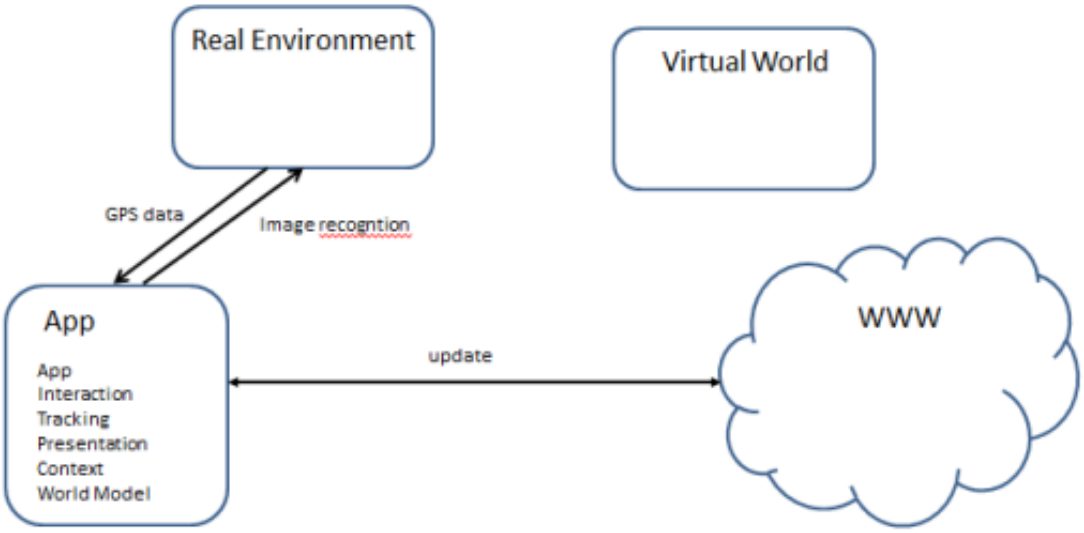


Figura 9 – Arquitetura *Standalone* (Ben Butchart, 2011)

Esta aplicação está estruturada de acordo com a arquitetura *Standalone* apresentada na Figura 9 que, como o próprio nome indica mantém apenas o subsistema *App*, incorporando todos os subsistemas na própria aplicação. A principal desvantagem desta arquitetura é o fato de que a própria informação de contexto estará integrada na aplicação, o que obriga a que a mesma seja atualizada sempre que estiverem disponíveis dados mais recentes. Torna-se ainda complicado qualquer integração com fornecedores de POI's externos o que pode limitar os

dados que a aplicação consegue apresentar ao utilizador. Por outro lado, como os dados já estão embutidos na aplicação, tem a vantagem de não necessitar de uma conexão de rede para os obter e apresentar (Ben Butchart, 2011), com todas as vantagens ao nível do consumo de dados de rede, e respetivos custos que isso implica.

## 3.4 Conclusões

Apesar de a arquitetura *Web* ser talvez a que oferece um melhor compromisso entre dependência estrutural e a escalabilidade das aplicações, cada uma das três arquiteturas apresentadas tem as suas vantagens e desvantagens, não se podendo afirmar que uma é melhor ou pior do que outra. Tudo depende das necessidades do utilizador ou dos objetivos do programador.

Segundo Ben Butchart (2011), os desenvolvimentos futuros levarão ao surgimento de novas variações destas arquiteturas, nomeadamente algumas soluções avançadas no capítulo anterior, como a utilização de imagens da câmara do dispositivo para determinar a localização e orientação do utilizador podem levar à passagem do *tracking* para o subsistema *Platform*.

# 4 Análise de Soluções

Conforme referido em 2.3.2 e em 2.4.3, existe um conjunto de desafios por ultrapassar no desenvolvimento de uma aplicação de Realidade Aumentada, mais especificamente um *Ar Browser*. Algumas das soluções propostas não são exequíveis sem uma determinada estrutura, que dificilmente poderia ser montada no contexto desta tese, uma vez que o módulo proposto será incorporado numa aplicação, o *coolplaces mobile*, que conta com uma estrutura já desenvolvida limitando assim a implementação dessas mesmas soluções. Um exemplo prático dessa limitação diz respeito à utilização de algoritmos de *nature feature tracking*, o que exigiria alterações ao nível de base de dados para que a cada POI pudesse ser associada a respetiva imagem, de forma a permitir a sua identificação.

No entanto, como referido no ponto 2.6, um determinado conjunto das soluções apresentadas tem sido desenvolvido e implementado por aplicações de Realidade Aumentada existentes no mercado, nomeadamente as abordadas no capítulo 3. Neste capítulo vamos analisar as API's e SDK's disponibilizados de acordo com as funcionalidades que oferecem e analisando em que medida vão ao encontro dos desafios anteriormente identificados.

## 4.1 Layar API

Esta API, disponibilizada pela aplicação descrita em 3.1 segue a estrutura lógica da arquitetura *Gateway* aí apresentada. Numa primeira fase, é necessário registar uma conta de programador na plataforma do *Layar*, posteriormente dá-se início ao processo de criação de uma *Layer* que tem que ser devidamente configurada de acordo com um conjunto de parâmetros. De seguida é preciso desenvolver um *webservice* que devolva os POI's de acordo com as especificações da plataforma, indicando o respetivo endereço nas definições. Após a *Layer* estar devidamente configurada e testada, a plataforma fica responsável pelo processo de *tracking* e por toda a gestão de dados sempre que essa *Layer* for requisitada. De seguida apresentamos a Figura 10 que diz respeito a esse mesmo processo de configuração.

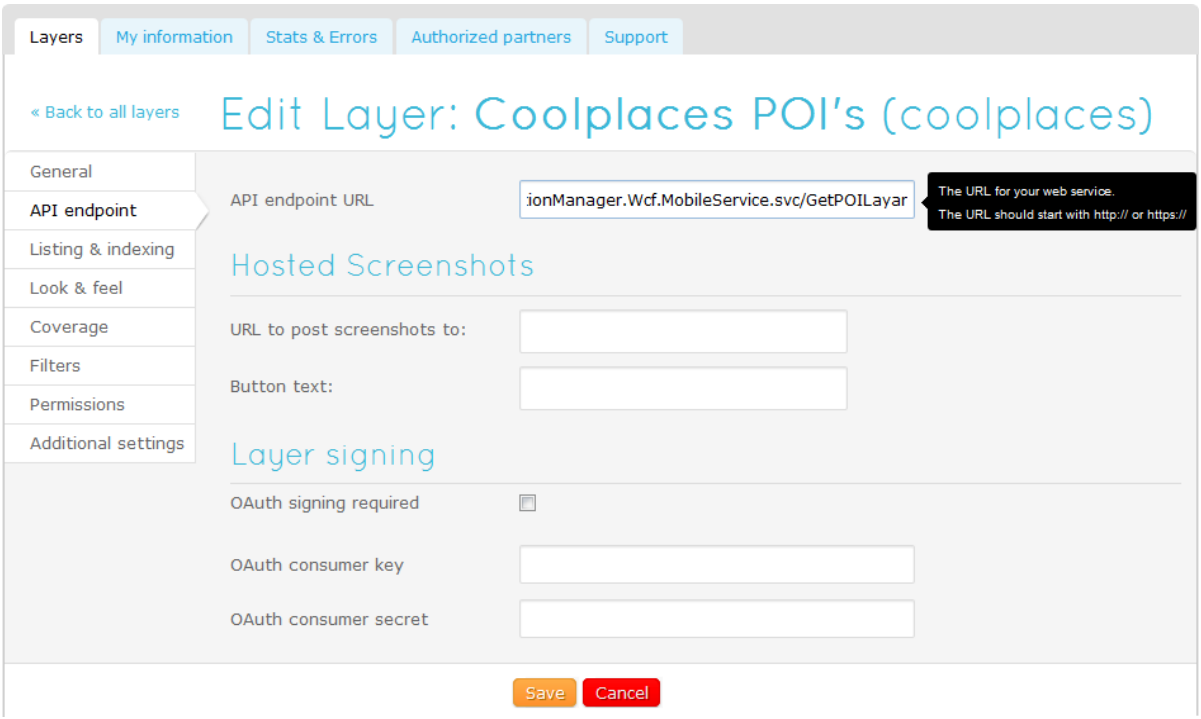


Figura 10 – Configuração de uma *Layer*

Conforme podemos observar na Figura 10 o processo de configuração de uma *Layer* é composto por oito passos, cada um com diferentes propriedades. No decorrer desses passos são requisitadas algumas configurações obrigatórias, como a indicação do endereço do *webservice* responsável pela obtenção dos dados dos POI's, nome da *Layer*, pais ou países a que se destina, entre outras. Existem também um conjunto de configurações obrigatórias como o *look* & *feel*, ou seja, o visual da *Layer* perante o utilizador (logotipos, esquema de cores, etc…), ou ainda a indicação de um protocolo de autenticação caso a *Layer* se destine apenas a membros de um determinado serviço que necessite dessa mesma autenticação.

Uma das principais vantagens desta aplicação é o fato desta assumir a gestão do *tracking* e da visualização dos POI's, para além de estar disponível sem custos. Outra vantagem importante é a de que, caso a *Layer* seja aceite e aprovada pela plataforma, esta permite a geração de um *launcher* da aplicação *Layar* exclusivamente dedicado à *Layer* desenvolvida, ou seja, é possível iniciar a aplicação devidamente configurada por omissão, permitindo assim algum grau de integração com outra aplicação. Por outro lado a aplicação fica parcialmente dependente de terceiros, uma vez que a L*ayer* tem que ser aprovada pela plataforma e a resposta em JSON do *webservice* tem que seguir um formato pré estabelecido pelo *Layar*. É ainda de referir que devido ao endereço do *webservice* ser parcialmente gerado pela plataforma, tal dificulta a utilização de parâmetros personalizados, uma vez que não é possível acrescentar determinados parâmetros ao endereço de forma a afinar e enriquecer a informação devolvida pelo mesmo, a menos que estes estejam disponíveis na plataforma. Por fim outra desvantagem importante é o fato de não ser possível customizar a vista da informação, uma vez que a informação visual apresentada ao utilizador segue um formato pré-determinado, não sendo possível escolher o nível de destaque de determinada informação.

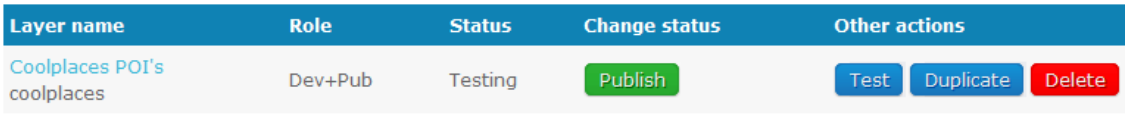


Figura 11 – Opções de gestão de uma *Layer*

Esta aplicação, conforme descrito no ponto 3.1 possui uma plataforma que oferece a possibilidade do programador efetuar testes às *Layers* desenvolvidas antes de as submeter para aprovação. Na Figura 11 podemos visualizar o ecrã que nos permite, entre outras opções, submeter para aprovação as *Layers* e iniciar a plataforma de testes. Plataforma essa que permite testar as *Layers* desenvolvidas munindo o programador de ferramentas para alterar várias configurações e criar diversos cenários de teste diferentes.

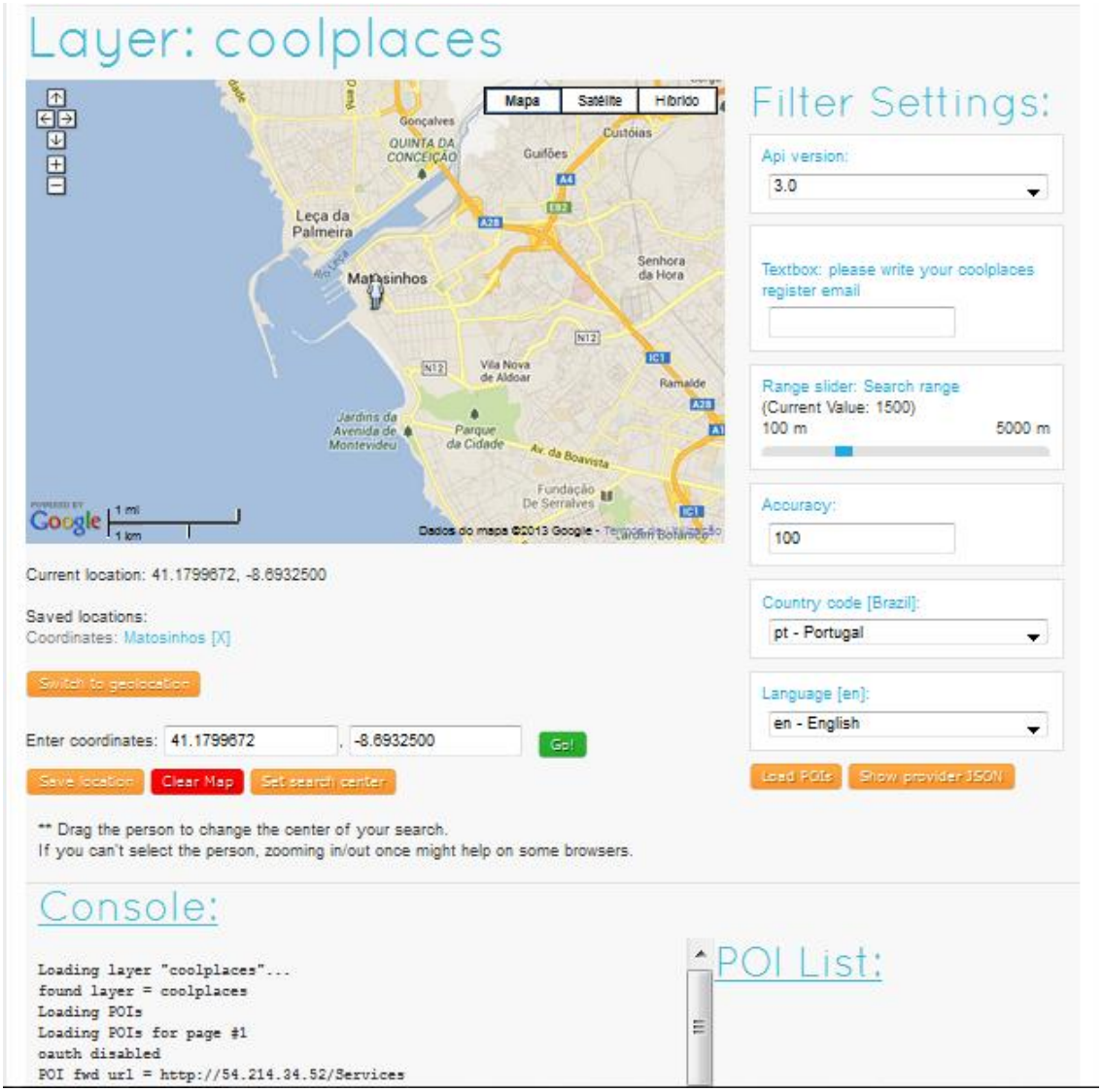


Figura 12 – Ferramentas de teste de uma *Layer*

Conforme podemos verificar na Figura 12, é possível alterar as coordenadas GPS, o alcance até ao qual devem ser procurados resultados, entre outras opções. Os resultados são depois apresentados numa consola de texto para poderem ser analisados permitindo aos programadores identificarem eventuais problemas com o pedido ao *webservice*, problemas nas estruturas de dados e, em caso de sucesso, as informações que são devolvidas por cada POI.

## 4.2 Metaio

*Metaio* (www.metaio.com) é uma empresa sediada em Munique que disponibiliza de forma gratuita uma solução bastante completa ao nível das opções de *tracking* que oferece e tem a particularidade de estar disponível em dois modos de implementação diferentes. Um SDK que permite uma maior integração com uma aplicação já existente e uma aplicação denominada *Junaio* cuja estrutura é semelhante à apresentada no ponto 4.1. Aplicação essa que já foi alvo de quase dois milhões de *downloads*, e permite localização tanto em ambientes exteriores (ruas, etc…) como dentro de edifícios, novamente caso os edifícios estejam previamente preparados com códigos QR (Saenz, 2011).

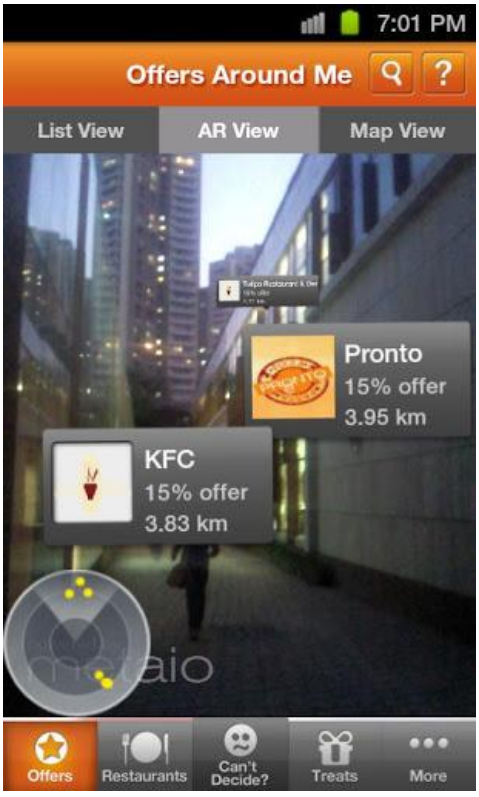


Figura 13 – Exemplo de utilização do *Metaio*

Conforme podemos verificar na Figura 13, a aplicação oferece um conjunto de indicadores visuais capazes de apresentar texto e imagem com o devido destaque consoante a distância do utilizador em relação ao POI. Além da vista de Realidade Aumentada, permite ainda aceder

a uma lista dos POI's encontrados e a um mapa bidimensional com a localização desses mesmos POI's.

### 4.2.1 SDK

Através do SDK disponibilizado pela aplicação os programadores podem aceder a um conjunto de métodos que permitem, desde registar um conjunto de pontos de interesse, a atualizar a respetiva posição relativa de cada um deles consoante as mudanças de posição do dispositivo. Basta assim obter da base de dados os pontos de interesse de acordo com uma dada localização e associar cada um deles a uma informação visual, tipicamente uma imagem com texto embutido, que será devidamente apresentada no local correspondente às coordenadas GPS do POI.

São vantagens desde SDK, o fato de assumir a gestão do *tracking*, o grau de integração que possibilita, uma vez que os métodos disponibilizados podem ser utilizados na lógica de uma outra aplicação, e ainda o fato de ser gratuito sendo apenas visível uma marca de água. Por contra ponto o registo dos POI's no mundo virtual tem que ser implementado na nossa aplicação fazendo uso dos já referidos métodos.

### 4.2.2 junaio Plugin

Como já foi referido, o *junaio plugin* permite uma implementação semelhante à apresentada no ponto 4.1, com apenas duas diferenças: a resposta do *webservice* em vez de retornar a informação no formato JSON, deve retorná-lo no formato XML[11] e o conceito *Layer* assume a denominação *Channel*.

Mais uma vez uma das vantagens passa pelo fato da aplicação assumir a gestão do *tracking* e, tal como o SDK, estar disponível sem custos apenas com uma marca de água. Em relação às desvantagens, é de salientar o fato da aplicação ficar parcialmente dependente de terceiros, uma vez que à semelhança da solução apresentada no ponto 4.1, requer aprovação prévia da plataforma e a resposta XML do *webservice* tem que seguir um formato pré estabelecido pela mesma.

Esta solução parece ter conseguido atrair uma grande parte dos entusiastas da Realidade Aumentada e dos *AR Browser* em particular, uma vez que em três anos terá conquistado nada menos do que cinco mil criadores de conteúdo que são responsáveis por cerca de setenta e cinco por cento dos *Channels* disponibilizados ao utilizador (Saenz, 2011). Foi inclusivamente desenvolvida uma ferramenta que permite incorporar anúncios publicitários nos *Channels*, desta forma os utilizadores são ainda mais encorajados a produzir conteúdo para a comunidade *Metaio*.

[11] Um das formatos de intercâmbio de dados mais utilizado

## 4.3 Droidar

O funcionamento do projeto *Droidar* (https://github.com/bitstars/droidar) é semelhante à API referida em 4.2.1, com a diferença de que é um projeto O*pen Source* que não pertence a nenhuma empresa. Suporta localização por GPS e por marcadores, sendo que uma das principais vantagens é que oferece métodos, dedicados à melhoria da precisão da localização dos POI's, ficando o *tracking* a cargo da API. Relativamente às desvantagens, além de caber à nossa aplicação o registo dos POI's, existe uma grande falta de documentação relativamente a todos os aspetos desta API, uma vez que é um projeto que carece de uma liderança profissional, sendo totalmente desenvolvido de forma voluntaria e de acordo com contribuições da comunidade.

## 4.4 Mixare

O mixare (www.mixare.org), cujo nome é um acrónimo de *mix Augmented Reality Engine*, é um projeto *Open Source* desenvolvido pela *Peer Internet Solutions* (www.peer.biz/en/) que, apesar de ter uma equipa de desenvolvimento a tempo inteiro, está aberto a qualquer contribuição da comunidade. A aplicação está disponível para os sistemas *Android* e *iPhone OS* e disponibiliza uma solução para o desenvolvimento de outras aplicações autónomas.

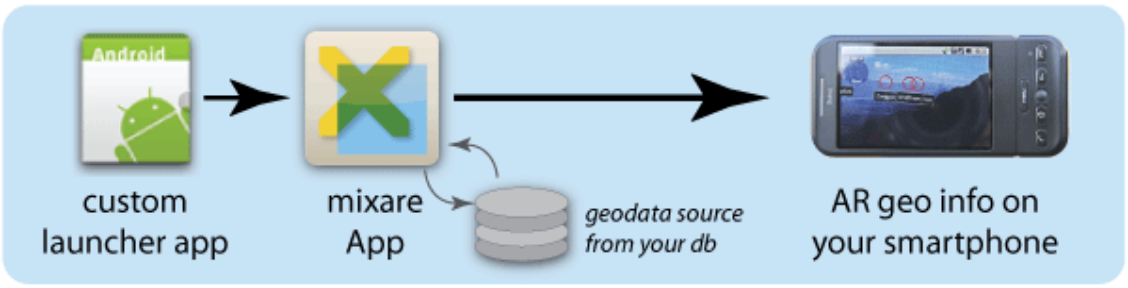


Figura 14 – Fluxo lógico da solução *Mixare* (mixare, n. d.)

Conforme indicado na Figura 14, esta solução tem um funcionamento muito particular, na medida que permite que a própria aplicação *Mixare* seja lançada a partir de outra, utilizando opcionalmente uma base de dados diferente da que é utilizada por omissão para obtenção da informação sobre os POI's. Para tal basta apenas indicar o endereço de um *webservice* que devolva essa mesma informação no formato JSON e com uma determinada estrutura para que a aplicação *Mixare* interprete e mostre os dados. Para além de ser gratuita e de assumir a gestão do *tracking* e da visualização dos POI's, a principal vantagem é a sua fácil implementação como podemos verificar no seguinte extrato de código.

```
Intent i = new Intent();
i.setAction(Intent.ACTION_VIEW);
i.setDataAndType(Uri.parse(serviceURL), "application/mixare-json");
startActivity(i);
```

Código 1 – Extrato de código da implementação da solução *Mixare*

Conforme demonstrado pelo extrato de Código 1, com apenas quatro linhas de código é possível lançar a aplicação utilizando a nossa própria base de dados de POI's. Em primeiro lugar é declarado um *Intent*, um objeto de comunicação da API do sistema operativo *Android* que facilita a troca de dados entre aplicações encorajando a colaboração e a reutilização de componentes (A. P. Felt, 2011), o que neste caso vai permitir a comunicação entre a nossa aplicação e a aplicação *Mixare*. De seguida são configuradas algumas propriedades desse mesmo *Intent*, nomeadamente a indicação da aplicação com a qual vamos comunicar e a indicação de um endereço de um *webservice* responsável pela obtenção dos dados dos POI's. Por fim é iniciada uma *Activity*, uma classe do sistema *Android* responsável pela criação da janela onde vamos visualizar a aplicação Mixare com os respetivos dados retornados pelo *webservice* especificado anteriormente.

O custo da fácil implementação reflete-se numa dependência da própria aplicação *Mixare* uma vez que além de definir que base de dados usar, todo o resto do processo fica a cargo do *Mixare* sem qualquer intervenção do programador, o que impede qualquer customização da visualização da informação. Outra desvantagem é o fato da resposta em JSON do *webservice* ter que seguir um formato pré estabelecido para que os dados possam ser corretamente apresentados.

## 4.5 Wikitude SDK

O *Wikitude* SDK, disponibilizado pela aplicação *Wikitude* apresentada no ponto 3.3, é bastante completo e permite uma implementação mais tradicional semelhante à abordada em 4.3. A grande diferença são as tecnologias utilizadas, uma vez que este SDK baseia-se em tecnologias web como HTML[12], *JavaScript*[13] e CSS[14], permitindo aos programadores desenvolver experiências de Realidade Aumentada multiplataforma.

O processo de registo dos POI's fica a cargo da nossa aplicação, sendo que os dados dos mesmos necessitam de ser estruturados de acordo com uma estrutura disponibilizada pelo SDK. Uma das suas mais-valias é o suporte a reconhecimento de imagem, de modo a aperfeiçoar o processo de *tracking*, sendo que esse processo é efetuado de acordo com

[12] *HyperText Markup Language*, linguagem para a criação e apresentação de conteúdos num *browser*.
[13] Linguagem de programação que permite executar código do lado do cliente e comunicação assíncrona, isto é, não é necessário ficar à espera que uma transmissão termine para continuar a execução do código.
[14] *Cascading Style Sheets*, linguagem de estilo que permite definir a apresentação de conteúdos escritos em linguagens como o HTML ou o XML.

recursos nativos do próprio SDK, isto é, utiliza bases de dados de próprias para o efeito. Infelizmente a utilização deste SDK apenas é gratuita para projetos educacionais.


Figura 15 – Exemplo de utilização do *Wikitude*

Em comparação com o que foi descrito no ponto 4.2, podemos verificar na Figura 15 a existência de algumas diferenças, nomeadamente nos indicadores visuais que apresentam apenas imagens enquanto o texto apenas é disponibilizado numa vista de detalhe apenas acedida após clique no indicador correspondente.

## 4.6 ARViewer SDK

O *ARViewer* SDK é disponibilizado pela aplicação analisada no ponto 3.2, e a sua estrutura é bastante semelhante à que vimos em 4.4, com a diferença que ao invés de um dos parâmetros ser o endereço do *webservice* que devolve os POI's, requer uma lista já com os resultados obtidos. É necessária a instalação prévia da aplicação *ARViewer* que depois fica responsável pelo processo de *tracking* e pela respetiva visualização e interação com os POI's. Estando disponível gratuitamente, um dos seus grandes trunfos é o facto de permitir um grande grau de customização da apresentação da informação, uma vez que se trata de uma aplicação *Open Source,* o que permite que os programadores acedam gratuitamente ao seu código de modo a poderem efetuar as alterações necessárias do ponto de vista visual.

```
GenericLayer mylayer = new GenericLayer(0, "", "My Layer", "Description",
null, null, null, null, null, null);
ArrayList<GeoNode> mNodeList = new ArrayList<GeoNode>();
mNodeList.addAll (getLayerNodes(0, "", "0", latitude, longitude, 10.0, 0,
5));
mylayer.setNodes(mNodeList);

Intent i = new Intent("com.libresoft.apps.ARviewer.VIEWER");
i.putExtra("LAYER", mylayer);
i.putExtra("LATITUDE", latitude);
i.putExtra("LONGITUDE", longitude);
startActivity(i);
```

Código 2 – Extrato de código da implementação da solução *ARViewer*

Como podemos ver no extrato de Código 2 apresentado, a sua implementação é relativamente simples. Primeiro é declarado um *Intent* com a indicação da aplicação a utilizar sendo configuradas três propriedades: a latitude, a longitude e um objeto da API denominado *GenericLayer*, ao qual é atribuído previamente uma lista de objetos *GeoNode*, estrutura de dados disponibilizada pela API e que representa o POI contendo as respetivas propriedades para posterior visualização na aplicação. Após estas configurações é iniciada uma *Activity* do sistema *Android* que permite assim criar uma nova janela onde é executada a aplicação *ARViewer*.

## 4.7 ARLab SDK

O *ARLab* SDK, desenvolvido pela companhia *ARLab* (www.arlab.com), tem um modo de implementação mais tradicional e oferece um sem número de funcionalidades tais como: suporte de elementos vídeo, edição de POI's em tempo real, gestão da visualização dos POI's no ecrã, suporte para *tablets*; performance e gestão de memória otimizadas, entre outras. Fácil de integrar, permite o desenvolvimento de uma aplicação totalmente independente fazendo a gestão das funcionalidades mais complexas como o *tracking* e a visualização dos POI's. Seria à partida uma forte candidata a ser a solução escolhida, não fosse o fato de infelizmente não estar disponível gratuitamente, pelo que não nos vamos alongar mais na respetiva análise.

## 4.8 Análise dos resultados

Com vista á escolha de uma solução para posterior implementação do módulo de Realidade Aumentada, foram efetuados testes às soluções apresentadas no decorrer deste capítulo. Numa primeira fase, foram desenvolvidos protótipos cujo propósito era o de pelo menos conseguir obter e apresentar um POI através de Realidade Aumentada. Não foi possível obter resultados satisfatórios, na medida em que não foi possível apresentar visualmente um único POI, com a API *DroidAr*, o SDK *Metaio* e o *junaio Plugin*, pelo que não foram incluídos nesta análise. Em relação ao SDK *ARLab*, conforme referido no ponto 4.7, uma vez que este não

estava disponível gratuitamente não foi sequer equacionado. Em relação às restantes soluções, foi possível apresentar POI's tendo sido desenvolvidos quatro protótipos. Através da sua utilização, foi possível medir e comparar quatro parâmetros: precisão, visualização da informação, integração e conteúdos multimédia.

A precisão da localização dos POI's diz respeito à comparação entre o posicionamento real de um POI e o seu posicionamento virtual no ecrã da aplicação, a visualização da informação diz respeito ao modo como a mesma é apresentada ao utilizador, a integração diz respeito ao grau da mesma com a aplicação *coolplaces mobile* que a respetiva solução permite e, por fim, o parâmetro multimédia representa os conteúdos multimédia que cada solução disponibiliza para associar a cada POI.

Os valores da precisão foram obtidos em parte através de informações disponibilizadas pelas próprias API's e SDK's e através da comparação entre a posição de um POI de localização conhecida e a respetiva representação virtual no dispositivo, tendo para o efeito sido executados em dez ocasiões diferentes os quatro protótipos testados com o dispositivo na mesma posição e orientado para um determinado POI, sendo estimada a respetiva precisão pelos desvios apresentados pelos indicadores visuais em relação à localização física do POI. Relativamente à visualização de informação, o critério utilizado medido, foi a ação, ou falta dela que o utilizador tinha que efetuar para visualizar informação referente a um POI. Por fim, as classificações do parâmetro integração têm o seguinte significado: mínima, módulo acessível por uma aplicação externa; parcial, módulo acessível por uma aplicação externa mas executada pela nossa aplicação e com controlo sobre a apresentação de dados; total, módulo desenvolvido na estrutura da nossa aplicação, sem ser necessário executar uma aplicação desenvolvida por terceiros.

De seguinte apresentamos uma comparação direta entre os valores dos parâmetros anteriormente mencionados referentes a cada uma das soluções que foram implementadas com sucesso.

Tabela 2 — Análise comparativa das soluções

| | **Precisão (m)** | **Visualização Inf** | **Integração** | **Multimédia** |
|---|---|---|---|---|
| **Layar** | Aprox. 40 | Após clique POI | Mínima | Texto e Imagem |
| **Mixare** | Aprox. 10 | Imediata | Parcial | Texto, web |
| **ArViewer** | Aprox. 10 | Imediata | Parcial | Audio e Imagem |
| **Wikitude** | Aprox. 10 | Imediata | Total | Texto |

Conforme se pode verificar pela Tabela 2, existem três soluções com resultados muito semelhantes e uma quarta com resultados mais divergentes. A aplicação *Layar*, tendo em conta uma menor precisão (fator identificado como um dos principais problemas) e o fato dos conteúdos multimédia não serem diretamente visualizados foi a primeira a ser descartada.

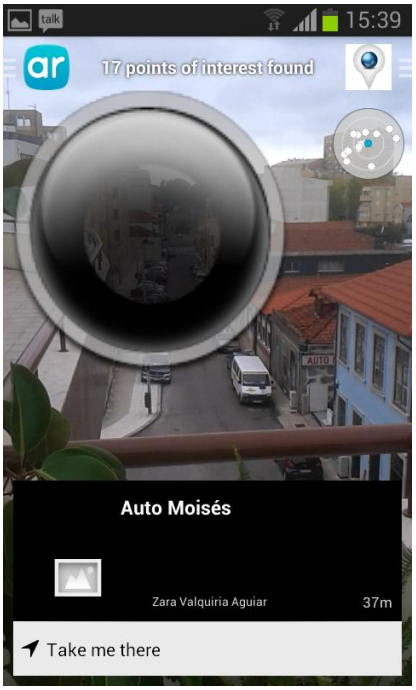


Figura 16 – Exemplo de utilização do protótipo desenvolvido com a solução *Layar*

Conforme podemos verificar na figura anterior, o protótipo desenvolvido através da solução *Layar* não permite uma correta identificação do POI uma vez que só clicando no indicador visual e acedendo à vista de detalhe é que temos acesso à informação referente ao respetivo POI. É também possível verificar a falta de precisão desta solução, uma vez que o POI se encontra do lado direito da imagem enquanto o indicador visual se encontra no topo superior esquerdo da mesma.

Em relação às três soluções restantes, o SDK da *Wikitude* permite uma integração a nível de código superior às outras duas soluções, no entanto além da sua implementação ser ligeiramente mais complexa, fica a perder na comparação em termos de atributos multimédia. Finalmente, as soluções *Mixare* e *ArViewer* são bastante semelhantes, até no modo de implementação, no entanto, o SDK da *ArViewer* permite associar a cada POI uma maior gama de elementos multimédia, o que torna a experiência de utilização mais completa, o que contribuiu decisivamente para que a escolha recaísse nesta solução.

Este processo de análise de soluções ficou assim concluído com a escolha a recair sobre a solução *ArViewer* SDK, solução essa utilizada no desenvolvimento de um protótipo final cujos detalhes de implementação serão explanados no capítulo seguinte.

# 5 Desenvolvimento do Módulo AR do coolplaces mobile

Escolhida a solução a utilizar, cabe agora explicar o processo de desenvolvimento e implementação de um protótipo mais completo do módulo proposto no âmbito desta tese, um modo de pesquisa e exploração de POI's em Realidade Aumentada, que faça uso da base de dados e respetiva informação de contexto da aplicação *coolplaces mobile*.

Conforme descrito no ponto 4.6, a implementação do módulo de Realidade Aumentada da aplicação *ArViewer* pode ser alcançada fazendo uso das seguintes linhas de código:

```
Intent i = new Intent("com.libresoft.apps.ARviewer.VIEWER");
i.putExtra("LAYER", mylayer);
i.putExtra("LATITUDE", latitude);
i.putExtra("LONGITUDE", longitude);
startActivity(i);
```

Código 3 – Extrato de código responsável pela ativação da aplicação *ARViewer*

Embora o processo de fornecer os dados e posteriormente executar a aplicação *ARViewer* pareça simples, antes dos dados poderem ser apresentados tem que ser obtidos, tratados e customizados pela nossa aplicação. Só posteriormente é que o extrato de Código 3 apresentado pode ser utilizado com sucesso, sendo os dados adicionados a uma variável do tipo *GenericLayer* (*mlayer*) assim que devidamente formatados de acordo com a estrutura requerida pela aplicação. Por sua vez, os dados obtidos vão depender do cálculo da latitude e da longitude que é efetuado recorrendo aos sensores do dispositivo.

Para melhor explicar o processo de obtenção dos dados e de cálculo da latitude e longitude, importa realizar uma breve apresentação da estrutura da aplicação *coolplaces mobile*.

## 5.1 Estrutura coolplaces mobile

A aplicação *coolplaces mobile* é uma aplicação LBS que foi desenvolvida na linguagem de programação *Java* para a plataforma *Android*, oferecendo como tal uma série de funcionalidades baseadas na localização do utilizador. A sua estrutura pode ser explicada através do seguinte cenário de uso, que diz respeito ao processo de utilização do módulo cuja implementação este capítulo se propõe a detalhar.

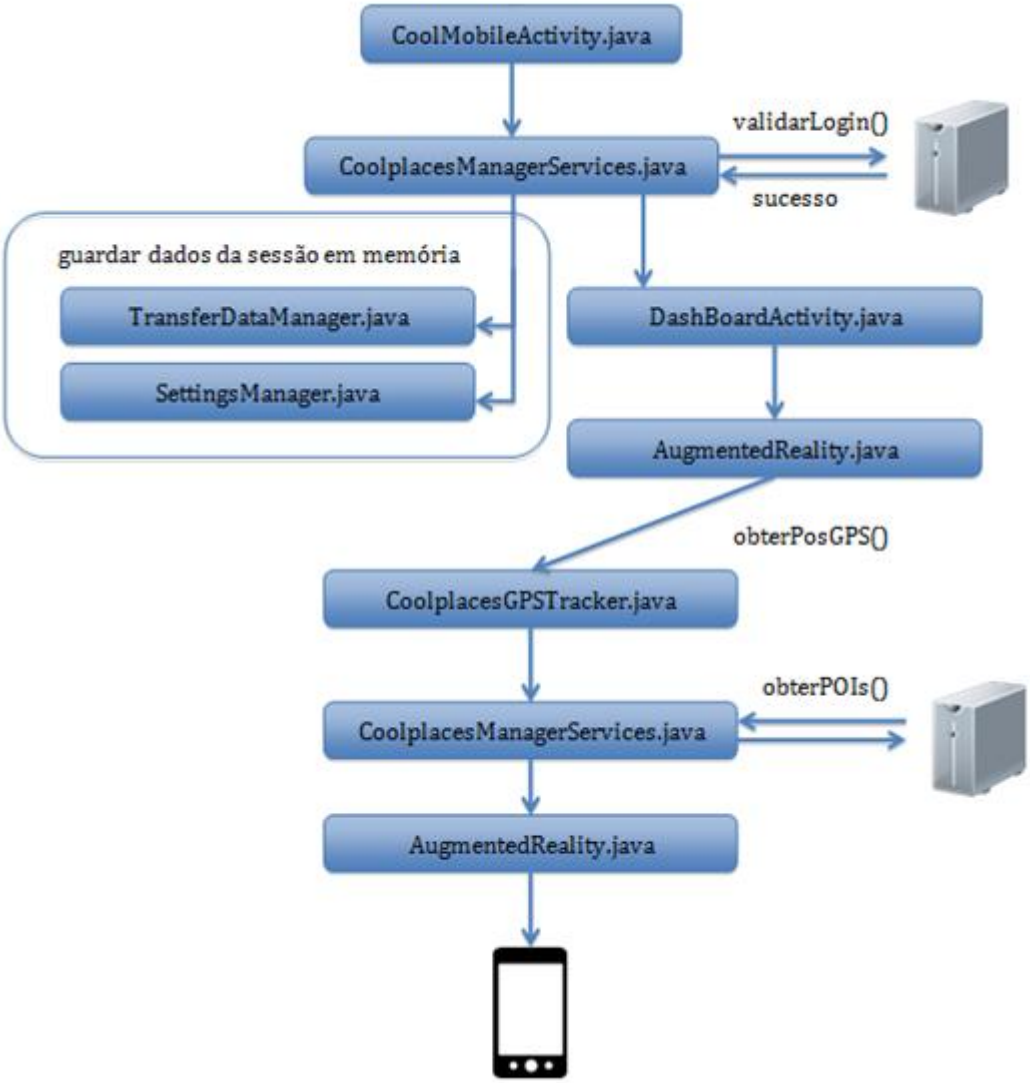


Figura 17 – Fluxo lógico da utilização do módulo de Realidade Aumentada desenvolvido para a aplicação *coolplaces mobile*

Ao iniciar a aplicação é requisitado ao utilizador que inicie sessão na plataforma *coolplaces*, sendo as credenciais verificadas via *webservice* e em caso de sucesso os dados de sessão, informação sobre o utilizador e outros indicadores de atividade da rede *coolplaces*, são armazenados através de duas classes, *TransferDataManager* e *SettingsManager*, de forma que estes dados estejam sempre acessíveis durante a utilização da aplicação. O ecrã principal da aplicação é apresentado através da *Activity DashBoardActivity*, a partir do qual podemos aceder a todas as funcionalidades da aplicação, sendo que a lógica de cada funcionalidade está organizada dentro da respetiva *Activity* numa relação funcionalidade → *Activity*, isto é, cada funcionalidade está implementada numa *Activity* própria, com exceção do novo módulo, que como já vimos é lançado numa *Activity* da própria API. Em relação à obtenção das coordenadas GPS, necessárias em grande parte das funcionalidades, incluindo no módulo desenvolvido, a principal classe é a classe *CoolplacesGPSTracker*. Foi desenvolvida de forma a

permitir a localização através de sinal GPS ou internet (WiFi ou GSM), sendo que preferencialmente utiliza o sinal GPS se este estiver disponível, utiliza a internet caso, esta esteja disponível, apenas como alternativa. Por fim, existe uma camada de acesso a dados cuja principal classe é denominada de *CoolplacesManagerServices,* classe responsável por invocar os *webservices* e da gestão da comunicação com o servidor para obtenção e publicação de dados. Os *webservices* invocados são do tipo WCF (*Windows Communication Foundation*), desenvolvidos em C# e alojados num servidor externo retornam os dados no formato JSON. Tal permite devolver objetos mais pequenos obtendo-se assim ganhos de tempo e menores taxas de transferências de dados, o que se traduz num menor custo para o utilizador. Um diagrama de classes que permite verificar o modo como estas classes estão relacionadas foi incluído em anexo e pode ser consultado na Figura 31.

De referir ainda a existência de uma camada de objetos de negócio, que contém as classes refentes aos objetos retornados pelos vários *webservices* da classe acima mencionada sendo posteriormente utilizada uma biblioteca desenvolvida na linguagem de programação *Java*, denominada *Gson* (http://code.google.com/p/google-gson/), para interpretar a informação devolvida no formato JSON e convertê-la nos respetivos objetos de negócio. O extrato de Código 16 incluído em anexo permite verificar a aplicação da referida biblioteca *Gson* e o respetivo processo de interpretação e conversão de dados.

## 5.2 Estruturas de dados

No ponto 5.1 descrevemos como os dados são obtidos, no entanto para a aplicação *ArViewer* os interpretar é necessário uma conversão para uma estrutura de dados. O *ArViewer* suporta mais do que um objeto no entanto cada tipo tem por base um objeto *GeoNode*. Torna-se assim necessário, por cada objeto *Place* devolvido pelo *webservice*, realizar uma conversão tendo por base o objeto *GeoNode*.

### 5.2.1 Objeto Place

Este é o objeto base que representa cada POI na estrutura do *coolplaces mobile*. No contexto desta implementação as principais propriedades são as seguintes.

```
public String comercial_name;
public String gps_coord_x;
public String gps_coord_y;
public double distance;
public String friends;
public byte[] image;
```

Código 4 – Extrato de código das principais propriedades do objeto *Place*

Conforme podemos verificar no extrato de Código 4, a latitude e a longitude correspondem à propriedade *gps_coord_x* e *gps_coord_y* respetivamente, a propriedade *distance* diz respeito

à distância entre o utilizador e o POI, a propriedade *friends* contêm os nomes dos amigos do utilizador que recomendam o POI separados por um delimitador (;), por fim, o objeto *image* corresponde a uma imagem do respetivo POI caso ela exista na base de dados.

### 5.2.2 Objeto GeoNode

```
private Integer mId;
private Double mLatitude;
private Double mLongitude;
private Double mAltitude;
```

Código 5 – Extrato de código das principais propriedades do objeto *GeoNode*

Este objeto, representado no extrato de Código 5, é a base da estrutura de dados do *ArViewer*, pelo que é composto pelas propriedades básicas de um POI tendo em conta a definição da sua localização: um identificador, a latitude, a longitude e a altitude. Todos os outros objetos de negócio do *ArViewer* herdam estas propriedades do *GeoNode*.

### 5.2.3 Objeto Photo

De forma a podermos visualizar a foto de um POI, caso ela exista na base de dados, foi selecionado o objeto *Photo*, que além de acrescentar as propriedades nome e descrição, permitindo assim identificar o POI e apresentar a informação de contexto, diferencia-se dos outros objetos pela propriedade *mPhotoUrl*, que permite a indicação e posterior visualização da foto do POI, conforme podemos constatar no extrato de Código 6.

```
private String mName = null;
private String mDescription = null;
private String mPhotoUrl = null;
```

Código 6 – Extrato de código das propriedades do objeto *Photo*

No entanto, as imagens dos POI armazenados na base de dados do *coolplaces* não são acessíveis por endereço URL, sendo inclusivamente esses dados retornados na forma de vetor de *bytes* pelo *webservice* utilizado. Mas numa análise mais detalhada ao objeto foi possível verificar que este converte o endereço URL recebido para um formato igual ao mencionado, sendo assim, foi efetuada uma alteração ao objeto *Photo*, de forma a realizar uma atribuição direta entre o vetor de *bytes* do nosso POI e a respetiva propriedade *mByteBitMapImage* do objeto. Essa alteração consistiu no desenvolvimento do método apresentado no extrato de Código 7.

```
public void  setImage(byte[] image) {
      mByteBitMapImage = image;
}
```

Código 7 – Extrato do código que permite atribuir o vetor de *bytes* ao objeto *Photo*

## 5.3 Novo paradigma

No decorrer da análise de soluções e numa fase inicial desta implementação, foi identificado um objeto *User* da solução *ArViewer*, que graças às suas propriedades e ao fato de poder ser configurado e modificado potenciou o surgimento de um novo paradigma.

Este novo paradigma consiste na apresentação, num módulo de Realidade Aumentada, de indicadores visuais baseados na localização de POI's, tendo por foco pessoas que tenham assinalado a sua presença nesses mesmos locais, ou seja, ao invés de assinalar POI's, os indicadores visuais indicariam pessoas na localização de um determinado POI. Seria assim possível para o utilizador identificar se alguém da sua lista de amigos do *coolplaces* teria uma grande probabilidade de se encontrar nas imediações, no momento em que estivesse a utilizar a aplicação. A lógica inerente a este paradigma é a seguinte: o utilizador 1 da rede *coolplaces mobile* ao efetuar *checkin* num dado local, está a sinalizar a sua presença nesse local, numa determinada data. Se a diferença entre essa data e a data em que um utilizador 2 estiver a utilizar o referido módulo for inferior a um determinado intervalo de tempo a definir (preferencialmente inferior a uma hora), essa relação tempo/espaço pode ser inferida e utilizada como informação de contexto.

A implementação segue a estrutura lógica do outro módulo, com a diferença de que o objeto utilizado foi o objeto *User*, ao invés do objeto *Place*, pois como o próprio nome indica esta mais vocacionado para a apresentação de dados de um utilizador. Sendo assim, foi necessário desenvolver um segundo método de *parsing* dos dados, bem como desenvolver um novo *webservice*, *getNearFriends*, que devolve uma lista de objetos *NearFriendInfo* com as seguintes propriedades.

```
public String place_name;
public String gps_coord_x;
public String gps_coord_y;
public double distance;
public ArrayList<UserCheckInInfo> users;
```

Código 8 – Extrato de código das propriedades do objeto *NearFriendInfo*

À semelhança do objeto *Place*, o objeto *NearFriendInfo,* apresentado no extrato de Código 8 contém informação relativa à latitude e a longitude nas propriedades *gps_coord_x* e *gps_coord_y* respetivamente, a distância entre o utilizador e o POI na propriedade *distance*, bem como o nome do POI referente às coordenadas GPS devolvidas. Por fim, destaque para o objeto *UserCheckInInfo* da propriedade *users*, que representa a informação dos amigos que assinalaram a sua presença no respetivo POI, assim como a indicação do tempo passado após essa mesma sinalização. Por sua vez a informação mais de detalhada dos amigos, entre as quais o nome e uma foto de perfil, é devolvida no objeto *UserEssentialInfo* que passamos a apresentar no extrato de Código 9.

```
public UserEssentialInfo user;
public String checkInElapsedTime;
```

Código 9 – Extrato de código das propriedades do objeto *UserCheckInInfo*

O modo como os objetos *NearFriendInfo*, *UserCheckInInfo* e *UserEssentialInfo* se relacionam entre si em termos de estrutura de classes pode ser observado no diagrama de classes apresentado na Figura 34 incluído em anexo.

O desenvolvimento deste módulo permite acrescentar ainda mais valor ao *coolplaces mobile*, permitindo ao utilizador descobrir se alguém da sua lista de amigos se encontra nas redondezas com a simples ação de apontar o telemóvel eu seu redor. Por outro lado, permite a um utilizador sinalizar a sua posição, se bem que indiretamente através de outra funcionalidade, se assim o entender para dar a conhecer aos seus amigos aonde se encontra.

## 5.4 Métodos de acesso aos dados

Depois de apresentadas as estruturas de dados importa explicar o processo de obtenção dos dados dos *webservices*. Como referido no ponto 5.1, estes foram desenvolvidos na linguagem de programação c# e estão alojados num servidor externo. De seguida apresentamos os interfaces dos dois *webservices* utilizados para obter os dados dos dois módulos desenvolvidos.

```
[OperationContract]
[WebGet(UriTemplate = "getPlaces/{x}/{y}/{email}")]
Stream GetNearPlaces(string x, string y, string email);
```

Código 10 – Extrato do interface do *webservice GetNearPlaces*

```
[OperationContract]
[WebGet(UriTemplate = "getNearFriends/{x}/{y}/{email}")]
Stream GetNearFriends(string x, string y, string email);
```

Código 11 – Extrato do interface do *webservice GetNearFriends*

Podemos verificar nos dois extratos apresentados que ambos os *webservices* requerem os mesmos parâmetros: latitude (x), longitude (y) e o *email* do utilizador. Cada um destes *webservices* tem um método correspondente na classe *CoolplacesManagerServices* responsável por efetuar o respetivo pedido ao servidor via HTTP[15] e de interpretar a respetiva resposta JSON, convertendo a respetiva informação numa lista de objetos *Place* ou *NearFriendInfo*. O método responsável pelo pedido ao *webservice GetNearPlaces* é o *getPlacesFromService* e o outro método responsável pelo pedido ao *webservice GetNearFriends* é o *getNearFriendsFromService*. Extratos de código dos dois *webservices* podem ser consultados no extrato de Código 14 e no extrato de Código 15 incluídos em anexo, bem como uma explicação detalhada da sua lógica.

[15] *Hypertext Transfer Protocol*, protocolo de comunicação entre sistemas de informação.

## 5.5 Parsing dos dados

De forma a fazer uso das estruturas de dados nativas do *ArViewer* foi necessário desenvolver um processo capaz de fazer corresponder as propriedades dos objetos devolvidos pelos métodos de acesso a dados, às propriedades das respetivas estruturas. De seguida apresentamos o método de *parsing* do objeto *Place*.

```
private Photo parsePhoto (Place place){
      String description = "";

       if(place.getCoolPoints() > 0)
              description = "CoolPoints: " + place.getCoolPoints() +
              "\n";

       if(place.getFriends() != null)
              description += "Friends that advise this place: "  +
              place.getFriends();

 return new Photo(0, Double.parseDouble(place.getGps_coord_x()),
      Double.parseDouble(place.getGps_coord_y()), 0.0, 0.0,
      place.getName(),description, "", "", null, "", place.getDistance()) ;
}
```

Código 12 – Extrato do código do método *parsePhoto*

Este método é chamado por cada objeto *Place* de uma lista devolvida pelo método *getPlacesFromService*. Após efetuado o *parsing* é devolvido um objeto *Photo*, com as propriedades principais do POI e com a indicação da informação de contexto que corresponde aos amigos do utilizador que o recomendam. No extrato de Código 12, podemos visualizar um objeto *Photo* a ser criado com base nas propriedades de um objeto *Place*. É aqui neste processo que escolhemos que informação destacar aquando da apresentação dos indicadores visuais dos POI's, a imagem e nome são os dois campos que são visionados dos respetivos indicadores sem ser necessário os selecionar. Posteriormente ao clicar nos respetivos indicadores visuais o utilizador tem acesso a dados de contexto mais detalhados sobre o POI correspondente. Neste caso foi feita uma construção entre o número de *coolpoints*, um sistema numérico de classificação de POI's do *coolplaces*, e a identificação dos amigos do utilizador que possuem o respetivo POI na sua lista de locais favoritos, dotando-o de informação de contexto perfeitamente identificada (não anónima) disponível para consulta se utilizador assim o desejar. É esta construção, esta informação de contexto que resulta da rede de amigos do utilizador que pretende ser um fator de decisão para este, acrescentando assim valor à típica aplicação *Ar Browser*.

Também para o novo módulo foi necessário desenvolver um método de *parsing* em tudo semelhante ao descrito anteriormente, sendo chamado por cada objeto *NearFriendInfo* de uma lista devolvida pelo método *getNearFriendsFromService*, devolvendo no final do processo um objeto *User*. Tal como no método *getPlacesFromService* é neste método que definimos que informação terá destaque no respetivo indicador visual.

```
String otherFriends = "";

for(int i = 1; i < friend.getUsers().size(); i++){
	if(i == friend.getUsers().size()-1)
		otherFriends += friend.getUsers().get(i).getUser().getUserName();
	else if (i == friend.getUsers().size()-2)
		otherFriends += friend.getUsers().get(i).getUser().getUserName() +
		" and " ;
	else
		otherFriends += friend.getUsers().get(i).getUser().getUserName() +
		", " ;
}

if(!otherFriends.equalsIgnoreCase("")){
	if(friend.getUsers().size() == 2)
		otherFriends += " was also there recently.";
	else
		otherFriends += " were also there recently.";
}
```

Código 13 – Extrato de código do método *parseUser*

Destaque para o extrato de Código 13 referente ao método de *parsing parseUser*, onde podemos verificar novamente a implementação de uma construção que vai permitir dotar os indicadores visuais com a respetiva informação de contexto, além do nome e da foto do utilizador que mais recentemente sinalizou a sua presença POI correspondente, bem como o nome desse mesmo POI.

## 5.6 Estrutura lógica da aplicação

Após a integração dos módulos desenvolvidos e da respetiva solução *ARViewer* na aplicação *coolplaces mobile* pode-se concluir que esta está estruturada de modo muito semelhante à arquitetura do estilo *Web* descrita no ponto 3.2. uma vez que é composta por um subsistema *App* que foi igualmente descrito anteriormente no ponto 3.1. Esse subsistema corresponde neste caso à aplicação *coolplaces mobile*, sendo os dados dos POI's obtidos diretamente da W*orld Wide Web*, através de *webservices*. No entanto existe uma diferença significativa a qual passamos a apresentar na Figura 18.

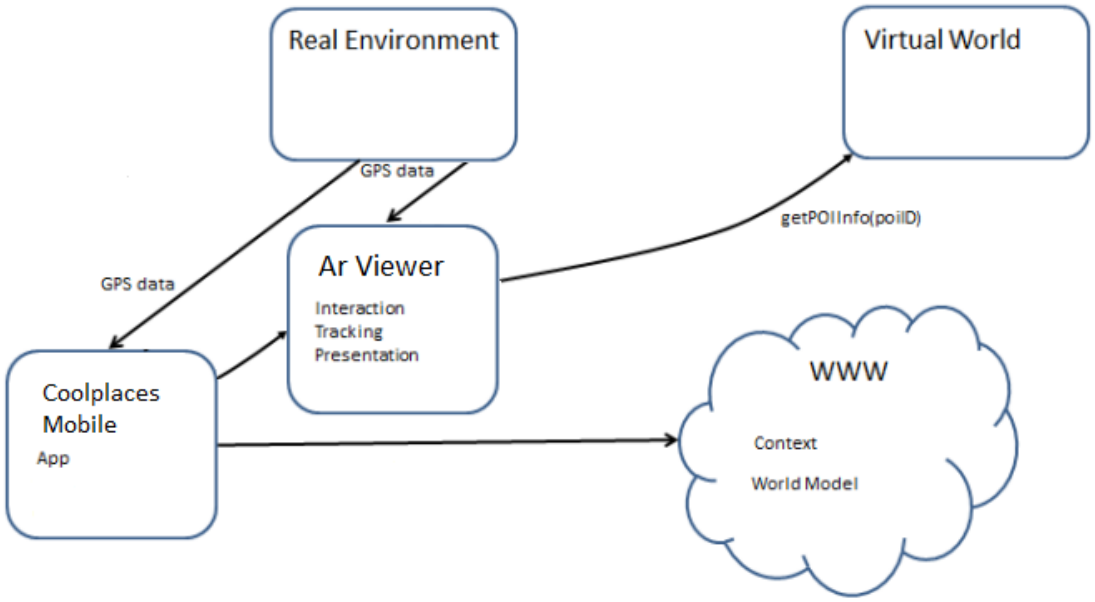


Figura 18 – Arquitetura dos módulos implementados

A grande diferença é a existência de um segundo subsistema *App*, neste caso a aplicação *ArViewer*, que é responsável pelo processo de *tracking*, pela apresentação dos conteúdos digitais e pela gestão da interação do utilizador com os mesmos. Toda a lógica da gestão da experiência de Realidade Aumentada fica assim isolada da aplicação *coolplaces mobile*, sendo esta que obtém os dados de contexto e os fornece à aplicação *ARViewer* sempre que um dos módulos de Realidade Aumentada é acionado pelo utilizador. No que diz respeito aos dados GPS ambas as aplicações conseguem obtê-los, o *coolplaces mobile* para efetuar o pedido aos *web services* dos dados de contexto, e o *ArViewer* de forma a efetuar corretamente o *tracking* dos mesmos.

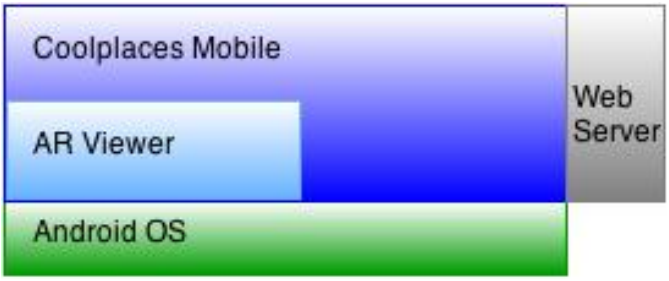


Figura 19 – Diagrama de arquitetura de blocos da aplicação *coolplaces mobile*

O diagrama da Figura 19 permite visualizar mais claramente a hierarquia existente entre a aplicação *ARViewer* e o *coolplaces mobile*. Embora ambas as aplicações tenham acesso ao sistema operativo *Android* é o *coolplaces mobile* que sinaliza ao sistema operativo que inicie a aplicação *ARViewer* sempre que tal é requerido. Podemos também verificar que não existe qualquer comunicação entre o *ARViewer* e os *webservices* alojados no servidor externo.

## 5.7 Funcionamento da aplicação

Neste ponto fazemos uma breve apresentação mais funcional da aplicação *coolplaces mobile* com destaque para os módulos desenvolvidos.

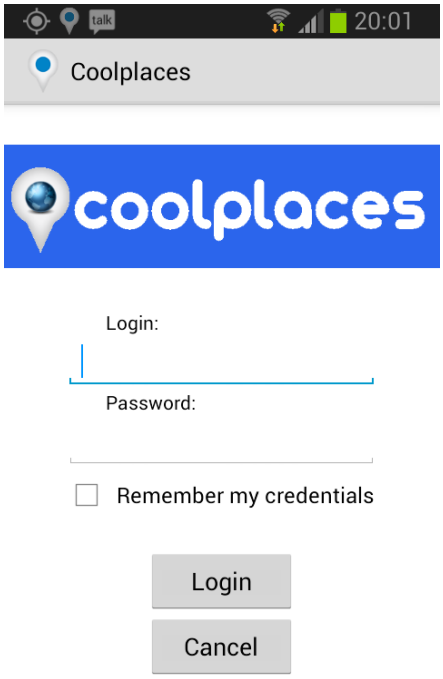


Figura 20 – Ecrã de início de sessão da aplicação

Após iniciarmos a aplicação, acedemos a um ecrã apresentado na Figura 20, onde nos é requerida a autenticação com os dados de acesso da plataforma *coolplaces*, sendo que se assim o desejarmos as respetivas credenciais podem ficar guardadas na aplicação de forma a automatizar o processo nas utilizações seguintes. Se os dados forem autenticados com sucesso temos acesso ao menu principal da aplicação (Figura 21), que é composto por um conjunto de botões que representam as funcionalidades da mesma.

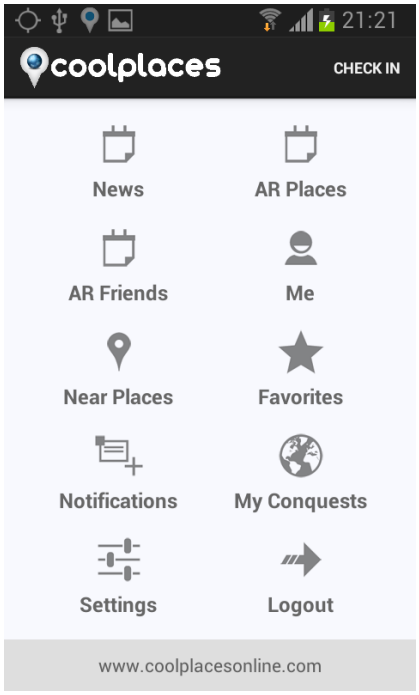


Figura 21 – Ecrã principal da aplicação

Podemos verificar na figura as funcionalidades existentes na aplicação, sendo de destacar as opções AR Places e AR Friends que dizem respeito aos módulos desenvolvidos no decorrer desta tese. A funcionalidade AR Places diz respeito ao módulo de exploração de POI's, enquanto a outra opção permite identificar amigos que tenham sinalizado a sua presença nas imediações da localização do utilizador.


Figura 22 – Módulo de exploração de POI's *AR Places*

O módulo de exploração de POI's é composto por uma vista da câmara do dispositivo móvel à qual são sobrepostas informações visuais nas localizações dos POI´s. As informações visuais de maior destaque são, o nome do POI e a imagem do mesmo. Caso esta não exista, é apresentada uma imagem genérica da API, como é o caso do POI "Fonte do Cuco" que podemos visualizar na Figura 22.

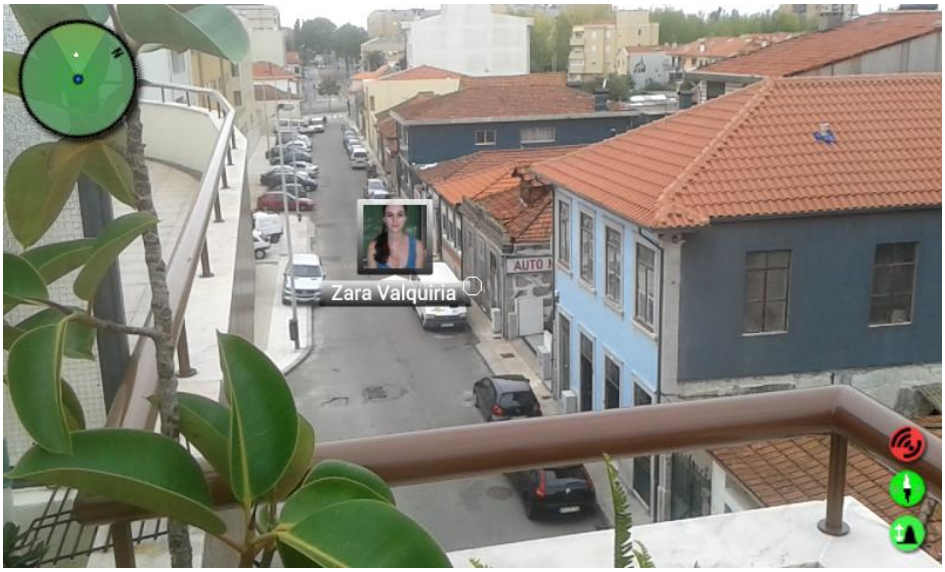


Figura 23 – Módulo de exploração *AR Friends*

O módulo de exploração que nos permite identificar amigos em redor do utilizador é em tudo semelhante ao módulo *AR Places*, sendo apenas o cariz das informações disponibilizadas diferente, assim como o modo como são obtidas. Novamente pode ser visualizada uma imagem, neste caso de um amigo ao invés de um POI e o seu respetivo nome, tudo isso localizado sob as coordenadas GPS do POI aonde a sua presença foi localizada conforme podemos verificar na Figura 23.


Figura 24 – Detalhe do módulo de exploração *AR Places*


Figura 25 – Detalhe do módulo de exploração *AR Friends*

Em ambos os módulos é possível clicar na informação visual de modo a visualizar o respetivo detalhe. Como mostra a Figura 25, o detalhe disponibilizado no módulo *AR Friends* é composto pela distância do utilizador ao ponto à qual a informação diz respeito, o nome do POI, o nome e foto do amigo que mais recentemente sinalizou a sua presença em destaque e o nome de outros amigos que também tenham sinalizado a sua presença. No módulo de vista *Ar Places* (Figura 24) o detalhe é composto pelo nome e foto do POI, a sua distância em relação ao utilizador e os nomes dos amigos que recomendaram esse POI na plataforma *coolplaces*. Em ambos os módulos é ainda possível navegar entre as várias vistas de detalhes sem ter que se selecionar as informações visuais uma a uma, bastando para isso selecionar uma e efetuar um movimento deslizante do dedo sobre a mesma para aceder à seguinte.

De modo a melhor explicitar a sequência de passos que o utilizador deve efetuar para aceder aos dois módulos desenvolvidos foram incluídos em anexo dois diagramas de caso de uso, a Figura 32 e a Figura 33, referentes ao módulo *AR Places* e *AR Friends* respetivamente.

Após a descrição do trabalho desenvolvido e respetiva implementação da solução escolhida, bem como da estrutura lógica e funcionamento da aplicação após a integração dos dois módulos desenvolvidos no âmbito desta tese avançamos para o capítulo final onde foram discutidos os resultados obtidos e retiradas as devidas conclusões relativamente aos objetivos propostos e trabalho alcançado.

# 6 Conclusões

Neste capítulo final, vamos então efetuar um balanço entre os objetivos iniciais e os resultados obtidos e suas limitações, tendo também em conta questões de usabilidade avaliadas por um pequeno grupo de teste. Para finalizar vamos discutir eventuais melhorias futuras para o trabalho desenvolvido tentando também estabelecer um paralelismo com as perspetivas de desenvolvimentos tecnológicos das tecnologias em questão.

Antes de avançar para a apresentação e análise dos testes de usabilidade é importante definir esse conceito. Neste contexto, usabilidade é a facilidade, ou falta dela, com que determinado sistema pode ser utilizado. O sistema cuja usabilidade vamos avaliar é a aplicação *coolplaces mobile*, mais particularmente os módulos *AR Friends* e *AR Places* desenvolvidos. A usabilidade advém de um conjunto de fatores a ter em conta a quando da utilização dos módulos a avaliar. Neste caso de teste os parâmetros avaliados foram os seguintes: facilidade de aprendizagem, ou seja, se é fácil ou difícil de aprender a utilizar o sistema; eficácia do sistema, se o sistema faz o que se propôs sem erros ou imprecisões; facilidade de memorização do sistema, a facilidade com que utilizador casual consegue usar o sistema esporadicamente; satisfação do utilizador, o grau de satisfação do utilizador relativamente ao funcionamento do sistema.

## 6.1 Testes de usabilidade

A aplicação foi testada por um grupo de dez indivíduos de várias idades (entre os vinte e dois e os sessenta e seis anos), todos com experiencia de utilização de *smartphones* embora três apenas utilizem normalmente as funções mais tradicionais como efetuar chamadas e enviar mensagens, não estando tão à vontade com o funcionamento de aplicações mais complexas, muito menos aplicações LBS e de Realidade Aumentada. Foi-lhes pedido que utilizassem os módulos desenvolvidos, num espaço exterior num local onde se sabia que pelo menos um POI seria identificado pela aplicação. Após os dados serem carregados e os indicadores visuais apresentados foi-lhes pedido que explorassem o ambiente através de movimentos laterais com o dispositivo até finalmente virarem a sua atenção para um POI à sua escolha e, posteriormente, o tentassem selecionar com o dedo no ecrã para poderem aceder à informação de detalhe do mesmo.

Antes de começarem os testes todos os indivíduos gostaram bastante do conceito e mostraram-se interessados e realçaram o potencial do mesmo. Os que utilizam aplicações móveis com regularidade mostraram mesmo interesse num uso continuado, opinião que mudou ligeiramente após a realização dos mesmos. De destacar duas reações distintas aquando da explicação do conceito do módulo *AR Friends*, pese embora quando conseguiram visualizar a foto de uma pessoa acharam extremamente útil para encontrar amigos num espaço de diversão noturna ao ar livre, cerca de metade dos indivíduos mostrou algumas

reservas quanto a questões de privacidade. Após a execução dos testes descritos, foi pedido a cada sujeito de teste que avaliasse cada um dos parâmetros de usabilidade apresentados no ponto anterior numa escala de um a cinco sendo um, uma má pontuação e cinco, uma boa pontuação. Podemos verificar a pontuação atribuída por cada sujeito de teste a cada um dos quatro parâmetros na Tabela 3 e a média de pontuação de cada parâmetro na Figura 26.

Tabela 3 — Avaliação dos parâmetros de usabilidade

| | **Facilidade de aprendizagem** | **Eficácia** | **Facilidade de memorização** | **Satisfação do utilizador** |
|---|---|---|---|---|
| **Sujeito Teste 1** | 5 | 4 | 5 | 4 |
| **Sujeito Teste 2** | 3 | 2 | 3 | 2 |
| **Sujeito Teste 3** | 4 | 3 | 4 | 4 |
| **Sujeito Teste 4** | 5 | 3 | 5 | 4 |
| **Sujeito Teste 5** | 3 | 2 | 3 | 2 |
| **Sujeito Teste 6** | 3 | 3 | 5 | 3 |
| **Sujeito Teste 7** | 4 | 3 | 5 | 3 |
| **Sujeito Teste 8** | 5 | 3 | 5 | 3 |
| **Sujeito Teste 9** | 5 | 4 | 5 | 4 |
| **Sujeito Teste 10** | 5 | 4 | 5 | 4 |

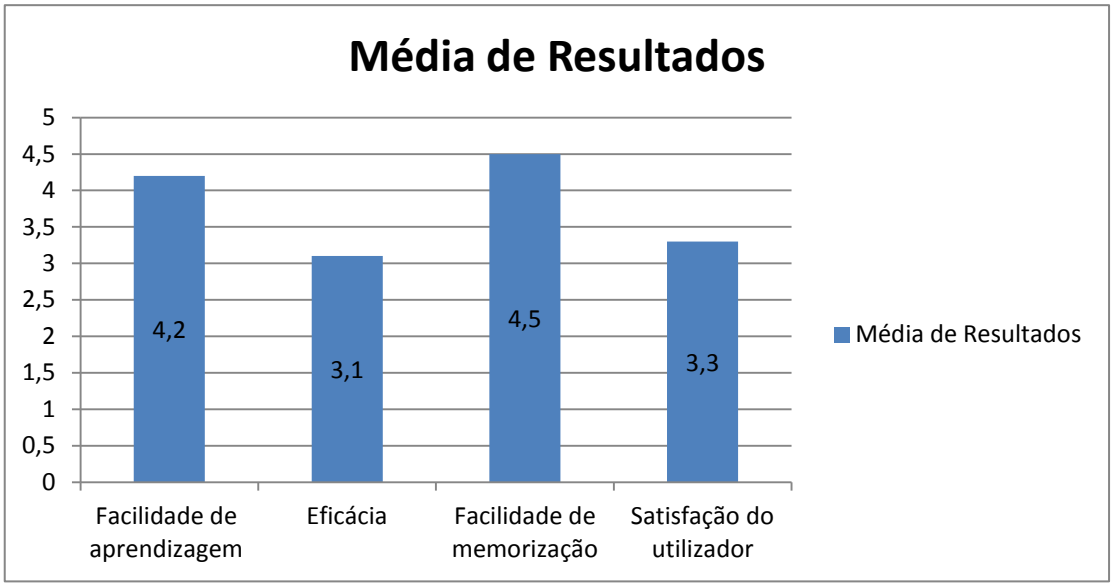


Figura 26 – Gráfico correspondente aos valores médios de avaliação de cada parâmetro

Os sujeitos de teste menos habituados a uma utilização regular de aplicações móveis evidenciaram algum receio e hesitação ao mexer na aplicação bem como apresentaram dificuldades em focar um determinado POI por terem problemas em manter o dispositivo fixo tendo efetuado alguns movimentos que fizeram oscilar os indicadores visuais, problema que também afetou outros indivíduos. A maioria dos sujeitos indicou que apesar de grande parte dos indicadores visuais estar corretamente posicionado no local onde devia a sua, a precisão deste posicionamento não era cem por cento eficaz, tendo sido identificados desvios de

alguns metros em alguns casos. Os resultados mostram precisamente que o parâmetro com pior classificação foi o da eficácia da aplicação provavelmente devido às já mencionadas oscilações dos indicadores visuais durante a exploração do meio envolvente na procura de POI's. Conforme podemos verificar na Figura 27 não houve nenhum sujeito de teste que tenha dado pontuação máxima, tendo a eficácia sido considerada como meramente suficiente pela maioria dos utilizadores.

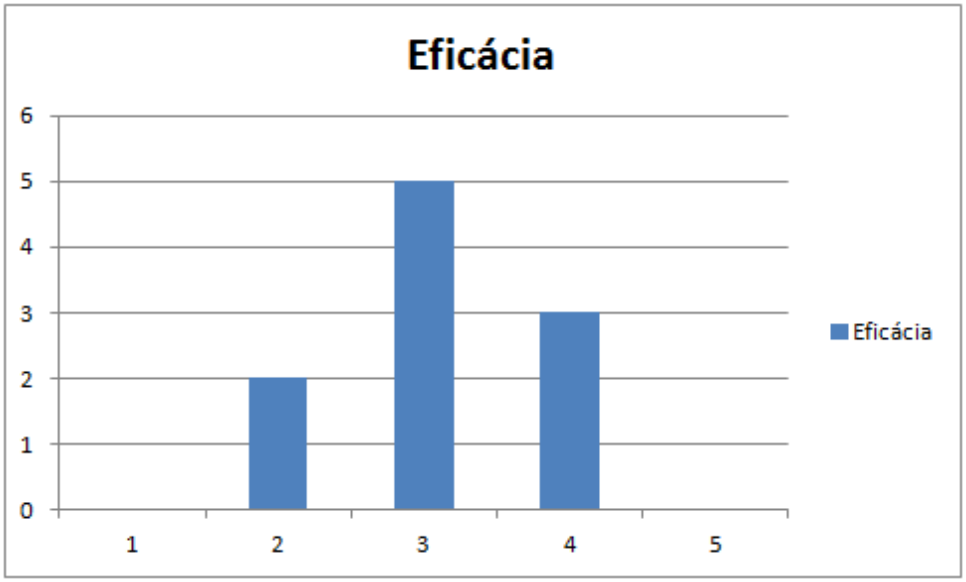


Figura 27 – Gráfico correspondente à avaliação da eficácia

Este parâmetro tem obviamente um grande impacto na satisfação do utilizador sendo que apenas dois sujeitos atribuíram uma nota diferente na comparação direta entre estes dois parâmetros sendo os resultados em tudo idênticos entre estes dois parâmetros conforme podemos verificar na Figura 26, na Figura 27 e na Figura 28. Ainda relativamente à Figura 26 os valores médios de ambos os parâmetros confirmam esse mesmo impacto.


Figura 28 – Gráfico correspondente à avaliação da satisfação do utilizador

Esta relação entre a eficácia da aplicação e a satisfação do utilizador vem aliás vem corroborar estudos já mencionados no ponto 2.3.4 que apontam falhas de usabilidade como um dos grandes entraves a uma utilização contínua deste tipo de aplicações.

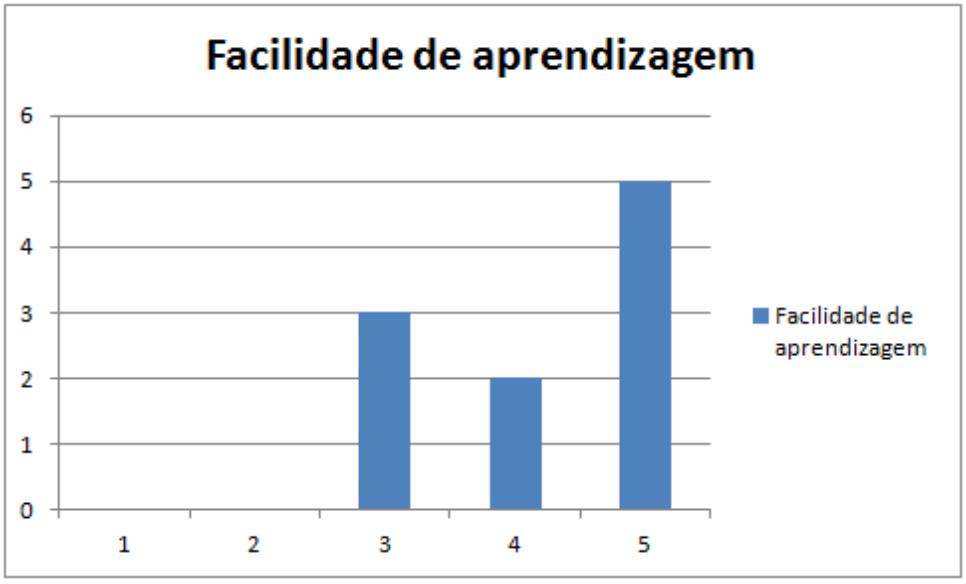


Figura 29 – Gráfico correspondente à avaliação da facilidade de aprendizagem

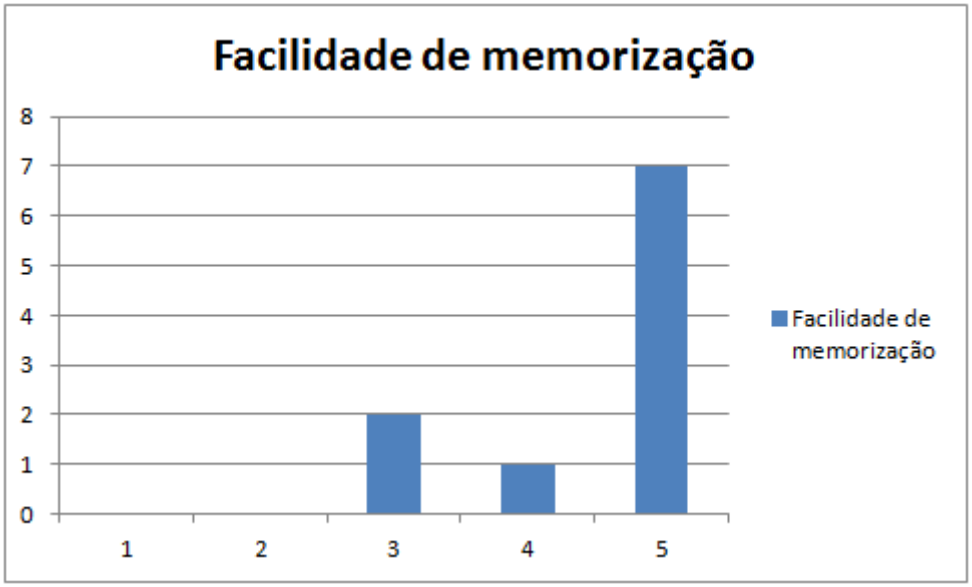


Figura 30 – Gráfico correspondente à avaliação da facilidade de memorização

Relativamente à facilidade de aprendizagem e de memorização que cujos resultados podemos visualizar na Figura 29 e na Figura 30, foi possível verificar que a maioria dos sujeitos de teste não teve quaisquer problemas nesses aspetos, exceção a três indivíduos que poderão corresponder eventualmente aos sujeitos de teste com menos à vontade na utilização de aplicações móveis, tendo inclusivamente a grande maioria dos indivíduos realçado o fato de não ser necessário efetuar muitos passos para rapidamente começarem a interagir com a aplicação, os módulos desenvolvidos e o seu conteúdo.

Os resultados obtidos vêm confirmar os estudos mencionados no ponto 2.3.4 que apontam a usabilidade como um fator chave que pode estar a atrasar a adesão às aplicações *Ar Browser* uma vez que na generalidade as pessoas gostam do conceito e afirmam antever as potencialidades do mesmo, no entanto aquando da sua utilização não ficam inteiramente satisfeitas com a sua performance. Numa nota pessoal, os resultados também reforçam a minha opinião de que ainda existe um longo caminho a percorrer até estas aplicações conseguirem conquistar os utilizadores casuais de aplicações móveis. Uma das dificuldades que senti aquando do processo de análise de soluções foi, além da falta de precisão das mesmas, o fato de que com qualquer pequeno desvio na posição do *smartphone* os indicadores visuais mudavam de posição por vezes chegando mesmo a desaparecendo totalmente do ecrã, baralhando-me. Efetivamente todas a pessoas a quem expliquei o conceito o adoraram, dando inclusivamente a entender que usariam uma aplicação que o conseguisse reproduzir, no entanto com questões como estas por resolver fica explicado a demora dos utilizadores em aderir a estas aplicações.

## 6.2 Discussão

Tendo em conta os objetivos propostos, de desenvolver um módulo de Realidade Aumentada de pesquisa de POI's dotados de informação de contexto, conclui-se que tal foi alcançado com sucesso. Foi possível desenvolver o módulo acima referido e integra-lo na aplicação *coolplaces mobile*, tornando assim possível ao utilizador usufruir desta experiencia inovadora. Posteriormente, com o surgimento do novo paradigma mencionado no ponto 5.3 foi ainda possível desenvolver um segundo módulo de Realidade Aumentada, enriquecendo ainda mais a aplicação, ao mesmo tempo que se aprofundou o estudo da solução escolhida.

De realçar que o levantamento do estado da arte efetuado permitiu aferir o atual estado de desenvolvimento da tecnologia de Realidade Aumentada e em particular das aplicações *AR Browser*. Permitiu também identificar quais as aplicações que lideram os desenvolvimentos no sentido de mitigar os desafios ainda existentes com principal impacto na usabilidade das mesmas. Limitações essas que o mesmo estudo permite concluir que advém principalmente de limitações de *hardware* dos dispositivos.

A solução escolhida, apesar das suas mais-valias, não era a solução que permitia uma melhor integração com a aplicação já existente, no entanto, como se trata de um projeto *Open Source*, o código da solução *ARViewer* está disponível para contribuições da comunidade, sendo possível efetuar alterações ao mesmo, pelo que após um estudo mais exaustivo da referida solução, ficam em aberto eventuais melhorias futuras, principalmente ao nível da já referida integração com a aplicação *coolplaces mobile*.

À semelhança dos módulos produzidos, também a tecnologia da Realidade Aumentada está ainda numa fase de evolução em que vários aspetos poderão vir a ser melhorados. Conforme referido no ponto 2.3.4 a Realidade Aumentada, principalmente aplicada a dispositivos móveis, é neste momento uma tecnologia emergente, sendo que existem algumas barreiras a ultrapassar, de índole técnica, ao nível de *hardware* e, ao nível da criação de padrões e de metodologias de desenvolvimento. Obviamente estes módulos poderão beneficiar bastante de desenvolvimentos vindouros no campo da Realidade Aumentada, principalmente ao nível da usabilidade, mas enquanto tal não sucede existem melhorias que não dependem de tais desenvolvimento e que apresentaremos de seguida.

## 6.3 Limitações

Conforme referido no ponto anterior, a solução escolhida não garante uma integração total com a aplicação *coolplaces mobile*, uma vez que quando o utilizador seleciona um dos dois módulos desenvolvidos, o que esta faz é lançar a aplicação *ArViewer* fornecendo-lhe os dados com um formato devidamente configurado de acordo com os dados que se pretende destacar. Uma das possíveis melhorias futuras passa por desenvolver esforços no sentido de obter uma integração mais completa, uma vez que tal seria benéfico não só a nível da usabilidade da aplicação como permitiria o desenvolvimento de outras funcionalidades. Ao analisar a Figura

18 e a Figura 19, chegamos rapidamente à conclusão que a aplicação *ARViewer* não tem acesso aos *webservices* do *coolplaces* o que limita o acesso e a atualização de dados de contexto, limitando também as possibilidade de interação do utilizador com o conteúdo uma vez que após os dados dos POI's serem fornecidos à aplicação *ARViewer* esta não tem como agir dinamicamente sobre eles, como obter mais detalhes, nem tão ponto como obter outros dados. O processo de integração seria no sentido de tornar a arquitetura mais semelhante à arquitetura *Web* apresentada no ponto 3.2 passando toda a lógica da gestão da Realidade Aumentada a estar concentrada num único subsistema *App*, que seria a aplicação *coolplaces mobile*. Um exemplo de um desenvolvimento que essa integração permitiria, seria a hipótese de o utilizador sinalizar a sua presença num dado POI com um simples toque na respetiva informação visual, desde que estivesse localizado dentro de um determinado alcance do mesmo. Seria também possível obter mais informações sobre determinado POI que não são incluídas nos serviços desenvolvidos no âmbito desta tese: horários, descrição mais elaborada, contactos, etc…

Uma outra limitação diz respeito à classe *CoolplacesGPSTracker* apresentada no ponto 5.1. Esta classe contém a lógica necessária ao cálculo da latitude e da longitude mas, a solução *ArViewer* requer a indicação da altitude além desses dois parâmetros para definir a localização de um POI. Uma vez que os POI's da base de dados do *coolplaces* não possuem indicação da altitude a que se encontram situados, será assim necessário efetuar algumas alterações à referida classe para que esta seja capaz de efetuar o cálculo desse mesmo parâmetro, de forma a ser possível indicá-lo corretamente. Após essa alteração seria possível calcular a altitude a que o dispositivo se encontra, definindo esse mesmo valor como a altitude padrão para todos os POI's ao alcance do utilizador, deste modo os indicadores visuais seriam sempre visíveis se o utilizador segurasse o dispositivo na sua frente, tendo apenas que se preocupar com movimentos laterais para os poder visualizar na sua totalidade.

Um aspeto a retirar das conclusões do trabalho efetuado é a existência de limitações de *hardware* que afetam negativamente as aplicações de Realidade Aumentada e os módulos desenvolvidos no decorrer desta tese não são exceção. Erros de alguns metros na precisão da localização dos POI's, oscilações momentâneas do giroscópio do dispositivo quando os módulos são utilizados em movimento, ambos são exemplos das limitações mencionadas que advêm dos dispositivos em si e não da qualidade do trabalho desenvolvido. No entanto com os contínuos desenvolvimentos efetuados na área dos sistemas móveis será talvez uma questão de tempo até que essas limitações sejam ultrapassadas.

## 6.4 Perspetivas futuras

Conforme já foi referido a adesão dos utilizadores a estas aplicações ainda é relativamente escassa, o fator novidade consegue efetivamente levar as pessoas a experimentar mas a taxa de retenção de utilizadores ativos é ainda muito baixa (cerca de um por cento). Existem questões de usabilidade que precisam de ser resolvidas mas estão em grande parte

dependentes de desenvolvimentos futuros ao nível do *hardware* dos dispositivos. Mas além desses desenvolvimentos, dos quais se espera um impacto positivo nestas questões, existem algumas melhorias que estão ao nosso alcance e que podem contribuir para um enriquecimento da aplicação *coolplaces mobile* no geral e dos módulos desenvolvidos em particular.

Uma das melhorias futuras que pode ser implementada passa por permitir ao utilizador definir qual o alcance desejado aquando da obtenção dos POI's em seu redor. Tal permitiria afinar a busca, aumentado o alcance caso os resultados fossem poucos ou insatisfatórias, ou diminuindo-o caso houvesse um excesso de informação no ecrã do dispositivo. Outra melhoria futura seria o desenvolvimento de uma funcionalidade de pesquisa que permitisse pesquisar por um POI em particular através do seu nome, tal poderia funcionar numa primeira instância sobre os POI's obtidos, sendo que se não fosse encontrado nenhum registo seria efetuado um novo pedido a um *webservice* desenvolvido para o efeito. Relativamente aos POI's, a informação de contexto disponível na plataforma *coolplaces* não se esgota na utilizada no decorrer desta tese, uma vez que existem mais dados passíveis de serem utilizados de forma a enriquecer a experiência de Realidade Aumentada. Alguns do desenvolvimentos futuros podem passar assim pela indicação de comentários dos amigos do utilizador, referentes a cada POI.

Saindo do âmbito da informação disponibilizada pelo *coolplaces*, outro caminho que pode ser seguido é a obtenção de mais dados através de bases de dados de conhecimento, de forma a complementar a informação apresentada ao utilizador. Tal informação pode ser obtida de forma gratuita recorrendo ao *Linked Open Data* de forma a enriquecer a informação de contexto apresentada ao utilizador. Trata-se de um conceito, de criação, integração e partilha de dados de forma gratuita cujos dados são disponibilizados por várias entidades tais como a BBC, o New York Times, a Newsweek, entre outros, sendo que desde 2006 mais de vinte biliões de itens foram disponibilizados (Vinny Reynolds, n. d.). Exemplos de dados geolocalizados disponíveis são os dados presentes em serviços como o *GeoNames* (www.geonames.org), *LinkedGeoData* (http://linkedgeodata.org) ou *DBpedia* (http://dbpedia.org), que incluem informações sobre POI's, localidades e respetivas populações (Vinny Reynolds, n. d.). A utilização destes dados permite o acesso a uma grande variedade de informação de contexto, que pode ser muito útil ao utilizador.

## 6.5 Considerações finais

O desenvolvimento desta tese e o trabalho despendido permitiu-me avaliar o atual estado das aplicações *AR Browser*, o que era um dos meus objetivos pessoais. Debati-me com algumas dificuldades, uma vez que os aspetos técnicos da Realidade Aumentada eram relativamente desconhecidos para mim mas aprendi bastante com a análise dos seus desafios e respetivas soluções encontradas, nomeadamente com o estudo das técnicas de visualização que me fizeram perceber muitos dos processos que uma aplicação que utilize Realidade Aumentada tem efetuar para nos poder apresentar os conteúdos virtuais que tanto entusiasmam os

utilizadores. Por outro lado, desconhecia totalmente o conceito de *AR Browser*, muito menos as aplicações existentes no mercado. Curiosamente depois de as analisar, correspondem ao que eu tinha visionado para um módulo de Realidade Aumentada da aplicação *coolplaces mobile*. Para terminar, outro aspeto onde me debati com dificuldades, tendo inclusivamente despendido uma parcela considerável de tempo foi na análise das API's e SDK's que efetuei. O processo de as tentar instalar, ver a seu modo de implementação, verificar que tipo de modificações me permitiam efetuar, e finalmente conseguir obter protótipos que me permitissem visualizar conteúdo fazendo uso da estrutura de dados do *coolplaces mobile*.

Ainda assim, após todo o trabalho efetuado creio que consegui situar corretamente o atual estado das aplicações *AR Browser*, dar a conhecer as mais relevantes em termos de mercado e o tipo de funcionalidades que cada uma oferece neste momento ao utilizador. Outra conclusão que consigo retirar foi o porquê de não ter ainda ocorrido uma adesão em massa dos utilizadores a este tipo de aplicações. Efetivamente a análise aos testes de usabilidade efetuada no ponto 6.1 vai ao encontro das conclusões dos estudos mencionados no ponto 2.3.4 e que sugerem que no atual estado de desenvolvimento destas aplicações, continuam a existir um conjunto de problemas que afetam diretamente a usabilidade afastando assim os utilizadores de um uso continuado.

# 7 Referências


A. P. FELT, E. C., S. HANNA, D. SONG AND D. WAGNER 2011. Android permissions demystified. *Proceedings of the 18th ACM conference on Computer and communications security.*

ALLIANCE, O. H. n. d. *Android Overview* [Online]. Available: http://www.openhandsetalliance.com/android_overview.html [Accessed 18/08/2013.

AZUMA, R. T. 1997. A Survey of Augmented Reality. *In Presence: Teleoperators and Virtual Environments,* 6**,** 355-385.

BEN BUTCHART 2011. Architectural Styles for Augmented Reality in Smartphones. *International AR Standards Meeting.*

BRAY, T. 2010. *What Android Is* [Online]. Available: http://www.tbray.org/ongoing/When/201x/2010/11/14/What-Android-Is [Accessed 18/08/2013.

BROOKS, F. 1996. The Computer Scientist as Toolsmith II. 61-68.

CHIN, N. J. P. 2012. *Critical Success Factors of Location-Based Services.* University of Nebraska - Lincoln.

D. MARIMON, C. S., P. CARRASCO, R. ÁLVAREZ, J. MONTESA, T. ADAMEK, I. ROMERO, M. ORTEGA AND P. GASCÓ n. d. MobiAr: Tourist Experiences through Mobile Augmented Reality.

D. PERRITAZ, C. S., D. GILLET, O. NAEF, J. BAPST, F. BARRAS, E. MUGELLINI AND O. ABOU KHALED 2009. 6th sense – toward a generic framework for end-to-end adaptive wearable augmented reality. 280-310.

DENIS KALKOFEN, C. S., SEAN WHITE AND DIETER SCHMALSTIEG 2011. Visualization Techniques for Augmented Reality. *In:* FURHT, B. (ed.) *Handbook of Augmented Reality.*

H. XU, H.-H. T., B. C. TAN AND R. ARGAWAL 2010. The Role of Push-Pull Technology. *Journal of Management Information Systems***,** 135-173.

HENRYSSON, A. 2007. *Bringing augmented reality to mobile phones.* Linköping university.

HERBERT BAY, A. E., TINNE TUYTELAARS AND LUC VAN GOOL n. d. Speeded-Up Robust Features.

IDC. 2013. *Android and iOS Combine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year, According to IDC* [Online]. Available: http://www.idc.com/getdoc.jsp?containerId=prUS23946013#.USuIPVrF2Bg [Accessed 19-10-2013.

J. B. GOTOW, K. Z., J. WHITE, AND DOUGLAS C. SCHMIDT n. d. Addressing Challenges with Augmented Reality Applications on Smartphones.

J. GRUBERT, T. L. A. R. G. 2011. Augmented Reality Browser Survey. Institute for Computer Graphics and Vision Graz University of Technology.

KAPLAN, R. 2011. *Location-Based Services Forum.* Washington D.C., United States.

LAYAR. 2011. *Layar Reality Browser* [Online]. Available: http://www.layar.com/ [Accessed 21/01/2012.

LESTER. 2012. *Readers Choice Awards 2012 – Results* [Online]. Available: http://www.augmentedplanet.com/2012/10/readers-choice-awards-2012-results/ [Accessed 19-10-2013.

LIBREGEOSOCIAL. n. d. *ARviewer SDK | LibreGeoSocial: Augmented Reality FLOSS* [Online]. Available: http://www.libregeosocial.org/node/24 [Accessed 6/07/2013.

MACWILLIAMS, A. 2005. *A Decentralized Adaptive Architecture for Ubiquitous Augmented Reality Systems.* Technische Universität München.
MAGAZINE, P. n. d. *tabler computer Definition from PC Magazine Encyclopedia* [Online]. Available: http://www.pcmag.com/encyclopedia/term/52520/tablet-computer [Accessed 18-10-2013.
MARCO ANISETTI, C. A. A., VALERIO BELLANDI, ERNESTO DAMIANI, MARIO DÖLLER, FLORIAN STEGMAIER, TILMANN RABL, HARALD KOSCH AND LIONEL BRUNIE 2011. Landmark-assisted location and tracking in outdoor mobile network.
MATTIAS ROST, H. C., NICOLAS BELLONI AND LARS ERIK HOLMQUIST n. d. Geolocation in the mobile web browser.
MICHAEL HALLER, S. D. A. W. H. 2003. A real-time shadow approach for an augmented reality application using shadow volumes. *In Proceedings of the ACM Symposium on Virtual Reality Software and Technology*.
MIXARE. n. d. *mixare | usage* [Online]. Available: http://www.mixare.org/usage/ [Accessed 3/08/2013.
OLIVIER HUGUES, J.-M. C. A. P. G. 2011. GIS and Augmented Reality : State of the Art and Issues. *Handbook of Augmented Reality.*
POELMAN, D. W. F. V. K. A. R. 2010. A Survey of Augmented Reality Technologies, Applications and Limitations. *The International Journal of Virtual Reality***,** 1-19.
RABINER, L. R. 1975. Theory and Application of Digital Signal Processing.
S. GAMMETER, A. G., L. BOSSARD, T. QUACK AND L. VAN GOOL 2010. Server-side object recognition and client-side object tracking for mobile augmented reality. *Computer Vision and Pattern Recognition Workshops (CVPRW).*
SAENZ, A. 2011. *Augmented Reality Browser Makes Your Mobile A Portal Into the Digital World* [Online]. Available: http://singularityhub.com/2011/09/09/augmented-reality-browser-makes-your-mobile-a-portal-into-the-digital-world/ [Accessed 17-10-2013.
SCHMALSTIEG, M. G. A. D. 2012. Anywhere Interfaces Using Handheld Augmented Reality.
SCOOP, P. n. d. *Smartphone definition* [Online]. Available: http://www.phonescoop.com/glossary/term.php?gid=131 [Accessed 18/08/2013.
SHUKLA, M. S. A. A. 2012. Implementation of Location based Services in Android using GPS and Web Services. *IJCSI International Journal of Computer Science Issues,* 9.
TAMARJAN, D. 2012. *Mobile AR browsers and what stops them from taking over the world* [Online]. Available: http://augmentedtomorrow.com/mobile-ar-browsers-and-what-stops-them-from-taking-over-the-world/ [Accessed 17-10-2013.
THOMAS, M. B. A. B. H. n. d. Mobile Collaborative Augmented Reality.
VÄÄNÄNEN-VAINIO-MATTILA, T. O. A. K. 2011. Expected User Experience of Mobile Augmented Reality Services. *13th International Conference on Human Computer Interaction with Mobile Devices.* Stockholm, Finland.
VARSHNEY, S. D. A. U. 2011. Challenges and business models for mobile location-based services and

advertising. *Communications of the ACM,* 54**,** 121-129.
VINNY REYNOLDS, M. H., AXEL POLLERES, MANFRED HAUSWIRTH AND VINOD HEGDE n. d. Exploiting Linked Open Data for Mobile Augmented Reality. National University of Ireland: Digital Enterprise Research Institute (DERI).
VOISARD, J. S. A. A. 2004. *Location-based Services.* San Francisco, CA.
WATSON, I. A. J. A. R. T. 2008. Location-based services. *Communications of the ACM,* 51**,** 65-69.
Z. YOVCHEVA, D. B. A. C. G. 2012. Overview of Smartphone Augmented Reality Applications for Tourism. *e-Review of Tourism Research (eRTR),* 10.

Z. ZHOU, J. K., D. HII, M. SCHNEIDER, W. LU AND S. WITTKOPF 2009. Robust pose estimation for outdoor mixed reality with sensor fusion. *UAHCI '09: Proceedings of the 5th International Conference on Universal Access in Human-Computer Interaction. Part III.* Berlin.

ZHOU, T. 2012. Examining location-based services usage from the perspectives of unified theory of acceptance and use of technology and privacy risk. *Journal of Electronic Commerce Research,* 13.

# Anexo 1 – Diagramas UML

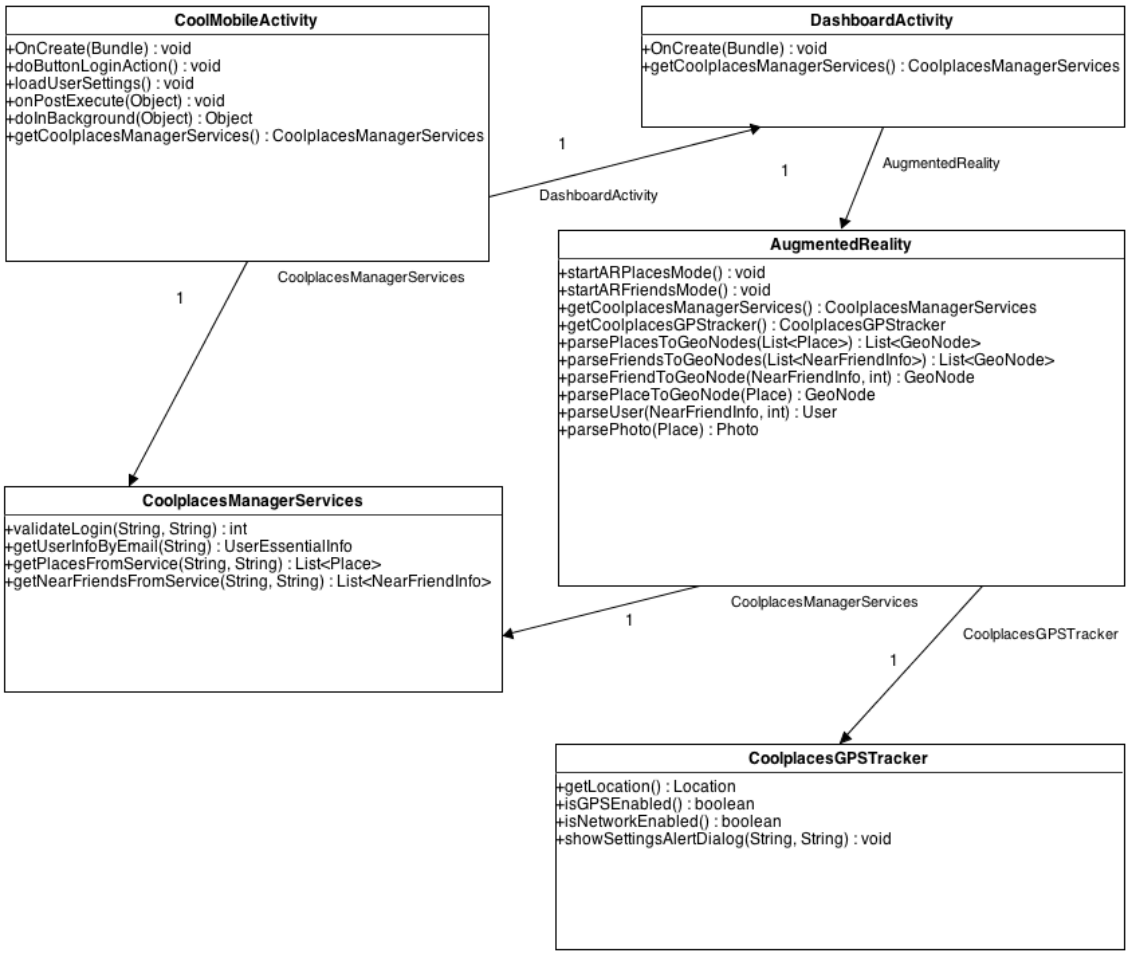


Figura 31 – Diagrama de classes da aplicação coolplaces mobile

O diagrama de classes da Figura 31 diz apenas respeito às classes utilizadas pela aplicação durante um caso de uso de um dos módulos de Realidade Aumentada desenvolvidos no âmbito desta tese. Os métodos da classe *CoolMobileActivity*, a classe respeitante ao ecrã inicial da aplicação, permitem ao utilizador autenticar-se na plataforma *coolplaces* e de seguida aceder ao ecrã principal implementado na classe *DashboardActivity*. Ao selecionar um dos dois módulos de realidade aumentada desenvolvidos, a lógica que permite a obtenção dos dados e posterior adaptação às estruturas de dados da solução *ArViewer* está implementada na classe *AugmentedReality*. A localização do dispositivo de modo a permitir obter os POI's da área circundante é calculada e obtida através da classe CoolplacesGPSTracker que é acedida pela classe *AugmentedReality*. Por fim a obtenção dos dados via *webservices* é efetuada através da classe *CoolplacesManagerServices* que é acedida tanto no processo de início de sessão pela *CoolMobileActivity*, como durante a configuração das estruturas de dados dos módulos de realidade aumentada pela classe *AugmentedReality*.

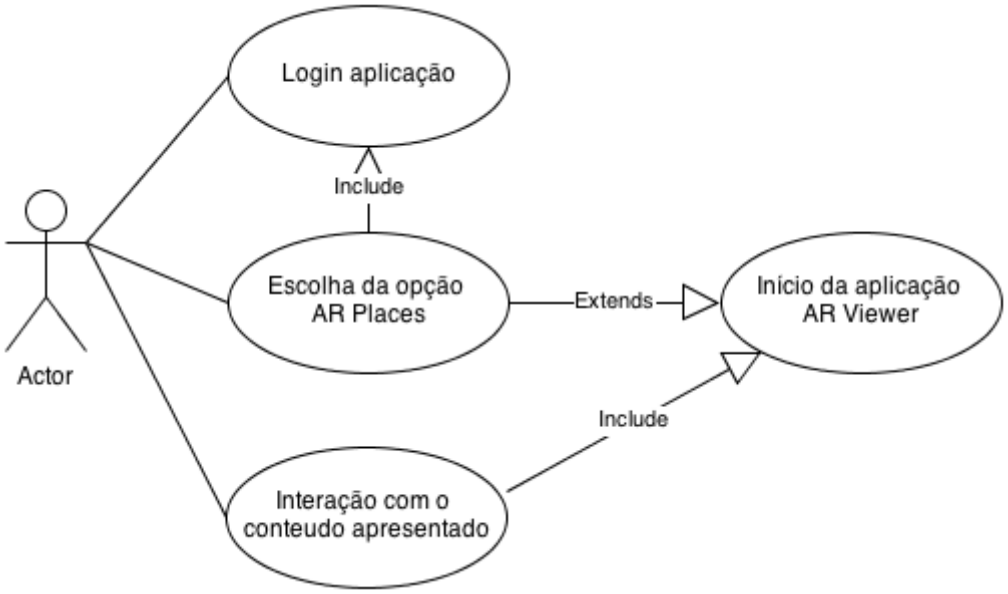


Figura 32 – Diagrama de caso de uso do módulo de exploração de POI's *AR Places*

O case de uso representado na Figura 32 representa a sequencia de ações que o utilizador tem que efetuar para aceder ao módulo *AR Places* e interagir com os indicadores visuais que representam os POI's.

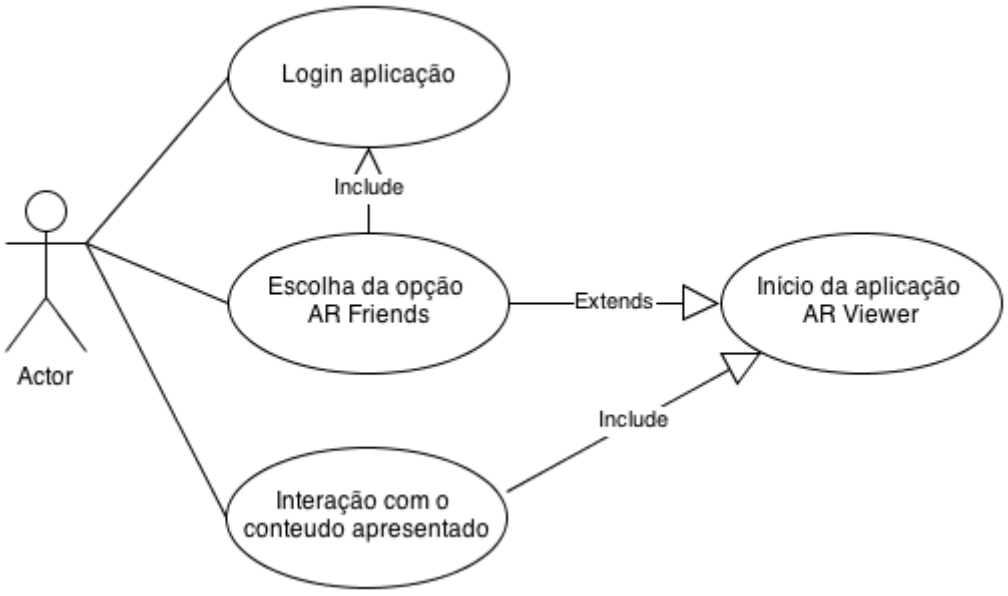


Figura 33 – Diagrama de caso de uso do módulo de localização de amigos *AR Friends*

O diagrama da Figura 33 é muito semelhante ao apresentado na Figura 32, uma vez que o processo para aceder ao módulo *AR Friends* é em tudo semelhante à sequência de ações que o utilizador tem que efetuar para aceder ao módulo *AR Places* divergindo apenas na opção selecionada.

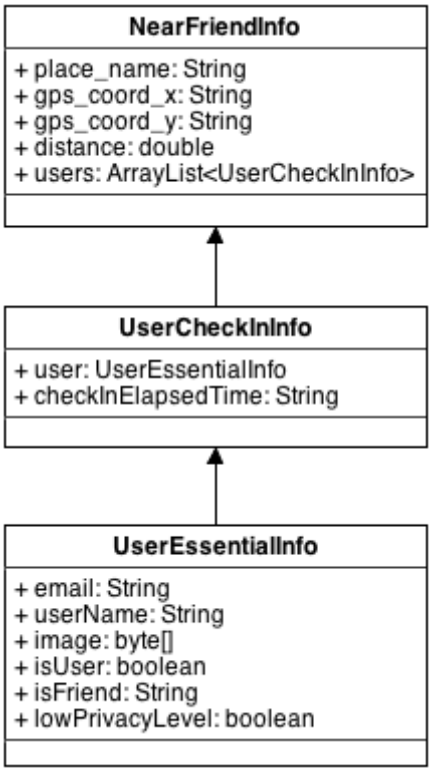


Figura 34 – Diagrama de classes que representa a relação das classes *NearFriendInfo*, *UserCheckInInfo* e *UserEssentialInfo*

# Anexo 2 – Extratos de código

Neste anexo estão incluídos alguns extratos de código relevantes para a compreensão do trabalho desenvolvido, uma vez que dizem respeito ao modo de obtenção dos dados sobre os POI's e da já referida informação de contexto que pode ser obtida através da base de dados do *coolplaces*.

```
List<CompaniesDomain> places;
List<FavouriteCompaniesDomain> favorites;
List<UsersDomain> friends;

friends = GetNearPlacesFromBd(email, gpsx, gpsy, 0.01, out favorites, out
places);

if (places.Count() > 0)
{
        foreach (CompaniesDomain place in places)
        {
               CompanyMobileInfo info = new CompanyMobileInfo();
               info.email = place.email;
               info.address = place.address;
               info.city = place.city;
               info.comercial_name = place.comercial_name;
               info.coolPoints = place.cool_points;
               info.country = place.country;
               info.gps_coord_x = place.gps_coord_x.ToString();
               info.gps_coord_y = place.gps_coord_y.ToString();
               info.phone = place.phone;
               info.distance = GetGPSPointsDistanceInMeters(gpsx, gpsy,
               place.gps_coord_x, place.gps_coord_y);

               if (place.profile_foto != null)
                      info.image = GetProfilePicture(place.profile_foto);

               if (favorites.Count() > 0)
               {
                       foreach (FavouriteCompaniesDomain friend in favorites)
                       {
                              if (friend.company_email == place.email)
                                info.friends += friends.Where(w => w.email
                                == friend.user_email).First().name + "; ";
                       }
               }
```

Código 14 – Extrato de código do *webservice GetNearPlaces*

Conforme é possível verificar no extrato de Código 14, é obtido um conjunto de dados referentes aos POI's (lista *places*) localizados na área circundante ao dispositivo, ao grupo de utilizadores que indicaram algum desses mesmos POI's como favoritos no *coolplaces* (lista *favorites*) e ao grupo de amigos do utilizador que está a utilizar a aplicação (lista *friends*). Estes dados são depois relacionados entre si para ser possível identificar a existência de amigos do

utilizador que constem da lista contendo indicações de preferência sobre determinado POI que por sua vez também esteja presente na referida primeira lista (lista *places*). Deste modo é possível fornecer esta informação ao utilizador quando ele estiver a utilizar o modo *AR Places*, sendo assim possível para ele verificar se determinado POI foi incluído como favorito por alguém da sua lista de amigos.

```
List<CompaniesDomain> places;
List<UsersDomain> friends;
List<CheckInsDomain> checkIns;

…

IList<string> placesCheckedIn = checkIns.Select(z =>
z.place_email).ToList();

places = session.Linq<onlinePlacesData.Domain.CompaniesDomain>().Where(w
=> placesCheckedIn.Contains(w.email) && w.gps_coord_x < gpsx + 0.01 &&
w.gps_coord_x > gpsx - 0.01 && w.gps_coord_y < gpsy + 0.01 &&
w.gps_coord_y > gpsy - 0.01).Take(20).ToList();

…

if (places.Count() > 0)
{
       foreach (CompaniesDomain place in places)
       {
              List<CheckInsDomain> checkInsByPlace = checkIns.Where(w =>
              w.place_email.Equals(place.email)).OrderByDescending(z =>
              z.check_in_date).ToList();

              NearFriendsMobileInfo info = new NearFriendsMobileInfo();
              info.distance = GetGPSPointsDistanceInMeters(gpsx, gpsy,
              place.gps_coord_x, place.gps_coord_y);
              info.gps_coord_x = place.gps_coord_x.ToString();
              info.gps_coord_y = place.gps_coord_y.ToString();
              info.place_name = place.comercial_name;
              info.users = new List<UserCheckInInfo>();

              foreach (CheckInsDomain checkIn in checkInsByPlace)
              {
                     UserCheckInInfo userInfo = new UserCheckInInfo();
                     userInfo.user = new UserEssentialInfo();
                     TimeSpan elapsedTime = (DateTime.Now -
                     checkIn.check_in_date);
                     string elapsedTimeString = string.Empty;
                     if (elapsedTime.Hours > 0)
                        if(elapsedTime.Hours == 1)
                            elapsedTimeString = elapsedTime.Hours + "h ";
                        else
                            elapsedTimeString = elapsedTime.Hours + "hs ";
                        if (elapsedTime.Minutes > 0)
                          if (elapsedTime.Minutes == 1)
                           elapsedTimeString = elapsedTime.Minutes + "m ";
                          else
                           elapsedTimeString = elapsedTime.Minutes + "ms ";

                        elapsedTimeString += "ago";
```

```
                    userInfo.checkInElapsedTime = elapsedTimeString;

                UsersDomain user = friends.Where(w =>
                w.email.Equals(checkIn.user_email)).First();

               userInfo.user.email = user.email;
               if (user.profile_foto != null)
                       userInfo.user.image =
                       GetProfilePicture(user.profile_foto);
               userInfo.user.isUser = true;
               userInfo.user.userName = user.name;
```

Código 15 – Extrato de código do *webservice GetNearFriends*

O *webservice* ao qual pertence o extrato de Código 15 e que devolve a informação de contexto utilizada no módulo AR Friends é ligeiramente mais complexo que o *webservice GetNearPlaces* descrito anteriormente. Inicialmente é obtido um conjunto de dados referentes ao grupo de amigos do utilizador na plataforma *coolplaces* (lista *friends*). De seguida são obtidas informações sobre os *checkins* que esses utilizadores tenham efetuado (lista *checkIns*) dentro de um determinado período de tempo (para efeitos de teste esse período ainda não foi implementado). Finalmente são obtidos os POI's (lista *places*) tendo em conta não apenas a sua localização, ou seja, se estão localizados dentro do raio de alcance do dispositivo móvel, mas também se constam na lista de *checkins* efetuados pelos amigos do utilizador. Caso existam POI's nessa condição, os dados obtidos são combinados de forma a identificar por casa POI qual ou quais os amigos do utilizador que efetuaram *checkin* nesse mesmo POI. Se existir mais do que um amigo por cada POI, o que tiver efetuado *checkin* mais recentemente, por comparação com a data do sistema, é colocado numa posição de destaque, com a sua foto e nome a serem colocadas no indicador visual posicionado na localização do POI, sendo o nome e o tempo passado desde o respetivo *checkin* dos restantes incluída como informação de contexto que pode ser acedida no modo de detalhe apresentado na Figura 25.

```
public ArrayList<NearFriendInfo> getNearFriendsFromService(String gpsx, String
gpsy) {

     MyHttpClient client = new MyHttpClient(MOBILE_SERVICE_URL);

     try {
             String email = SettingsManager.singleton().getUserEssencialInfo()
                          .getEmail();

             String Params = METHOD_GETNEARFRIENDS + gpsx + "/" + gpsy + "/" +
             email;

             client.ExecuteWithParams(0, Params);
     } catch (Exception e) {
             e.printStackTrace();
     }

     String jsonData = client.getResponse();

     Log.i("JSONDATA", jsonData);
```

```
	GsonBuilder gsonb = new GsonBuilder();
	Gson gson = gsonb.create();

	JSONArray j;
NearFriendInfo[] nearFriends = null;

try {
	j = new JSONArray(jsonData);
	nearFriends = gson.fromJson(j.toString(), NearFriendInfo[].class);
} catch (Exception e) {
	e.printStackTrace();
}

ArrayList<NearFriendInfo> listFriends = new ArrayList<NearFriendInfo>();
Log.i("NEAR FRIENDS", "-- " + nearFriends.length);

for (int i = 0; i < nearFriends.length; i++) {

	NearFriendInfo nearFriend = nearFriends[i];

	Log.i("CREATE FRIEND AT", nearFriend.getPlaceName());

	NearFriendInfo newFriend = new NearFriendInfo(
			nearFriend.getPlaceName(), nearFriend.getGps_coord_x(),
			nearFriend.getGps_coord_y(), nearFriend.getDistance(),
			nearFriend.getUsers());

	Log.i("ADD FRIEND AT", nearFriend.getPlaceName());

	listFriends.add(newFriend);
}

Log.i("LIST FRIENDS", "-- " + listFriends.size());

return listFriends;
}
```

Código 16 – Extrato de código do método *GetNearFriends* da classe *CoolplacesManagerServices*

O método apresentado no extrato de Código 16 serve como mero exemplo do processo utilizado pela aplicação para comunicar com os *webservices* e interpretar a respetiva resposta em formato JSON através da utilização biblioteca *Gson* convertendo os dados devolvidos em objetos de negócio. Inicialmente é efetuado um pedido HTTP contendo o endereço do *webservice* pretendido. Quando é obtida a resposta em texto, de acordo com o formato JSON, esta é posteriormente interpretada fazendo uso da já mencionada biblioteca *Gson* e os objetos de negócio correspondentes são preenchidos e devolvidos ao método que efetuou o pedido.