Mestrado em Engenharia Informática
Estágio
Relatório Final

Desenvolvimento e implementação de funcionalidades em Aplicações Móveis | MTG

Case study: JiTT | Just in Time Tourist

Djamilo Ascenção Monteiro
djamilo@student.dei.uc.pt

ORIENTADORES

DEI | FCTUC
Pedro Furtado

iClio | History for the New Media
Alexandre Pinto

Data: 31 de Agosto de 2012

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

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Pólo II, Pinhal de Marrocos, 3030 – 290 Coimbra

+351 239 790 000 | info@dei.uc.pt

iClio, LDA
History for the New Media

Instituto Pedro Nunes, Rua Pedro Nunes, 3030 – 199 Coimbra

+351 910 013 636 | geral@iclio.pt

Candidato
Nome: Djamilo Ascenção Monteiro
Número de Estudante: 2006130777
Contacto: djamilo@student.dei.uc.pt

Orientador | DEI
Nome: Pedro Furtado
Contacto: pnf@dei.uc.pt

Orientador | iClio
Nome: Alexandre Pinto
Contacto: alexandrepinto@iclio.pt

RESUMO

No negócio das aplicações móveis a redução de custos na mobilidade das pessoas (viagens Low Cost) abre novas oportunidades de negócio. Surgem no mercado aplicações que se direcionam para este tipo de clientes: viajantes em negócios ou independentes. Este é o mercado do Just in Time Turist (JiTT). Utilizando as funcionalidades de GPS dos equipamentos móveis, a aplicação JiTT permite que o utilizador tenha acesso a sugestões de percurso para o tempo que tem disponível para esse fim.

Este projeto de estágio tem como principal objetivo o desenvolvimento de soluções que acrescentam valor à disponibilização de um conjunto de conteúdos e funcionalidades como a integração de uma agenda cultural da cidade, uma nova solução para atualização de dados do JiTT na sua relação com o Backoffice da iClio, optimização da utilização de bateria melhorando a eficiência de utilização de recursos necessários ao funcionamento do guia e integração no JiTT um mecanismo de Realidade Aumentada.

Palavras-Chave

"MTG", "guias", "turismo", "aplicações", "usabilidade", "Just in Time Tourist", "JiTT", "android", "iOS", "developer"

Índice

Índice das Figuras

Índice das tabelas

Glossário

API	Aplication Programming Interface
App	Application
AR	Augmented Reality
CEO	Chief Executive Officer
DOM	Document Object Model
HTML	HyperText Markup Language
JiTT	Just in Time Tourist
JRE	Java Runtime Environment
JSON	JavaScript Object Notation
MTG	Mobile Tourist Guide
POI	Point Of Interest
RSS	Really Simple Syndication
SAX	Simple API for XML
SDK	Software Development Kit
XML	eXtensible Markup Language

1. Introdução

A iClio empresa de produção de conteúdos, tem o seu enfoque na transformação das oportunidades criadas pela tecnologia em potenciais negócios, sempre associada à produção de conteúdos.

O passado, a cultura, o património cultural da humanidade são uma fonte inesgotável de conteúdos, um filão de conteúdos que urge transformar em negócios rentáveis.

A iClio desenvolve o Just in Time Tourist (JiTT), uma aplicação direcionada para o viajante que em determinado momento da sua estadia se confronta com alguma disponibilidade temporal para visitar o que o rodeia e pretende sugestões adequadas. Utilizando as funcionalidades de GPS do Smartphone, a aplicação JiTT (Just in Time Tourist) permite que o utilizador tenha acesso a sugestões de percurso para o tempo que tem disponível para esse fim. Esses percursos podem depois ser personalizados, consoante os interesses e disponibilidades do utilizador.

Esta aplicação não atingiu o limite do seu potencial e o objectivo permanente da iClio passa por identificar e implementar continuamente novas funcionalidades por forma a ser referência em termos de inovação.

2. Metodologia de Trabalho e Planeamento

2.1. Metodologia

No decorrer do estágio utilizamos algumas ferramentas da metodologia scrum [6], um processo ágil, simples, leve, iterativo e incremental, centrando-se mais no software e em menos artefactos.

A empresa já possuía uma lista de funcionalidades que deveriam ser implementadas no JiTT. Essa lista, assemelha-se ao "Product Backlog" numa metodologia scrum. Não se encontrava organizada por prioridades, sendo apenas uma descrição dos requisitos que acrescentariam valor ao produto.

A partir do nosso "Product Backlog", foram calendarizados os processos de desenvolvimento, tendo sido acordado para o ano lectivo de 2011/2012 os

seguintes upgrades ao JiTT: integração de agenda cultural da cidade, uma nova solução para atualização de dados do JiTT na sua relação com o Backoffice da iClio, optimização da utilização de bateria melhorando a eficiência de utilização de recursos necessários ao funcionamento do guia e integração no JiTT Realidade Aumentada. A complexidade dos requisitos, o impacto que teriam no mercado e a possibilidade de serem implementados durante o ano lectivo, foram os critérios tidos em conta aquando da escolha dos upgrades que seriam desenvolvidos ao longo do estágio.

Os upgrades escolhidos foram então organizados de acordo com a complexidade de cada um, o nível de aprendizagem e conhecimento da arquitetura do JiTT necessários para a execução dos mesmos.

1. **Agenda de eventos –** trata-se de um upgrade simples, mas significativo no tipo de serviço disponibilizado ao cliente final, abrindo uma nova forma de abordagem a principais clientes/parcerias. Vai permitir a familiarização com a estrutura do código do JiTT e, ao mesmo tempo, pôr em prática os conhecimentos adquiridos a nível de programação para plataformas móveis.
2. **Atualização da base de dados –** é um upgrade de extrema importância para o JiTT e que irá melhorar a usabilidade, evitando que o cliente seja obrigado a descarregar toda a aplicação (aproximadamente 200MB) sempre que haja uma atualização a nível de conteúdos. Os conhecimentos adquiridos na fase anterior, à nível do código e da arquitetura do JiTT, traduzem-se numa mais valia para esta fase de desenvolvimento.
3. **Optimização da bateria –** tendo em conta a natureza da aplicação JiTT e dos recursos utilizados por este, nomeadamente o GPS, torna-se necessária uma análise do código e posterior implementação de soluções, que ao fim ao cabo, se traduzirão em mecanismos de decisão que determinarão quando o GPS deve ou não estar ligado.
4. **Realidade Aumentada –** este será a "cereja no topo do bolo". A complexidade de execução é maior e o conhecimento necessário para o seu desenvolvimento também, daí ser o último upgrade a ser implementado.

No final de cada semana reunimos presencialmente ("Review Meeting" numa metodologia scrum) para fazer uma revisão do trabalho que foi realizado durante a semana e planear os objectivos seguintes.
O acompanhamento diário é concretizado através da utilização da plataforma de gestão de projetos BASECAMPHQ (http://basecamphq.com). Esta permite calendarizar tarefas, definir objectivos, marcar reuniões, trocar ficheiros e promover uma constante relação entre colaboradores.
O CEO da iClio, Alexandre Pinto, acompanha o decorrer da execução das tarefas, assegurando o cumprimento dos objectivos e apontando as direções à seguir de modo a ter um resultado satisfatório e de qualidade. Perguntas de natureza mais técnica são dirigidas ao Gestor de Software da empresa, João Carvalho. Se pensarmos numa metodologia scrum temos o papel de "Product Owner" desempenhado pelo CEO e o de "Scrum Master" pelo Gestor de Software da iClio.

2.2. Calendarização

A iClio estrutura os seus trabalhos focada nos objectivos e através de uma relação constante entre todos os colaboradores, não determina calendarizações a curto ou médio prazo, mas apenas datas e objectivos de concretização final. A empresa estabeleceu com o orientando uma relação de confiança na gestão do tempo e processos, mas sempre acompanhada por forma a que a calendarização final fosse cumprida.

2.2.1. Primeiro Semestre

A fase inicial do estágio foi orientado no sentido de conhecer o mercado das aplicações móveis e do enquadramento do JiTT neste.
Por forma a preparar para os desafios que se iriam seguir, houve a necessidade de adquirir *know-how* em relação ao software de desenvolvimento para o sistema operativo iOS, o xCode, a linguagem de programação Objective C e as melhores práticas de desenvolvimento para os dispositivos móveis. Em relação à programação para plataformas Android, já possuía um background adquirido à disciplina de Sistemas Ubíquos (segundo semestre do mestrado).

Em acordo com o previsto (ver figura 1), até ao final do primeiro semestre, dever-se-ia ter desenvolvido e implementado a agenda de eventos. Ainda neste, dar-se-ia início ao desenvolvimento 2 - atualização da base de dados do JiTT na sua relação com o Backoffice da iClio, cujo término está calendarizado para o início do segundo semestre.

A documentação escrita de suporte ao estágio foi sendo elaborada em consonância com o desenvolvimento de software, por forma a manter um registo atualizado do desenvolvimento dos trabalhos.

2.2.2. Segundo Semestre

No segundo semestre continuou-se o desenvolvimento dos upgrades acordados no primeiro semestre. A calendarização feita no primeiro semestre (figura 2) sofreu algumas alterações: o upgrade 3, optimização no uso da bateria, foi retirado do plano de estágio por motivos que serão explicados mais a frente, na secção 6.4 Desenvolvimento III | Otimização da Bateria. Sendo assim, foi feita um novo plano (figura 3). De sublinhar que o desenvolvimento não foi sempre linear porque surgiram novos projetos fora do âmbito do estágio e que, serão, igualmente, elucidados na secção 6.4.

	Modo da	Nome da tarefa	Duração	Início	Término
1		**1º Semestre**	**113 dias**	**Qui 01-09-11**	**Seg 06-02-12**
2		**Estado da arte**	**17 dias**	**Sex 02-09-11**	**Seg 26-09-11**
3		**JiTT | Análise detalhada das funcionalidades**	**4 dias**	**Ter 27-09-11**	**Sex 30-09-11**
4		**Familiarização com as plataformas**	**22 dias**	**Sáb 01-10-11**	**Seg 31-10-11**
5		**Análise de Requisitos de desenvolvimento**	**3 dias**	**Qua 02-11-11**	**Sex 04-11-11**
6		**Agenda**	**34 dias**	**Seg 07-11-11**	**Qui 22-12-11**
7		Análise de Requisitos	5 dias	Seg 07-11-11	Sex 11-11-11
8		Desenho da Arquitectura	3 dias	Seg 14-11-11	Qua 16-11-11
9		Implementação	19 dias	Qui 17-11-11	Ter 13-12-11
10		Testes	4 dias	Qua 14-12-11	Seg 19-12-11
11		**Actualização da base de dados (inicio)**	**20 dias**	**Ter 20-12-11**	**Seg 16-01-12**
12		Análise de requisitos	6 dias	Ter 20-12-11	Ter 27-12-11
13		Desenho da Arquitectura	4 dias	Qua 28-12-11	Seg 02-01-12
14		Implementação (Início)	10 dias	Ter 03-01-12	Seg 16-01-12
15		**Revisão do relatório**	6 dias	**Ter 17-01-12**	**Ter 24-01-12**
16		**Apresentação**	9 dias	**Qua 25-01-12**	**Seg 06-02-12**

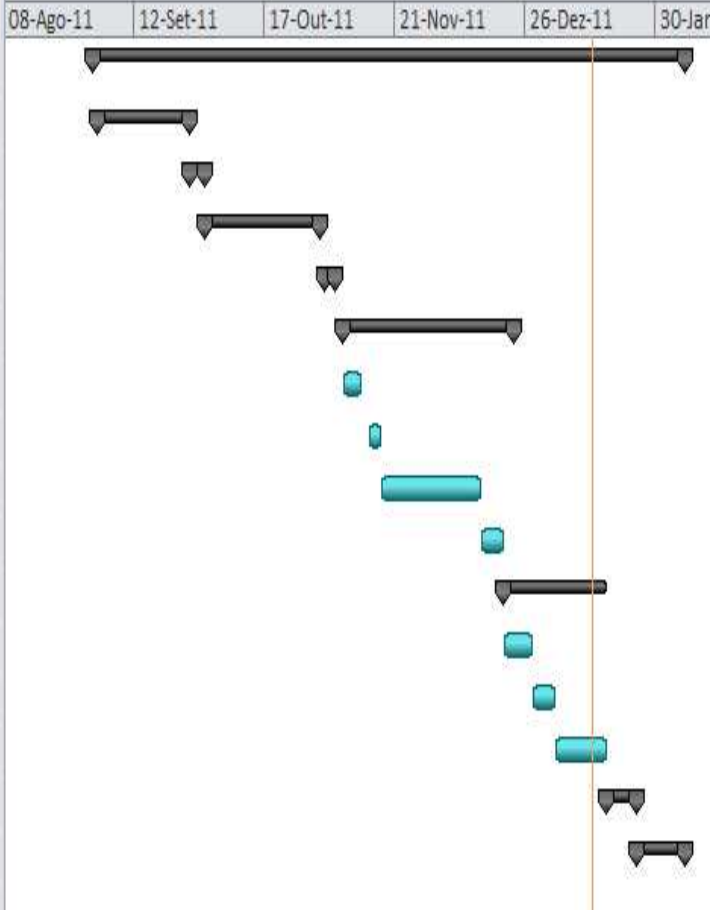

Figura 1 - Diagrama de Gantt 1º Semestre

	Mod da Tare	Nome da tarefa	Duração	Início	Término
1		**2º Semestre**	**122 dias**	**Ter 07-02-12**	**Qua 25-07-12**
2		**Actualização da Base de Dados (cont)**	**13 dias**	**Ter 07-02-12**	**Qui 23-02-12**
3		Implementação (cont)	9 dias	Ter 07-02-12	Sex 17-02-12
4		Testes	5 dias	Sáb 18-02-12	Qui 23-02-12
5		**Optimização do uso da Bateria**	**47 dias**	**Sex 24-02-12**	**Seg 30-04-12**
6		Análise do código JiTT	19 dias	Sex 24-02-12	Qua 21-03-12
7		Análise de requisitos	5 dias	Qui 22-03-12	Qua 28-03-12
8		Implementação das soluções	19 dias	Qui 29-03-12	Ter 24-04-12
9		Testes	5 dias	Qua 25-04-12	Ter 01-05-12
10		**Realidade Aumentada**	**43 dias**	**Qua 02-05-12**	**Sex 29-06-12**
11		Análise de requisitos	10 dias	Qua 02-05-12	Ter 15-05-12
12		Desenho da Arquitectura	4 dias	Qua 16-05-12	Seg 21-05-12
13		Implementação	23 dias	Ter 22-05-12	Qui 21-06-12
14		Testes	6 dias	Sex 22-06-12	Sex 29-06-12
15		**Revisão do relatório final**	**9 dias**	**Seg 02-07-12**	**Qui 12-07-12**
16		**Apresentação**	**9 dias**	**Sex 13-07-12**	**Qua 25-07-12**

Figura 2 - Diagrama de Gantt 2º Semestre (original)

	Modo da Tarefa	Nome da tarefa	Duração	Início	Término
1		**2º Semestre**	**121 dias**	**Ter 07-02-12**	**Ter 24-07-12**
2		**Actualização da Base de Dados (cont)**	**13 dias**	**Ter 07-02-12**	**Qui 23-02-12**
3		Implementação (cont)	12 dias	Ter 07-02-12	Qua 22-02-12
4		Testes	5 dias	Qui 23-02-12	Qua 29-02-12
5		**Realidade Aumentada**	**87 dias**	**Qui 01-03-12**	**Sex 29-06-12**
6		Análise de requisitos	15 dias	Qui 01-03-12	Qua 21-03-12
7		Desenho da Arquitectura	4 dias	Qui 22-03-12	Ter 27-03-12
8		Implementação	62 dias	Qua 28-03-12	Qui 21-06-12
9		Testes	6 dias	Sex 22-06-12	Sex 29-06-12
10		**Revisão do relatório final**	**17 dias**	**Seg 02-07-12**	**Ter 24-07-12**
11		**Apresentação**	**7 dias**	**Sáb 01-09-12**	**Seg 10-09-12**

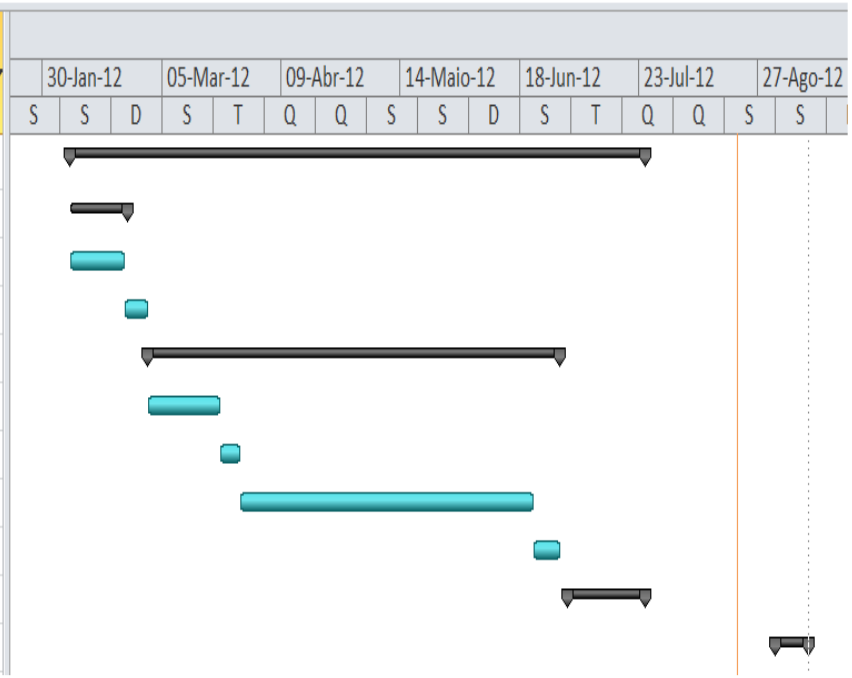

Figura 3 - Diagrama de Gantt 2º Semestre (atual)

3. Estado da Arte

3.1. Dispositivos Móveis | Tendências

Os dispositivos móveis praticamente fazem parte do nosso dia-a-dia e o papel que desempenham tem vindo a mudar a cada dia que passa. Desde consultar emails, comprar bilhetes, fazer transferências de dinheiro, tirar fotos, ver programas de televisão em direto, verificar os preços de mercadorias, etc, os dispositivos móveis têm se tornado numa ferramenta essencial que nos ajuda a navegar no nosso dia-a-dia.

Como podemos constatar no gráfico da figura 4, a previsão de crescimento na venda de dispositivos móveis, feita pela BI Intelligence, aponta um futuro de veras risonho para este sector do mercado, com os números a rondar 1.6 mil milhões de unidades em 2016.

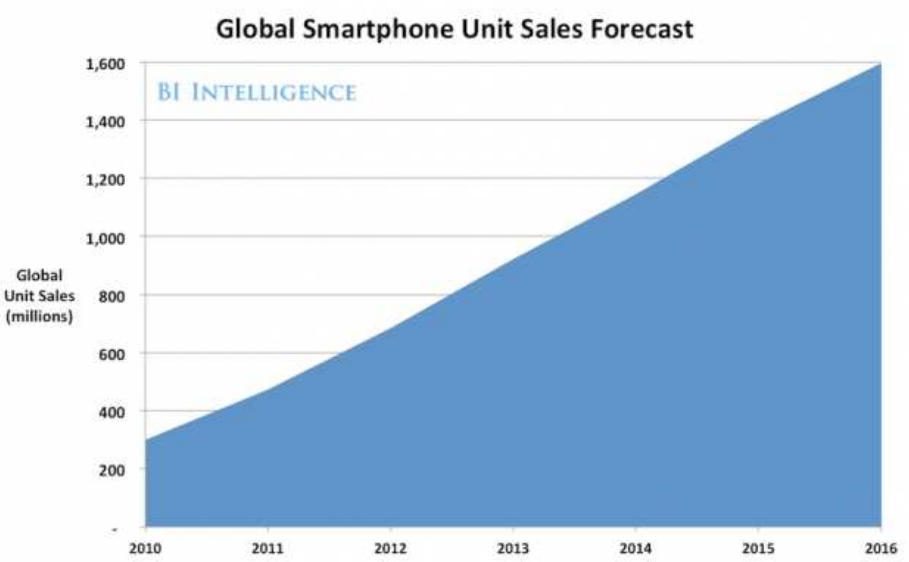

Figura 4 - Previsão de venda de Smatphones em todo mundo - em milhões de unidades (Fonte: http://www.androidauthority.com)

Um dos principais factores que contribuem para esta previsão de crescimento é claramente a queda de preço. Em 2011 o preço de um smartphone oscilou a volta dos 315 dólares e a previsão é que o custo médio caia para os 200 dólares nos dois anos que se seguem [26]. Esta queda nos preços faz com que os consumidores estejam cada vez mais inclinados a comprar estes dispositivos.

Segundo a Gartner, Inc., empresa de consultoria líder a nível mundial em pesquisas de informação tecnológicas, o total de dispositivos móveis vendidos em todo o mundo no ano de 2011 alcançaram os 472 milhões de unidades - representando 31 por cento de todas as vendas de dispositivos móveis. Enquanto isso, a International Data Corporation (IDC) acredita que 491,4 milhões de unidades de smartphones foram vendidas em 2011. Tendo em conta estes dados, a previsão de 1.6 mil milhões para 2016 não parece assim tão ridícula [1].

No presente o preço no acesso a dados via "Mobile" tem vindo a descer, assim como se torna, a cada dia, mais fácil e barato aceder à internet por diversos meios. Os custos na última década têm descido e a tendência é que este decréscimo continue. Isso também ajudou no crescimento dos pontos de acesso Wi-Fi. Praticamente hoje podemos encontrar redes sem fios em todos os lugares possíveis e imaginários desde cafés, restaurantes, praças até autocarros. Vivemos cada vez mais numa sociedade ligada a internet.

A internet contribuiu para o início de uma nova era em que os smartphones passarão a ser os dispositivos standard escolhidos pelos consumidores para se conectarem com os amigos, internet e o mundo. Se juntarmos o decréscimo dos preços, o aumento das capacidades e das aplicações para esses dispositivos, estamos perante o início de uma onda tecnológica gigante.

3.2. Aplicações Móveis | Tendências

As aplicações especialmente desenvolvidas para telemóveis fazem parte da nossa vida há mais de uma década, antes mesmo da popularização do termo "Application (App) Store". A distribuição de conteúdos e aplicações móveis muitas vezes era realizada através de portais que eram geridos pelas operadoras de rede. Estes portais tiveram algum sucesso na venda de conteúdos, mas o seu funcionamento não era assim tão claro e acabou por não atrair programadores e utilizadores suficientes para alavancar essa ideia.

A Apple App Store com uma forma fácil de encontrar aplicações e um modelo de negócio simplificado para os programadores, foi a força de ruptura que deu o pontapé inicial no mercado para as aplicações móveis. O sucesso foi imediato e na primeira semana de funcionamento, a App Store foi responsável por cerca de 10 milhões de downloads de aplicações. Hoje em dia todos os principais fornecedores de sistemas operativos e fabricantes de telemóveis têm seguido essa tendência e lançaram as suas próprias lojas de aplicações [2].

O gráfico da figura 5 mostra uma previsão de crescimento bastante elevado no número de downloads. Para o ano de 2015 prevê-se o download de cerca de 100 mil milhões de unidades – uma subida de cerca de 230% comparado com o ano de 2011.

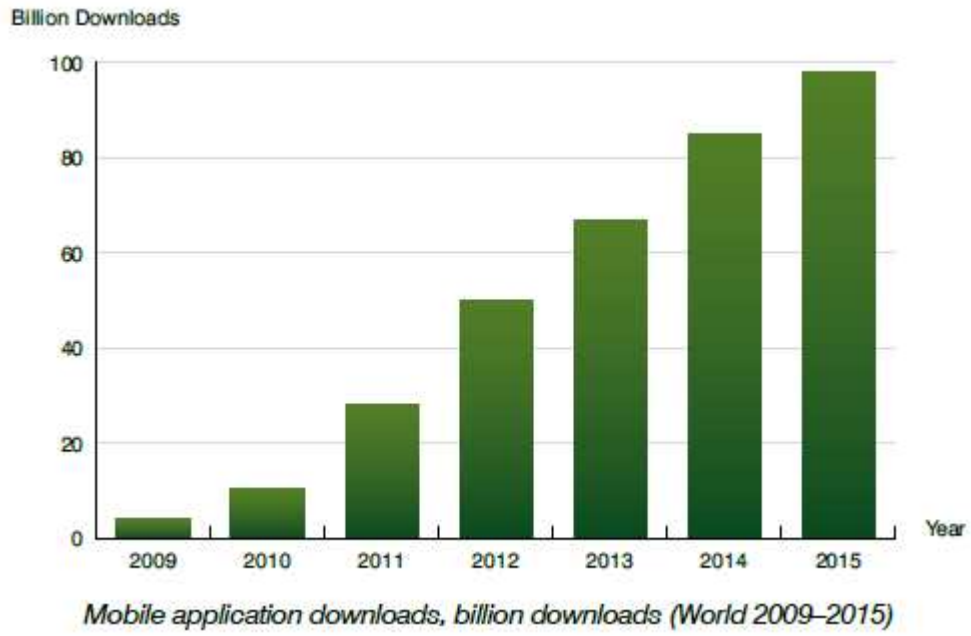

Figura 5 - Número de downloads de aplicações, em todo o mundo (em milhares de milhões) (Fonte: www.berginsight.com)

Por exemplo nos Estados Unidos da América o maior volume de downloads de aplicações gratuitas pode ser encontrado no Google Play onde o volume de downloads diário, entre o top 100 das aplicações gratuitas mais populares, foi de 4 Milhões em abril de 2012. Google Play foi seguido pela Apple App Store para iPhone (82% do volume do Google Play), a Apple App Store para iPad (20%), Amazon App Store (10%) e Windows Phone 7

Marketplace (2%)(ver figura 6). [25]

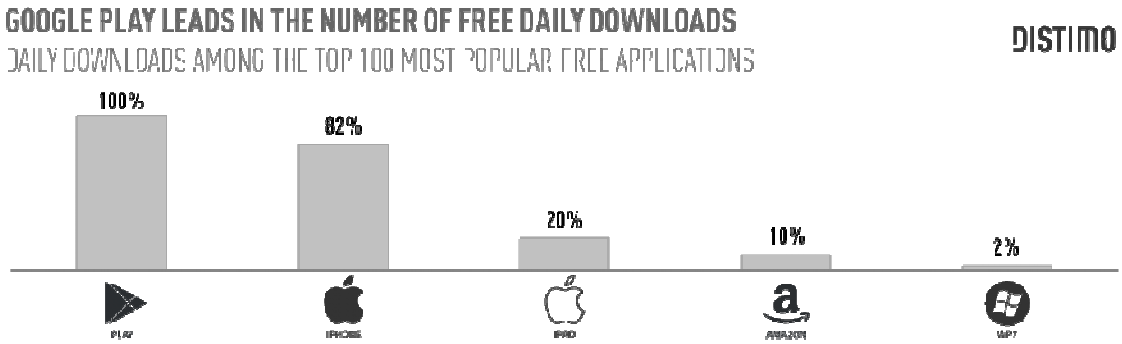

Figura 6 - Downloads de aplicações do top 100 das aplicações gratuitas mais populares (Fonte: www.distimo.com/blog)

Muitas das empresas que desenvolvem aplicações móveis começaram a desenvolver para múltiplas plataformas, como por exemplo a Rovio lançou o Angry Bird Space para Android, iOS e Mac ao mesmo tempo [25].

O top 300 das aplicações móveis gratuitas e o top 300 das aplicações pagas em abril, nas seis principais lojas de aplicações (ver figura 7), mostram que uma parte significativa das aplicações de sucesso já estão disponíveis em múltiplas plataformas. Por exemplo, 33% das aplicações mais populares da App Store da Apple para o iPhone também estão disponíveis no Google Play.

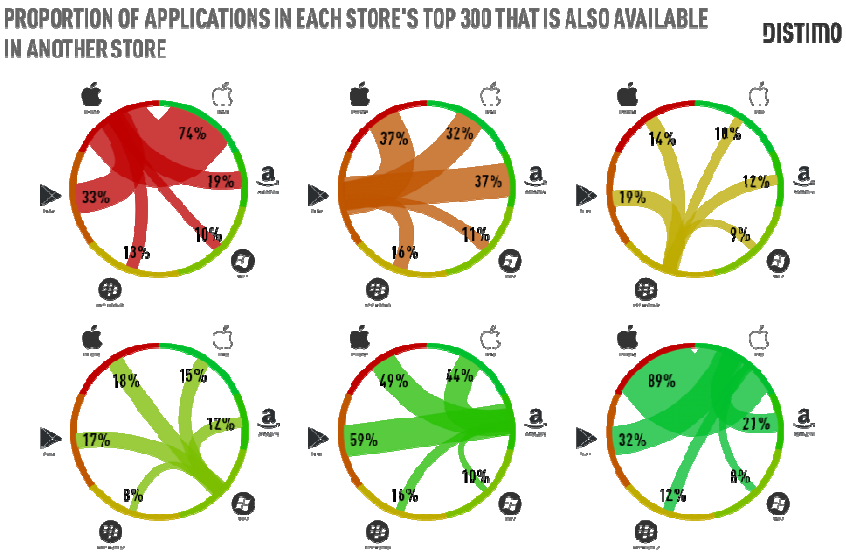

Figura 7 - Proporção de aplicações em cada Loja (Fonte: Fonte: www.distimo.com/blog)

No início de Maio de 2012 a Google Play já contava com cerca de 500 mil Aplicações em todo o mundo e na Apple App Store cerca de 600 Mil (finais de Abril). No gráfico da figura 8 são apresentados os números de aplicações nestas duas lojas desde Novembro de 2011 a Abril deste ano [25].

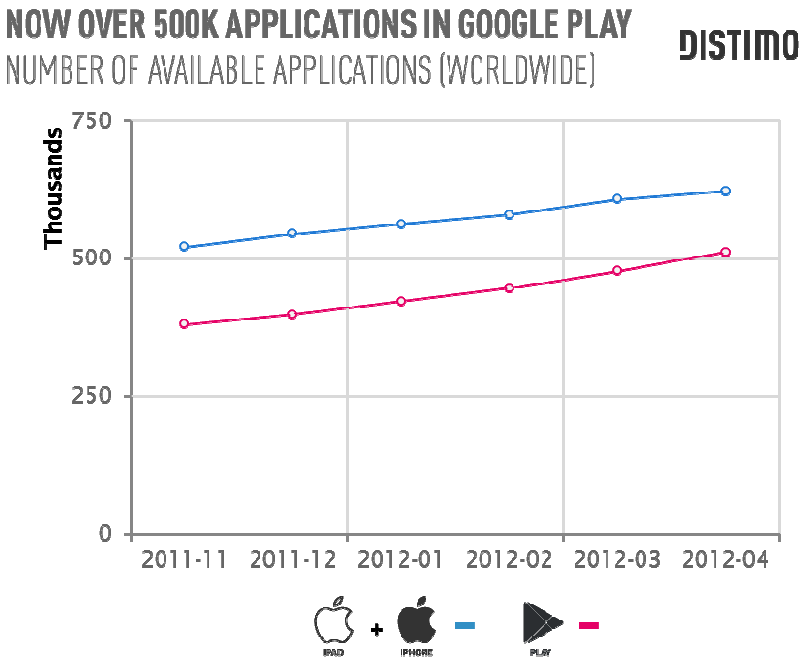

Figura 8 - Número de aplicações disponíveis (em todo o mundo) (fonte: www.distimo.com/blog)

O que se observa é que o Google Play aos poucos vai alcançando a Apple App Store para iOS, mas a diferença ainda é bastante grande (cerca de 110 Mil).
A maioria das aplicações ainda são gratuitas na primeira, apenas 28% do conteúdo é paga, enquanto que na segunda a maioria dos conteúdos são pagos [25].

4.MTG | Mobile Tour Guide

Nas últimas décadas a tecnologia evoluiu rapidamente e os computadores tornaram-se cada vez mais pequenos e disponíveis para uso pessoal, assim como para negócios. Estes avanços tecnológicos abriram uma porta a um novo mercado onde os computadores portáteis e tecnologias móveis podem contribuir para a inovação em áreas que se têm mantido idênticas ao longo de muitos anos, como é o caso do uso dos mapas tradicionais e dos guias.

Com a redução de custos na mobilidade das pessoas (viagens Low Cost) abre novas oportunidades de negócio no mundo das aplicações, como os sucessos de aplicações como Play&Tour City Guide, Navigaia PREMIUM, Lonely Planet, Rick Steves, mTrip, City Walks, jourist, Giracittá entre outros. Surgem no mercado aplicações que se direcionam para este nicho de mercado, onde os principais clientes são viajantes em negócios ou independentes. Este é o mercado do Just in Time Turist (JiTT).

O Mobile Tourist Guide tira vantagem das novas tecnologias e catapulta o turismo para um novo nível no que diz respeito a navegação e orientação dos turistas.

Mobile Tourist Guide (MTG) é a designação que se dá a aplicações para dispositivos móveis que se direcionam para área do turismo, ajudando a proporcionar uma melhor experiência aos turistas, sobre uma determinada região turística. São na sua maioria implementados como guias multimédia, mas o ênfase principal recai sobre os conteúdos.

Na verdade, nada se compara à experiência de uma visita conduzida por um guia humano altamente competente, e é isso que estas aplicações pretendem simular: um guia humano pessoal adaptado aos gostos, ao momento e às disponibilidades de quem o utiliza. A fasquia foi colocada num nível elevado, consequentemente, há sempre espaço para melhorias.

4.1. Comparação de aplicações MTG

Foram escolhidas empresas de acordo com a expressividade que têm no mercado das aplicações MTG.

Na primeira tabela comparamos as soluções existentes quanto ao número de guias e de cidades. É também dado a conhecer o preço médio de cada aplicação e a data em que ocorreu a última atualização.

O número de guias não é um fator que determina a qualidade ou o sucesso, é claro que quanto maior o número de cidades contempladas maior será a probabilidade de se ouvir falar de um determinado guia, já que consegue atingir um conjunto mais alargado de turistas. Mas para se tornar num guia de referência é preciso muito mais do que isso, pois, como se costuma dizer "quantidade não é qualidade". Novas empresas que se queiram entrar nesse mercado terão de apostar em conteúdos originais, implementar novas funcionalidades que as diferencia e pensar na melhor forma de apresentar as informações ao cliente final, portanto há que apostar também no design e na usabilidade.

Na segunda tabela apresentamos a comparação quanto aos seguintes aspectos:

- **Mapas** – aplicação integra mapas, com necessidade ou não de utilização de dados, ou seja se funciona ou não offline?
- **Imagens** – aplicação contém imagens dos locais localizados?
- **Áudio** – a aplicação oferece ao utilizador a possibilidade de ouvir instruções e narrações sobre um determinado ponto de interesse?
- **Vídeo** – a aplicação recorre de alguma forma a recursos de vídeo?
- **Offline** – a aplicação funciona offline ou precisa de uma conexão a internet para o efeito?
- **Conteúdos** – conteúdos originais, ou seja, não ser um *parsing* de sites como wikipedia ou similares.
- **Informação** – Conter informação dos pontos de interesse, como moradas, telefone, horários ou outros.
- **Categorias** – as informações estão agrupadas de acordo com a sua categoria, permitindo ao utilizador escolher entre as categorias?
- **Planeamento do tour** – a aplicação possui algum mecanismo que permite gerar automaticamente o tour?
- **Customização do tour** – o utilizador poderá editar o seu tour segundo as suas preferências (adicionar ou remover pontos de interesse)?
- **Realidade aumentada** – a aplicação permite ver a cidade, com os vários pontos de interesse, através da realidade aumentada?

(desenvolvimento e implementação no JiTT de AR)

- **Agenda de eventos** – a aplicação dispõe de alguma agenda de eventos que permite consultar os vários eventos numa determinada cidade?
  (desenvolvimento e implementação no JiTT de Agenda de Eventos)
- **iOS (iPhone & iPod), iOS(iPad) e Android** – existe uma versão da aplicação para estas plataformas?

Como podemos ver na tabela 2, a Agenda de Eventos, uma das funcionalidades a desenvolver durante o presente estágio e que apresenta uma forte utilidade, não é partilhada por mais nenhuma aplicação, o que a torna num factor inovador e diferencial.

Com o desenvolvimento incremental de guias para outras cidades e tendo em conta as qualidades apresentadas pelo JiTT neste momento, esta aplicação terá uma palavra a dizer neste nicho das aplicações MTG.

A lista completa das cidades e funcionalidades podem ser consultadas no Anexo E - Cidades e Funcionalidades.

Tabela 1 - MTG | Número de aplicações por marca

EMPRESA	**iClio**	**Lonely Planet**	**Lonely Planet**	**mTrip**	**Play&Tour City Guide**	**JOURIST**	**4 Emme Service SpA**	**Navigaia PREMIUM**	**Rick Steves**
MARCA	JiTT	Lonely Planet	Lonely Planet	mTrip	Play&Tour City Guide	JOURIST	Giracittà	Navigaia PREMIUM	Rick Steves
PREÇO MÉDIO	3,99 €	4,99 €	4,99 €	4,99 €	0,79 €	3,99 €	5,99 €	VARIOS	2,99 €
CIDADES	3	78	2	29	21	7	16	19	3
GUIAS	4	78	7	29	42	24	48	41	3
TIPO	AUDIO GUIDE	CITY GUIDE	AUDIO GUIDE	CITY GUIDE	AUDIO GUIDE	AUDIO GUIDE	AUDIO GUIDE	AUDIO GUIDE	AUDIO GUIDE
ATUALIZAÇÃO	2011	2011	2011	2011	2011	2011	2011	2011	2011
01.BARCELONA	✔	✔	✔	✔	✔	✔	✔	✔	✘
02.LONDON	✘	✔	✔	✔	✔	✔	✔	✔	✘
03.PARIS	✔	✔	✔	✔	✔	✔	✔	✘	✔
04.BERLIM	✘	✔	✘	✔	✔	✔	✔	✔	✘
05.MILÃO	✘	✔	✘	✘	✘	✘	✔	✘	✘
06.MADRID	✘	✔	✘	✔	✔	✘	✔	✔	✘
07.BRUXELAS	✘	✘	✘	✘	✔	✔	✘	✔	✘
08.PEQUIM	✘	✔	✘	✔	✘	✘	✘	✘	✘
09.VIENA	✘	✔	✘	✔	✘	✘	✔	✔	✘

✔ ***Possui um guia para a cidade***
✘ ***Não possui um guia para a cidade***

Tabela 2 - Funcionalidades oferecidas

MARCA	JiTT	Lonely Planet	mTrip	Play&Tour City Guide	JOURIST	4 Emme Service SpA	Navigaia PREMIUM
MAPAS	**OFFLINE**	**ONLINE**	**OFFLINE**	**OFFLINE**	**OFFLINE**	**ONLINE**	**OFFLINE**
IMAGENS	✔	✔	✔	✔	✔	✔	✔
ÁUDIO	✔	✘	✘	✔	✔	✔	✔
VÍDEO	✘	✘	✘	✘	✘	✘	✘
OFFLINE	✔	✘	✔	✔	✔	✔	✔
CONTEÚDOS	✔	✔	✔	✔	✔	✔	✔
INFORMAÇÃO	✔	✔	✔	✔	✔	✔	✔
CATEGORIAS	✘	✘	✔	✔	✔	✔	✔
PLANEAMENTO TOUR	✔	✔	✔	✘	✘	✘	✘
CUSTOMIZAÇÃO TOUR	✔	✘	✔	✘	✘	✘	✘
AUGMENTED REALITY (AR)	✔ *	✘	✔	✘	✘	✘	✘
AGENDA EVENTOS	✔ *	✘	✘	✘	✘	✘	✘
iOS (iPhone & iPod)	✔	✔	✔	✔	✔	✔	✔
iOS (iPad)	✔	✔	✔	✔	✔	✔	✔
Android	✔	✔	✔	✔	✔	✔	✔

✔ ***Possui a funcionalidade***

✘ **Não possui a funcionalidade**

*** Desenvolvido durante o presente estágio**

5. JiTT | Just in Time Tourist

O JiTT está disponível para as plataformas iOS e Android. Não é um simples guia áudio, nem uma aplicação que acede a "Pontos de Interesse" em torno do utilizador. O guia não pretende fornecer grandes quantidades de informação, que na prática confundem um utilizador quando condicionado por questões de tempo. Apresenta, em função das escolhas individuais do utilizador, uma solução viável e capaz de produzir uma experiência inovadora e de reconhecida qualidade.
O factor diferencial dessa aplicação, comparado à aplicações do mesmo género no mercado, é claramente a originalidade dos conteúdos, que aliados a dimensão estética, funcional e afectiva nas formas de consumo de informação, fazem com que este seja um concorrente com margem de crescimento, num mercado muito competitivo.

A ideia do JiTT não é "bombardear" o turista com muitas informações, mas sim, dar-lhe a conhecer pequenos trechos da história de uma determinada cidade ou "Pontos de Interesse" que faz com que ele se sinta um habitante da cidade e não um turista.
Uma outra característica importante é o facto de funcionar totalmente "Offline", portanto sem custos de dados, factor relevante quando estamos a pensar em turistas de outros países com contratos em diferentes operadoras. Esta característica deriva do facto de os custos de aceder em roaming à rede de dados serem sempre elevados para o utilizador final. Esta é uma mais-valia determinante para viabilizar um guia sofisticado que não esteja permanentemente ligado à rede. Tipicamente, o utilizador obtém o guia antes de partir, ou, mais provavelmente, em hotéis ou outros lugares em que dispõem de acesso Wi-Fi durante a sua viagem.
Para além das características supracitadas, o JiTT apresenta as seguintes funcionalidades:

- Seleção de Tempo: Esta aplicação é direcionada para um tipo de turista específico, aquele que por algum motivo ou outro se depara com a situação de ter algum tempo livre e querer aproveitar esse

tempo para conhecer um pouco a cidade. O JiTT permite traçar um tour baseado no tempo disponível que esta pessoa tem – 1h, 2h, 4h ou 8h.

- Utilizar ou Evitar Interiores: O utilizador poderá querer ou não que seja considerado interiores de edifícios no seu tour, como por exemplo um museu. Isto será levado em conta aquando da geração do percurso.
- Utilizar ou Evitar Transportes: O JiTT permite que o turista especifique se pretende ou não utilizar os transportes, no caso tram, metro ou transportes públicos. O sistema, também com base nesse input, calcula a rota mais eficiente.
- Voltar ou não ao ponto de partida: Será da vontade do viajante voltar ao ponto onde se encontra naquele momento ou então não se importar de terminar a viagem em outro ponto qualquer da cidade.
- Escolher o ponto de partida: A pessoa escolhe a partir de que ponto quer iniciar a sua viagem, este pode ser o ponto em que se encontra naquele exato momento ou então a partir do centro da cidade. No caso do utilizador se encontrar a uma grande distância da referida cidade, a aplicação mostra uma mensagem de alerta e pergunta se quer mesmo que o trajeto seja gerado.
- Visualização do trajeto no mapa: Todo o percurso pode ser visualizado no mapa. O turista consegue interagir com o mapa, fazendo zoom in, zoom out ou então carregando sobre os pontos de interesse (Points Of Interest – POIs).
- Visualizar uma lista com os POIS: Para além do mapa pode-se ver também os pontos de interesse numa lista. A cada POI está associado uma imagem, informação institucional, como chegar lá, um pequeno texto (apenas disponível em tablets) e um ficheiro áudio com uma descrição de até 400 palavras.
- Personalizar o percurso: O turista não se deve limitar a seguir apenas os pontos sugeridos pela aplicação, consegue incluir novos pontos na sua viagem ou eliminar aqueles que por alguma razão não quer visitar. Tendo em conta estas alterações, o sistema trata de gerar uma nova rota.

- Obter instruções sobre os transportes: Muitas vezes, para chegar a um determinado destino, dentro de uma cidade, uma pessoa tem de mudar várias vezes de transporte. Por isso é de extrema importância o turista saber quando deve mudar, em que estação deve isso acontecer e para qual das plataformas deve se dirigir para apanhar o próximo meio de transporte.

A iClio tirou proveito das novas tendências e acrescentou valor ao mercado das aplicações móveis, estando neste momento presente na loja de aplicação da Apple e Google, esperando em breve ver as suas aplicações aprovadas na loja de aplicações Amazon e Vodafone.

6. Software

As tecnologias evoluíram e evoluem rapidamente, o que ontem era novidade hoje é passado, o que torna crucial o acompanhar dessa evolução e o aproveitamento do que de melhor esse fenómeno tem para nos oferecer em cada momento. Não se pode dar nada por garantido, temos que pautar pela inovação e encarar cada produto como uma matéria prima que poderá ser transformada em algo ainda melhor.

O JiTT não obstante de ser uma boa aplicação, de já ter o seu lugar no mercado e constar da lista das dez melhores aplicações para visitar Barcelona e Roma[1], deve ser tratado como um produto inacabado, o que traduz numa oportunidade de acrescentar novas funcionalidades e melhorar ainda mais esse guia.

Nesta secção iremos analisar quatro funcionalidades, de uma lista de muitas, que carecem de ser implementadas no JiTT. Estamos a falar de uma agenda com atualização de eventos, mecanismo de atualização da base de dados, optimizações no uso da bateria e integração da Realidade Aumentada (AR).

[1] *De acordo com o site http://www.10travelapps.com*

6.1. Tecnologias

Esta secção descreve as tecnologias estudadas e implementadas nas três fases de desenvolvimento.

Android SDK [19]

Android SDK contem um conjunto de ferramentas e documentação para o desenvolvimento de aplicações para a plataforma Android. Estas ferramentas contem API´s para todas as versões de Android atualmente existentes (a partir da versão 1.5 até o mais recente, a 4.1), assim como um emulador para testes.

xCode 4.2 [20]

xCode é um ambiente de desenvolvimento integrado da Apple que permite criar aplicações para Mac, iPhone e iPad. Inclui ferramentas de análise, simulador iOS e o SDK do iOS. A interface do xCode permite a edição de código, contem o Interface Builder para o design da Interface Gráfica (UI), permite testes e debugging. Está dotado do compilador LLVM da Apple que sublinha os erros a medida que são digitadas as linhas de código, e possui um mecanismo inteligente de corrigir automaticamente esses erros.

XML [28]

eXtensible Markup Language (XML) é uma linguagem que define um conjunto de regras para a codificação de documentos em um formato que seja entendida pelos humanos e pelas máquinas. É definida na especificação XML 1.0 elaborado pela W3C[2] (World Wide Web Consortium).

JSON [19]

Java Script Object Notation (JSON) é um formato de troca de dados que representa estruturas de dados e matrizes associativas. É uma alternativa ao formato XML, e está ganhando cada vez mais importância, pois é mais leve que o anterior, o que é importante quando se trata de transmitir dados em conexões de rede.

[2] *World Wide Web Consortium (W3C) é uma comunidade internacional que desenvolve padrões abertos de modo a garantir o crescimento a longo prazo da Web[29].*

Fragment API [19]

Fragment API é um componente autónomo com a sua própria interface de utilizador e "*lifecicle*" e pode ser utilizado em diferentes partes da interface gráfica da aplicação. Foi incluída a partir do Android 3.0.

Google Geocoding[3] API [30]

Google Geocoding API oferece uma forma direta de aceder a um "*geocoder*" via HTTP, o que permite converter endereços em coordenadas geográficas. Este serviço também permite fazer a conversão contrária, ou seja, obter os endereços a partir das coordenadas. Este processo é conhecido como "*reverse geocoding*".

Simple API for XML (SAX) [32]

Simple API for XML é uma API que permite aceder aos documentos XML. É usado frequentemente por servlets e programas orientados para a rede, para transmitir e receber documentos XML, isto porque é atualmente um dos mecanismos mais rápido e que consome menos memória, quanto a manipulação de documentos XML.

System Configuration Framework [20]

System Configuration Framework (iOS) fornece funções que permitem determinar se um *host* está acessível. Isto pode ser feito de forma síncrona ou assíncrona. Fornece também mecanismos de detecção de erros.

ThreadPoolExecutor [19]

Não se trata de uma API, faz parte do Android SDK e geralmente fornece um melhor desempenho ao executar um grande número de tarefas assíncronas, devido a uma overhead reduzida na invocação, e fornecem também um

[3] *Geocoding é o processo de conversão de endereços (por exemplo "1600 Amphiteatre Parkway, Mountain View, CA") em coordenadas geográficas (como por exemplo latitude 37.423021 e longitude -122.083739) usadas para colocar marcadores e posicionar o mapa.*

mecanismo para limitar e gerir os recursos, incluindo a execução das threads.

NSOperation [31]

NSOperation tem um comportamento semelhante ao ThreadPoolExecutor mas para a plataforma iOS.

Android Camera API (android.hardware.Camera package)[19]

Android Camera API permite aceder as funcionalidades da câmara num dipositivo móvel android. Possui a classe Câmara que é usada no acesso as imagens captadas pela câmara, permite começar/parar uma visualização e obter as frames do vídeo. Esta classe atua como um cliente do Camera Service, que faz a gestão do hardware da câmara propriamente dita.

Android Location API (android.location package, LocationManager class) [19]

A Location API oferece as ferramentas necessárias para que uma aplicação possa determinar a localização do dispositivo. A localização consiste na latitude e longitude e, opcionalmente, informações sobre a altitude e a velocidade.

Android Sensor Framework (android.hardware.SensorManager package)[19]

Sensor Framework permite o acesso a uma variedade de tipos de sensores, nomeadamente: o sensor do campo magnético, o acelerómetro e o vetor de rotação.

Mixare – Open Source Augmented Reality Engine

Mixare (mix Augmented Reality Engine) [33] - é uma livraria de código aberto de realidade aumentada, publicado sob a licença GPLv3[4]. Está disponível para Android e iPhone 3GS e versões mais recentes.

[4] http://www.gnu.org/licenses/gpl-3.0.html

6.2. Desenvolvimento I | Agenda JiTT

Numa aplicação como o JiTT é uma mais valia permitir que os viajantes estejam a par dos eventos que acontecem numa determinada cidade. Com isso desenvolveu-se uma agenda que permite ao utilizador:

- consultar a lista dos eventos semanais em decurso numa determinada cidade;
- atualizar a lista sempre que necessário;
- consultar informações sobre um determinado evento;
- procurar mais informações sobre um evento caso estiver interessado (link direto para página do evento);
- ver a localização do evento no mapa.
- ver os eventos que ocorrem nos arredores dos pontos de interesse, assim como a distância a esses pontos.
- ser notificado de eventos na sua vizinhança.

Informações mais detalhadas sobre os requisitos podem ser visualizados no Anexo A - Requisitos da Agenda.
Nas secções que se seguem expõe-se com mais detalhes as componentes que constituem o módulo Agenda de Eventos, assim como as decisões tomadas.

6.2.1. Arquitetura do sistema

Em **iOS** foi criada uma diretoria "agenda" dentro do projeto JiTT onde todo o código relativo a agenda foi implementado. Esta é acionada a partir da classe principal, com a adição de um novo botão ao ecrã inicial.

Na plataforma **Android** foi seguida praticamente a mesma abordagem. Entretanto, nesta versão, a interface sofreu uma alteração maior. Utilizando o Fragment 3.0 API do Android conseguiu-se dar uma nova aparência ao ecrã principal do JiTT e mudou-se totalmente a forma como o utilizador interage com a aplicação.

A usabilidade foi tida em conta, pois este fator melhorou muito com a novo design. A agenda, em android, passou a estar associada a um Fragment[5] cujo comportamento assemelha-se ao mecanismo utilizado atualmente pelos browsers - o separador ou aba.

As versões das interfaces podem ser consultados no Anexo F - Interfaces.

A figura 9 ilustra os vários componentes que compõem a agenda, assim como as interações entre eles.

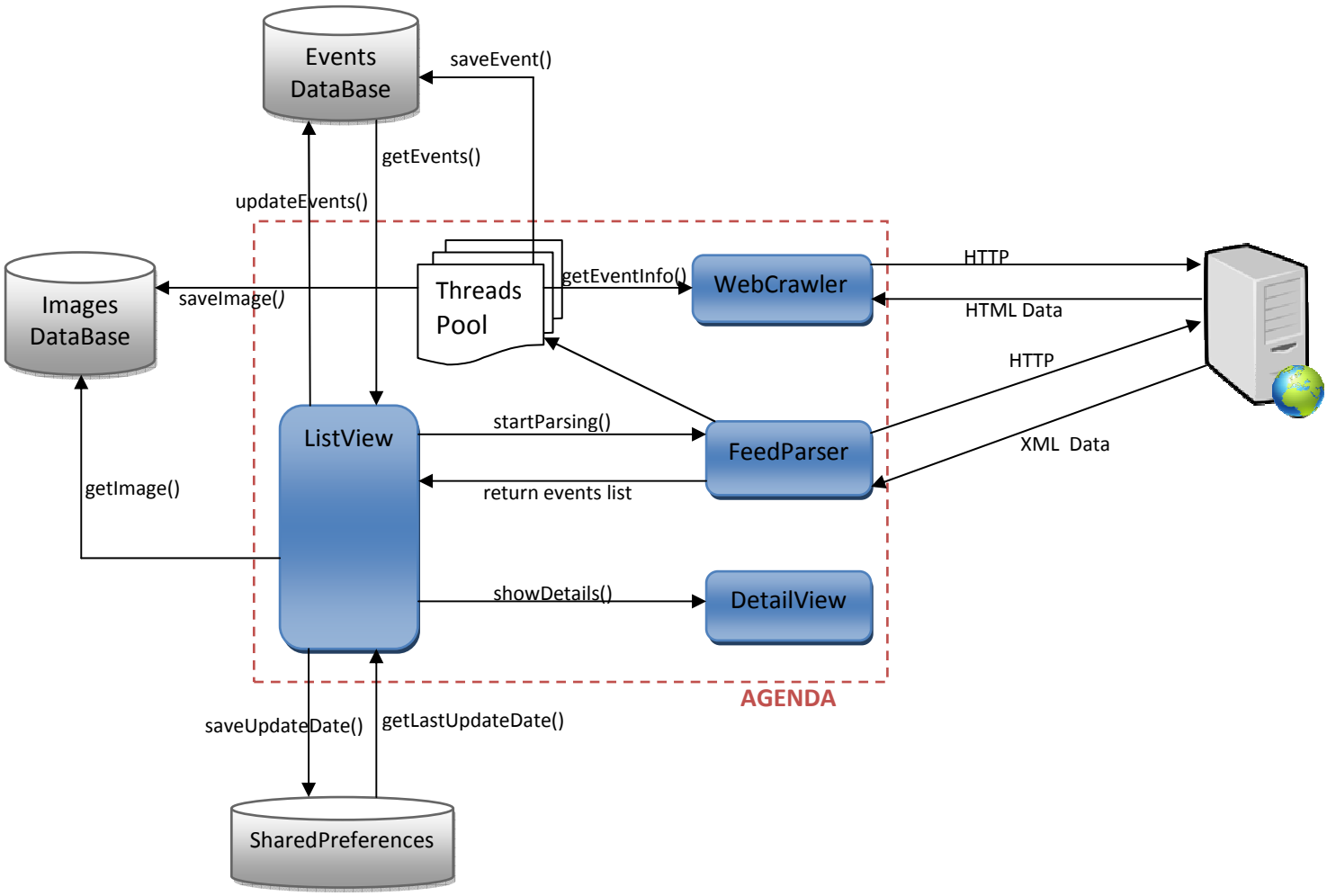

Figura 9 - Arquitetura do sistema

[5] *Fragment é uma componente autónoma com a sua própria interface de utilizador e lifecicle e pode ser utilizado em diferentes partes da interface gráfica da aplicação (mais informações em http://android-developers.blogspot.com/2011/02/android-30-fragments-api.html)*

Feed Parser

A solução ideal nesse caso seria usar um backoffice de eventos da própria empresa. Assim toda a informação necessária (data, local, endereço de imagem, etc) estaria estruturada no ficheiro XML. Entretanto não é do interesse da empresa ter o tal backoffice e a solução passa por procurar na internet por páginas fidedignas que oferecem feeds de eventos.

A forma como a agenda foi desenvolvida facilita a integração com outras páginas de eventos, requerendo para isso poucas alterações, já que os RSS feeds praticamente seguem a mesma estrutura, como se pode ver pelas figuras 10 e 11.

```
                <item>
                        <title>Real Madrid v FC Barcelona - Copa del Rey Clasico</title>
                        <link>http://www.barcelonayellow.com/bcn-events/details/1458-real
                        <guid>http://www.barcelonayellow.com/bcn-events/details/1458-real
                        <description><![CDATA[<font color = 'green'>Start date:</font> 20
<font color = 'green'>Type:</font> FCB calendar<br>]]></description>
                        <pubDate>Wed, 18 Jan 2012 07:00:00 +0000</pubDate>
                </item>
```

Figura 10 - Estrutura do feed da página www.barcelonayellow.com

```
▼<title>
    <![CDATA[ 34 Cursa del Barri de Sant Antoni ]]>
  </title>
▼<description>
  ▼<![CDATA[
      El 22 de enero de 2012 a las 09:30 de la mañana, el barrio de
      enmarca dentro de la fiesta Mayor del barrio. Se trata de una
      llega con más fuerza que nunca. Y es que el objetivo es llega
      participación que ha tenido esta carrera desde el primer día.
      abierta a la participación de todo el mundo. <BR>Esta carrera
      del barrio, la salida será en la calle Floridablanca con Comp
    ]]>
  </description>
▼<guid isPermaLink="false">
    http://www.barcelonaturisme.com/visAgenda?idAgenda=7277&lang=2
  </guid>
▼<link>
    http://www.barcelonaturisme.com/visAgenda?idAgenda=7277&lang=2
  </link>
  <pubDate>2011-12-23 12:45:32</pubDate>
  <category>Deportes</category>
</item>
```

Figura 11 - Estrutura do feed da página www.barcelonaturisme.com

ListView

Tanto no Android como em iOS são utilizadas classes do próprio SDK para apresentar a lista dos eventos, ListFragment e UITableViewController respetivamente.

Figura 12 - Android ListView

Esta classe faz a interface com o utilizador. Verifica em primeiro lugar se há uma ligação a internet, usando a classe ConnectivityManager e o System Configuration Framework, no Android e iOS, respetivamente. Em caso afirmativo, executa o método parse() da classe FeedParser, que retorna uma lista de eventos.

Fica a espera que todas as threads na pool (ContentParsers) acabem de executar. Utiliza um timeout de 60s e ao fim deste tempo é fechada a pool de threads (Neste caso a data de publicação do feed não é guardada nas preferências). Depois faz uma query a base de dados por eventos que ocorrem numa determinada semana.

Esta classe também utiliza uma thread, executada após o parsing, e que é responsável por percorrer a lista dos eventos e, através da utilização da API do Google Maps, obter as coordenadas do lugar de cada evento . Esta informação será depois usada para determinar os eventos que ocorrem nas proximidades de um determinado POI. Se a thread em background for interrompida, é gerada uma exceção e o estado em que terminou é guardado numa variável de preferência. Numa execução posterior é verificada essa variável de modo a decidir se é preciso fazer o download das coordenadas.

O carregamento das imagens do disco leva tempo, por isso, na plataforma android, foi necessário implementar um carregador de imagem assíncrono,

com um mecanismo para desenhar a imagem só depois de carregada. Na plataforma iOS, é utilizada uma cache com as imagens, portanto uma abordagem diferente. Em ambos os casos a performance da ListView é ótima o que faz com que o scroll pela lista seja muito mais fluido.

FeedParser

A classe FeedParser recebe como parâmetro o URL do feed, a data da última atualização do ficheiro XML e a data de publicação do evento mais recente na base de dados. Estas datas são guardadas utilizando as classes do SDK, SharedPreferences e UserDefaults, em android e iOS respetivamente. Estas classes guardam pares de valores <chave,valor> que só são acedidas pela aplicação que as criou.

Antes de começar o parsing dos eventos, verifica-se se houve atualização do documento no servidor, comparando as datas de publicação - a data do documento com a recebida como parâmetro. Caso não tenha havido atualizações o parsing é abortado, caso contrário, inicia-se o parsing e compara-se a data de publicação de cada evento com a do evento mais recente na base de dados. Havendo eventos com a data de publicação mais recente, são criadas novas threads, que recebem um objeto evento como parâmetro e, são adicionadas a uma pool de threads.

Estas threads, chamadas de "ContentParsers acionam o WebCrawler, passando o link para página do evento, e este, a partir de expressões regulares, procura na referida página por padrões, tais como data, link de imagens, local do evento, etc.

As ContentParsers também são responsáveis por fazer o download das respetivas imagens e guardar os eventos na base de dados, utilizando uma instância da classe DBHandler, que trata das operações sobre a base de dados. Mesmo não havendo espaço em memória para guardar as imagens, o evento é adicionado a base de dados.

XML Parser - *Plataforma Android*

Um dos pontos fortes da plataforma Android é o facto de utilizar a linguagem de programação Java. O SDK do Android não oferece tudo que está disponível no padrão Java Runtime Environment (JRE), mas suporta uma fração muito significativa do mesmo. A plataforma Java suporta muitas formas diferentes de trabalhar com XML, e a maioria dos APIs do Java para XML são totalmente suportados no Android, como por exemplo:

- Simple API Java para XML (SAX)[6] e o Document Object Model (DOM)[7] estão ambos disponíveis no Android.
- API de streaming para XML (StAX) não está disponível no Android, no entanto, este fornece uma biblioteca funcionalmente equivalente.
- Java XML Binding API também não está disponível no Android. Esta API certamente poderia ser implementada nesta plataforma, no entanto, tende a ser pesado, com muitas instâncias de classes diferentes necessárias para representar um documento XML. Por isso, não é o ideal para um ambiente restrito, como é o caso dos dispositivos móveis [3].

De acordo com [3] escolhemos o Pull Parser que funciona de forma similar ao StAX e pode ser encarada como uma forma simplificada do SAX. Possui as vantagens de ser um parser rápido, de minimizar o consumo da memória e de abortar o parsing, descartando o resto do documento. Este é um ponto importante já que os feeds dos eventos estão ordenados por datas de publicação, o que possibilita extrair só os eventos mais recentes, reduzindo o tempo de parsing. Este tipo de optimização é muito importante para os dispositivos móveis, onde a velocidade da conexão pode ser lenta.

XML parser - Plataforma iOS

Há várias opções referentes ao parsing de XML para iOS. O SDK inclui duas livrarias para este fim - NSXMLParser e o libxml2 - entretanto existem várias

[6] *Um parser SAX é aquele onde o código é notificado a medida que o parser percorre a árvore XML, e temos de manter o controlo do estado e construir os objetos que precisamos.*

[7] *Um parser DOM lê todo o documento e cria uma representação em memória e podemos fazer uma query aos diferentes elementos.*

outras livrarias de terceiros, tais como TBXML, TouchXML, KissXML, TinyXML, and GDataXML [4].

O quadro a seguir (tabela 3) compara o desempenho de cada uma dessas livrarias no parsing de um documento de aproximadamente 900KB que contem 300 músicas que estão no top da loja iTunes.

Tabela 3 - Comparação entre os vários parsers XML para iOS (Fonte: http://www.raywenderlich.com)

	NSXML	libxml2 – SAX	TBXML	TouchXML	KissXML	TinyXML	GDataXML	libxml2 – DOM
Included with SDK?	Yes	Yes	No	No	No	No	No	Yes
Seconds to Parse	1.87	1.19	0.68	1.1	1.37	1.27	1.07	0.84
Peak Memory Usage	3.11	3.01	3.07	6.5	5.25	4.8	4.15	4.97
Parse While Downloading?	No	Yes	No	No	No	No	No	No
Edit/Save XML?	No	No	No	No	Yes	Yes	Yes	Yes
XPath Support?	No	No	No	Yes	Yes	Yes*	Yes	Yes
C or Obj-C	Obj-C	C	Obj-C	Obj-C	Obj-C	C	Obj-C	C
License	Apple	MIT	MIT	MIT	MIT	ZLib	Apache	MIT

* = with TinyXPath

Tendo em conta os resultados descritos na tabela acima optamos pela libxml2-SAX porque trata-se de uma livraria que já vem incluída no SDK, é rápida e consome pouca memória, pois não tem de guardar toda estrutura do documento em memória. Embora o TBXML tenha mostrado uma performance superior ao libxml2-SAX, aquele, para além de ser proprietário, é mais indicado para documentos muito grandes que não é o nosso caso já que os feeds dos eventos, no geral, não costumam ser tão grandes. Por exemplo, na aplicação que andamos a trabalhar, JiTT Barcelona, o feed do documento XML é de apenas 44KB.

WebCrawler

A WebCrawler é a componente encarregada pela extração de informações da página de cada evento, através da técnica do Web scrapping[8]. Utiliza expressões regulares para extrair da página de cada evento o endereço da imagem, a data, a hora e o local. Foi escolhida essa abordagem por ser relativamente muito rápida e flexível e torna muito simples a adaptação do

[8] *Web scrapping é uma técnica de software utilizada para extrair informações de web sites*

código para outras páginas web, mudando apenas as expressões regulares em causa. Essa particularidade foi levada em conta, já que a agenda JiTT será implementada em vários guias para cidades diferentes.

A figura 13 mostra um exemplo de expressões regulares utilizadas para extrair as informações para o JiTT de Barcelona.

```
<!-- EVENTS -->
<string name="event_date_pattern"><![CDATA[<span class=\"textocontent4\">(.*?)<br />(.*?)<br />]]></string>
<string name="event_img_pattern"><![CDATA[<img src=\"/files/(.*?)\" alt]]></string>
<string name="event_place_pattern"><![CDATA[<div class=\"parrafo1\">(.*?)<strong>(.*?)</strong>(.*?)<br/>]]></string>
<string name="event_hour_pattern"><![CDATA[<div class=\"parrafo2\">(.*?)<strong>(.*?)</strong>(.*?)<br/>]]></string>
<string name="event_description_pattern"></string>
<string name="language_pattern"><![CDATA[<div id=\"idioma2\"><a href=\"(.*?)\"]]></string>
```

Figura 13 - Expressões regulares | JiTT Barcelona

Estas expressões regulares foram definidas, no ficheiro strings.xml em Android e no ficheiro Localized.string na plataforma iOS. Como cada guia possui um ficheiro deste tipo, é possível definir as suas próprias expressões regulares, que depois são acedidas a partir da classe WebCrawler.

No Android, cada aplicação possui também um ficheiro arrays.xml, onde é possível definir uma lista de URLs (figura 14). Assim, podem ser feitas downloads de eventos de múltiplas fontes. Uma condição é que os dados deverão ser extraídos utilizando as mesmas expressões regulares.

```
<!-- Used in EventListSupport -->
<string-array name="feeds_urls">
    <item>http://feeds.visitlondon.com/WhatsOnInLondon?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonSportsGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonSpecialEventsGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonTheatreGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonMusicGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonDanceGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonComedyGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonClubsGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/LondonArtsGuide?format=xml</item>
    <item>http://feeds.visitlondon.com/NextMonthInLondon?format=xml</item>
</string-array>
```

Figura 14 - Array de URLs | JiTT Londres - Android

O ponto crítico da componente WebCrawler é claramente o download da página Web. Num universo de mais ou menos 40 eventos, estabelecer uma conexão à página de cada um deles e descarregar o documento, não é de todo uma tarefa que demore pouco tempo. Tanto na implementação para

Android como para iOS são utilizadas threads, que executam em paralelo esse trabalho. O ganho proveniente disso é considerável, mas o uso da memória do dispositivo é maior. No entanto optou-se por essa abordagem já que lista de todos os eventos só é descarregada uma vez e nas execuções seguintes esse problema não se coloca já que só são descarregadas as páginas dos eventos mais recentes.

Esta componente poderia não existir caso a empresa possuísse um servidor de eventos próprio, ou seja, toda a informação necessária estaria devidamente estruturada no documento XML. Este seria uma otimização muito importante pois, para além de ser mais rápida (resumia-se apenas ao parsing do ficheiro XML), evitava o trabalho de andar a procura na internet por páginas de eventos que obedeçam a uma estrutura semelhante e evitava ter que definir novas expressões regulares para cada nova página encontrada.

DetailView

A DetailView está encarregada de uma tarefa relativamente simples, recebe um objeto evento da lista e apresenta ao utilizador as respetivas informações. Mostra a imagem do evento, o local, o título, horas em que o local se encontra aberto ao público, uma pequena descrição e dois botões: um que permite consultar a página do evento e outro que permite ver o local no mapa (figura 15).

Figura 15 - DetailView Android

Em ambas as implementações, Android e iOS, quando acionada o botão "ver no mapa", é executada a aplicação Google maps, que já vem instalada nos dispositivos, passando como parâmetro o endereço do local. Esta, caso houver uma ligação a internet, faz a query e apresenta no mapa o local do evento.
No caso da consulta da página do evento, utilizou-se uma abordagem idêntica, com recurso ao Browser do próprio dispositivo, que recebe o URL da página.

Bases de Dados

Base de dados dos Eventos

Tanto o Android como o iOS utilizam o SQLite - uma biblioteca em linguagem C, de tamanho reduzido, aproximadamente 275KB, no entanto eficiente e robusta na criação e manipulação de bases de dados relacionais; é de domínio público e é uma boa alternativa para os dispositivos móveis [5].

Como o JiTT é um guia que opera praticamente em offline, todos os eventos são guardados na base de dados para uso a posteriori. Com isso há garantias de que a agenda pode ser consultada a qualquer momento, o que evita custos elevados de acesso em roaming a uma rede de dados e a situação do viajante andar sempre a procura de locais com acesso a uma rede Wi-Fi.

A base de dados consiste numa única tabela, "events", que contem 14 colunas. Os campos, bem como o tipo de dados podem ser visualizadas na figura 16. A tabela pode ser estendida para desenvolvimento futuro. A classe DBHandler dispõe de uma constante que determina a versão da base de dados. Se houver alguma modificação - adição/remoção de colunas - a incrementação da variável faz com que seja removida a base de dados antiga e criada a nova.

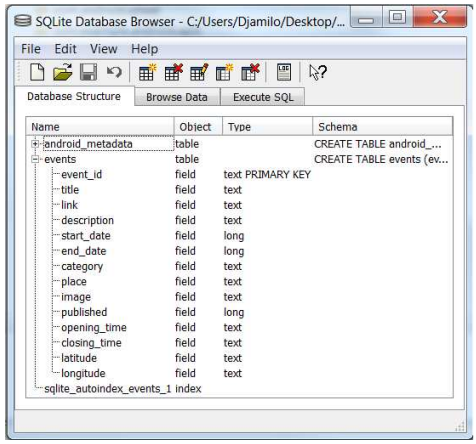

Figura 16 - Base de dados SQLite

Base de dados das Imagens

As imagens são guardadas no cartão de memória, numa pasta que é removida caso a aplicação seja desinstalada. O endereço da imagem é guardado na variável "image" do objeto Evento.

6.2.2. Testes

O teste consistiu em:

1. Verificar como se comporta a agenda quando é acionada a primeira vez e o dispositivo não se encontra ligado a rede.
2. Analisar o comportamento da agenda quando o dispositivo não está ligado a rede mas com a base de dados dos eventos já criada.
3. Verificar o comportamento caso haja uma falha na rede aquando do download dos eventos
4. Certificar de que toda a informação guardada está de acordo com a página dos eventos

Tabela 4 - Agenda | Resultados dos testes

Teste	Resultado	Descrição
1	✓	O sistema apresenta uma mensagem de alerta informando o sucedido e mostra o botão "Retry" para que se possa carregar os eventos novamente
2	✓	Passa despercebido ao utilizador final. O sistema carregou os eventos da base de dados e apresentou no ecrã a lista dos mesmos

3	✓	Utilizando um mecanismo de timeout, a agenda interrompe o parsing ao fim deste tempo e caso já tiver algum evento na base de dados, este é apresentado ao utilizador. No entanto, da próxima vez que voltar a ligar retoma a atualização a partir da ultima data de atualização com sucesso
4	✓	O sistema carrega todos os eventos da página. Foram escolhidos 15 eventos aleatoriamente que foram analisados e comparados com os que estavam na base de dados de modo a certificar a consistência dos dados.

A versão do JiTT de Barcelona e Londres, para Android e iOS, já se encontram nas respetivas lojas. No Android Play Store há seis meses e na Apple App Store há 4 meses. Até então apenas um erro foi detetado e corrigido.

Tabela 5 - Agenda | bug

Erro	Problema	Correção
A agenda, em Android, deixava de carregar novos eventos entre atualizações da aplicação.	De uma versão para a outra, a base de dados tinha sido modificada, com a inserção de mais duas colunas (longitude e latitude). Portanto, a base de dados antiga era apagada e era criada a nova, no entanto, as variáveis de preferência com as datas da última atualização dos eventos não eram inicializadas.	Sempre que há uma modificação na estrutura da base de dados, todas as variáveis que controlam as datas de atualização são inicializadas, assim todos os eventos são carregados de novo.

Os screenshots das mensagens de alerta podem ser consultados no Anexo D - Resultado dos testes.

6.2.3. Outras alterações

No Android foram feitas alterações na classe POIViewActivity e no seu respetivo layout de modo a que possa mostrar os 3 eventos mais próximos de um determinado ponto. Esta classe faz uma chamada ao método getNearByEvents() da classe DBHandler, passando as coordenadas do ponto, e este, retorna uma lista com os 3 eventos mais próximos. O método getNearByEvents executa a seguinte query:

```
db.query(EVENTS_TABLE,new String[] {EVENT_ID, TITLE, LINK, DESCRIPTION, START,
      END,PLACE,CATEGORY,IMAGE,PUBLISHED,OPENINGTIME,CLOSINGTIME,LATITUDE,LONG
ITUDE},LATITUDE+" > 0 AND "+LONGITUDE+" > 0", null, null, null,
"abs("+LATITUDE+" - "+lat+") + abs("+LONGITUDE+" - "+lng+") asc LIMIT 3");
```

Foram escolhidos os três eventos mais próximos por opção da empresa e de modo a evitar que os POIs contenham praticamente os mesmos eventos na vizinhança. A figura 17 ilustra o layout antes e depois das alterações.

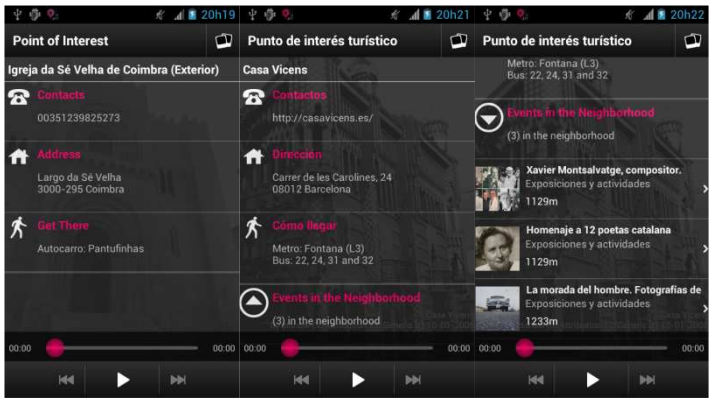

Figura 17 - POIView antes e depois das alterações

6.2.4. Trabalhos futuros

A componente agenda pode ainda ser melhorada tendo como base o trabalho feito até agora. De seguida são apresentadas algumas propostas de melhorias a implementar:

- Notificação de eventos na proximidade: embora especificado como um dos requisitos a desenvolver, não chegou a ser implementado na prática. No futuro poderá servir para enriquecer ainda mais o JiTT.

- Apresentação dos eventos no mapa em modo offline: invés de recorrer a aplicação do Google maps, que precisa de uma conexão a internet, os eventos poderão ser apresentados no mapa do JiTT em modo offline, já que cada evento possui as suas coordenadas.

6.3. Desenvolvimento II | Atualização Base de Dados

Um dos maiores problemas do JiTT é a falta de um mecanismo de atualização da base de dados. Caso seja alterado algum conteúdo, mesmo se tratando de uma ínfima palavra numa descrição de um determinado ponto de interesse, é necessário que toda aplicação seja submetida a Apple App Store e ao Google Play, ou seja, a cada atualização o utilizador terá de fazer o download de aproximadamente 200 MB de dados, o que é incómodo nos dispositivos móveis, onde a velocidade da conexão pode ser lenta. Sem contar com o incómodo que traz a empresa porque a equipa da Apple costuma demorar cerca de uma semana a avaliar a aplicação até ser disponibilizado ao público. Essa situação não se coloca no dispositivos Android, pois a aplicação fica disponível quase de imediato.

Embora, de início, este se tratar de um upgrade importante para a empresa, esta resolveu não dar continuidade ao desenvolvimento deste upgrade. Chegou-se a conclusão que a atualização da base de dados não era algo que era feita regularmente, portanto, era desnecessário implementar um mecanismo de atualização.

A maioria do trabalho já tinha sido feito. O upgrade para Android já estava praticamente finalizado, portanto nas seções que se seguem são descritas as várias componentes que fazem parte do sistema de atualização da base de dados, assim como as interações entre elas.

.

Informações mais detalhadas sobre os requisitos podem ser consultadas no Anexo B - Requisitos da Atualização da BD.

6.3.1. Arquitetura do sistema

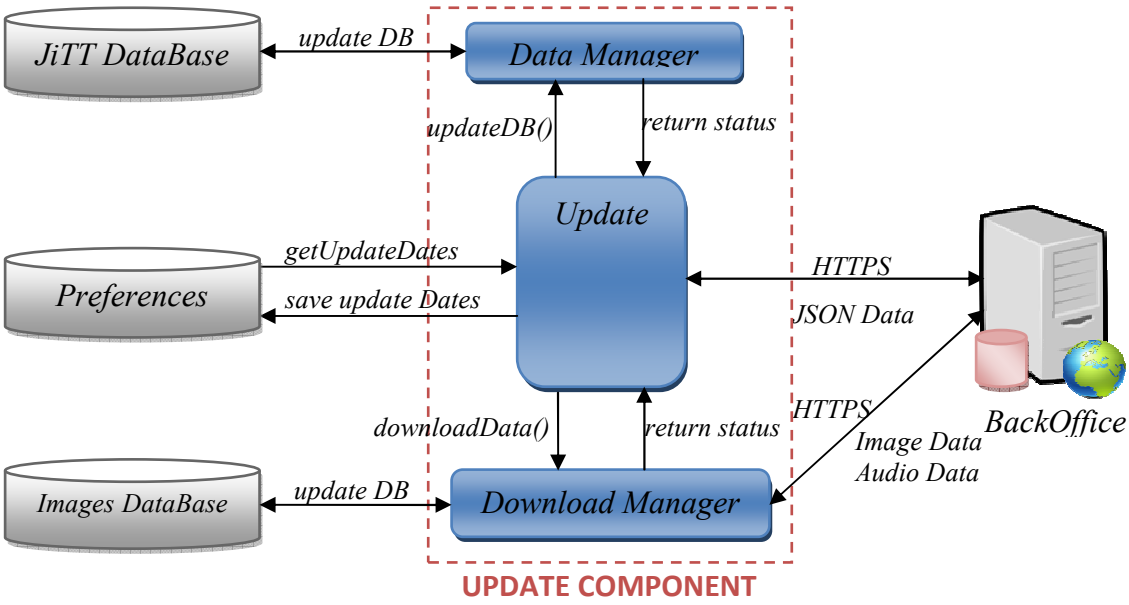

Figura 18 - Arquitetura do sistema

BackOffice

O servidor de BackOffice já se encontrava funcional. A implementação de funcionalidades do lado do servidor está fora do âmbito deste estágio e qualquer modificação necessária do lado deste é feita pelo Gestor de Software da iClio. Através da plataforma de gestão de projetos online, Basecamphq, são acordadas as modificações.

Uma das alterações propostas foi precisamente a adição de um campo "modified" as tabelas do ficheiro exportado, com a data da última modificação. Este será usado para verificar quais as tabelas que serão processadas pela aplicação.

Como não podemos medir a quantidade de ligações BackOffice e de modo a que este não seja muito sobrecarregado, é exportado um único ficheiro JSON, com todas as informações da base de dados, que depois será acedido pelas aplicações JiTT. Foi escolhida essa solução, em detrimento de uma query a base de dados e geração em tempo real dos dados necessários, o

que podia afetar o desempenho do servidor caso o número de acessos fosse muito elevado.

Update

Utilizamos a API JSON em ambas as implementações (iOS e Android) tendo em conta que o servidor de BackOffice já possui a funcionalidade de exportar os dados da base de dados para esse formato. A componente Update sempre que acionada irá verificar se o ficheiro foi modificado, no caso afirmativo faz o download do documento e dos recursos necessários.
Esta componente é composta por três classes principais: Update, DataManager e DownloadManager.

Classe Update

Esta é a classe principal, responsável pela inicialização das outras duas. É acionada pelo utilizador a partir do botão "update".
Quando acionada, é executada o método startUpdating que comporta da seguinte forma:

1º. Verifica se existe uma conexão a internet. Caso não haja é retornado um "status code" que identifica a causa do problema;
2º. Verifica se o servidor não está em baixo. Caso esteja, ou seja, a aplicação não consegue estabelecer uma conexão Java HTTPS ao Backoffice, é retornado um "status code" que identifica a causa do problema.
3º. Faz a autenticação e o download do ficheiro JSON. Em caso de erro retorna um "status code".
4º. Compara a data do ficheiro recebido do servidor (variável "modified") com a data guardada no dispositivo. Se não houver atualizações desde a última ligação é retornado um "status code" que identifica a mensagem a ser apresentada ao utilizador.
5º. Faz o parsing do ficheiro JSON. Começando pelo jsonarray "order" que determina a ordem pela qual as atualizações devem ser feitas, de modo a salvaguardar as relações entre as tabelas da base de dados.

6º. Para cada tabela, compara-se as datas e verifica-se se houve atualizações. No caso afirmativo faz-se então a atualização da tabela na base de dados e o download dos dados (imagens e áudio).

Para cada uma das tabelas existe um método (parsing<Nome da tabela>(String tableName)), que é chamado pela ordem especificada no array "order". Os métodos extraem cada campo referente as tabelas, e criam um novo objeto, que depois é passado para os respetivos métodos da classe DataManager, que encarrega de escrever as alterações na base de dados.
Se todos os passos forem executados com sucesso, ou seja, não é detetada nenhuma exceção, é guardada, para cada tabela e para o documento, a data da última atualização (utilizou-se as variáveis de preferência para o efeito).
Para controlar as atualizações é utilizada uma variável de preferência (boolean) que guarda informações sobre a atualização - se foi bem sucedida ou não. Esta variável é verificada sempre que o utilizador executar a aplicação JiTT. Caso a última atualização não tenha terminado com sucesso, o sistema apresenta uma alerta ao utilizador que terá de fazer uma nova atualização de modo a continuar a usar a aplicação.
Este é um mecanismo muito importante já que o processo de atualização pode ter sido interrompido e alguns campos atualizados e outros não. Serve para garantir a consistência dos dados e a relação entre as tabelas.

Classe DataManager

A classe DataManager é responsável pelas operações sobre a base de dados (insert, delete, update). Para cada tabela na base de dados, foi criado um método para tratar das operações sobre a respectiva tabela. Estes métodos retornam uma variável do tipo boolean que determina se a operação de atualização ocorreu com sucesso. Caso ocorra algum erro a atualização é abortada.

Classe DownloadManager

A classe DownloadManager possui dois métodos: um para o download das imagens e outro para o download dos ficheiros de áudio. O Funcionamento destes não é muito complexo, estabelecem uma conexão ao backoffice e autenticam-se utilizando um mecanismo de login e password. Para cada

ficheiro da tabela multimédia é feita o download para a base de dados do dispositivo (uma diretoria no cartão de memória).
Em primeiro lugar é verificada se o ficheiro já existe na base de dados e se o espaço disponível é suficiente para guardar o ficheiro. No caso de não haver espaço é enviada uma mensagem de falha para a classe Update e a atualização é interrompida e o utilizador notificado do sucedido.

Base de Dados

O esquema da base de dados do JiTT foi criada utilizando a Core Data Framework da Apple que determina onde e como os dados são guardados, faz o caching de dados e a gestão da memória. Possui uma interface gráfica onde o programador define o schema da base de dados (as entidades e as suas relações) e o código é gerado automaticamente. Portanto, a base de dados foi inicialmente criada para iPhone (iOS) e depois, o ficheiro sqlite gerado, foi transportado para a plataforma Android.

A base de dados do JiTT é constituída por 20 tabelas. De modo a salvaguardar os interesses da empresa a estrutura da base de dados não foi aqui exposta.

A geração automática do esquema da base de dados constituiu um dos problemas na implementação do mecanismo de atualização. Enquanto que na plataforma iOS a Core Data Framework faz todo o gerenciamento da base de dados, em Android todo o processo terá se ser feito a mão e temos de garantir a relação entre as várias tabelas.

6.3.2. Diagrama de atividade

O diagrama da figura 19 modela o processo de atualização da base de dados do JiTT. Consegue-se ter uma visão geral do processo de atualização.

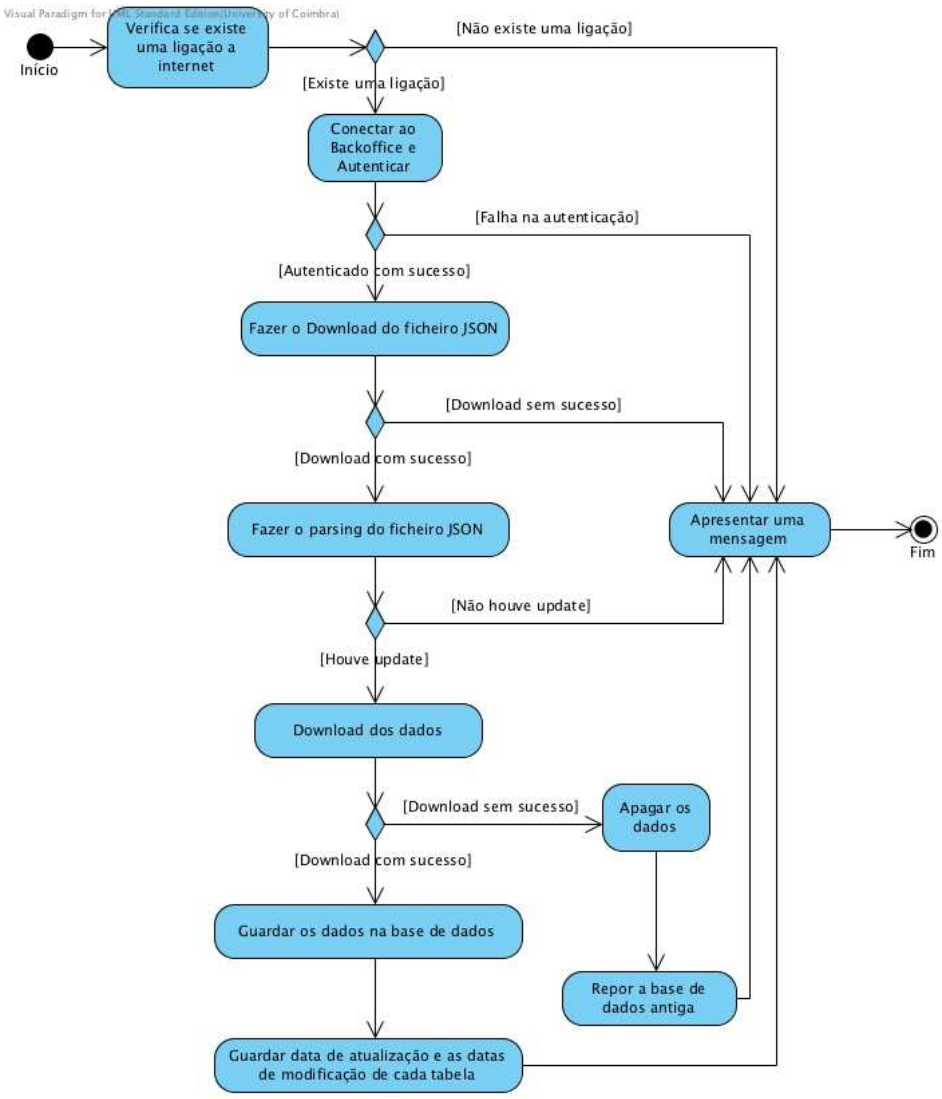

Figura 19 - Diagrama de atividade

6.3.3. Testes

Os testes consistiram em verificar as seguintes situações:

1. Quando não existe uma ligação a rede
2. Quando existe uma atualização (são transferidos apenas os dados que foram atualizados ou todos os dados?)
3. Quando há uma interrupção na conexão ou quando o utilizador cancela o processo de atualização

Tabela 6 - Atualização da Base de dados | Resultado dos testes

Teste	Resultado	Descrição
1	✓	O sistema apresenta uma mensagem de alerta informando o utilizador que deverá verificar a sua conexão a internet.
2	✓	São processadas apenas as tabelas que foram modificadas e apenas os dados (imagem, áudio) são transferidos.
3	✓	O estado é guardado numa variável de preferência e a mensagem é apresentada ao utilizador. Quando a aplicação for executada novamente, esta variável é verificada e se a atualização foi interrompida o utilizador terá de atualizar antes de continuar a usar a aplicação.

Os screenshots das mensagens de alerta podem ser consultados no Anexo D - Resultado dos testes.

6.4. Desenvolvimento III | Otimização de Bateria

A otimização no uso da bateria é uma das características a ter conta quando se desenvolve aplicações para telemóveis. Desenvolver aplicações que fazem muito o uso da bateria, principalmente quando sabemos que não há nenhum lugar perto onde possamos recarregar o mesmo, não será nada agradável para o consumidor final, particularmente para alguém que se encontra numa cidade desconhecida.

A optimização da bateria é uma das preocupações da iClio no que respeita ao JiTT. Este utiliza recursos tais como o GPS, áudio e imagens, que não são propriamente amigos da bateria, sobretudo o primeiro.

Esta foi um dos upgrades anteriormente planeado para a execução durante o decorrer do segundo semestre do estágio. No entanto, este upgrade teve de ser retirado dos planos devido ao escasso tempo necessário para a execução do desenvolvimento da Realidade Aumentada e porque existiam outros projetos da empresa que teriam de ser desenvolvidas, como por exemplo a aplicação Casa Angola (Android e iOS), aplicação para as Águas de Coimbra (tablet Android) e o Corão de ouro para iPad.

De salientar que a aplicação Águas de Coimbra percorrerá as ruas de Coimbra nos finais de Agosto - início de Setembro (figura 20).

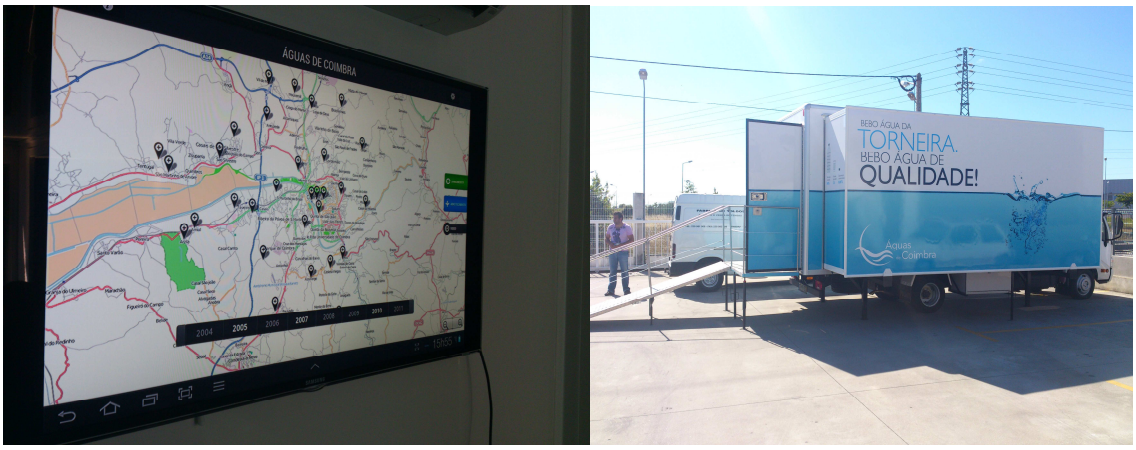

Figura 20 - Camião das águas de Coimbra

6.5. Desenvolvimento IV | Realidade Aumentada

A definição de Ronald Azuma [28] sobre a Realidade Aumentada (RA) é a descrição melhor aceite. Ela ignora um subconjunto do objetivo inicial da RA, porém é entendida como uma representação de todo o domínio da RA: Realidade Aumentada é um ambiente que envolve tanto realidade virtual como elementos do mundo real, criando um ambiente misto em tempo real. Por exemplo, um utilizador da RA pode utilizar óculos translúcidos, e através destes, poderia ver o mundo real, bem como imagens geradas por computador projetadas no mundo.

A implementação dessa tecnologia no JiTT permitirá ao utilizador visualizar os pontos de interesse (POIs) e os eventos, assim como toda a informação associada a esses pontos, tais como: o nome, a descrição, a distância a que se encontram do utilizador, imagens, etc. A ideia é permitir ao utilizador direcionar a câmara para uma determinada direção e ver todos os pontos que se encontram na referida direção e a uma determinada distância escolhida por aquele.

Nas seções que se seguem, são descritas todas as componentes pertencentes a arquitetura do sistema da RA, as decisões tomadas, os testes

efetuados e sugestões para possíveis melhorias. De salientar que este upgrade só foi implementado em Android, portanto as descrições sobre a implementação, dizem respeito a essa plataforma. Informações mais detalhadas sobre os requisitos podem ser consultadas no Anexo C - Requisitos RA.

As versões das interfaces podem ser consultados no Anexo F - Interfaces.

6.5.1. Arquitetura do Sistema

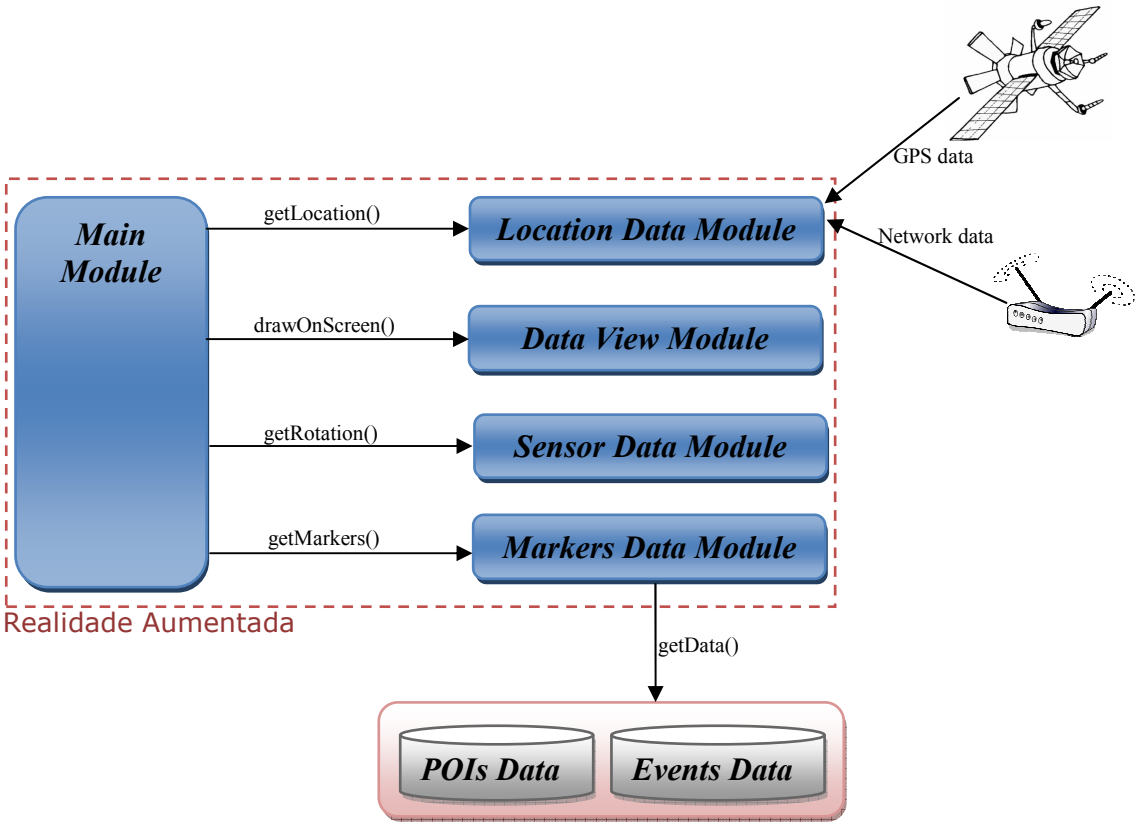

Figura 21 - Arquitetura do sistema

Antes da implementação foram adicionadas as seguintes permissões no ficheiro manifest.xml:

```
<uses-permission android:name="android.permission.CAMERA" />
      <uses-feature android:required="false"
android:name="android.hardware.camera.autofocus"></uses-feature>
      <uses-feature android:required="false"
android:name="android.hardware.camera.flash"></uses-feature>
      <uses-feature android:required="false"
```

```
android:name="android.hardware.camera.front"></uses-feature>
      <uses-permission
android:name="android.permission.WAKE_LOCK"></uses-permission>
```

Com estas permissões a aplicação poderá aceder aos recursos da câmara do dispositivo e ao wake_lock que permite manter o ecrã ativo.

Também foi adicionado um novo botão ao layout do MapActivity do JiTT para acionar o modulo de realidade aumentada.

A classe principal deste modulo foi igualmente acrescentada ao ficheiro manifest.xml.

```
<activity android:name=".ARActivity"
                android:label="@string/app_name"
                android:theme="@android:style/Theme.NoTitleBar"
                android:screenOrientation="landscape"/>
```

Todas as subclasses da Activity, que fazem interação com o utilizador, são adicionadas ao manifest.xml, de modo a que possam ser chamadas pelo método Context.startActivity() do Android.

Main Module

É o módulo principal, composta pela classe ARActivity, responsável pela inicialização dos outros módulos e pela interação com o utilizador. Inicializa as componentes gráficas que são apresentadas ao utilizador e, com recurso a câmara do dispositivo, mostra a imagem em tempo real.

Começa por criar o barra de zoom (SeekBar) que inicialmente é invisível e de seguida a barra com os botões de interação com o utilizador. Cada uma dessas decomponentes estão organizadas numa FrameLayout que depois serão adicionadas a uma View.

Utilizando a Camera API, é criada a View principal que será a imagem captada através da câmara. A esta é adicionada uma outra, onde serão desenhados todos os objetos (o módulo DataView utiliza esta View para desenhar os marcadores e o radar), mas antes disso é preenchida a lista com todos os marcadores.

A classe ARActivity implementa o método onTouchEvent() da interface OnTouchListener. Este método deteta o ponto de toque (x,y) sobre o ecrã, e envia as coordenadas para o módulo DataView que se encarrega de tratar deste evento.

Este módulo está completamente integrada no JiTT, faz o uso das classes existentes anteriormente, e integra o módulo Agenda. O utilizador poderá carregar sobre o marcador dos eventos e ver os detalhes, poderá carregar nos POIs e ouvir o áudio associado e também ver os eventos e os POIs numa lista.

Location Data Module

Este módulo é composto por uma única classe: ARLocation. Esta, implementa os métodos da Interface LocationListener e utiliza uma instância da classe LocationManager (componente central da framework de localização), que fornece APIs para determinar a localização do dispositivo.

Para obter uma instância dessa classe foi feita uma chamada ao método getSystemService(Context.LOCATION_SERVICE). Com a referida instância criada, conseguimos:

- Fazer uma query aos "fornecedores de localização" pela última localização conhecida;
- Registar/cancelar o registo para atualizações periódicas de localização.

Foram utilizados dois provedores:

1. *GPS_PROVIDER* - Esse provedor determina a localização por meio de satélites. Requer que a permissão, android.permission.ACCESS_FINE_LOCATION, seja definida no ficheiro manifest.xml.
2. *NETWORK_PROVIDER* - Esse provedor determina a localização com base na disponibilidade de uma antena do provedor de serviços de telefonia móvel e pontos de acesso wi-fi. Requer que uma das permissões, android.permission.ACCESS_COARSE_LOCATION ou

android.permission.ACCESS_FINE_LOCATION seja definida no ficheiro manifest.xml.

Sempre que há uma mudança na localização é acionada o método onLocationChanged(Location location) que guarda a nova localização numa instância da classe Application, podendo, assim, ser acedida a partir de qualquer instância de uma Activity. A localização será depois utilizada para determinar os pontos que serão apresentados ao utilizador .

Sensor Data Module

Este módulo, também composto por uma única classe, Sensor, implementa o método onSensorChanged(SensorEvent event) da interface OnSensorListener. Utiliza dois tipos de sensores:

1. *TYPE_MAGNETIC_FIELD* [19] - Mede o campo geomagnético do ambiente em todos os três eixos (x, y, z) em μT (micro Tesla).
2. *TYPE_ACCELEROMETER* [19] - Mede a força de aceleração em m/s^2 que é aplicada a um dispositivo em todos os três eixos físicos (x, y e z), incluindo a força da gravidade.

Os outputs destes dois sensores são utilizados no cálculo da matriz de inclinação e da matriz de rotação, transformando um vetor a partir do sistema de coordenadas do dispositivo para o sistema de coordenadas do mundo real.

A matriz de rotação resultante é depois invertida. Esta será usada para transformar os pontos das coordenadas do mundo real para as do dispositivo.

Markers Data Module

Este módulo é constituído pelas duas classes que representam os marcadores - POIMarker e EventMarker - e a classe ARDataManager encarregue pela operação de leitura das bases de dados.

As classes POIMarker e EventMarker são estendidas a partir da classe Marker (biblioteca Mixare). Possui o método draw responsável por desenhar o marcador no ecrã, assim como o bloco de texto.

A ARDataManager possui uma instância da classe DBHandler, desenvolvida no módulo Agenda, para operações sobre a base de dados dos eventos. Os POIs do JiTT não são lidas a partir da base dados, mas sim, a partir da classe Application, que guarda um objeto do tipo Tour que contêm a lista dos POIs pertencentes ao percurso gerado.

Data View Module

Este módulo trata dos marcadores a serem desenhados no ecrã e das interações sobre os mesmos. Acede a lista dos marcadores a partir da variável estática definida na classe principal do módulo Main e calcula, para cada marcador, a distância a que se encontra da posição atual do utilizador. A lista é depois ordenada, de modo ascendente, de acordo com a distância. Entretanto, os marcadores são desenhados a partir do fim para o início da lista, ou seja, os pontos mais distantes são desenhados em primeiro lugar.

De referir que para cada marcador a ser desenhado, é verificado se a distância está dentro dos limites do raio de vizinhança definido.

Quando há uma interação sobre um marcador, a lista com os marcadores é percorrida de modo ascendente, e o primeiro que satisfazer as condições é escolhida, ou seja, no caso de haver um marcador atrás do outro, o que estiver a frente é selecionado.

6.5.2. Testes

Para a realidade aumentada foram especificadas os seguintes testes a serem efetuados:

1. Verificar o comportamento quando o dispositivo não tem o GPS ligado
    a. O sistema deverá apresentar uma mensagem ao utilizador informando que são necessários dados do GPS.

2. Usabilidade
    a. Servirá para conferir se as opções tomadas em termos de interação com o utilizador são as mais corretas e com base nisso melhorar a experiência do mesmo. Serão pedidas a vários utilizadores que testem a aplicação de modo a ter dados suficientes para a formulação de novas soluções.

3. Testes de campo
    a. O JiTT possui um guia para a cidade de Coimbra, portanto este será usado para testar a navegação utilizando o modulo da RA. Será gerado um tour que servirá como forma de certificar que as direções dadas pelo módulo estão de acordo com a realidade.

Tabela 7 - Realidade Aumentada | Resultado dos testes

Teste	Resultado	Descrição
1	✓	É apresentada uma mensagem informando ao utilizador do problema. O utilizador poderá escolher o centro da cidade como o ponto inicial.
2		No momento da escrita deste relatório não conseguimos que outros utilizadores testassem a aplicação com o módulo de realidade aumentada, porque ainda não se encontra totalmente finalizado.
3	✓	Foi gerado um tour para a cidade de Coimbra em que tivemos a oportunidade de testar a orientação dada pela aplicação.

6.5.3. Trabalhos futuros

O módulo de Realidade Aumentada só foi desenvolvido para a plataforma Android. A versão para iOS será desenvolvida numa fase posterior.

Este módulo encontra-se praticamente finalizado. A maioria das decisões tomadas, tanto a nível do design e das opções apresentadas , são da minha inteira responsabilidade, ou seja, ainda não foi validado pela empresa devido

ao facto do CEO se encontrar fora do país. Depois de aprovado e feitas possíveis alterações, será preciso fazer mais testes de campo e de usabilidade.

7.Conclusão

O trabalho feito durante o presente estágio permitiu o desenvolvimento de três upgrades para o JiTT. Inicialmente foram previstos quatro: uma agenda de eventos, mecanismo de atualização da base de dados, otimização no uso da bateria e mecanismo de realidade aumentada. No entanto, o desenvolvimento três, otimização no uso da bateria, foi retirado do plano porque surgiram novos projetos que careceram da nossa atenção.

O upgrade dois foi interrompido já que a empresa chegou a conclusão de que afinal a introdução de um mecanismo de atualização da base de dados não era algo tão crucial. Sendo assim, todo o trabalho desenvolvido até ao referido ponto, foi aqui exposto.

Durante o primeiro semestre desenvolveu-se a Agenda de eventos, que viu o seu código revisto e a corrigido, durante o segundo semestre. Numa primeira fase, este upgrade só tinha sido introduzido numa versão grátis do JiTT, JiTT Barcelona em espanhol (plataforma Android), para, no fundo, fazer um teste de campo. Na segunda fase foi então introduzido no JiTT Barcelona em inglês, JiTT Londres, para as plataformas iOS e Android.

No segundo semestre deu-se a continuidade no desenvolvimento, em que o plano foi revisto e alterado. Novos projetos apareceram e que estavam fora do âmbito do estágio, o que levou a remoção de um dos upgrades que estava inicialmente planeado. Foi desenvolvido o mecanismo de Realidade Aumentada que coincidiu com fase final do estágio.

Este estágio traduziu-se num grande desafio, já que houve fases em que estive envolvido em três projetos ao mesmo tempo, numa área em que, até então, não tinha muita experiência.

8. Referências

[1] Gartner Inc - Online
http://www.gartner.com/it/page.jsp?id=1622614

[2] Berg Insight - Online
http://www.berginsight.com/ReportPDF/ProductSheet/bi-app1-ps.pdf

[3] Working with XML on Android - Online
http://www.ibm.com/developerworks/opensource/library/x-android

[4] How To Choose The Best XML Parser for Your iPhone Project - Online
http://www.raywenderlich.com/553/how-to-chose-the-best-xml-parser-for-your-iphone-project

[5] SQLite, in Wikipedia - Online
*http://en.*wikipedia*.org/wiki/SQLite*

[6] Scrum, in Wikipedia - Online
http://en.wikipedia.org/wiki/Scrum_(development)

[7] Just in Time Tourist - Online
http://www.justintimetourist.com

[8] Lonely Planet - Online
http://www.lonelyplanet.com

[9] mTrip - Online
http://www.mtrip.pt

[10] BeeLoop - Online
http://www.beeloop.com

[11] Play&Tour City Guide - Online
http://www.playandtour.com

[12] Jourist - Online
http://www.jourist.info

[13] City Speaker - Online
http://www.cityspeaker.com

[14] 4 Emme Service Spa - Online
http://www.giracitta.eu

[15] Navigaia PREMIUM - Online
http://www.navigaia.com

[16] iJourneys - Online
http://www.ijourneys.com

[17] TreckExchange - Online
http://www.treckexchange.com

[18] Rick Steves - Online
http://www.ricksteves.com/news/audio-tours.htm

[19] Android Developers - Online
http://developer.android.com

[20] Apple Developers
http://developer.apple.com/

[21] Mobile Tuts+
http://mobile.tutsplus.com/

[22] 10 Travel Apps - Online
http://www.10travelapps.com

[23] AppBrain - Online
http://www.appbrain.com/stats

[25] Distimo - Online
http://www.distimo.com/blog

[26] Android authority - Online
http://www.androidauthority.com

[27] Wikipedia - Online
http://pt.wikipedia.org/wiki/Realidade_aumentada

[28] Wikipedia XML
http://en.wikipedia.org/wiki/XML

[29] World Wide Web Consortium
http://www.w3.org/

[30] Geocoding API
https://developers.google.com

[31] NSOperation tutorial
http://www.cimgf.com

[32] Simple API for XML
http://java.sun.com