Novopca Mobile

1050550 Ricardo Manuel Silva Alves

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Área de Especialização em

Arquitectura, Sistemas e Redes

Orientador: Doutor Luís Miguel Pinho Nogueira

Co-orientador: Eng. Frederico António Sá Oliveira Pinho

Júri:

Presidente:

Maria de Fátima Coutinho Rodrigues, Professora Coordenadora, Instituto Superior de Engenharia do Porto

Vogais:

Luís Miguel Moreira Lino Ferreira, Professor Adjunto, Instituto Superior de Engenharia do Porto

Luís Miguel Pinho Nogueira, Professor Adjunto, Instituto Superior de Engenharia do Porto

Frederico António Sá Oliveira Pinho, Gestor de Projectos, Novopca Construtores Associados S.A.

Porto, Outubro 2011

Aos meus pais, irmão, sobrinho e à minha cara-metade.

Agradecimentos

Em primeiro lugar quero agradecer à minha família e à minha namorada por toda a paciência e apoio ao longo de todos estes anos. Em especial aos meus pais que sempre estiveram a meu lado e fizeram com que fosse possível a minha formação no ensino superior. Ao meu irmão que sempre teve uma palavra de apoio e incentivo. À minha namorada por, apesar de tudo, me continuar a aturar mesmo nos momentos mais difíceis. Ao meu sobrinho por ser o bebé mais bonito e simpático do planeta.

Quero ainda agradecer a todos os colaboradores e ex-colaboradores da Novopca que me apoiaram e ajudaram na realização deste projecto, nomeadamente os engenheiros Miguel Neves, Diogo Velho, Manuel Duarte e Jacinto Barbosa.

Quero agradecer especialmente ao supervisor e ao orientador do projecto aqui documentado; designadamente o Eng. Frederico Pinho, pelo acompanhamento e toda a ajuda prestada ao longo do projecto, com o objectivo de corresponder às expectativas da empresa; e o Professor Doutor Luís Nogueira pela sua enorme disponibilidade e ajuda.

Por fim quero agradecer a todos os colegas do ISEP que conheci e com quem convivi ao longo destes anos.

A todos os referidos, um sincero obrigado.

Resumo

A constante evolução da tecnologia obriga a que o mundo empresarial tenha que alocar esforços para se manter actualizado. Os equipamentos tecnológicos que se encontram actualmente na vanguarda da tecnologia são os dispositivos móveis, que cada vez mais nos rodeiam e acompanham com o intuito de nos facilitar a execução de tarefas rotineiras.

O ramo de construção e obras públicas em particular é um sector que obriga a que os seus trabalhadores estejam constantemente em movimento, e que muito têm a ganhar com a utilização de ferramentas de trabalho em equipamentos móveis.

Neste projecto pretendeu-se identificar qual o sistema operativo para dispositivos móveis que melhor se enquadra com as necessidades empresariais, assim como quais os mecanismos de comunicação inter-máquinas que melhor se enquadram com o cenário em estudo. Foi então desenvolvida uma plataforma para dispositivos móveis que permite agilizar a execução de processos que normalmente são morosos ou que obrigam à utilização do computador em locais menos próprios, nomeadamente o ambiente de obra. A solução proposta e implementada envolve uma aplicação modular quer permita a fácil inserção e remoção de módulos. Foram também realizados os estudos e implementações de quatro módulos, designadamente: auto-afectação, conferência documental, fornecedores e presenças em obra.

Concluído este projecto, ficam reunidas as condições para o fácil desenvolvimento de novos módulos, visando inovação e optimização dos processos de trabalho da empresa.

Palavras-chave: dispositivos móveis, *Web Services*, Android, sistema aplicacional modular, *middleware*

Abstract

The constant evolution of technology forces the corporal world to allocate further efforts to remain updated. All technological equipment “in vogue” nowadays, as we acknowledge as mobile devices, keep up with the current needs of one to keep daily tasks easy and quick to execute.

The construction industry and public works sector in particular is one which requires constant mobility from its employees and therefore much to gain from using work tools on mobile devices.

The main goal of this project focuses on identifying which from the various mobile devices’ operating systems best fit with current corporal needs, as well as inter-machine communication mechanisms that also best fit with the scenario under study. Therefore, it was then developed a platform for mobile devices that allows faster execution of processes that are usually time-consuming or that require using a computer in less own particular work environment. The proposed and implemented solution involves a modular application that allows easy insertion and removal of modules. There were also conducted studies and implementations of four modules, namely: “Self-Allocation”, “Documental Conference” , “Suppliers” and “Attendance on work site”.

Given this project as concluded, we gather all the required conditions for easy development of new modules, in order to head towards innovation and optimization of the company’s work processes.

Key-words: mobile devices, Web services, Android, Modular Application System, Middleware

Índice

Índice de Figuras

Índice de Tabelas

Notação e Glossário

ADT Acrónimo de Android Development Tools. É um *plugin* para o Eclipse que fornece ferramentas para desenvolvimento aplicacional.

API Acrónimo de A*pplication Programming Interface*, é um conjunto específico de regras e especificações que os programas de computador utilizam para comunicarem entre si.

CRUD Acrónimo de *Create, Read, Update and Delete*. Representa as operações que se podem realizar sobre informação armazenada.

CSV Comma-Separated Values, em português Valores Separados por Vírgula, é um formato de documento que armazena dados tabelados.

DAL Data Access Layer, em português Camada de Acesso a Dados.

GPS Acrónimo de Global Positioning System, é um sistema de navegação por satélite.

HEAP É um local de memória onde é realizada a alocação dinâmica de memória para armazenamento temporário de dados.

ID Número identificador.

Endereço IP Endereço de um nó numa rede de computadores.

NFC Acrónimo de Near Field Communication, é uma tecnologia de troca de informação entre dispositivos electrónicos de curta distancia.

RAM Acrónimo de Random Access Memory, é normalmente a memória principal de sistemas computacionais.

SD-Card Acrónimo de Secure Digital Card, é um cartão de memória utilizado para armazenamento permanente de informação.

URL Uniform Resource Locator, representa um endereço de um Website.

URI Uniform Resource Identifier, é um conjunto de caracteres utilizados na identificação de nomes ou recursos na Internet.

WiFi É um mecanismo de comunicação sem fios entre dispositivos eléctricos.

1 Introdução

"Vamos entrar em 2011 mergulhados na maior crise de que há memória. Já há oito anos que o sector da construção não cresce e este ano que passou o sector registou uma quebra na produção de 36 por cento" [Destak, 2011].

Esta afirmação proferida por Reis Campos, presidente da Confederação Portuguesa da Construção e do Imobiliário, demonstra bem as dificuldades pelo qual o sector da construção está a passar. Devido à forte crise económica que está a atingir as empresas de construção civil e obras públicas, estas vêem-se obrigadas a reduzir custos, seja através do aumento da eficiência dos processos de trabalho, seja através da utilização de diferentes materiais de construção ou através da redução do número de recursos humanos.

Apesar das constantes perdas, o sector da construção representa ainda um papel preponderante na economia nacional, sendo responsável por 10,7% do emprego, cerca de 5,2% do PIB e 48,8% do total do investimento feito em Portugal [AICCOPN, 2011].

Perante este cenário pouco animador, a empresa Novopca Construtores Associados pretendeu inovar os processos de trabalho dos seus colaboradores. É neste âmbito que se enquadra o projecto aqui documentado.

O Novopca Mobile pretende ser uma solução móvel para S*martPhones* que permita a execução de vários tipos de trabalhos em qualquer parte do mundo, principalmente em locais de obra. Este projecto poderá constituir uma grande mais-valia na revolução dos métodos de trabalho da empresa, tornando-os mais eficientes.

Actualmente existe uma gama quase infindável de S*martPhones* no mercado, existindo diferentes ofertas tanto ao nível de *hardware* como de *software*. Tendo em conta que o sector da construção obriga a que os trabalhadores se encontrem muitas vezes a laborar em condições agrestes, os equipamentos devem estar desenhados para resistir a estas condições.

1.1 Enquadramento

A elaboração desta dissertação visa descrever o projecto ocorrido nas instalações da empresa Novopca, desenvolvido no âmbito da disciplina de Tese/Dissertação/Projecto de Mestrado, pertencente ao plano de Bolonha do Mestrado em Engenharia Informática do ISEP.

A empresa onde decorreu o estágio possui, ao nível de *hardware*, vários servidores e uma vasta rede que interliga os seus colaboradores. Ao nível de *software*, são utilizadas diversas plataformas, tendo vindo a privilegiar sempre que possível o uso de software livre.

O objecto do projecto realizado focou-se no desenvolvimento de uma solução para dispositivos móveis que permita uma interligação com os sistemas já existentes.

1.2 Objectivos

As soluções para dispositivos móveis implementadas visam equipar os colaboradores da empresa com ferramentas que permitam uma rápida e célere execução das suas tarefas de trabalho.

Os colaboradores das empresas de construção são normalmente obrigados, como já referido, a deslocar-se para locais de obra, com condições normalmente limitadas para a realização de trabalhos com computadores. Os colaboradores ficam também afastados da sede das respectivas empresas, onde normalmente se encontra toda a informação nuclear e onde são executadas as actividades de apoio à laboração das organizações. Tendo em conta estes factos, a aplicação desenvolvida visa criar uma plataforma, denominada NovopcApp, capaz de ser escalável ao longo do tempo através da adição de módulos. Estes módulos serão as ferramentas de trabalho dos colaboradores. Os módulos implementados ao longo deste trabalho foram os módulos de auto-afectação, de conferência documental, de fornecedores e de presenças em obra. A arquitectura modular aqui proposta e implementada permite que os colaboradores escolham quais os módulos que necessitam para a realização das suas tarefas, permitindo também que novos módulos sejam adicionados à plataforma sem que seja necessário que os colaboradores se desloquem dos locais aos quais estão afectos.

Os objectivos concretos do Novopca Mobile são:

- Criação de uma plataforma móvel capaz de ser escalável com a adição de novos módulos.
- O desenvolvimento de módulos que tragam mais-valia aos colaboradores da organização, nomeadamente os módulos de auto-afectação, de conferência documental, de fornecedores e de presenças em obra.
- O módulo de auto-afectação será responsável por localizar os colaboradores em obras e por comunicar a sua afectação às mesmas.
- O módulo de conferência documental deverá permitir a visualização de documentos por conferir assim como realizar a respectiva conferência.

- O módulo de fornecedores deverá permitir a pesquisa e consulta de dados relacionados com fornecedores.
- O módulo de presenças em obra deverá permitir que seja realizado o controlo de presenças dos colaboradores subcontratados (externos à empresa) em obra.
- Implementação de *middleware* que permita a comunicação da plataforma móvel com a base de dados já existente; comunicação essa que deverá ser realizada o mais rapidamente possível e com o mínimo de tráfego gerado, de forma a maximizar a experiência dos utilizadores.

A Novopca Mobile, apesar de ser uma solução móvel muito orientada aos processos de laboração da empresa Novopca, poderá em muitas das suas características ser implementada noutra organização, seja ou não do sector de construção.

1.3 Novopca Construtores Associados, SA

A NOVOPCA é uma empresa de Construção Civil e Obras Públicas fundada em 1947. Desde a sua fundação, tem mantido a sua identidade, desenvolvendo a sua cultura própria no respeito pelo espírito que norteou a actuação dos seus fundadores.

Alicerçada na sua tradição e experiência, a empresa procura constantemente novos desafios, ensaiando tecnologias inovadoras e adaptando-se às mutações cada vez mais frequentes das condições de mercado.

A empresa é constituída por dois pontos estratégicos: a Sede que se situa na Senhora da Hora, distrito do Porto, e a Delegação Sul situada na cidade de Lisboa. Face ao seu ramo de actividade, é alimentada pelas obras que se distribuem por todo o país.

No ano de 2003 existiu um reforço de investimento no sector de sistemas de informação através da implementação de um Website interno designado por Intranet, ao qual se seguiram posteriormente plataformas *Sharepoint* distribuídas pelas obras. Estas plataformas *Sharepoint* permitem, entre outras funcionalidades, o arquivo de documentação das obras. Este reforço de investimento tem como principais pressupostos a optimização das ferramentas de trabalho dos seus colaboradores e a melhoria das condições de negócio da empresa.

Com o forte declínio da económica no sector da construção, associado a fortes investimentos realizados, a Novopca Construtores Associados viu-se obrigada a realizar um pedido de insolvência em meados de 2011. Este pedido colmatou no despedimento colectivo de um grande número de colaboradores; embora a empresa continue a laborar com as condições mínimas indispensáveis à realização dos projectos que ainda estão em curso.

1.4 Organização do documento

A presente dissertação é constituída por oito partes principais, designadamente:

1. **Introdução:** neste capítulo são apresentados os conteúdos gerais do projecto, onde é feito o enquadramento do projecto, assim como os objectivos que pretende alcançar. Neste capítulo é também realizada uma breve descrição da empresa onde o projecto foi desenvolvido assim como a organização deste documento.
2. **Estado da Arte:** neste capítulo é feito o levantamento do estado da arte relativo a várias temáticas. Inicia-se pelo levantamento das vantagens e limites da utilização de dispositivos móveis no sector da construção, seguido pela apresentação de sistemas comerciais existentes para este sector e por fim é realizado o estado da arte relativo a dispositivos móveis.
3. **Sistemas Operativos Móveis:** neste capítulo é apresentado o estudo realizado sobre os sistemas operativos móveis. Este estudo contempla a decisão de qual o sistema operativo a utilizar na implementação do projecto e o estudo mais aprofundado desse mesmo sistema.
4. **Comunicação:** neste capítulo é apresentado um estudo sobre qual a melhor forma de trocar informação entre um sistema móvel e um servidor.
5. **Arquitectura a implementar:** este capítulo é responsável por apresentar a arquitectura proposta para a implementação da solução móvel pretendida.
6. **Implementação:** este capítulo apresenta com detalhe todo o desenvolvimento realizado. Aqui é colocada em prática a implementação da arquitectura proposta no capítulo anterior, assim como a descrição da implementação dos módulos que integram a plataforma móvel.
7. **Testes:** neste capítulo são apresentados os testes realizados à plataforma implementada e respectivos resultados obtidos.
8. **Conclusões:** o último capítulo pretende explicar as conclusões obtidas com a realização deste projecto, nomeadamente quais os objectivos cumpridos, limitações detectadas e potenciais desenvolvimentos futuros.

2 Estado da arte

Neste capítulo é feito o levantamento do estado da arte relativo à utilização de sistemas móveis no sector da construção civil e obras públicas. Mais concretamente serão abordadas as vantagens que estes sistemas apresentam para o desenvolvimento do sector, os limites destes sistemas, um conjunto de sistemas empresariais existentes no mercado e por fim o estado actual dos dispositivos móveis.

Ao observarmos as características do sector da construção facilmente nos apercebemos que existem diferentes trabalhadores a actuarem em diferentes locais. A realização eficaz dos seus respectivos trabalhos, neste ambiente distribuído em que estão inseridos, só é viável se estes estiverem acesso constante a toda a informação de que necessitam [Deibert et al., 2009]. A automatização da captura de informações e acesso via dispositivos móveis é considerado o primeiro passo para melhorar o desempenho dos processos neste contexto [Deibert et al. 2008]. A utilização destes dispositivos deverá de reduzir redundância de trabalho, melhorar a comunicação e reduzir os tempos de obtenção de informação.

Actualmente existem várias aplicações para dispositivos móveis utilizadas em local de obra, porém estas são na sua generalidade apenas vocacionadas para utilização por parte dos gestores de obra, ficando portanto por resolver os problemas com que os restantes trabalhadores se deparam todos os dias.

2.1 Vantagens dos Dispositivos Móveis no Sector da Construção

A introdução de tecnologia móvel no sector da construção traz um alargado número de vantagens tanto para as construtoras como para os clientes. Esta tecnologia tem o potencial de reduzir tempos de construção, custos, defeitos, acidentes e desperdícios, e aumentar a previsibilidade e produtividade [Bowden et al., 2006]. Estas melhorias podem ser alcançadas através do fornecimento de acesso a informação actualizada e relevante em local de obra, de identificação automática de materiais, de comunicação facilitada entre colaboradores em obra e colaboradores fora da obra e através de informação de progresso em tempo real [Vilkko et al., 2008].

Vantagens para os colaboradores:

- Maior rapidez na execução de tarefas;
- Acesso a informação actualizada em qualquer parte do mundo;
- Maior fluidez no fluxo de trabalho entre colaboradores;

Vantagens para a empresa:

- Aumento de produtividade, eficácia e eficiência;
- Redução de custos;
- Redução dos tempos de construção;
- Maior facilidade na gestão da informação;
- Aumento de competitividade em relação a outras empresas;

Vantagens para o cliente:

- Diminuição do custo final das obras;
- Diminuição das possibilidades de ocorrência de derrapagens orçamentais;
- Diminuição do tempo de construção;
- Obtenção de um produto com maior qualidade de construção;

2.2 Limitações dos Dispositivos Móveis no Sector da Construção

Os dispositivos móveis, apesar de todas as vantagens que acarretam, apresentam limites em várias características. Estes dispositivos apresentam limites ao nível da memória, de poder de processamento e de espaço em disco quando comparados com computadores normais. Ao nível das comunicações, as ligações que estes disponibilizam, seja WiFi, 3G ou outro tipo de ligação, são susceptíveis a desconexões e a variações nas taxas de transferência de dados. Uma das principais limitações que apresentam é ao nível da autonomia, pois o funcionamento destes dispositivos está sempre dependente das baterias que incorporam [Satyanarayanan, 1996].

Os dispositivos móveis mais utilizados actualmente, intitulados *SmartPhones*, são produzidos com o principal objectivo de serem fáceis de transportar e guardar, ou seja, são produzidos de forma a serem o mais pequenos e leves possíveis. Esta característica levanta limites relativos à sua usabilidade através de ecrãs e teclados físicos de dimensão reduzida. Os tamanhos reduzidos dos ecrãs levantam grandes desafios ao desenvolvimento de *software*, pois estes têm que ser funcionais, intuitivos e devem de apresentar todas as informações aos utilizadores num pequeno espaço visual.

Como as limitações existentes não fossem por si só suficientes, o sector da construção apresenta ambientes agrestes para estes dispositivos, obrigando a que estes estejam preparados para funcionarem neste tipo de ambientes. Os locais de obra apresentam condições que podem variar em termos de temperatura, de humidade, de barulho, de sujidade e de luz. Os dispositivos devem também de ser resistentes ao ponto de conseguirem aguentar lesões físicas, como por exemplo quedas e colisões com todo o tipo de materiais [Vilkko et al., 2008].

2.3 Sistemas Comerciais

Neste capítulo são apresentados dois sistemas comerciais para dispositivos móveis que têm como principal objectivo agilizarem a execução de várias tarefas realizadas em local de obra.

2.3.1 Penta Mobile

A Penta Technologies (2011) é uma empresa especializada no desenvolvimento de *software* para o sector da construção. O seu principal produto, designado PENTA Construction Software, é um sistema de gestão de todo o negócio de uma empresa de construção. Este sistema, para além de oferecer funcionalidades ao nível de gestão, é composto por uma plataforma móvel que permite agilizar processos de trabalho e a comunicação entre os locais de obra e os escritórios das empresas.

A plataforma móvel desenvolvida pela Penta Technologies é acedida através do *Browser*, criando assim a possibilidade de ser utilizada em qualquer dispositivo móvel que disponibilize um *Browser*. Esta plataforma subdivide-se em dois produtos: Mobile Field Reporting e Mobile Field Service [Penta, 2011].

O Mobile Field Reporting permite que os trabalhadores em obra troquem informações com os servidores centrais das organizações, nomeadamente: horas de trabalho dos trabalhadores, horas de trabalho das máquinas e controlo de tarefas.

O Mobile Field Service possibilita a sincronização de informação entre os gestores das organizações e os técnicos que estejam em local de obra, nomeadamente: actualizações do estado dos trabalhos, historial de serviços e encomenda de materiais.

2.3.2 Jonas Mobile

A Jonas Mobile [Jonas, 2011] é uma aplicação desenvolvida para sistemas operativos Windows. A aplicação tem como principais objectivos o aumento da produtividade dos trabalhadores, da satisfação dos clientes e a diminuição dos tempos de ciclo de facturação.

A Jonas Mobile permite que ordens de trabalho sejam trocadas entre os escritórios das organizações e os trabalhadores que se encontram em local de obra de uma forma fácil e eficiente. As suas principais funcionalidades são [Jonas, 2011]:

- Gestão de tarefas;
- Listagem de equipamentos, clientes e inventários;
- Controlo de horas de trabalho;
- Sincronização automática com o *software* Jonas Dispatch Board.

Esta aplicação traz vantagens tanto para os expedidores de tarefas como para os técnicos responsáveis pela realização dessas tarefas. Os expedidores reduzem o tempo de atribuição de tarefas e de comunicação entre o escritório e os técnicos. Os técnicos, por sua vez, reduzem os tempos gastos com a gestão de documentação, aumentam a eficiência na gestão de tarefas e eliminam a necessidade de deslocação para o escritório para a resolução de tarefas burocráticas.

2.4 Dispositivos Móveis

Os dispositivos móveis actualmente podem ser divididos em duas grandes áreas: *SmartPhones* e *Tablets*. Enquanto os *SmartPhones* são actualmente os dispositivos móveis mais utilizados em todo o mundo, os *Tablets* ganharam expressão desde o lançamento do iPad pela Apple em 2007.

Actualmente no mercado existe uma oferta quase infindável de dispositivos móveis, sendo por vezes o mais difícil escolher qual o que melhor satisfaz as necessidades dos utilizadores. As ofertas existentes podem ser separadas por uma das características mais importantes destes dispositivos: o sistema operativo incorporado.

O projecto aqui documentado tem como base a criação de uma plataforma que será utilizada em *SmartPhones*, como consequência neste capitulo apenas será feito o estado da arte relativa a estes equipamentos, organizado por sistema operativo que incorpora.

2.4.1 iOS

O sistema operativo da Apple para dispositivos móveis é pioneiro na sua interface intuitiva e apelativa. Totalmente desenvolvido para utilização sem teclados físicos, o iOS interage com o utilizador através do seu ecrã ultra sensível que abrange também funcionalidades de multi-toque [Apple Inc, 2011].

O iOS, que se encontra na versão 4.3.5, apenas funciona em dispositivos fabricados pela própria Apple, sendo o mais conhecido o iPhone (actualmente na versão 4) (Figura 1).

Figura 1 – iPhone 4 [Apple Inc, 2011]

O iPhone foi responsável pela introdução no mercado de várias características inovadoras como o já referido ecrã com multi-toque, a utilização de acelerómetros, a interface de utilizador e a App Store. A App Store é uma loja virtual de venda de aplicações que permite aos criadores de *software* colocarem as suas aplicações à venda num local homogéneo, onde todos os utilizadores têm acesso e a possibilidade de as adquirir.

O iPhone 4 está equipado com um processador ARM Cortex-A8 de 1 Ghz, 512Mb de memória RAM e 16 ou 18 GB (dependendo da versão) de armazenamento interno [Apple Inc, 2011].

2.4.2 Android

O sistema operativo da Open Handset Alliance é baseado em Linux e como tal tem a vantagem teórica de poder ser utilizado em qualquer equipamento físico [Open Handset Alliance, 2011].

Actualmente na sua versão 2.3.7 (nome de código *Gingerbread*) para *SmartPhones*, o Android fornece todo o conjunto de ferramentas disponibilizadas pela Google, nomeadamente: pesquisas via Google, GMail, YouTube e Google Maps.

A distribuição de aplicações é realizada através da utilização do Market que apresenta um funcionamento semelhante à App Store da Apple.

O Nexus S (Figura 2) é o *SmartPhone* criado pela parceria entre a Google e a Samsung e é actualmente um dos principais dispositivos móveis equipados com Android. Equipado com um *design* elegante, um processador Hummingbird de 1 GHz, 16 GB de memória interna e uma unidade de processamento gráfico dedicada, o Nexus S é um dos *SmartPhones* mais rápidos da actualidade [Google Inc, 2011b].

Figura 2 – Nexus S [Google Inc, 2011b]

Uma das inovações introduzidas pelo Nexus S é a incorporação de funcionalidades de comunicação de curta distancia (NFC). Esta funcionalidade permite que este dispositivo consiga ler informação de etiquetas “inteligentes” que incorporem chips NFC.

2.4.3 Symbian

O Symbian é um dos sistemas operativos móveis mais utilizados em todo o mundo. Apesar de estar a entrar na fase final do seu ciclo de vida (deixará de obter actualizações a partir de 2016), este continua a fazer parte de muitos equipamentos desenvolvidos pela Nokia.

O Symbian era um sistema operativo vocacionado para telemóveis básicos, que apresentam poucas funcionalidades em comparação com os *SmartPhones*. Com a expansão do mercado deste tipo de dispositivos e o constante desaparecimento dos tradicionais telemóveis, o Symbian foi obrigado a actualizar-se para responder à exigência do mercado. Foi então lançado a sua actual versão, o Symbian Belle [Nokia Inc, 2011].

O Symbian Belle está equipado com uma interface de utilizador completamente diferente das antigas versões do Symbian. Este engloba vários ecrãs virtuais onde podem ser colocados ícones e *wigdets*. Uma das principais novidades que apresenta é o facto de ser possível trocar imagens, documentos, músicas e qualquer outro tipo de ficheiros através de um simples gesto entre S*martPhones [Nokia Inc, 2011].*

A distribuição de aplicações desenvolvidas por terceiros é realizada através da Ovi Store, que é uma loja virtual semelhante ao Market do Android e à App Store do iOS.

Actualmente existem três dispositivos equipados com o Symbian Belle: o Nokia 600, o Nokia 700 e o Nokia 701 (Figura 3).

Figura 3 – Nokia 701 [Nokia Inc, 2011]

O Nokia 701 é, de acordo com a Nokia, o dispositivo que incorpora o ecrã mais cristalino do mercado. Este dispositivo apresenta um *design* elegante e robusto construído à base de alumínio. Incorpora também a tecnologia NFC que faz uso das funcionalidades disponibilizadas pelo Symbian Belle, um processador de 1 Ghz e capacidade de armazenamento de 8 Gb [Nokia Inc, 2011].

2.4.4 Windows Phone

O Windows Phone [Microsoft Inc, 2011a] é um sistema operativo para dispositivos móveis desenvolvido pela Microsoft e é o sucessor do Windows Mobile. Ao contrário do Windows Mobile, que tinha como principal alvo o mercado empresarial, o Windows Phone é destinado aos utilizadores comuns.

Actualmente na versão 7.10.7720, o Windows Phone oferece uma nova interface de utilizador denominada Metro. A Metro é baseada num conjunto de princípios que visam criar interfaces leves, simples e rápidas. Uma das características que mais impacto visual tem para o utilizador são as transições animadas muito bem conseguidas entre ecrãs e aplicações [Microsoft Inc, 2011b].

A distribuição de aplicações é realizada, à semelhança dos outros sistemas operativos móveis desta geração, através de uma loja virtual: a Marketplace.

Actualmente existem já vários dispositivos equipados com o Windows Phone 7, sendo um dos mais conhecidos o HTC HD7 (Figura 4).

Figura 4 – HTC HD7 [HTC, 2011]

O HTC HD7 está equipado com um ecrã de 4,3 polegadas, uma câmara que grava em alta definição (720p), um processador Scorpion de 1 GHz e 576MB de memória RAM. Este dispositivo está vocacionado para o mercado do entretenimento [HTC, 2011].

3 Sistemas Operativos Móveis

Neste capítulo é apresentada uma análise ao historial dos dispositivos móveis, seguida de uma comparação entre sistemas operativos móveis e uma análise mais detalhada à plataforma Android.

Um Sistema Operativo Móvel é responsável pelo controlo de um dispositivo móvel, tais como um PDA, um *Smartphone* ou um *Tablet*. Estes sistemas operativos começaram a ganhar força com o aparecimento dos *Smartphones*, já que estes oferecem mais capacidade de computação e de conectividade que os telemóveis básicos. Os *Smartphones* permitem aos seus utilizadores a instalação de aplicações da mesma forma que um *desktop* permite, para além de permitirem efectuar chamadas da mesma forma que os telemóveis convencionais.

3.1 Nota histórica

O primeiro *Smartphone* surgiu em 1992 e tinha a designação de IBM Simon, este dispositivo oferecia ao utilizador várias aplicações e em vez do típico teclado físico, estava equipado com um ecrã táctil, o que para a época representava um avanço tecnológico significativo [Schneidawind, 1992].

Apesar do lançamento do IBM Simon em 1992, o termo *Smarthphone* foi utilizado pela primeira vez em 1997 com o lançamento do Ericsson GS88 [Stockholm, 2011].

Tendo sido iniciada a era dos *Smartphones*, no ano 2000 foi lançado o primeiro dispositivo móvel, com o nome de Ericsson R380, a utilizar um sistema operativo aberto: o Symbian OS [Baker, 2005], o qual é ainda hoje um dos sistemas operativos móveis mais utilizados no mundo. Em 2001 a Microsoft apresentou o Windows CE (*Compact Edition*), um sistema operativo de código fechado destinado a sistemas embebidos, nos quais se enquadram os *Smartphones*. A RIM (Research In Motion) lançou também o seu sistema operativo, o BlackBerry OS (conhecido também por RIM, nome da empresa criadora) que teve um grande sucesso nos Estados Unidos da América. Estes três sistemas operativos foram os que tiveram maior aceitação durante vários anos.

Em 2007 o mundo dos *Smartphones* foi revolucionado pela introdução do iPhone, equipado com o sistema operativo iOS, pela Apple [Macworld, 2007]. O iPhone estava equipado com um ecrã totalmente táctil e com uma interface bastante intuitiva, o que levou a que a sua quota de mercado rapidamente começasse a ganhar expressão. Foi devido ao sucesso do iPhone que os *Smartphones* começaram a ganhar força em relação aos tradicionais telemóveis; apesar dos preços praticados pela Apple serem altos este facto não foi

impeditivo para o crescimento sucessivo das vendas. O iOS tem, porém, o senão de apenas ser compatível com equipamentos da Apple, não sendo possível a sua utilização em outros equipamentos. Em 2008 a Apple lançou a App Store, uma loja virtual de venda de aplicações, que foi, a par do iTunes, uma das razões do sucesso do iPhone.

No ano de 2008, e em resposta à crescente procura do mercado por *Smartphones*, a Google lançou o Android. Este sistema operativo foi originalmente desenvolvido pela Android Inc., adquirida pela Google no ano de 2005 [Businessweek, 2005]. De forma a ganhar expressão no mercado, a Google liderou a criação da Open Handset Alliance (aliança de várias empresas ligadas a dispositivos móveis, com a intenção de criar padrões abertos nesta área). Um dos factores que levou ao sucesso deste sistema operativo foi o facto de os utilizadores terem total controlo sobre o sistema, podendo alterá-lo da forma que entenderem, ao contrário do que acontece, por exemplo, com o iOS. O facto de o Android ter como base os serviços da Google, bastante difundidos no mercado, também ajudou ao seu sucesso. Ao contrário do iOS, o Android não é dependente da plataforma, podendo ser utilizado em quase todos os equipamentos existentes.

De forma a combater a constante perda de mercado e as novidades apresentadas pelos sistemas operativos da Apple e Google, a Microsoft lançou em 2010 o Windows Phone. Este sistema baseado no Windows CE, marcou uma mudança de paradigma por parte da empresa de Redmond: este tem como objectivo principal chegar ao mercado dos utilizadores comuns, ao contrário dos seus antecessores, que estavam claramente desenvolvidos para o mercado empresarial.

3.2 Comparação entre Sistemas Operativos móveis

De forma a ser possível a realização do projecto aqui documentado, foi necessário fazer a selecção de qual o sistema operativo móvel a utilizar. Para tal, foram analisados diversos factores, sendo os mais preponderantes as actuais quotas de mercado e as linguagens de programação utilizadas no desenvolvimento aplicacional.

Em termos de quota de mercado, o sistema operativo que apresenta mais vendas realizadas no segundo trimestre de 2011, com um total de 43% de mercado de acordo com a Gartner Inc. [2011], é o Android. O iOS por sua vez também continua a ganhar mercado mas de uma forma menos acentuada, apresentando um total de 18% de mercado. O sistema da RIM, o Symbian e o Windows Phone têm marcado tendência contrária, com perdas de quotas de mercado tanto para o Android como para o iOS [Gartner Inc, 2009], [Gartner Inc, 2011]. Para contrariar a tendência de queda, a Microsoft e a Nokia formaram, já em 2011, uma parceria que visa dotar os equipamentos Nokia com Windows Phone, como também outra importante implicação: o Symbian começará aos poucos a desaparecer dos terminais da Nokia, tendo a

empresa informado que o seu sistema operativo só continuará a receber actualizações até 2016. Na Figura 5 está representada a quota de mercado registada nas vendas destes sistemas no segundo trimestre de 2011.

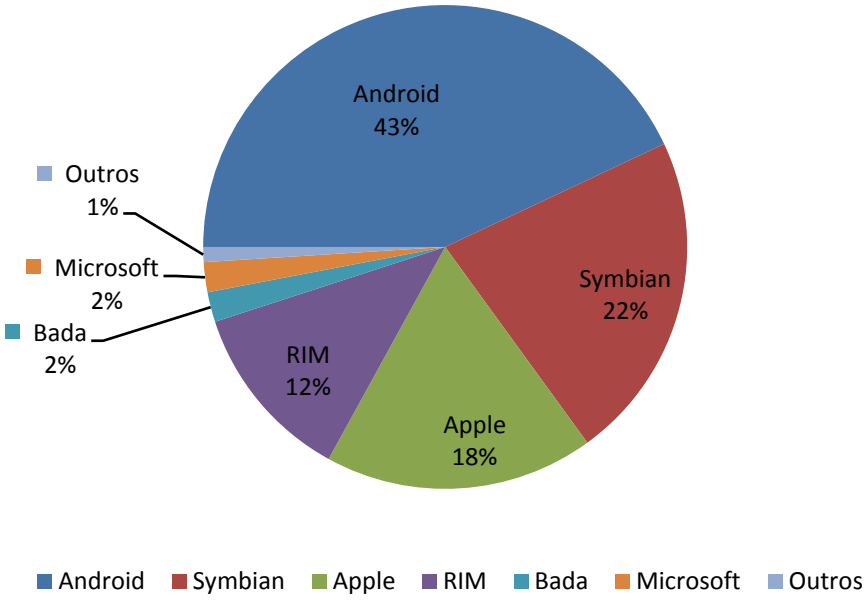

Figura 5 – Quotas de mercado no segundo trimestre de 2011 [Gartner Inc, 2011]

Os sistemas mencionados são muito diferentes entre si, tanto ao nível de sistema operativo base, passando pelas licenças até às linguagens de programação com o qual é desenvolvido *software* para a camada aplicacional. De entre os cinco sistemas analisados, apenas o Android é mantido por mais do que uma empresa, sendo, como já referido, suportado pela Open Handset Alliance. As linguagens de programação utilizadas no desenvolvimento de aplicações para cada um dos sistemas não são muito dispares: para Android utiliza-se C,C++ e Java; para o iOS C,C++ e Objective-C; para Windows Phone 7 .NET; BlackBerry OS utiliza Java; e por fim o Symbian utiliza também C++. Pode-se verificar que as linguagens predominantes são o C, C++ e Java. As licenças sobre o qual cada sistema está protegido são divididas em dois grupos: as licenças proprietárias e as open-source. O Android é o único com licença open-source, já os restantes (iOS, Windows Phone 7, BlackBerry OS e Symbian - que já esteve com licença open-source) estão sobre licenças proprietárias.

Tabela 1 – Características dos Sistemas Operativos Móveis [Wikipédia, 2011a]

Característica	Android	iOS	Windows Phone 7	BlackBerry OS	Symbian
Empresa	Open Handset Alliance	Apple	Microsoft	RIM	Nokia
Família SO	Linux	Mac OS X/Unix-like	Windows CE 7	-	-
Linguagem Programação	C, C++, Java	C, C++, Objective-C	.NET	Java	C++
Licença	Gratuito e de código aberto	Proprietária	Proprietária	Proprietária	Proprietária

Analisados todos os factores, a escolha recaiu sobre o Android. Este sistema operativo foi o que mais crescimento teve nos últimos anos e o que deverá dominar o mercado nos anos vindouros; para além disso, é suportado por uma aliança que conta com grandes empresas da área sendo esta liderada pela Google. O facto de ser um Sistema Operativo de código aberto é também um factor abonatório. No Android as aplicações são desenvolvidas utilizando a linguagem de programação Java, a qual possui um nível de maturidade elevado, uma grande versatilidade e é também a que neste momento mais popularidade tem no mundo da programação [TIOBE Inc, 2011].

3.3 Android

A plataforma Android é, como já referido, apoiada pela Open Handset Alliance que tem como objectivo acelerar a inovação dos dispositivos móveis, assim como oferecer aos utilizadores uma melhor, mais rica e menos dispendiosa experiência com estes dispositivos [Open Handset Alliance, 2011].

Esta plataforma inclui um sistema operativo (também conhecido com o mesmo nome) e aplicações chave para os utilizadores. Para desenvolvimento de aplicações esta plataforma disponibiliza um Software Development Kit (SDK) que inclui várias ferramentas de apoio ao desenvolvimento, sendo uma das mais importantes a possibilidade de emular o sistema operativo de forma a ser possível o desenvolvimento de aplicações sem a necessidade de manusear um dispositivo Android real.

O desenvolvimento aplicacional é preferencialmente realizado através do Eclipse IDE. Para tal, existe o Android Development Tools (ADT) que consiste num *plugin* para o Eclipse, desenhado de forma a oferecer aos programadores um ambiente poderoso e integrado para a criação de aplicações. O ADT permite a criação de projectos Android, a criação de

interfaces gráficas, a adição de componentes baseados no Android Framework API, testar as aplicações com as ferramentas disponibilizadas pelo SDK e compilar estas mesmas aplicações de forma a ser possível a sua distribuição.

3.3.1 Arquitectura

A arquitectura do Android, retratada na Figura 6, é composta pelas seguintes camadas: Linux Kernel, Libraries, Android Runtime, Application Framework e Applications [Google Inc, 2011a].

O Kernel de Linux é responsável pela abstracção do hardware dos dispositivos, o que permite que as camadas mais acima comuniquem através de d*evice drivers*. Esta camada é também responsável pela gestão de memória, gestão de processos, pilha de rede, pela segurança e por outros serviços da responsabilidade do sistema operativo.

Na camada acima encontramos as camadas Libraries e Android Runtime. A camada de livrarias fornece importantes funcionalidades para as aplicações e para outras componentes do sistema Android. As livrarias mais importantes são:

- Livraria C padrão, optimizada para dispositivos baseados em Linux;
- Motores gráficos 2D e 3D, responsáveis pela renderização dos gráficos.
- Framework Multimédia que suporta a reprodução e gravação de muitos formatos de vídeo e de imagens, nomeadamente os formatos: MPEG4, H.264, MP3, AAC, AMR, JPG, e PNG.
- SQLite que é um sistema gestor de base de dados poderoso e leve.

Ao mesmo nível encontramos o Android Runtime, que disponibiliza conjuntos de livrarias fundamentais assim como uma máquina virtual com o nome de Dalvik. Esta máquina virtual foi desenvolvida inicialmente por Dan Bornstein e foi projectada especialmente para dispositivos com pouca capacidade de memória e de processamento. Cada aplicação correrá como um processo independente e terá uma instância única da máquina virtual, isto porque a Dalvik foi desenvolvida de forma a permitir a execução de várias máquinas virtuais sem perdas de eficiência. A Dalvik é uma máquina virtual baseada em registos que executa classes Java previamente compiladas e transformadas para o formato Dalvik Executable (*.dex*) pela ferramenta “dx” incluida no SDK. A utilização do formato *.dex* deve-se ao facto de este ser optimizado para consumos mínimos de memória.

Na camada seguinte encontramos a *framework* aplicacional, que disponibiliza aos programadores de aplicações um conjunto alargado de componentes essenciais ao desenvolvimento das mesmas. Os componentes mais importantes são:

- Um conjunto alargado de Views, nomeadamente: listas, grelhas, caixas de texto, botões, caixas de selecção e muitos mais;
- Content Providers que permite a partilha de informação entre aplicações;
- Resource Manager que disponibiliza o acesso a recursos externos ao código da aplicação como imagens e ficheiros de interface de utilizador (*layout*);
- Notification Manager que permite que as aplicações apresentem notificações na barra de estado do Android;
- Activity Manager responsável pela gestão do ciclo de vida das aplicações e a funcionalidade de "navegar para trás" dentro das aplicações.

Os componentes disponibilizados nesta camada são essenciais para o desenvolvimento aplicacional, e por isso esta será alvo de um estudo mais extenso que as restantes camadas.

Por fim, na camada do topo temos as Aplicações; nesta camada estão já presentes um conjunto de aplicações essenciais ao sistema, como por exemplo, cliente de email, navegador de internet, calendário, entre outros. Nesta camada é onde se situarão todas as aplicações desenvolvidas pelo Android, aplicações essas que são escritas na linguagem Java.

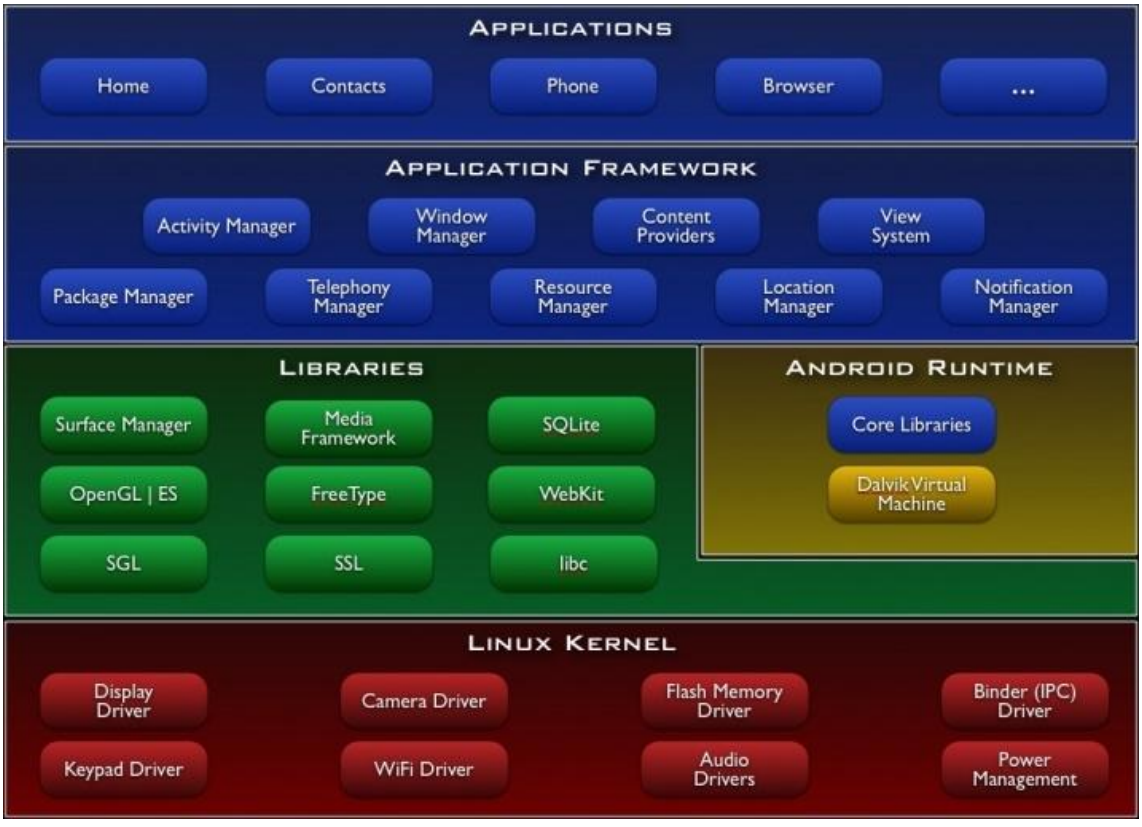

Figura 6 – Arquitectura Android [Google Inc, 2011a]

3.3.2 Views

O sistema de *views* do Android tem como objectivo dar às aplicações vários elementos de criação de interfaces de utilizador. Para que tal seja possível é disponibilizado o conceito de View, que é o bloco de construção mais elementar da interface de utilizador e serve como base para os *widgets*. Os *widgets* são elementos visíveis da interface, tais como botões, caixas de texto, rótulos, imagens, entre outros.

As interfaces de utilizador das aplicações raramente são constituídas apenas por um elemento, sendo necessário o agrupamento de vários elementos que constituem a interface. Desta forma o Android disponibiliza os ViewGroup que permitem a criação de uma árvore de *views* (Figura 7), que na realidade será a interface gráfica de utilizador final.

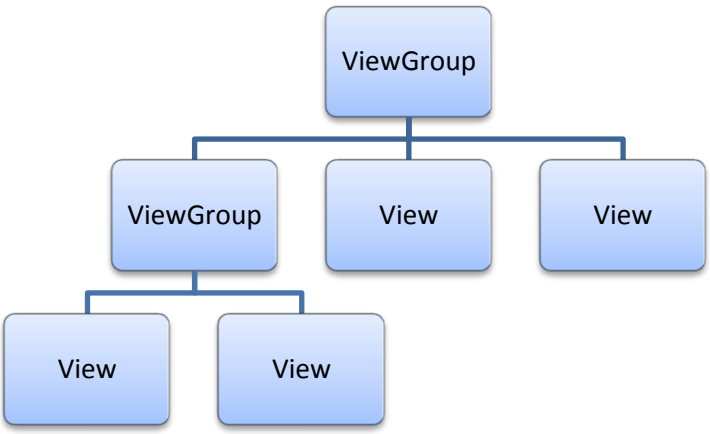

Figura 7 – Composição de Views [Google Inc, 2011a]

Os conjuntos de *widgets* existentes nas aplicações são colocados nas respectivas posições do ecrã através de utilização de *layouts*. O Android disponibiliza já vários tipos de *layouts*, entre os quais se encontram o LinearLayout e o RelativeLayout.

Uma grande vantagem do Android é o facto de permitir que a interface de utilizador seja definida em XML, ficando assim separada do código, para além de poder ser definida através de código. Os ficheiros XML que definem as interfaces têm que ser colocados no directório *res/layout* de forma a que estes possam ser acedidos através do código.

As *views* e *widgets* existentes nas interfaces podem ser acedidas através de um id que é atribuído pelo programador como sendo uma propriedade do objecto e deverá ser único na interface em causa.

No início de cada *Activity* é necessário indicar ao sistema qual o *layout* que por esta será utilizado, esta operação é realizada através da chamada à função *setContentView()*. Esta função pode receber como parâmetro uma *view* criada programaticamente, como também pode receber um número de um recurso da aplicação, sendo que este recurso

tem que ser um *layout* ou uma *view*. Os números atribuídos aos recursos são criados automaticamente pelo ADT, sendo que é mantida uma ligação entre o número criado pelo ADT e o *id* atribuído pelo programador. Desta forma o programador acede aos componentes da interface através de um *id* por si atribuído, ao invés do número automático criado pelo ADT.

3.3.3 Activities

Uma *activity* pode ser entendida como sendo um ecrã ou uma janela, fazendo a comparação com o sistema operativo Windows. Por exemplo, numa aplicação de e-mails existirão pelo menos duas *activities*, sendo uma para a listagem dos emails e a outra para a visualização desses mesmos emails. A classe *android.app.Activity* permite a criação de uma janela onde será inserida a interface de utilizador.

As *activities* são manipuladas como uma pilha (*stack*); quando uma *activity* é executada, esta é colocada no topo da pilha, passando a ser a *activity* em execução. A *activity* anterior é então colocada em background e só sairá deste estado quando estiver de novo no topo da pilha (Figura 8).

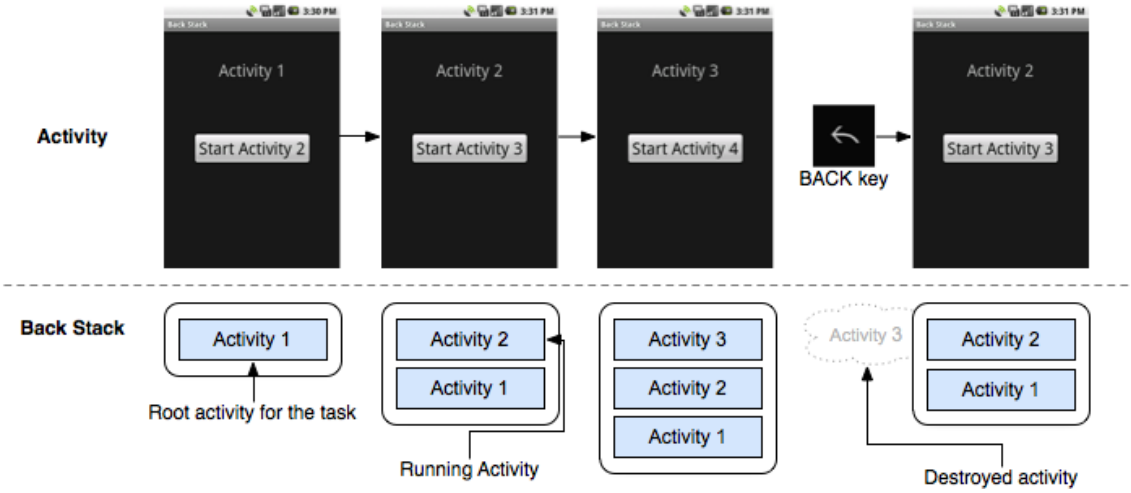

Figura 8 – Funcionamento da pilha de Activities [Hodin, 2011]

As *activities* têm um ciclo de vida (Figura 9) que pode ser definido em três tipos:

- Activa ou em execução: a *activity* está no topo da pilha, ou seja, está visível no ecrã do dispositivo;
- Pausada: a *activity* ainda está visível mas perdeu o foco para outra *activity*; isto acontece quando outra *activity* fica no topo da pilha mas não ocupa todo o ecrã ou é transparente. Neste caso a *activity* é mantida em memória sendo preservada a sua ligação ao gestor de janelas, porém, a sua execução podendo ser interrompida em casos de extrema carência de memória;

- Parada: a *activity* foi colocada neste modo depois de perder a sua posição no topo da pilha para outra *activity*; a ligação ao gestor de janelas é perdida, no entanto, a activity é mantida em memória de forma a poder retornar do ponto onde se encontrava. Em casos de necessidade de memória, esta pode ser interrompida.

Para cada alteração de estado o Android envia uma notificação para a *activity* em causa através da chamada de métodos específicos, designadamente:

- onCreate(): Invocado quando a *activity* é criada pela primeira vez, neste método é onde se deve iniciar a interface de utilizador. Este método pode também receber por parâmetro um Bundle, que é um conjunto de informação sobre o estado anterior desta activity, caso tenha existido.
- onRestart(): Método invocado depois da *activity* ter ficado parada e quando esta a ser iniciada novamente;
- onStart(): Este método é invocado quando a *activity* está a ficar visível para o utilizador;
- onResume(): Método invocado quando a *activity* está a iniciar a interacção com o utilizador. Neste momento esta está no topo da pilha de *activities*;
- onPause(): Invocado quando o sistema está prestes a retomar outra *activity*; este método deve ser utilizado para gravar definitivamente alterações a informação persistente, para parar animações e todo outro tipo de computação que consuma muito processador. Este método tem que ser de muito rápida execução, pois a próxima *activity* não será retomada enquanto este método não terminar;
- onStop(): Método invocado quando a *activity* deixa de estar visível para o utilizador, devido a outra *activity* ter sido posta por cima desta na pilha de *activities* ou simplesmente por esta ter sido terminada pelo utilizador. Uma nota importante é o facto de este método poder nunca ser invocado, isto porque quando o sistema necessita de mais memória os processos podem simplesmente ser interrompidos, sem existência de notificações por parte do sistema.
- onDestroy(): Este método é o último a ser invocado antes da *activity* ser destruída; seja porque está a terminar (através do uso do método *finish()*) ou porque o sistema está temporariamente a destruí-la de forma a ganhar espaço de memória.

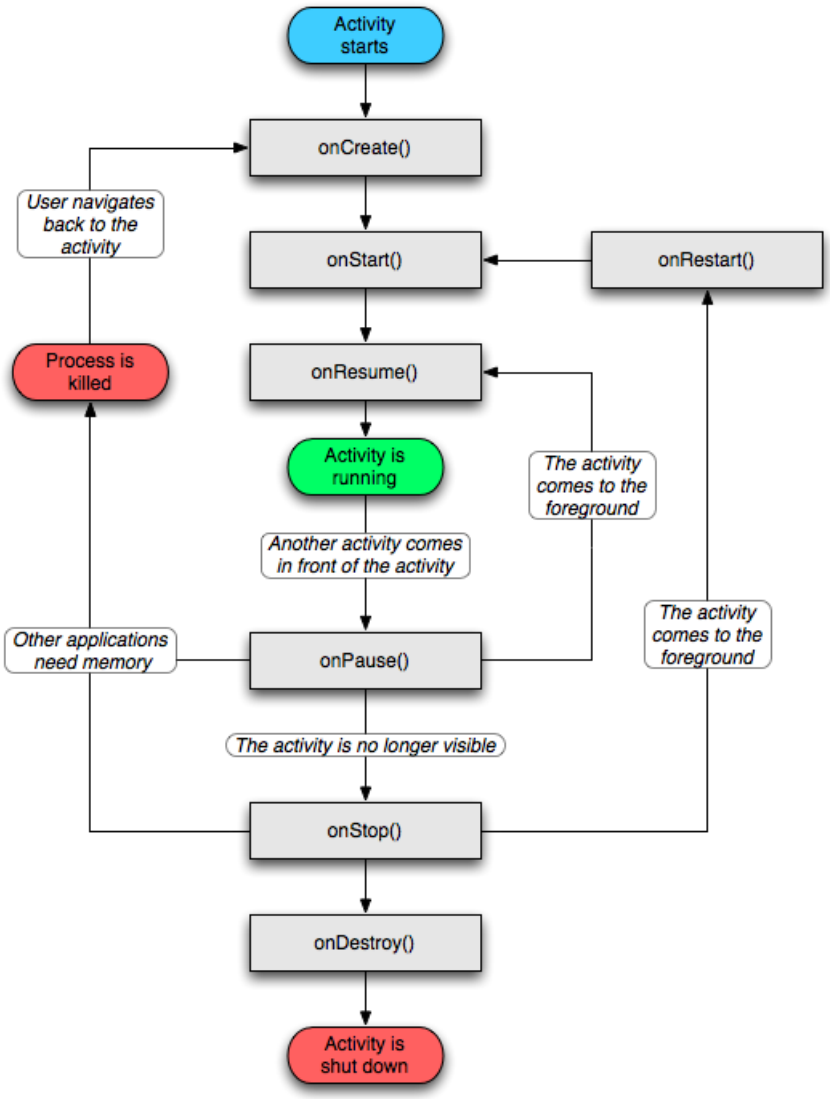

Figura 9 – Ciclo de vida de uma Activity [Google Inc, 2011a]

3.3.4 Intents

Intent é um componente do Android muito versátil, pois pode ser utilizado num vasto conjunto de diferentes situações. Estes podem representar mensagens de sistema que notificam as aplicações para o acontecimento de vários eventos, desde mudanças ao nível do *hardware*, passando por chegada de informação, até eventos das próprias aplicacionais, como também podem ser utilizados para representarem acções. Podemos utilizar um intent para visualizarmos por exemplo um ficheiro PDF, sendo que será da responsabilidade do Android determinar quais as aplicações que estão aptas a executar a acção pretendida. Este componente é também utilizado quando pretendemos iniciar uma *activity*. Os *intents* podem ser respondidos pela aplicação que o recebeu, simulando desta forma um modelo similar ao padrão *publish-subscrive*.

3.3.5 Content Providers

Content Provider é um componente que cria uma abstracção no acesso a qualquer tipo de informação armazenada ou não num dispositivo, passando qualquer tipo de informação a ser acedida sempre da mesma forma. A informação fica disponível através de um mecanismo do tipo Represantional State Transfer (REST) e como tal são utilizados *Uniform* Resource Identifier (URI) para o acesso à informação.

Os Content Providers permitem também a partilha de informação entre aplicações. O modelo de desenvolvimento do Android encoraja expressivamente a utilização deste componente de forma a disponibilizarmos informação, seja para outras aplicações ou dentro da nossa própria aplicação.

A abstracção criada pelos Content Providers permite que sejam alteradas as formas de acesso à informação apenas dentro destes, sendo essas alterações transparentes a todos os outros componentes que acedam à informação a partir deste componente. Por exemplo, podemos ter o nosso Content Provider a disponibilizar informação que está numa base de dados local ou que está armazenada num servidor externo (é completamente indiferente para a aplicação que obtém a informação através deste componente).

3.3.6 Services

Os serviços no Android são muito semelhantes aos serviços existentes nos tradicionais sistemas operativos, como é o caso do Windows. Estes são processos que são executados em segundo plano e que têm como principal característica o potencial de estarem em execução durante largos períodos de tempo, ao contrário das *activities* que têm um ciclo de vida muito curto.

O Android define dois tipos de serviços, os locais e os remotos. Os serviços locais são componentes que apenas podem ser acedidos pela aplicação detentora do mesmo, por outro lado, os serviços remotos podem ser acedidos por qualquer aplicação existente no sistema.

Um exemplo de um serviço existente no Android é o serviço responsável por reproduzir músicas na aplicação de música, que é uma das aplicações fundamentais já incluídas no próprio sistema. A aplicação é constituída por várias *activities*, desde a listagem de músicas até aos ecrãs de controlo da reprodução. Mas, como já visto anteriormente, o ciclo de vida destas *activities* é curto e caso estas percam o foco a sua execução é pausada, podendo por vezes ser terminada caso o sistema necessite de memória. Desta forma seria impossível o utilizador estar a ouvir música sem que estivesse com a *activity* de reprodução visível no ecrã; para contornar este problema, a reprodução da música é realizada através de um

serviço que corre em segundo plano. Para que tal seja possível, as *activities* comunicam com o serviço para que o utilizador consiga controlar a reprodução das músicas.

3.3.7 Broadcast receivers

Um *broadcast receiver* é um componente que responde a transmissões de anúncios. Muitos dos anúncios são transmitidos pelo próprio sistema, como por exemplo com a indicação de que a ligação *WiFi* está disponível ou que o dispositivo está com pouca bateria disponível. Porém, é possível que as aplicações transmitam anúncios entre elas ou entre *actitivies* da mesma aplicação. Como exemplo temos o caso de uma *activity* que apresenta o progresso de transferência de um determinado ficheiro enquanto um serviço é responsável pela respectiva transferência; a *activity* implementa a classe *BroadcastReceiver* de forma a receber anúncios do estado da transferência, enquanto o serviço de transferência transmite anúncios com essa indicação.

Os *broadcast receivers* não contêm interface de utilizador, sendo que a única forma de apresentar informação ao utilizador é através da criação de notificações na barra de estado do sistema.

3.3.8 Resources

Os recursos, tais como textos e imagens, devem estar sempre separados do código fonte das aplicações. Ao realizar esta separação, torna-se possível utilizar diferentes recursos para diferentes dispositivos, tendo em conta por exemplo o tamanho do ecrã ou a linguagem do mesmo; desta forma, torna-se muito simples a criação de, por exemplo, aplicações multi-linguagem.

Todos os recursos existentes nas aplicações devem ser colocados no directório *res/*, e dentro de subdirectórios que agrupem os recursos por tipos e por configurações.

Existem portanto dois tipos de recursos no Android: os por defeito e os alternativos. Os recursos por defeito são aqueles que são utilizados quando a configuração do dispositivo não é relevante ou quando esta não se encaixa com nenhuma configuração dos recursos alternativos. Os recursos alternativos são dependentes da configuração do sistema, por exemplo, podemos ter um recurso para quando o ecrã está na vertical e outro para quando o ecrã está na horizontal.

3.3.9 Android Manifest

O ficheiro AndroidManifest.xml define os conteúdos e comportamentos da aplicação. Aqui é necessário indicar todos os componentes existentes na nossa aplicação. Para além dos componentes normais, como as *activities* e os serviços, é também necessário declarar mais informação no Android Manifest, nomeadamente:

- Indicação das permissões que a aplicação requer, como por exemplo acesso à internet ou ao *SD-Card* do dispositivo;
- Indicação do nível mínimo da API necessário à execução da aplicação;
- Indicação do *hardware* ou software utilizado ou necessário à execução da aplicação, como por exemplo *bluetooth* ou câmara fotográfica.
- Indicação de outras livrarias necessárias para além das livrarias já disponibilizadas pelo Android, como é o caso da livraria do Google Maps.

4 Comunicação

Nos dias de hoje o acesso constante a informação é um bem essencial no nosso quotidiano [Metodista, 2011]. Este acesso é realizado pelas mais variadas formas, seja através da simples comunicação verbal entre pessoas ou através da leitura de um jornal utilizando um *Tablet*.

Com a explosão da internet, foi necessário criar meios de entendimento entre os sistemas informáticos, meios esses designados como protocolos de comunicação. Através destes protocolos, os vários sistemas ficam habilitados a comunicarem entre si, sabendo a forma como transmitir e como receber informação. De acordo com Comer (2000) os protocolos estão para a comunicação como as linguagens de programação estão para a computação.

No projecto em estudo o acesso à informação é um factor indispensável. Como tal foi necessário determinar qual o método de comunicação que melhor se enquadra no projecto. Uma das soluções mais conhecidas e utilizadas hoje em dia para a comunicação inter-máquinas designa-se por serviços Web, do termo anglo-saxónico *Web Services*. Os *Web Services* dividem-se em dois tipos, os quais foram alvo de estudo: os *Big Web Services* (também conhecidos como *SOAP Web Services*) e os *RESTful Web Services*. Neste último é possível utilizar vários formatos de informação a transmitir, pelo que foram estudados os dois principais formatos de representar informação existentes: o XML e o JSON.

4.1 Web Services

A W3C define um *Web Service* como sendo um sistema informático desenvolvido de forma a suportar a interoperabilidade entre máquinas sobre uma rede, mais concretamente [W3C, 2011a]:

"Um *Web Service* é um sistema de *software* projectado para suportar interoperabilidade entre máquinas numa rede de computadores. Este contempla uma interface descrita num formato processável por máquinas (especificamente WSDL). Outros sistemas interagem com os *Web Services* numa forma definida pela sua própria descrição através do uso de mensagens SOAP, normalmente transmitidas através de HTTP com uma serialização XML em conjunto com outros padrões relacionados com a Web."

Os serviços Web podem, como já referido, ser divididos em duas áreas distintas: *Big Web Services* e *Restful Web Services*.

4.1.1 Big Web Services

Os "*Big Web Services*" utilizam mensagens XML que seguem o padrão SOAP e são mais populares entre empresas tradicionais. Nestes sistemas existem descrições dos serviços e operações oferecidas pelo serviço através de *Web Service Description Language* (WSDL). O WSDL não é um requisito de SOAP, mas é uma condição necessária por parte de várias SOAP *frameworks* para a geração automática de código do lado do cliente.

Concretamente, este tipo de serviços assenta sobre uma pilha de protocolos que permitem a definição, localização e implementação dos mesmos. A pilha de protocolos é constituída por:

- Simple Object Access Protocol (SOAP): protocolo para troca de informação estruturada na implementação de serviços Web numa rede de computadores;
- Web Services Description Language (WSDL): linguagem baseada em XML utilizada na descrição dos serviços fornecidos por um serviços Web;
- Universal Description, Discovery and Integration (UDDI): protocolo que define um método padrão de publicar e descobrir *Web Services*.
- Extensible Markup Language (XML): é uma linguagem de marcação que define um conjunto de regras para a criação de documentos capazes de serem analisados por máquinas.

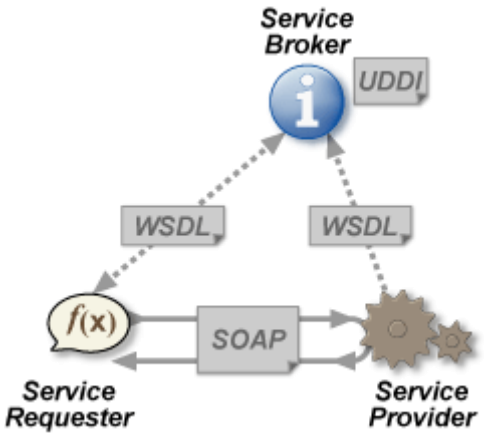

Figura 10 – Arquitectura de um Web Service [Wikipédia, 2011b]

Ao nível de transporte da informação os *Web Services* são independentes de qualquer protocolo, sendo possível a utilização de HTTP, SMTP ou até FTP para o efeito. Apesar desta independência, é prática comum, por várias razões, a utilização do protocolo HTTP.

A segurança neste tipo de serviços é realizada através da extensão *WS-Security*, publicada pela *Organization for the Advancement of Structured Information Standards* (OASIS). O protocolo especifica como garantir a integridade e confidencialidade das mensagens trocadas entre clientes e servidores, e permite a comunicação de vários formatos de segurança baseados em *token*s, como Kerberos e X.509 [Oasis, 2011]. A utilização deste

flexível e poderoso protocolo tem porém uma grande desvantagem: este adiciona uma sobrecarga significativa no processamento das mensagens SOAP devido ao aumento do tamanho das mensagens e à necessidade de processamento criptográfico associado [Lascelles and Flint, 2006].

Apesar de largamente utilizados, este tipo de *Web services* tem sido criticado por apologistas dos RESTful Web Services, principalmente devido à grande complexidade envolvida no desenvolvimento deste tipo de serviços [Bray, 2004].

4.1.2 RESTful Web Services

A *Representational state transfer* (REST) foi introduzida e definida no ano 2000 na dissertação de doutoramento de Roy Fielding. É apresentada como sendo um "*software architectural style*", que é normalmente entendido como a definição dos componentes de um software, as suas propriedades externas e o seu relacionamento com outros softwares. A arquitectura REST surgiu através da exploração de tecnologias e protocolos já existentes na Web, como são os casos do HTTP e o XML.

O conceito da arquitectura REST baseia-se na arquitectura cliente-servidor e, ao contrário dos Big Web Services onde os serviços podem ser entendidos como métodos, a REST é composta por recursos que estão disponíveis através de um URI. A maior implementação de um sistema em conformidade com a arquitectura REST é a World Wide Web.

O estilo de arquitectura REST descreve várias restrições aplicadas à arquitectura, nomeadamente [Fielding, 2000]:

- *Client-server*: os clientes estão separados dos servidores por uma interface uniforme, o que leva a que cada um deles seja independente do outro, sendo apenas necessário garantir a não alteração da interface que os une;
- *Stateless*: a comunicação entre cliente e servidor é efectuada com a restrição de nenhuma informação sobre o cliente ser armazenada entre pedidos, isto significa que cada pedido do cliente contém toda a informação necessária para que o servidor consiga responder ao pedido; qualquer informação de estado deverá portanto ser armazenada no cliente;
- *Cacheable*: a informação trocada entre cliente e servidor pode ser passível de ser colocada em cache, melhorando desta forma a escalabilidade e a performance dos intervenientes;
- *Uniform interface*: a interface definida pela arquitectura REST é guiada por quatro princípios:
    - Identificação de recursos: recursos individuais são identificados através de URIs. Os recursos ao serem transmitidos para o cliente são convertidos para

um formato que seja por este entendido, por exemplo os dados que estejam numa base de dados no servidor são transformados numa representação em XML antes de serem enviados ao cliente;
    - Manipulação de recursos através das representações: quando o cliente possui uma representação de um recurso e respectivas informações adicionais (metadados), esta representação é o suficiente para que o cliente altere ou elimine o recurso no servidor;
    - Mensagens auto-descritíveis: cada mensagem deve incluir informação que descreva a forma como esta deve ser processada;
    - Hipermédia como o motor do estado da aplicação: o cliente interage com o servidor inteiramente através de hipermédia disponibilizada dinamicamente pelo servidor. Esta restrição leva a que o cliente não necessite de ter conhecimento prévio de como interagir com o servidor para além de uma compreensão genérica de hipermédia;
- *Layered system*: de forma a melhorar a escalabilidade dos serviços, o cliente e o servidor não têm conhecimento de todas as camadas existentes entre eles, tendo apenas conhecimento da camada adjacente.
- *Code on demand*: é uma restrição opcional que permite que os servidores enviem lógica para os clientes, ou seja, enviem código passível de execução por parte dos clientes.

A dominação "RESTful Web Services" é utilizada para identificar serviços simples que utilizam o protocolo HTTP e que utilizam os princípios da arquitectura REST. Esta é uma colecção de recursos com quatro aspectos distintos [Fielding, 2000]:

- Uma URI base para o serviço Web;
- A definição do *internet media type* da informação suportada pelo serviço Web, normalmente JSON ou XML;
- O conjunto de operações suportadas pelo serviço Web através de métodos HTTP (POST, GET, PUT e DELETE);
- A API tem que ser orientada à hipermédia.

Os métodos HTTP: GET, POST, PUT e DELETE são utilizados para manipulação de recursos no servidor, servindo respectivamente para leitura, actualização, criação e eliminação destes mesmos recursos.

A forma de implementar segurança não é especificada pela arquitectura REST, sendo a sua implementação da responsabilidade dos programadores. Sendo o protocolo HTTP utilizado na comunicação entre cliente e servidor são normalmente utilizados os métodos de

segurança que este fornece. A segurança na comunicação ponto a ponto é normalmente garantida através da utilização de protocolos criptográficos como o SSL e o TLS. Ao nível da autenticação são utilizados normalmente dois métodos especificados pelo protocolo HTTP: *Basic e Digest Access Authentication* [Richardson and Ruby, 2007]. O método *Basic* é, como o nome indica, muito simples de usar, porém tem a grande desvantagem de enviar as credenciais de autenticação sem qualquer tipo segurança aplicada, é por esta razão que este método deve ser usado apenas em conjugação com a utilização de SSL ou TLS. O método *Digest* por sua vez já utiliza criptografia para proteger as credenciais de autenticação e não necessita assim da utilização dos protocolos SSL ou TLS para o efeito.

A REST é indicada como sendo mais simples que a abordagem SOAP, que requer a criação de aplicações clientes que entendam os contratos de interface disponibilizados pelos serviços. Por outro lado, a pilha de protocolos dos "Big Web Services" estão mais desenvolvidas e oferecem mais soluções no desenvolvimento de aplicações.

4.2 Representação da informação

A comunicação entre máquinas é, como já referido, um aspecto importantíssimo nos dias de hoje. Para que esta comunicação seja possível é necessária a existência de formatos de representação da informação que as máquinas sejam capazes de ler e criar, ou seja, de entender. Em conformidade com esta necessidade, foram criados vários formatos para troca de mensagens entre máquinas, desde ficheiros de texto com uma determinada lógica imposta (por exemplo os CSV) até aos bem definidos e auto explicativos documentos XML. O surgimento deste último veio revolucionar a forma como as máquinas comunicam, porém, tudo tem vantagens e desvantagens, e para combater as desvantagens do XML existe o JSON.

Como já referido, os dois principais formatos de representação de informação são o XML e o JSON e como tal foram estes dois formatos alvo de análise no desenvolvimento do presente projecto.

4.2.1 XML

XML (Extensible Markup Language) é uma linguagem de marcação derivada da linguagem SGML (ISO 8879), que é representada em forma de texto e se apresenta como sendo simples e muito flexível [W3C, 2011b]. A concepção deste formato foi muito focada para a representação de documentos, porém este é muito utilizado na representação de vários tipos de informação, sendo largamente utilizado em *Web Services*.

O formato XML foi criado com vista a atingir vários objectivos [W3C, 2008], sendo os principais:

- Deve ser fácil de criar programas que analisem documentos XML;
- Documentos XML devem ser possíveis de ler por humanos e razoavelmente claros;
- Documentos XML devem ser criados facilmente;
- O formato XML deve suportar uma larga variedade de aplicações;

Apesar destes objectivos de concepção, é de notar que a especificação XML não indica claramente a forma como os programadores devem criar programas capazes de processarem documentos XML. Desta forma foram desenvolvidas várias interfaces de programação que visam a análise de documentos no formato XML. Estas interfaces seguem normalmente dois tipos de abordagens:

- *Stream-based* ou *event-based*: esta abordagem leva a que a API analise os documentos XML sequencialmente e envie um sinal à aplicação sempre que são encontrados novos componentes no documento. Este tipo de abordagem tem, teoricamente, a vantagem de necessitar de pouca memória disponível e de ser mais célere na análise de documentos.
- *Tree-based*: neste tipo de abordagem, os documentos XML são convertidos para um conjunto de objectos em memória representados através de uma árvore de objectos. Como todo o documento é colocado em memória, este tipo de abordagem não é aconselhada para leitura de grandes documentos, pois pode levar a que o sistema fique sem memória disponível e à consequente interrupção da análise. A grande vantagem que apresenta perante as *stream-based* APIs é a possibilidade de acesso a qualquer parte do documento de forma aleatória.

Existem várias interfaces de programação para análise de documentos XML, sendo as mais importantes:

- *Simple API for XML* (SAX): é uma API, baseada em eventos (*event-based*), de acesso sequencial aos conteúdos de um documento XML. Esta API analisa o documento de forma sequencial do início ao fim, enviando sinais à aplicação sempre que documentos são encontrados;
- *Document Object Model* (DOM): é uma API que utiliza a abordagem *tree-based* e que permite a navegação dentro de um documento no formato HTML, XHTML ou XML, como se este fosse uma árvore de objectos.
- *Streaming API for XML* (StAX): esta API, *stream-based*, foi criada de forma a situar-se entre o que as APIs SAX e DOM oferecem. O StAX funciona de forma semelhante a um iterador de objectos, o que dá ao programador o poder de controlar a forma como o documento é analisado, podendo este, por exemplo, interromper a análise a

qualquer momento. Esta API não tem os problemas de memória que a DOM API apresenta, nem obriga à total análise do documento como a SAX API;

- *Transformation API for XML* (TrAX): esta API tem como principal objectivo a transformação de documentos XML através de XLST (XML Stylesheet Language for Transformations). A análise de documentos é efectuada através de XPATH (XML Path Language), que é uma linguagem de consulta que permite a selecção de nós de documentos XML.

Todas as APIs referidas têm as suas vantagens e desvantagens, sendo que cada uma deve ser aplicada a um determinado contexto. Por exemplo, caso necessitemos de realizar acessos aleatórios ao conteúdo de um documento XML, a melhor escolha passaria pela DOM API, mas se o nosso dispositivo ou computador apresenta pouca capacidade de memória as SAX ou StAX APIs são as que apresentam as melhores soluções. A TrAX API por outro lado, apresenta funcionalidades muito peculiares e para situações muito específicas, sendo esta normalmente utilizada na transformação de documentos XML.

Na Tabela 2 é apresentada uma comparação entre estas quatro APIs.

Tabela 2 - Comparação entre APIs de análise de documentos XML [Oracle, 2011]

Característica	StAX	SAX	DOM	TrAX
Tipo de API	*Pull, streaming*	*Push, streaming*	Árvore em memória	Regra XSLT
Facilidade de utilização	Alta	Média	Alta	Média
Compatibilidade XPath	Não	Não	Sim	Sim
Eficiência CPU e memória	Boa	Boa	Variável	Variável
Apenas para a frente	Sim	Sim	Não	Não
Leitura de XML	Sim	Sim	Sim	Sim
Escrita de XML	Sim	Não	Sim	Sim
Operações CRUD	Não	Não	Sim	Não
Incluído no Android	Não	Sim	Sim	Não

O sistema operativo Android é projectado para ser executado em dispositivos móveis que apresentam geralmente poucas capacidades de memória e de processamento, como tal a DOM API só deverá ser utilizada em casos especiais. A StAX e a TrAX API não estão incluídas nativamente na plataforma Android, ao contrário da SAX e DOM, o que é um ponto a favor destas duas APIs.

4.2.2 JSON

JSON (JavaScript Object Notation) é um formato leve de intercâmbio de dados computacionais. O formato JSON é de fácil leitura e escrita por humanos e de fácil geração e análise por parte de computadores. JSON é representado em formato de texto e é completamente independente de linguagens de programação [JSON, 2011].

O formato JSON é composto por duas estruturas:

- Uma colecção de pares de nomes e respectivos valores, muito semelhante ao representado através de uma matriz associativa;
- Uma ordenada lista de valores, muito semelhante aos conhecidos vectores ou matrizes não associativas;

O formato JSON é apresentado como sendo a alternativa “sem gordura” do XML, invocando que a mesma informação representada nos dois formatos requer mais metadados no formato XML. Com base neste facto, o JSON auto intitula-se como sendo mais eficiente de analisar por computadores que o XML [JSON, 2011].

O Android engloba nativamente um analisador deste formato, não sendo portanto necessária a inclusão de livrarias externas para o efeito.

4.3 Testes

Neste capítulo serão apresentados os resultados obtidos através da realização de testes às formas de comunicação apresentadas nos capítulos anteriores. Estes resultados servem como base para a escolha das tecnologias a utilizar na implementação do projecto aqui apresentado.

Concretamente, foram testadas três tipos de comunicações: Big Web Services, RESTful Web Services com XML e RESTful Web Services com JSON. De forma a obter resultados que permitam uma análise mais cuidada foram realizados três testes. O primeiro teste é composto por um único registo de dados de uma pessoa, onde são incluídas informações tais como nome, endereço de correio electrónico, morada, entre outras. O segundo teste será composto por 200 registos e, por fim, o último teste será composto por um total de 500 registos.

Os resultados serão analisados em três componentes distintas: quantidade de informação trocada entre cliente e servidor, tempo de execução e quantidade de memória utilizada pela aplicação cliente. A quantidade de informação trocada é o somatório dos tamanhos dos dados trocados entre o cliente e o servidor, incluindo aqui a quantidade de informação gerada por todos os protocolos de comunicação envolvidos. O tempo de execução de todo o processo é na verdade o total de tempo de execução da aplicação cliente, ou seja, é o

tempo decorrido desde que o cliente faz o pedido ao servidor até ao ponto em que o cliente tem já os dados obtidos armazenados correctamente em memória. A medição de tempos de execução está sempre sujeita a vários factores, variando de execução para execução. Desta forma este teste foi realizado cinco vezes, sendo depois considerada a média aritmética dos resultados obtidos. Por fim, a quantidade de memória utilizada é o máximo de memória utilizada pelo cliente ao longo do tempo, ou seja, é obtida no ponto em que a aplicação necessita de mais memória.

O ambiente de testes foi composto por um cliente e um servidor (Figura 11). O cliente é constituído por uma aplicação a executar no emulador Android em ambiente Windows. Do lado do servidor, temos um servidor Web Apache com PHP e MySQL também em ambiente Windows.

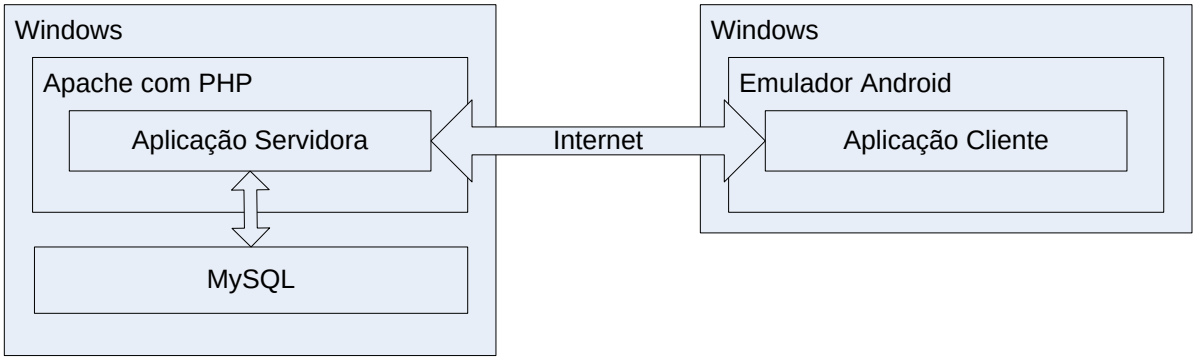

Figura 11 – Constituição do ambiente de testes

O tempo de execução foi obtido através da utilização da classe *TimingLogger* presente no Android, que foi desenhada exactamente com o propósito de medição de tempos de execução das aplicações ou métodos de aplicações.

A obtenção do máximo de memória utilizado pela aplicação foi realizada através da ferramenta DDMS (Dalvik Debug Monitor Server), que fornece dados sobre a utilização da *heap* de cada aplicação que esteja em execução.

Por fim, a obtenção da quantidade de informação trocada entre cliente e servidor foi realizada através da utilização de duas ferramentas: o *tcpdump* e o W*ireshark*. O *tcpdump* é uma ferramenta que permite a criação de relatórios de todo o tráfego de rede de um sistema operativo, neste caso o Android. O *Wireshark* por sua vez é uma ferramenta de análise de tráfego de uma rede de computadores; através desta ferramenta é possível analisar os relatórios gerados pelo *tcpdump*.

A plataforma Android não incorpora nativamente livrarias para consumo de serviços Web, sejam do tipo SOAP ou REST. Esta decisão da Google é no mínimo estranha, pois muitas das aplicações para dispositivos móveis necessitam de consumir serviços disponibilizados

por várias entidades na Web. Apesar deste facto, a Google incorporou nativamente livrarias para utilização do protocolo HTTP assim como para análise de documentos XML e JSON, o que já é um ponto de partida importante. Sendo a linguagem de desenvolvimento JAVA é possível a obtenção de livrarias externas que permitem o consumo de serviços Web baseados em SOAP e em REST.

4.3.1 Big Web Services

Como já referido, o Android não incorpora livrarias para consumo deste tipo de serviços, como tal, foi necessário obter uma livraria externa para o efeito. A livraria escolhida foi a kSOAP2 para Android, que se auto intitula como sendo uma livraria SOAP leve e eficiente [kSoap, 2011]. O kSOAP2 é amplamente utilizado pela comunidade Android.

A aplicação servidora foi construída através da utilização da livraria NuSOAP, que permite a criação de clientes e servidores de SOAP Web Services em PHP.

Os resultados obtidos, apresentados na Tabela 3, mostram que ao aumentar o número de registos, apesar da memória utilizada não aumentar de forma significativa, tanto o tempo de execução da aplicação cliente como o tamanho total da informação trocada aumenta significativamente. A obtenção de 500 registos, apesar de ser uma situação rara, leva a que o cliente fique em execução durante mais de 6 segundos, o que num dispositivo móvel não é aceitável em termos de usabilidade. A quantidade de tráfego registada também apresenta valores pouco animadores para quem utiliza maioritariamente ligações 3G, onde as operadoras de telecomunicações cobram por cada *kilobyte* de tráfego. A quantidade de tráfego tem também impacto directo na autonomia dos dispositivos móveis, seja qual for a interface de ligação sem fios à Internet [Sharkey, 2009].

Tabela 3 – Resultados dos testes a Big Web Services

Registos	Memória utilizada	Tempo de execução	Quantidade de tráfego
1	2,754 MB	460,8 ms	1,7 kb
200	3,258 MB	2821,8 ms	70,4 kb
500	3,695 MB	6217,8 ms	169,1 kb

4.3.2 RESTful Web Services com XML

No desenvolvimento da aplicação servidora deste tipo de serviços não foram utilizadas nenhumas livrarias adicionais. O PHP contempla já várias ferramentas e classes que permitem a criação de um serviço REST que forneça dados em formato XML.

A aplicação cliente, por sua vez foi constituída por uma Framework, denominada Restlet, que permite o consumo de serviços do tipo REST. A utilização desta Framework é simples e permite o desenvolvimento mais célere de aplicações.

O XML obtido na resposta do servidor foi analisado através do método SAX que está nativamente incorporado na plataforma Android. Porém, este método obriga a que o programador mantenha informação sobre o local onde o analisador se encontra no documento, o que leva a um aumento de complexidade de código.

Os testes realizados mostram bons resultados (Tabela 4) quando é obtido apenas um registo do servidor. Com 200 registos, o aumento de memória é pouco significativo, ao contrário do tempo de execução que sobe quase para os 3 segundos e a quantidade de tráfego que passa de menos de 1kb para 51,3kb. A obtenção de 500 registos é feita com o registo do mesmo aumento significativo, tanto no tempo de execução como na quantidade de tráfego. Apesar da quantidade de tráfego ser aceitável, o tempo de execução é claramente o ponto negativo dos resultados obtidos.

Tabela 4 – Resultados dos testes a RESTful Web Services com XML

Registos	Memória utilizada	Tempo de execução	Quantidade de tráfego
1	3,258 MB	771,8 ms	0,8 kb
200	3,758 MB	2667,2 ms	51,3 kb
500	3,945 MB	5466,8 ms	125,0 kb

4.3.3 RESTful Web Services com JSON

A aplicação de testes responsável pelo fornecimento deste tipo de serviços foi, da mesma forma que a anterior, desenvolvida sem a incorporação de livrarias externas ao PHP.

A aplicação cliente, por sua vez, utilizou a já mencionada Framework Restlet para o consumo do serviço Web. A informação em formato JSON foi analisada através do uso das classes já incorporadas no Android. É importante referir que o desenvolvimento desta aplicação necessitou de menos esforço que as aplicações anteriores. As classes para análise de JSON disponibilizadas pelo Android são muito intuitivas e simples de utilizar.

Os resultados obtidos, apresentados na Tabela 4, mostram que este método de troca de informações tem uma utilização de memória significativa, porém esta desvantagem perde força ao verificarmos que os tempos de execução são muito aceitáveis para o que se espera de um dispositivo móvel. A quantidade de tráfego gerada é também reduzida o que leva a tempos de transferência de informação reduzidos.

Tabela 5 – Resultados dos testes a RESTful Web Services com JSON

Registos	Memória utilizada	Tempo de execução	Quantidade de tráfego
1	3,258 MB	827,2 ms	0,7 kb
200	3,695 MB	1464 ms	33,7 kb
500	4,133 MB	1912,8 ms	81,1 kb

4.3.4 Comparação de resultados e conclusões

Os resultados obtidos, como já mencionado, servem como base à escolha de quais as tecnologias a utilizar na implementação deste projecto. Todas as componentes testadas são importantes, porém existe um factor que tem mais peso que os restantes. Este factor é a quantidade de informação trocada entre o cliente e o servidor. A justificação é simples: os equipamentos móveis, apesar de estarem hoje em dia equipados com antenas WiFi, utilizam em grande parte ligações 3G que estão muitas vezes "presas" a contratos que delimitam um plano de tráfego disponível por mês ou semana. Caso o limite de tráfego delimitado pelo plano seja ultrapassado, é necessário pagar por todo o tráfego extra. Tendo em conta a conjuntura actual, as empresas não se podem dar ao luxo de gastarem dinheiro em tráfego móvel só porque as aplicações estão mal feitas e geram mais tráfego que o necessário. É por esta razão que o factor da quantidade de informação trocada terá mais peso que os restantes dois factores.

Em termos de memória necessária à execução da aplicação cliente, os resultados, representados na Figura 12, evidenciam que o cliente que utiliza *Big Web Services* é mais eficiente que os restantes, sendo que o pior resultado verifica-se na combinação de REST com JSON. A diferença entre o melhor e o pior resultado é aproximadamente de apenas 1 *megabyte*, o que para os dispositivos actuais é pouco significativo.

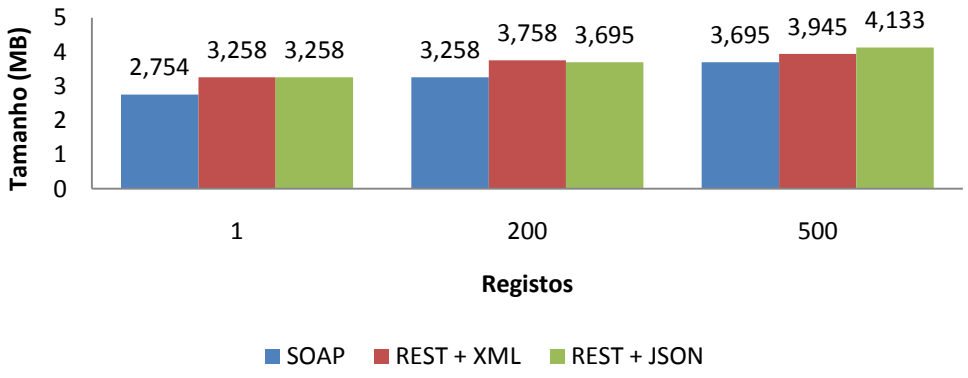

Figura 12 – Comparação de resultados - Uso de memória

Ao nível do tempo de execução, como comprovado na Figura 13, os *Big Web Services* são os que melhores resultados apresentam para o caso em que apenas existe um registo a analisar, tendo a combinação *REST* e *JSON* a pior pontuação neste caso, ficando a combinação *REST* com *XML* muito próximo deste. Perante o aumento de número de registos porém as situações invertem-se. O incremento da quantidade de registos é acompanhado por grande aumento do tempo de execução dos clientes que utilizam serviços Web baseados em SOAP e baseados em REST com XML. Aqui nota-se claramente que a conjugação de REST com JSON é a que apresenta melhores resultados em termos de tempos de execução.

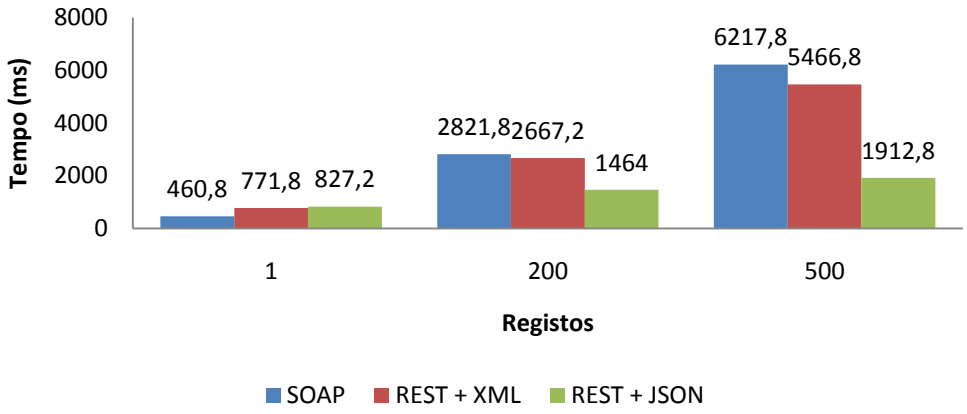

Figura 13 – Comparação de resultados - Tempo de execução

Por fim, a análise dos resultados dos tamanhos de tráfego gerada pelas operações (Figura 14) mostra que o protocolo SOAP que utiliza XML é muito pesado em relação à utilização de XML simples ou de JSON. Neste campo a utilização de REST mais a codificação dos dados em JSON é a que melhores resultados apresenta. Com a utilização de 500 registos e em comparação aos serviços Web baseados em SOAP, a quantidade de tráfego deste chega a ser um pouco mais do dobro em relação a REST com JSON.

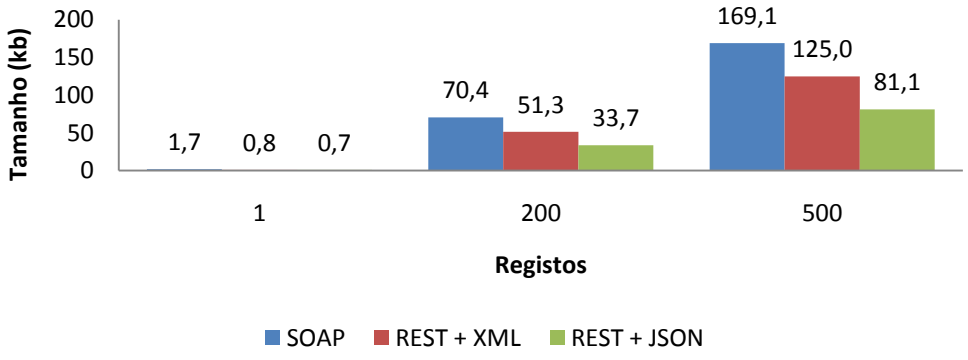

Figura 14 – Comparação de resultados - Quantidade de tráfego

Realizados todos os testes e todas as análises, é possível afirmar que a combinação de RESTful Web Services com XML apresenta sempre resultados medianos, ficando este quase sempre entre os resultados obtidos nos outros dois testes. A utilização de Big Web Services tem a vantagem de ser a que menos memória necessita para a realização do processamento dos dados, porém, é a que pior se comporta em termos de tempo de processamento e tamanho de tráfego gerado quando o número de registos é significativo. Apesar da utilização de JSON em RESTful Web Services ser a que piores resultados apresenta ao nível de utilização de memória, é a que apresenta significativamente melhores resultados nos restantes testes.

De acordo com Pautasso *et al*. (2008) os serviços baseados na arquitectura REST apenas devem ser utilizados na integração de aplicações com a Web, e os serviços baseados em SOAP devem ser utilizados na integração de aplicações empresariais. Esta conclusão foi retirada de uma análise extensa aos dois estilos de comunicação. Porém, os resultados obtidos dos testes práticos aqui apresentados dão uma indicação contrária. No caso concreto de integração de dispositivos móveis com sistemas empresariais, a escolha deverá recair sobre a utilização de RESTful Web Services em conjunção com o formato JSON, pois é a que dá mais garantias de melhor experiência para os utilizadores. É então esta conjunção que será utilizada no desenvolvimento da plataforma Novopca Mobile.

5 Arquitectura a implementar

Neste capítulo é descrita a arquitectura que serve de base à implementação da plataforma Novopca Mobile. São também descritos alguns dos aspectos importantes que constituem esta arquitectura, como é o caso da criação de um sistema modular, os processos de autenticação única e de economia de tráfego.

A arquitectura geral, representada na Figura 15, é baseada no modelo da arquitectura cliente-servidor *Three-Tier*. O modelo *three-tier* é uma arquitectura e um padrão de *design* de *software* composto por três componentes que normalmente se encontram em diferentes plataformas físicas [Eckerson, 1995]. Esta separação permite que os componentes tenham evoluções diferentes e independentes, podendo estes serem actualizados ou substituídos de forma transparente. É porém necessário garantir que as interfaces de comunicação entre os componentes não sejam alteradas, pois a alteração de uma interface obriga à actualização de todos os componentes que a utilizam.

Os três componentes que integram a arquitectura a implementar são: base de dados, *middleware* e dispositivo móvel. A base de dados é o local no servidor onde toda a informação está armazenada. A aplicação de *middleware* será responsável pelo estabelecimento das ligações entre a base de dados e os dispositivos móveis. Por fim, os dispositivos móveis serão os clientes que realizam pedidos de dados ao servidor. Em alguns casos as aplicações clientes poderão ter que fazer uso de uma base de dados interna para o armazenamento temporário de dados.

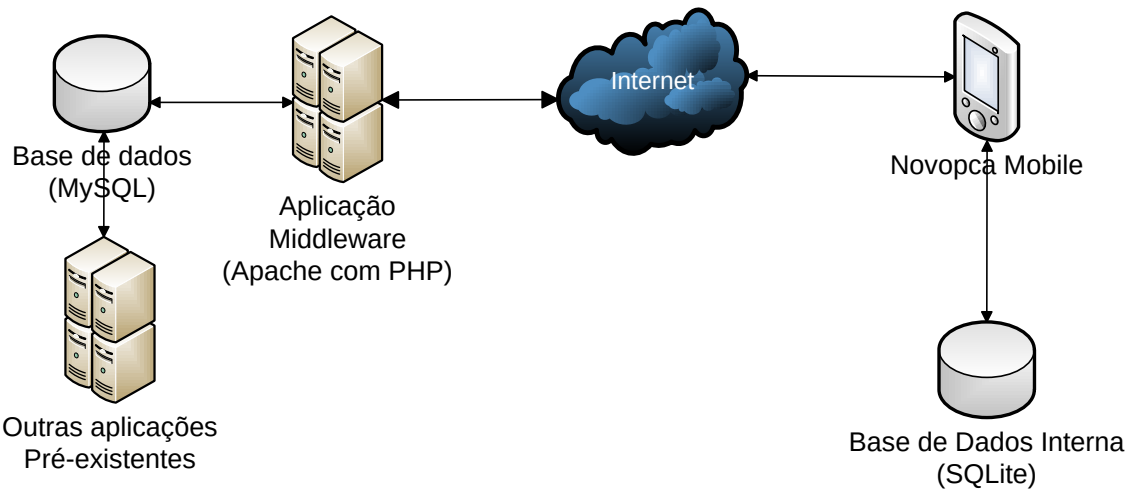

Figura 15 – Arquitectura do Novopca Mobile

A arquitectura é portanto constituída por um cliente e um servidor, como representado em mais detalhe na Figura 16. O servidor será constituído por três elementos: um gestor de comunicações, responsável pela comunicação com o cliente, seguindo a interface definida

nos RESTful Web Services; uma camada de acesso a dados, que será composta por classes que fazem a abstracção do acesso à base de dados; e por fim a própria base de dados onde está toda a informação que será acedida pelo cliente. As aplicações cliente serão mais complexas, sendo compostas por cinco elementos distintos: interface de utilizador, interface de acesso a dados, gestor de comunicações, camada de acesso a dados e base de dados. A interface de utilizador é o conjunto de ecrãs visíveis pelos utilizadores e é através dela que os utilizadores interagem com as aplicações. A interface de acesso a dados tem como principal objectivo abstrair a interface de utilizador sobre a obtenção dos dados que esta necessita, ou seja, a aplicação será capaz de obter informação do servidor, através do gestor de comunicações, ou da base de dados, através da camada de acesso a dados, dependendo do processo em vigor. A interface de acesso a dados será ainda responsável pela implementação de métodos de economia de tráfego que serão abordados no capítulo 5.3, e para este efeito fará uso de uma base de dados local.

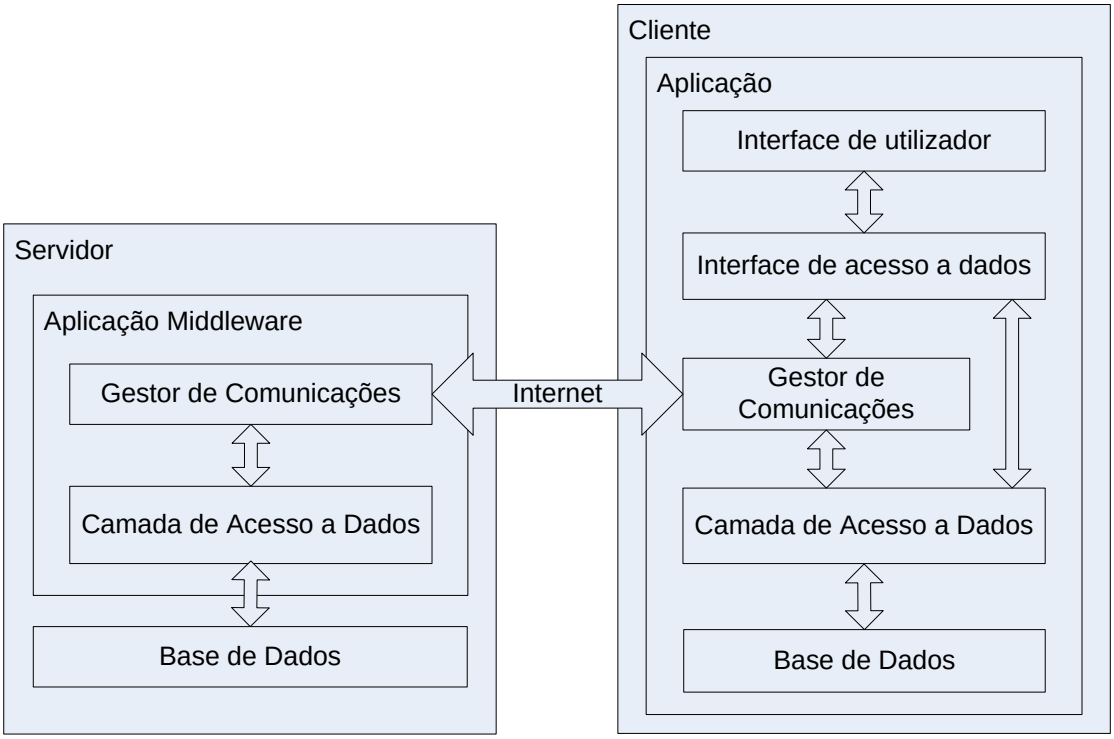

Figura 16 – Arquitectura detalhada da plataforma Novopca Mobile

O gestor de comunicações do cliente, para além de ser responsável pela comunicação com o servidor, é também responsável pela gestão do processo de autenticação. Um dos objectivos deste projecto é a criação de uma plataforma modular que permita que os diferentes módulos possam ser actualizados, adicionados ou eliminados de forma independente uns dos outros. A arquitectura que visa atingir este objectivo será explicada no capítulo 5.1. Apesar dos módulos serem independentes uns dos outros, estes terão um ponto em comum: a autenticação. O processo de autenticação só é possível caso o

utilizador introduza as suas credenciais de acesso. A obtenção das credenciais apenas é realizada pela aplicação NovopcApp, sendo assim possível que o utilizador apenas se autentique uma única vez no sistema e que todos os módulos que estejam instalados ou que sejam instalados no futuro utilizem a NovopcApp para obtenção das credenciais de autenticação. A arquitectura proposta para este processo será apresentada no capítulo 5.2.

Os actuais dispositivos móveis vêm equipados com equipamentos para utilização de redes WiFi ou de 3G. De acordo com os objectivos deste projecto, a rede WiFi será preferencialmente utilizada em detrimento das ligações 3G. No entanto estas ligações à internet têm sempre um custo: a perda de autonomia do dispositivo. Para maximizar esta autonomia as ligações 3G ou WiFi apenas deverão de ser utilizadas quando restritamente necessário; a redução da utilização destas ligações é possível através da utilização de sistemas que serão abordados com mais detalhe no capítulo 5.3.

5.1 Sistema Modular

A arquitectura a implementar, de um ponto de vista geral, será, como já referido, baseada na arquitectura *three-tier*; onde existem os componentes de dados, de interligação e de apresentação dos dados devidamente separados. Porém, a componente de apresentação de dados deverá de ter uma arquitectura modular que permita a adição, actualização e remoção de módulos de uma forma fácil e eficiente. Para que tal seja possível foi pensado o desenvolvimento de uma aplicação principal, já referida neste documento como NovopcApp, que será responsável por toda a gestão de módulos. Olhando para a plataforma Android estes módulos serão também aplicações, com a diferença de que não poderão ser executados sem a existência da NovopcApp no sistema, por estarem dependentes desta para a realização do processo de autenticação e são por isso denominados como módulos.

Esta aplicação deverá de ser capaz de:

- Obter a lista de módulos disponíveis para transferência;
- Transferir módulos existentes no servidor;
- Invocar o instalador do Android para os módulos obtidos;
- Fazer o controlo de quais os módulos instalados no sistema;
- Permitir a execução, desinstalação e actualização de módulos;
- Informar o utilizador sempre que existam actualizações de módulos disponíveis no servidor;
- Fazer o controlo de auto-actualização, avisando o utilizador sempre que exista uma actualização da própria aplicação;

A implementação de um sistema modular é uma grande inovação ao nível aplicacional, sendo que ainda não existem ofertas ao nível empresarial que permitam que o utilizador decida quais as ferramentas que necessita ou que pretende utilizar na realização do seu trabalho. Os criadores de módulos terão também a possibilidade de realizarem alterações a estes componentes individualmente sem afectar a plataforma como um todo.

5.2 Autenticação única

O sistema modular onde irá assentar a aplicação cliente da plataforma Novopca Mobile traz, como já referido, várias vantagens tanto para os utilizadores como para os programadores. A utilização deste tipo de sistemas levanta contudo um problema: obriga a que o utilizador insira credenciais de acesso em cada um dos módulos. A resolução do problema é realizada através da implementação de um mecanismo de comunicação entre os módulos e a aplicação gestora. Sendo assim, a aplicação deverá de ser responsável por receber as credenciais do utilizador, por as armazenar e por as tornar disponíveis aos módulos existentes. Por sua vez, os módulos terão que obter da aplicação principal as credenciais previamente introduzidas pelo utilizador sempre que necessitem de realizar um pedido ao servidor. Os módulos deverão porém de serem independentes da aplicação principal na implementação dos métodos de autenticação com o servidor.

O método de autenticação que a plataforma utilizará será o método *Digest access authentication* [RFC2617, 1999]. Como já referido neste documento, este método tem a vantagem de não enviar as credenciais em formato de texto como acontece no *Basic access authentication* [RFC2617, 1999], sendo realizados métodos criptográficos que permitem transformar as credenciais de acesso, mais concretamente o par de valores *username* e *password*, em cadeias de caracteres irreconhecíveis [Richardson and Ruby, 2007].

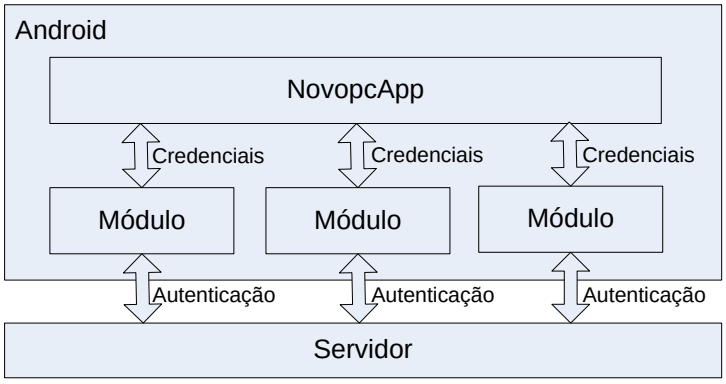

Figura 17 – Arquitectura Geral de Autenticação

O procedimento aqui proposto permitirá que o utilizador apenas se autentique na primeira vez que utilizar a aplicação, sendo que todos os módulos posteriormente instalados utilizarão as credenciais armazenadas pala aplicação NovopcApp.

5.3 Economia de tráfego

A comunicação entre cliente e servidor, como já referido, é uma característica indispensável na plataforma Novopca Mobile. A existência desta necessidade comunicacional levanta outros problemas que devem de atenuados; seja por questões de economia de bateria [Sharkey, 2009] ou por questões de custo associados à utilização de tráfego 3G. De forma a atenuar estas desvantagens, são neste capitulo apresentadas dois sistemas: um sistema para a recepção de dados e outro para o envio de dados, ambos com aplicabilidade na aplicação cliente.

5.3.1 Recepção de dados

A aplicação NovopcApp e módulos integrantes, terão como base a apresentação de dados obtidos do servidor. Uma grande parte dos dados existentes sofrem actualizações muito esporadicamente e por isso não faz sentido obter estes dados do servidor sempre que estes são requisitados pela interface de utilizador. Por outro lado, mesmo que por vezes os dados sofram alterações, a não replicação imediata destas alterações para a aplicação cliente é por vezes aceitável.

Com base nestes factos, para os casos em que é aplicável, a interface de acesso a dados, ao receber um pedido, deverá de seguir a seguinte ordem de métodos de obtenção de dados: obter os dados do servidor utilizando ligação WiFi, obter os dados armazenados previamante na base de dados local e por fim obter os dados do servidor utilizando outro tipo de ligação à Internet disponível. Pela ordem apresentada, cada vez que um dos métodos falhe deverá de ser utilizado o método seguinte; caso todos os métodos falhem, deverá de ser retornada informação de falha na obtenção de dados.

A Figura 18 representa o fluxograma do processo proposto para a atenuação dos problemas levantados pela necessidade constante de obtenção de dados.

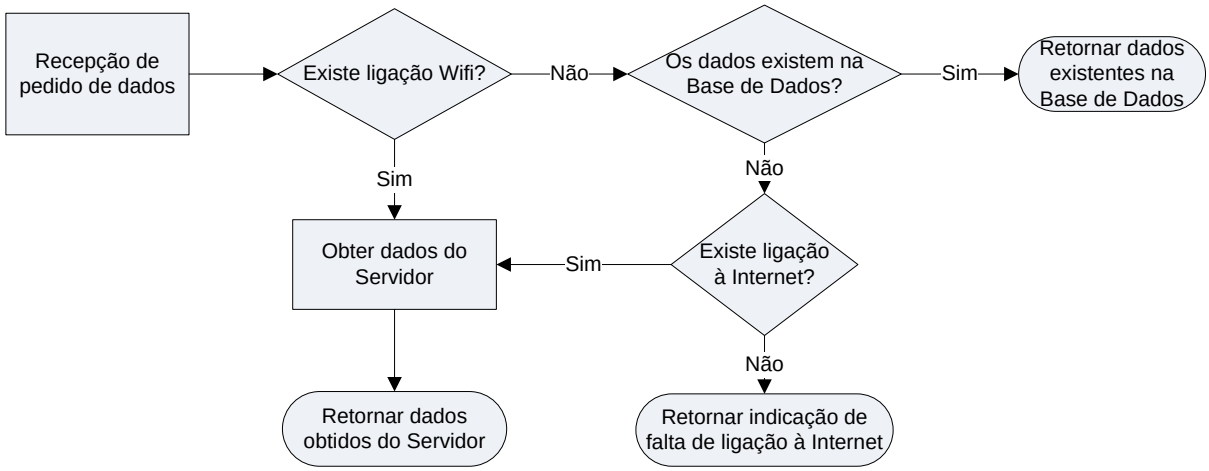

Figura 18 – Fluxograma do processo de economia de tráfego na recepção de dados

5.3.2 Envio de dados

O envio de dados para o servidor será condição necessária para os módulos que recebam dados introduzidos pelo utilizador ou dados fornecidos pelo Android, como por exemplo dados sobre a localização geográfica do dispositivo. Analogamente ao que acontece na recepção de dados, existem casos em que os dados não necessitam ser enviados para o servidor imediatamente depois de serem recolhidos pelos respectivos módulos.

Com base nestes factos, para os casos em que é aplicável, a interface de acesso a dados ao receber um pedido de envio de dados para o servidor deverá de verificar se existe uma conexão à Internet via WiFi e, caso exista deverá então de proceder ao respectivo envio dos dados. Caso esta conexão não exista, os dados deverão ser armazenados temporariamente na base de dados local. Na Figura 19 está representado o fluxograma deste processo.

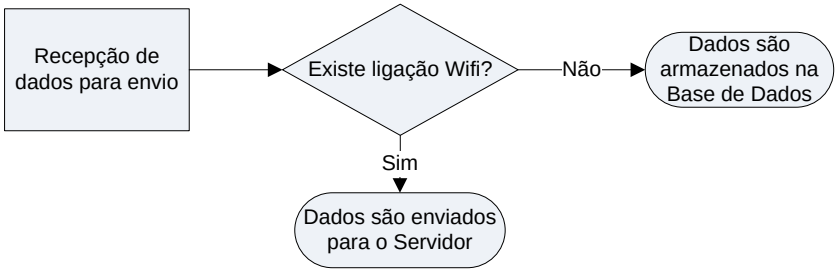

Figura 19 – Fluxograma do processo de economia de tráfego no envio de dados

Todos os dados armazenados localmente deverão seguir o seu rumo para o servidor assim que fique disponível no dispositivo uma ligação WiFi. Nesta situação será utilizado outro componente, assim denominado *Network Receiver*, que será responsável pela obtenção das notificações (*broadcasts*) de alteração na ligação à Internet enviadas pelo Android. Ao ser notificado da existência de uma ligação WiFi, o *Network Receiver* invocará o processo (Figura 20) que permite a obtenção dos dados armazenados temporariamente, o envio destes para o servidor, e por fim a eliminação dos dados temporários da base de dados.

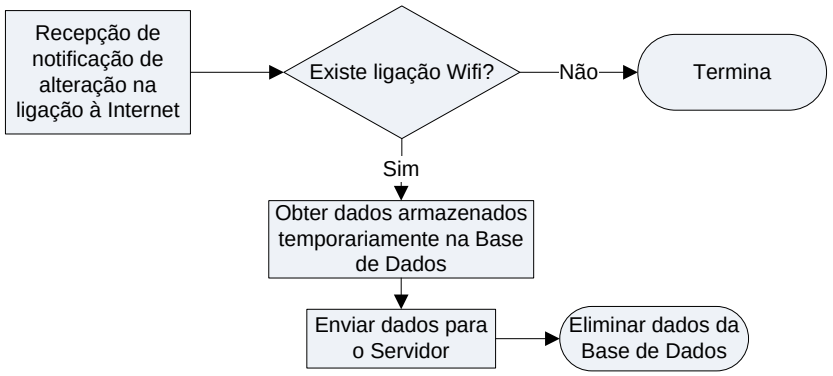

Figura 20 – Fluxograma do processo de economia de tráfego no envio de dados armazenados temporariamente

6 Implementação

Neste capítulo é apresentado todo o trabalho de implementação relacionado com o projecto documentado na presente dissertação. O capítulo, sendo o mais importante de todo o documento, descreve todos os componentes desenvolvidos que englobam a plataforma Novopca Mobile.

A plataforma Novopca Mobile é constituída, como já referido, por uma arquitectura do tipo cliente-servidor. O servidor é composto por uma base de dados e por uma aplicação de middleware. Por sua vez a aplicação cliente é constituído por um sistema modular.

O desenvolvimento da aplicação de *middleware* realizou-se num servidor Windows equipado com o servidor Web Apache versão 2.2.17 e com PHP versão 5.3.5. O sistema gestor de base de dados utilizado no armazenamento de informação é o MySQL versão 5.5.8.

A aplicação cliente gestora de módulos (NovopcApp) e respectivos módulos foram desenvolvidos em Eclipse versão 3.5.2 com o *plugin* de desenvolvimento ADT (Android Development Tools). Apesar do sistema operativo Android se encontrar já na versão 2.3 (*Gingerbread*) as aplicações cliente desenvolvidas tem como alvo a versão 1.6 (D*onut*). Esta decisão está ligada a questões de compatibilidade, pois todas as aplicações desenvolvidas para a versão 1.6 poderão ser executadas nas versões superiores a esta sem problemas, já o contrário não é possível. Apesar desta vantagem existe a desvantagem clara da impossibilidade de utilização dos novos componentes disponibilizados pela última versão do sistema.

6.1 Middleware

A aplicação *middleware* é, como já referido, constituída por um gestor de comunicações e uma camada de acesso a dados e utiliza uma base de dados para armazenamento de informação (Figura 21).

O gestor de comunicações é responsável por responder a todos os pedidos efectuados pelas aplicações cliente que se autentiquem através do método *Digest access authentication*. Neste projecto optou-se pelo uso de RESTful Web Services em conjunção com JSON e como tal, o gestor de comunicações tem que implementar este tipo de serviços Web e colocar a informação a enviar em formato JSON. A camada de acesso a dados faz a abstracção da comunicação entre o gestor de comunicações e a base de dados, transformando registos da base de dados em estruturas de dados mais eficientes. Este tipo de abstracção traz várias vantagens, sendo a principal a possibilidade de mudança do gestor de base de dados sem necessidade de alteração do gestor de comunicações. A base

de dados é composta por um conjunto de tabelas que contém toda a informação necessária à execução das aplicações cliente.

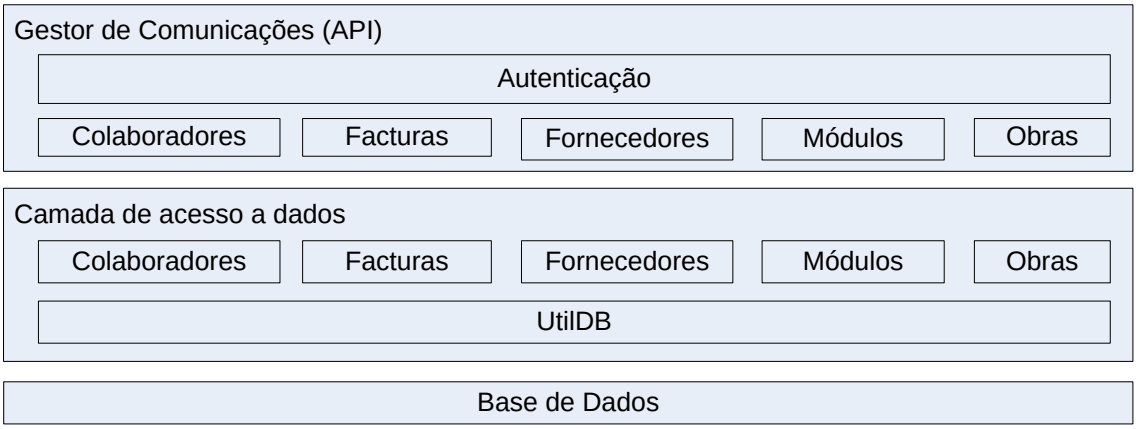

Figura 21 – Constituição da aplicação middleware

A base de dados utilizada por esta aplicação é uma agregação de tabelas pré-existentes na base de dados da empresa com algumas novas tabelas necessárias à execução da plataforma. No desenvolvimento deste projecto apenas foram consideradas as tabelas pré-existentes necessárias ao desenvolvimento da aplicação NovopcApp e dos respectivos módulos. As principais tabelas reutilizadas para este projecto são: Colaboradores, Obras, Fornecedores e Documentos. As novas tabelas são as de afectação automática às obras, e a tabela de módulos. O esquema de toda a base de dados está representada na figura Figura 22, todas as tabelas são facilmente identificadas e auto-explicativas.

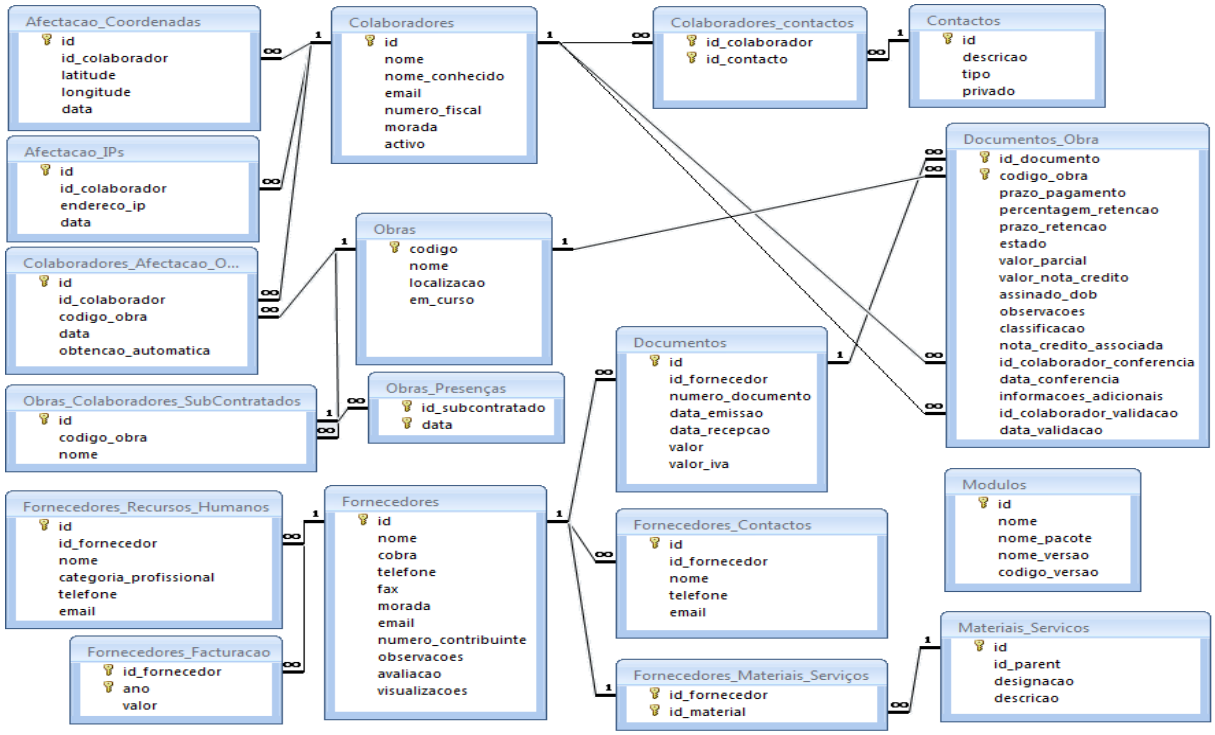

Figura 22 – Base de dados do Servidor

6.2 NovopcApp

A NovopcApp é a aplicação central da plataforma Novopca Mobile. É a partir desta que o utilizador poderá fazer toda a gestão de módulos existentes. As principais funcionalidades que esta aplicação deve de implementar para permitir a criação de um sistema modular foram já mencionadas no capítulo 5.1.

Tendo em conta que a NovopcApp será uma aplicação que estará sempre disponível nos terminais móveis dos colaboradores, esta deverá também conter um componente adicional que é utilizado por todos: listagem de colaboradores da empresa com respectivos contactos. Este componente deverá de ser capaz de:

- Obter a lista de colaboradores e respectivos dados adicionais do servidor;
- Permitir a execução de várias tarefas com base na informação dos colaboradores, como por exemplo: permitir realizar chamada ou enviar SMS com base no número de telefone, permitir o envio de email com base no endereço de correio electrónico, permitir a visualização da morada do colaborador num mapa e por fim, permitir a adição à lista de contactos do telefone os contactos que se pretenda.

As funcionalidades disponíveis ao utilizador estão representadas na Figura 23, através de um diagrama de casos de uso.

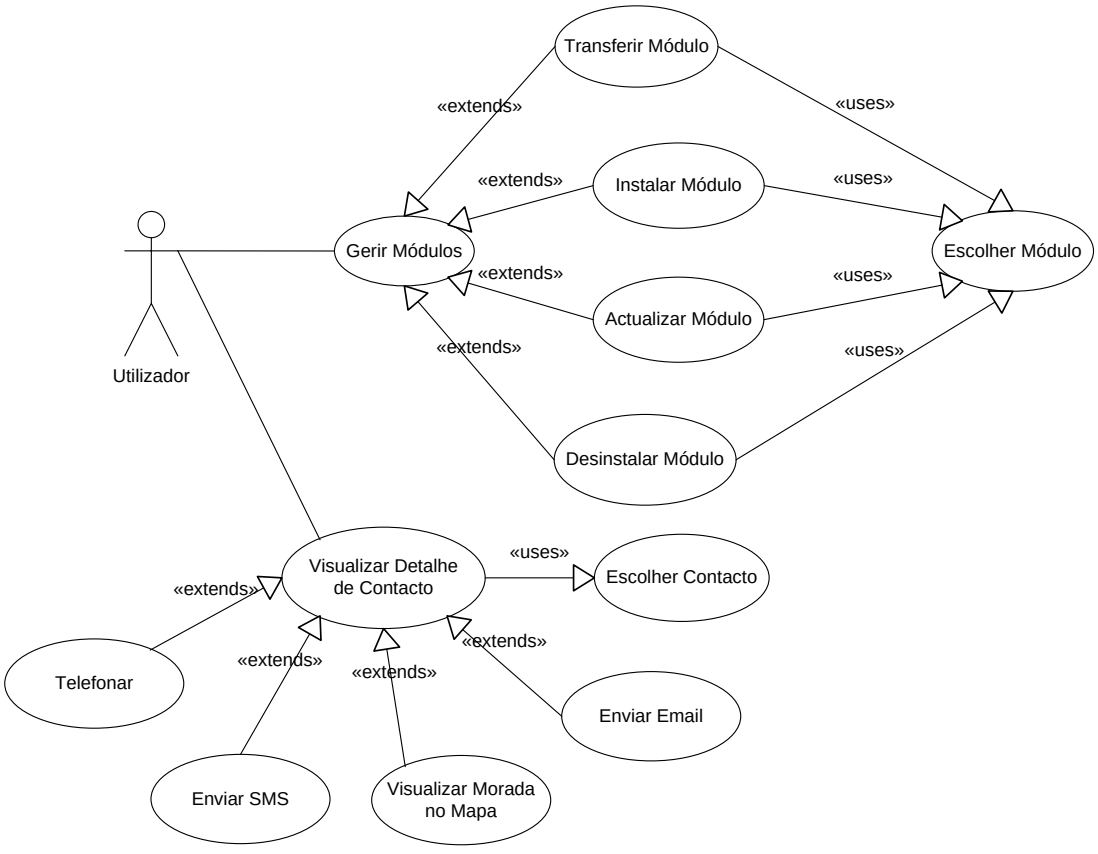

Figura 23 – Caso de uso da aplicação NovopcApp

A aplicação permite de uma forma fácil e eficaz que o utilizador faça a gestão dos módulos existentes, podendo instalar e desinstalar os que pretende. De igual forma pode visualizar os contactos de todos os colaboradores da empresa e a partir destes realizar um conjunto de operações, já referidas anteriormente, que utilizam na perfeição as características do Android.

A instalação de módulos é realizada da mesma forma que uma aplicação normal no Android; a NovopcApp apenas é responsável pela obtenção dos módulos do servidor e pela invocação do instalador do Android. Este fluxo obriga a que a NovopcApp tenha controlo sobre todas as aplicações que são instaladas no sistema; realizando-se através da recepção de notificações (mais concretamente *broadcasts*) enviadas pelo sistema Android, com indicações da ocorrência de instalação ou desinstalação de aplicações.

Ao nível de comunicações a aplicação NovopcApp implementa o procedimento apresentado no capítulo 5.3.1. Na sua primeira execução são realizadas três operações, cada uma executada no início da respectiva *activity*: a obtenção de módulos previamente instalados no sistema, obtenção de novos módulos disponíveis na plataforma Novopca Mobile e a obtenção de todos os contactos dos colaboradores da empresa, sendo que estes dois últimos apenas são possíveis caso exista uma ligação à internet. Todos os dados obtidos são armazenados na base de dados da aplicação. As execuções futuras, por outro lado, têm fluxos de funcionamento distintas da primeira; caso exista ligação à internet por WiFi os dados de novos módulos e dos contactos dos colaboradores são obtidos novamente do servidor e é actualizada a base de dados da aplicação; caso a ligação à internet seja realizada por 3G ou similar, a aplicação apresenta a informação previamente guardada na base de dados.

6.2.1 Composição

A aplicação NovopcApp é composta por vários blocos principais de construção (*Main Building Blocks*) fornecidos pelo Android [Gargenta, 2011]. Os blocos utilizados na sua implementação foram: *activities*, *content provider, services e broadcast receivers* (Figura 24).

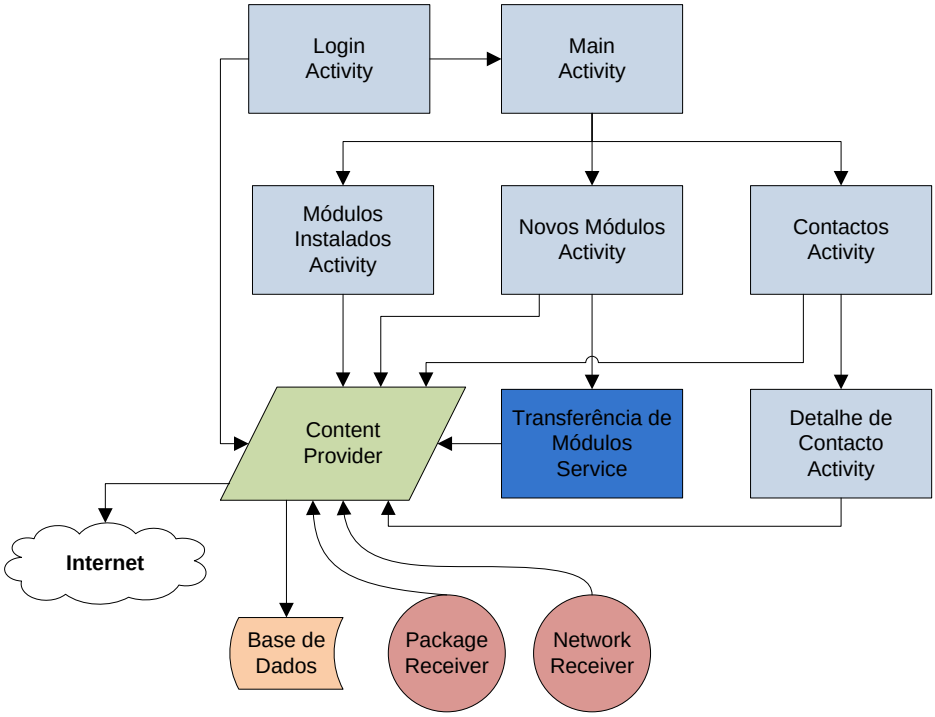

Figura 24 – Composição da aplicação NovopcApp

As *activities* são responsáveis pela composição da interface com o utilizador, permitindo que este visualize informação e interaja com a aplicação.

A transferência de novos módulos do servidor é realizada através de um serviço (*ModuleDownloadService*). A utilização de um serviço para esta finalidade tem como vantagem permitir que os módulos sejam transferidos em segundo plano, não sendo esta transferência interrompida mesmo que o ciclo de vida da *activity* que o iniciou termine.

O controlo de todas as aplicações instaladas ou desinstaladas do sistema é realizada através do *Package Receiver*, que implementa as funcionalidades de um *broadcast receiver*. A plataforma Android envia um *broadcast* sempre que ocorrem alterações ao nível de aplicações (mais concretamente *packages*), seja instalação, desinstalação, actualização, entre muitas outras. A NovopcApp apenas necessita de receber notificações de instalação, actualização ou desinstalação de aplicações para que seja capaz de actualizar a sua base de dados quanto aos módulos que estão instalados no sistema. Contudo, o *Package Receiver* irá receber notificações de alterações ocorridas a qualquer aplicação, sendo apenas relevantes as alterações realizadas a módulos. Os módulos são identificados a partir do seu *package name*, pois estes têm obrigatoriamente de ser no formato "*nome_modulo.novopca.android.pt* ".

O componente *Network Receiver* que implementa também as funcionalidades de um *broadcast receiver*, é responsável por obter dados relativos a actualizações de módulos disponíveis no servidor. O *Network Receiver* recebe uma notificação do Android com

indicação de que a ligação à internet foi alterada; seguidamente verifica se a ligação existente é do tipo WiFi, caso seja então obtém do servidor os dados relativos a actualizações de módulos, informando o utilizador através do *Notification Manager* caso existam.

O *Content Provider* abstrai a interface de utilizador para a obtenção de informação. O *Content Provider* é responsável por decidir se pedirá informação ao servidor ou se utilizará informação previamente armazenada em base de dados. Esta componente tem uma vantagem indispensável à estrutura modular da plataforma Novopca Mobile. Todos os módulos terão de obter credenciais de autenticação previamente introduzidas na NovopcApp. Esta obtenção de dados é realizada através do *Content Provider*, que para além de fornecer uma interface de acesso a dados dentro da própria aplicação, cria também uma interface de acesso a dados para as outras aplicações.

6.2.2 Interface de utilizador

A interface de utilizador da NovopcApp é constituída por um total de seis *activities*: *login*, main, módulos instalados, novos módulos, contactos e detalhe de contacto (Figura 24).

A *login activity*, representada na Figura 25, irá surgir na primeira vez que a aplicação for executada. O utilizador terá que introduzir credenciais de acesso válidas de forma a iniciar sessão e ter acesso às restantes *activities*. As credenciais introduzidas pelo utilizador serão guardadas na base de dados através do *Content Provider*, o que permitirá que o utilizador não necessite de fazer login novamente na próxima vez que iniciar a aplicação.

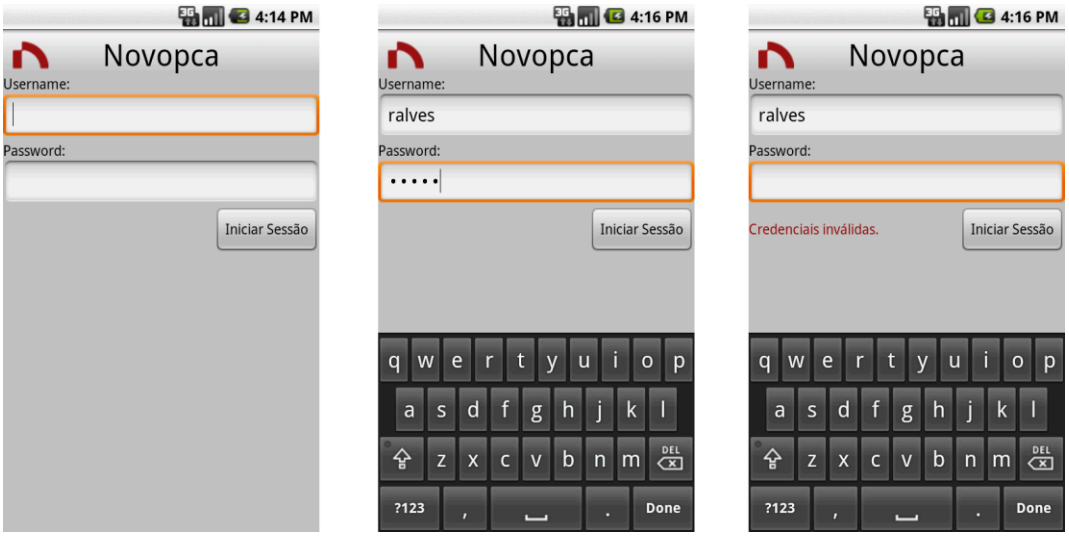

Figura 25 – NovopcApp - Login Activity

Realizado com sucesso o início de sessão, surge no ecrã a *main activity* em junção com a *activity* de listagem de módulos instalados (Figura 26 a)). A existência de duas *activities* no ecrã é neste caso possível através da utilização do componente TabHost. O TabHost é um

contentor que permite a criação de uma interface baseada em abas [Google Inc, 2011a]; este contentor armazena dois componentes: um *TabWidget* e uma *FrameLayout*. O *TabWidget* permite a criação de abas que nesta aplicação são: Módulos, Novos e Contactos. O *FrameLayout* será o local onde serão carregadas as activities com base na aba seleccionada. As *activities* módulos instalados, novos módulos e contactos serão carregadas neste componente.

Figura 26 – NovopcApp activities a) módulos instalados b) novos módulos c)contactos

A *activity* de listagem de módulos instalados, representada na Figura 26 a), apresenta, caso existam, os módulos instalados no sistema. O utilizador poderá aqui executar, desinstalar ou actualizar (caso esteja disponível uma actualização) os módulos que pretenda.

A *activity* de módulos disponíveis para transferência apresenta ao utilizador uma listagem de módulos que este poderá transferir e instalar. Ao seleccionar um módulo é iniciada a transferência deste através do serviço de transferências de módulos. O serviço de transferências envia, ao longo da transferência do módulo, *broadcasts* com indicação do progresso da transferência e actualiza, de igual forma, a *status bar* do android com notificações sobre o progresso de transferência (Figura 27 c)). A *activity* de novos módulos implementa as funcionalidades de um *broadcast receiver* e recebe os *broadcasts* enviados pelo serviço de transferências; com a informação recebida são actualizadas as *progress bars* que representam o progresso da transferência (Figura 27 a)). Quando a transferência é terminada fica disponível a acção de instalar os módulos transferidos (Figura 27 b) e c)). O ficheiro transferido terá o nome no formato *NomeModulo.apk* e será armazenado na raiz do *SD Card* do dispositivo.

Figura 27 – NovopcApp - a) Transferência de módulos b) módulos prontos a instalar c) notificações

O utilizador ao seleccionar um módulo que esteja pronto a instalar despoleta a invocação do instalador do Android para o módulo seleccionado. O instalador é invocado da seguinte forma:

```
String fileName = Environment.getExternalStorageDirectory() + "/" + moduleFileName;
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setDataAndType(Uri.fromFile(new File(fileName)),
        "application/vnd.android.package-archive");
context.startActivity(intent);
```

Ao ser instalado um módulo o *Package Receiver* irá receber uma notificação enviada pelo Android. O *Package Receiver* irá então verificar se é realmente um módulo através do *package name* e, caso seja, irá obter os dados associados ao módulo instalado e armazena-los na base de dados com indicação de que este está instalado no sistema. A obtenção dos dados do módulo instalado é realizada da seguinte forma:

```
PackageInfo p = context.getPackageManager().getPackageInfo(package_name, 0);
Module module = new Module();
module.setAppName(p.applicationInfo.loadLabel(context.getPackageManager()).toString());
module.setPackageName(p.packageName);
module.setVersionName(p.versionName);
module.setVersionCode(p.versionCode);
module.setIcon(p.applicationInfo.loadIcon(context.getPackageManager()));
module.setStatus(Status.INSTALLED);
```

Concluído o processo de obtenção dos dados do módulo, o *Package Receiver* envia uma notificação via *broadcast* de forma a que as *activities* de módulos instalados (Figura 28 a)) e

módulos disponíveis para transferência (Figura 28 b)) actualizem a sua interface com os novos dados.

Ao ser detectada uma actualização de um módulo disponível no servidor é apresentada uma notificação ao utilizador com essa informação (Figura 28 c)).

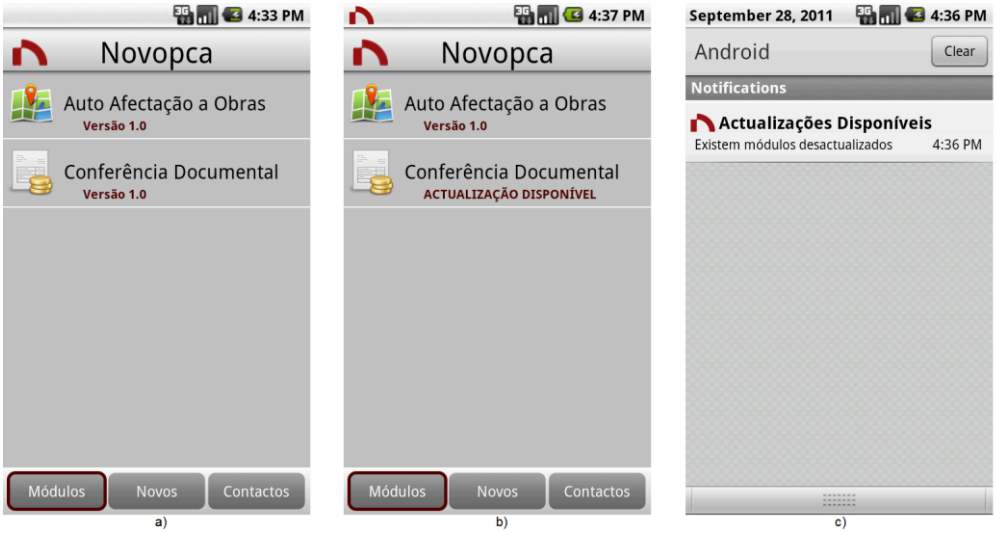

Figura 28 – NovopcApp - a) módulos instalados b) actualização disponível c) notificação actualização disponível

O ecrã com detalhe de um contacto, representado na Figura 29, fornece a informação existente desse contacto: fotografia, nome, telefone, endereço de correio electrónico e morada. Através desta informação a aplicação permite a execução de várias operações, nomeadamente: telefonar ao contacto, enviar mensagem escrita, enviar e-mail, visualizar a morada do contacto no mapa ou adicionar o contacto à lista de contactos do dispositivo.

Figura 29 – NovoppcApp - Detalhe de Contacto

A obtenção de dados do *Content Provider* é realizada de forma assíncrona, impedindo assim que a *thread* com a interface de utilizador bloqueie enquanto aguarda pelos dados.

Esta operação é realizada através da utilização da classe AsyncTask disponibilizada pelo Android, que permite a utilização correcta e fácil da thread de interface de utilizador. A AsyncTask permite que sejam realizadas operações em segundo plano e que os resultados obtidos sejam publicados na thread de interface de utilizador sem a necessidade de manipulação directa de *threads* e *handlers* [Google Inc, 2011a].

6.3 Módulos

Neste capítulo serão apresentados os módulos implementados na realização deste projecto. A Novopca é uma grande empresa de construção, onde existem vários processos que podem resultar na criação de módulos que resulte no aumento de eficiência dos mesmos. No âmbito deste projecto foi planeada a implementação de quatro módulos distintos, designadamente: auto-afectação, conferência documental, fornecedores e presenças em obra.

Os módulos implementados foram escolhidos por visarem o aumento imediato da eficiência na realização de processos importantes realizados diariamente pelos colaboradores da organização.

Os módulos, como já referido, são dependentes da aplicação NovopcApp para a realização do mecanismo de autenticação e como tal não é possível que estes sejam executados sem a existência do NovopcApp. De forma a prevenir situações em que o utilizador execute um módulo sem ter a aplicação principal instalada, cada módulo verifica no início da sua execução se a NovopcApp se encontra instalada e, caso não se verifique, interrompe a sua execução normal e apresenta ao utilizador uma interface que permite que este a transfira e instale no seu dispositivo (Figura 30).

Figura 30 – Módulos - Transferência e instalação da NovopcApp

6.3.1 Auto-Afectação

Um dos problemas com que as empresas de construção e obras públicas se debatem constantemente é o de não conseguirem ter total controlo sobre a localização dos colaboradores que se encontram em obras. Por exemplo, um director coordenador de obra pode visitar num único dia um vasto leque de obras para realização de controlo de projecto. O director coordenador será obrigado periodicamente a indicar em quais das obras esteve e em que dias, para que seja possível determinar a afectação deste às obras para atribuição de custos e realização de facturação.

O módulo de auto-afectação tem como principal objectivo eliminar a necessidade dos colaboradores indicarem as obras em que estiveram, assim como agilizar o processo de registo de afectação. Serão aqui utilizados duas características existentes nos dispositivos Android: o GPS e o WiFi. O GPS permitirá obter a localização dos utilizadores, enquanto que a existência de uma ligação WiFi permite, através do endereço IP, e com alguma certeza, obter de igual forma o local onde se encontra o dispositivo. A obtenção destes dados será realizada por um serviço existente no módulo e deverá de ser transparente para o utilizador. As funcionalidades disponíveis ao utilizador estão representadas na Figura 31, através de um diagrama de casos de uso.

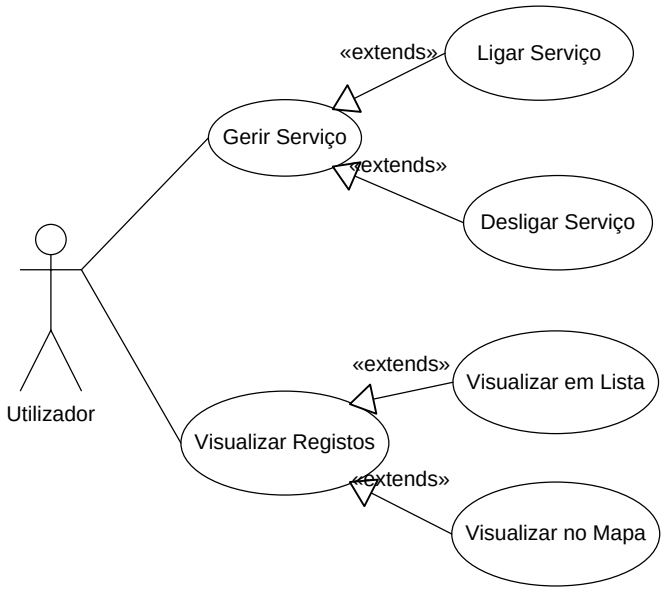

Figura 31 - Caso de uso do módulo de Auto-Afectação

O utilizador poderá ligar ou desligar o serviço de obtenção de informação relativa à sua localização, assim como poderá visualizar os registos obtidos. A visualização destes registos pode ser realizada de duas formas: através de um mapa, onde são apresentadas as

localizações obtidas via GPS, e por uma listagem de dados, onde são apresentados todos os dados obtidos por GPS e todos os endereços IP obtidos através de ligações WiFi.

6.3.1.1 Composição

O módulo de auto-afectação é composto por vários componentes fornecidos pela plataforma Android, nomeadamente: *activities, content provider, broadcast receivers, alarm manager* e base de dados local. A composição e interacção entre os componentes estão representadas na Figura 32.

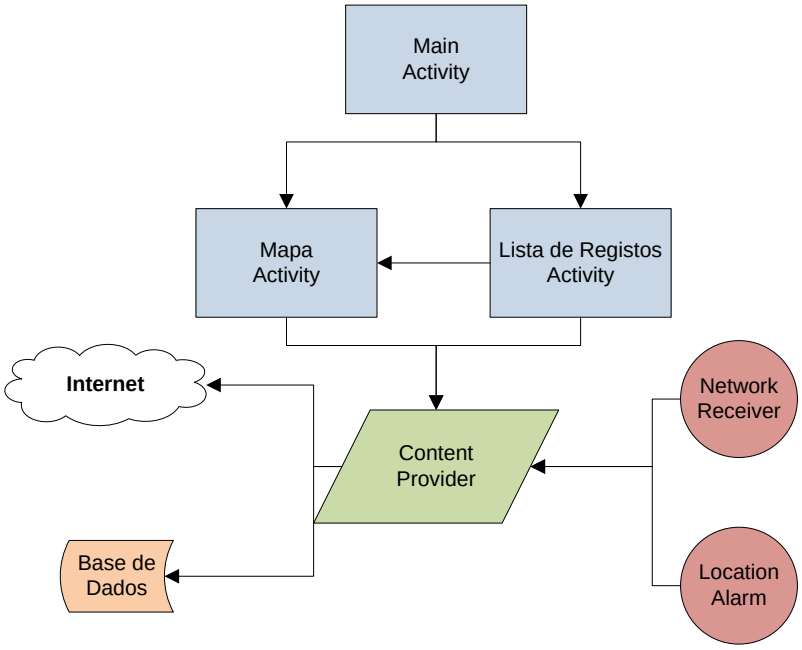

Figura 32 – Composição do módulo de auto-afectação

A interface de utilizador é constituída por três *activities*: a principal (denominada main), a de visualização de registos no mapa e a de visualização de registos em lista. A *activity* principal surge no início do módulo e através dela é dado ao utilizador a possibilidade de gerir (ligar ou desligar) o serviço de obtenção de localização para além da possibilidade de navegar para as restantes *activities*. A visualização de registos no mapa permite ao utilizador ter uma noção gráfica dos locais registados pelo serviço de localização. Por outro lado, a visualização dos registos em lista permite ao utilizador ter mais informação dos registos de localização, designadamente: data e hora de obtenção, coordenadas geográficas no sistema de graus decimais e, se possível, a morada correspondente a essa localização. São também apresentadas informações relativas aos endereços IP atribuídos por ligações WiFi, designadamente: hora e data de obtenção e endereço IP.

O componente *Network Receiver*, que implementa as funcionalidades de um *broadcast receiver*, é responsável por obter o endereço IP associado à interface de rede WiFi. Sempre

que fica disponível uma ligação WiFi o *Network Receiver* guardará na base de dados o endereço IP associado.

O *Location Alarm* implementa as funcionalidades da classe *AlarmManager* que permite a execução de código num determinado ponto no futuro. Os trabalhadores têm horários de trabalho normalmente compreendidos entre as 9 e as 19 horas e por isso a obtenção da sua localização deverá ser realizada durante este período temporal. O *Location Alarm* será executado de hora em hora durante este período de tempo e só durante os dias semanais de trabalho. A cada execução este componente tentará obter a localização do dispositivo através de GPS e caso consiga, as coordenadas geográficas são armazenadas na base de dados através do Content Provider.

O *Content Provider* integrado neste módulo implementa o sistema apresentado no capítulo 5.3.2 no envio dos dados para o servidor. Não existe neste módulo a necessidade de obtenção de qualquer tipo de informação do servidor. O armazenamento dos dados relativos a localizações e endereços IP obtidos automaticamente por este módulo é também realizado através deste componente.

6.3.1.2 Interface de utilizador

A interface de utilizador é constituída, como já referido, por um total de três *activities*: main, mapa e lista de registos (Figura 33).

Através da *activity* principal o utilizador poderá interagir com o serviço de obtenção de localizações, tendo a possibilidade de ligar (Figura 33 b)) ou desligar (Figura 33 a)) o seu funcionamento. Esta é uma característica importante, pois os utilizadores poderão olhar para esta funcionalidade de uma forma errada, considerando que podem alegar que a organização apenas os quer controlar de forma restrita. Esta característica fornece ao utilizador a hipótese de desligar o serviço quando este achar que as localizações que serão obtidas não serão relevantes para cálculos de afectação a obras.

Figura 33 – Auto-afectação - a) serviço desligado b) serviço ligado c) mapa com registos d) listagem de registos

O serviço de localização ao obter informações necessárias à obtenção das localizações dos utilizadores, seja através de *GPS* ou da rede *WiFi*, armazena-os localmente na base de dados.

A *activity* que contempla o mapa do mundo (Figura 33 c)) é na realidade uma *MapActivity.* Uma *MapActivity* realiza a gestão de uma *activity* que apresenta um *MapView,* que é uma *view* específica para exibição de mapas no ecrã. A *MapView* obtêm os dados relativos a mapas através do serviço fornecido pelo Google Maps. Esta *activity* permite que o utilizador visualize geograficamente os locais obtidos pelo serviço de localização, locais estes representados no mapa através de marcadores que após seleccionados indicam a data e hora em que esse registo foi obtido. Este tipo de visualização em mapa é muito interessante porque depois de obtidos vários registos de localização, o utilizador pode visualmente ter a noção dos locais que mais visitou, ou aqueles que nunca visitou. Os registos de endereços IP não são aqui representados, pois a conversão destes endereços para locais físicos apenas é realizada no servidor. A visualização destes registos pode ser realizada na *activity* de listagem de registos. Nesta *activity* são apresentados todos os registos existentes na base de dados assim como toda a informação associada a estes (Figura 33 d)). Um registo de localização ao ser seleccionado faz com que a *MapActivity* se inicie e apenas apresente o marcador deste registo no mapa. As informações sobre o registo seleccionado são enviadas para a MapActivity através de parâmetros extra adicionados ao *Intent* responsável pelo seu início de execução:

```
Intent i = new Intent(this, Map.class);
i.putExtra("latitude", lr.getLatitude());
i.putExtra("longitude", lr.getLongitude());
i.putExtra("date", lr.getDate());
startActivity(i);
```

A inserção de marcadores sobre o mapa é realizada através da utilização de uma lista de objectos do tipo *Overlay*. Esta classe permite a exibição de marcadores sobrepostos ao mapa numa *MapView*. A adição de marcadores é realizada da seguinte forma:

```
List<Overlay> mapOverlays = map.getOverlays();
Drawable drawable = this.getResources().getDrawable(R.drawable.marker);
MapItemizedOverlay itemizedoverlay = new MapItemizedOverlay(drawable,this);
OverlayItem overlayitem = new OverlayItem(new GeoPoint(latitude ,longitude), "", "");
itemizedoverlay.addOverlay(overlayitem);
mapOverlays.add(itemizedoverlay);
```

6.3.2 Conferência Documental

O processo de conferência de documentos é de extrema importância para a Novopca. Todas as notas de crédito ou de débito são introduzidas numa plataforma informática que permitem que as várias obras as confiram. A conferência das notas de débito é passo obrigatório para o futuro pagamento da mesma, pelo que por vezes é necessário uma quase imediata conferência destas para o desbloqueio de situações complicadas que possam surgir.

É importante referir que o processo de conferência documental já existente na empresa é constituído por dois passos. No primeiro passo o administrativo da obra terá que conferir o documento, verificando se os dados estão correctos e adicionando novos dados necessários. O segundo passo é realizado pelo director de obra que valida ou não a conferência realizada pelo administrativo. Concluído o processo de conferência, para o caso das notas de débito, é realizado o pagamento pelo sector de tesouraria.

Analisemos uma situação que possa acontecer na realidade. A empresa tem em sua posse várias notas de crédito, enviadas pela empresa Y há vários meses referentes à obrigatoriedade de pagamento de contas de luz do estaleiro da obra X. A conferência destes documentos nunca foi, por lapso, realizado e por isso o seu pagamento está pendente. A empresa Y, não tendo recepcionado qualquer tipo de pagamento, faz o corte de luz para o estaleiro da obra e parte do seu funcionamento é interrompido. Para desbloquear esta situação os documentos terão que ser rapidamente conferidos e pagos. O administrativo que está no estaleiro, encontra-se sem luz e por isso tem que se deslocar a outro local para que possa realizar o primeiro passo da conferência dos documentos. O director de obra que se situa no meio da obra, que tem uma extensão de dezenas de quilómetros, terá também ele que se deslocar a um local onde tenha acesso à plataforma de conferência de documentos e processar o passo final das respectivas conferências. Todo este processo

resultou num aumento de custos para a empresa, o deslocamento dos colaboradores tem custos ao nível de transporte e custos associados ao tempo consumido pelos dois recursos humanos durante as deslocações.

O módulo de conferência documental visa equipar os colaboradores com uma ferramenta que permite conferir documentos de uma forma fácil e intuitiva em qualquer parte do mundo. O presente módulo deverá de permitir que os administrativos confiram os documentos que se encontram no primeiro passo do processo, e que os directores de obra validem os documentos anteriormente conferidos pelos administrativos. Na Figura 34 está representado através de um diagrama de caso de uso as funcionalidades disponíveis aos utilizadores.

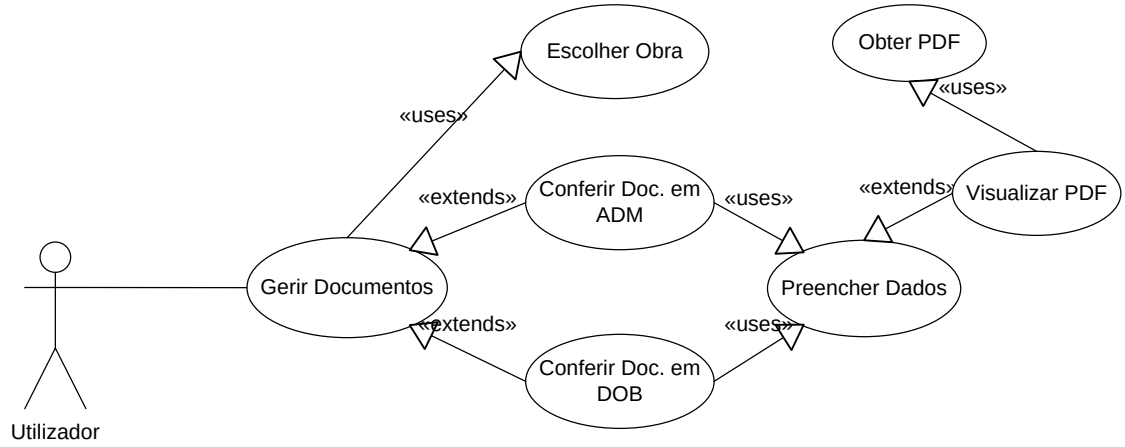

Figura 34 – Caso de uso do módulo de Conferência Documental

Todas as informações apresentadas e obtidas dos utilizadores deverão de estar sempre actualizadas com o servidor. O funcionamento do módulo permitirá que a conferência de documentos possa ser realizada paralelamente entre este e o sistema de conferência documental que já existe. Por exemplo, é possível que um administrativo utilize o dispositivo móvel para conferir um documento e que de seguida o director de obra utilize o sistema de conferência documental através de um computador para validar esta conferência; sendo o contrário também aplicável.

6.3.2.1 Composição

O módulo de conferência documental apresenta uma estrutura constituída apenas por um conjunto de *activities* e um *content provider*, como representado na Figura 35.

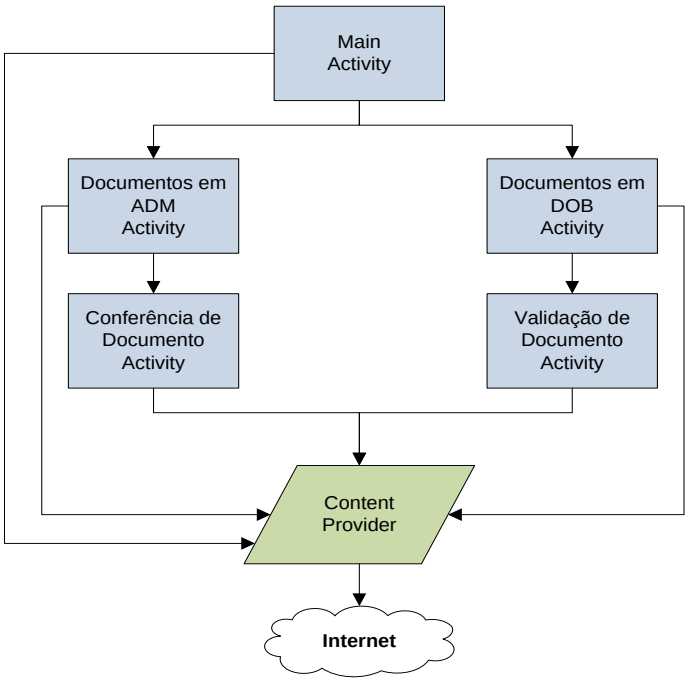

Figura 35 – Composição do módulo de Conferência Documental

A interface de utilizador é constituída por um total de cinco *activities* que serão abordadas com mais detalhe no capítulo 6.3.2.2.

O *Content Provider* implementado neste módulo é responsável apenas pela abstracção da comunicação da obtenção dos dados entre a interface de utilizador e a aplicação *middleware*. Como a informação tem que estar sempre actualizada, não é utilizado, neste módulo, nenhum dos métodos de economia de tráfego propostos no capítulo 5.3.

6.3.2.2 Interface de utilizador

A interface de utilizador é constituída, como já referido, por um total de cinco *activities*: *main*, documentos em ADM (administrativo), documentos em DOB (director de obra), conferência de documento e validação de documento.

A *activity* inicial (Figura 36 a)) é constituída por uma listagem das obras que têm documentos por conferir, com indicação de quantos documentos aguardam conferência e quantos aguardam validação. O utilizador ao seleccionar um dos botões azuis, que representam os documentos que aguardam conferência pelos administrativos, irá ser conduzido a uma listagem destes documentos (Figura 36 b)). Cada item da lista representa um documento por conferir, sendo indicado o nome da entidade que o emitiu, o seu número interno, o número de dias desde a sua introdução no sistema e o valor monetário associado ao documento. Caso seja seleccionado um dos botões verdes, que representam os documentos que aguardam validação pelos directores de obra, é apresentado ao utilizar uma lista visualmente semelhante à dos documentos que aguardam conferência, com o

acréscimo de informação relativa à conferência do documento, mais concretamente o nome do colaborador que conferiu o documento e a data em que foi conferido (Figura 36 c)).

Figura 36 – Conferência Documental – a) main b) documentos em ADM c) documentos em DOB

A conferência por parte dos administrativos é realizada através de três passos: identificação do documento, estado do documento e classificação do documento. A implementação visual destes três passos foi realizada através da utilização do componente *ViewFlipper* que agrega um conjunto de *views* e permite a animação na transição entre as mesmas [Google Inc, 2011a]. De todas as *views* que agrega, apenas uma fica visível no ecrã. No nosso módulo, cada passo da conferência será uma *view* e a transição entre passos será realizada através de navegação com botões, utilizando os métodos fornecidos pelo *ViewFlipper*. A transição animada entre a *view* actual e a próxima é realizada da seguinte forma:

```
ViewFlipper vf_phase = (ViewFlipper)findViewById(R.id.vf_phase);
vf_phase.setInAnimation(inFromRightAnimation());
vf_phase.setOutAnimation(outToLeftAnimation());
vf_phase.showNext();
```

No primeiro passo (identificação do documento) é apresentado ao utilizador a informação previamente inserida no sistema (Figura 37 a)), sendo que este apenas pode alterar os campos respectivos a prazos de pagamento, percentagem de retenção e prazo de retenção.

No segundo passo (estado do documento) o utilizador deverá indicar se o documento confere ou não confere. Ao seleccionar que o documento confere deverá indicar se este é totalmente válido, se apenas é válido parcialmente ou se fica sujeito a nota de crédito. Ao ser seleccionada a opção de parcialmente válido, ficam visíveis dois campos adicionais: o valor monetário parcial que deve de ser conferido por esta obra e a obra a que pertence o

resto do valor. Se por sua vez o documento necessitar de uma nota de crédito, o utilizador terá que introduzir o valor monetário dessa respectiva nota. Caso o documento não confira, são apresentadas três opções: devolver documento, documento pertence a outra obra e erros na identificação do documento. Se o documento pertencer a outra obra, o utilizador terá de seleccionar de uma lista a obra a que este pertence. Neste segundo passo são ainda dadas mais duas opções ao utilizador: indicar que o documento real já foi assinado pelo director de obra e por isso não necessita que este valide a conferência, e introduzir as observações que ache necessário.

Os componentes de escolha de valores, como por exemplo, a escolha do estado do documento, são do tipo *Spinner*. Os *Spinners* são *widgets* que permitem a escolha de um item dentro de um conjunto de itens. O seu comportamento é semelhante às tradicionais listas de *dropdown* [Google Inc, 2011a].

O terceiro e último passo (classificação do documento) tem como objectivo indicar se o documento é uma nota de crédito ou nota de débito. Caso seja uma nota de crédito o utilizador terá que indicar qual a nota de débito a que esta está ligada. Para terminar o processo de conferência o utilizador apenas terá que seleccionar o botão *conferir*, caso não o pretenda fazer pode seleccionar o botão *recuar* ou pressionar o botão físico *back* existente no dispositivo.

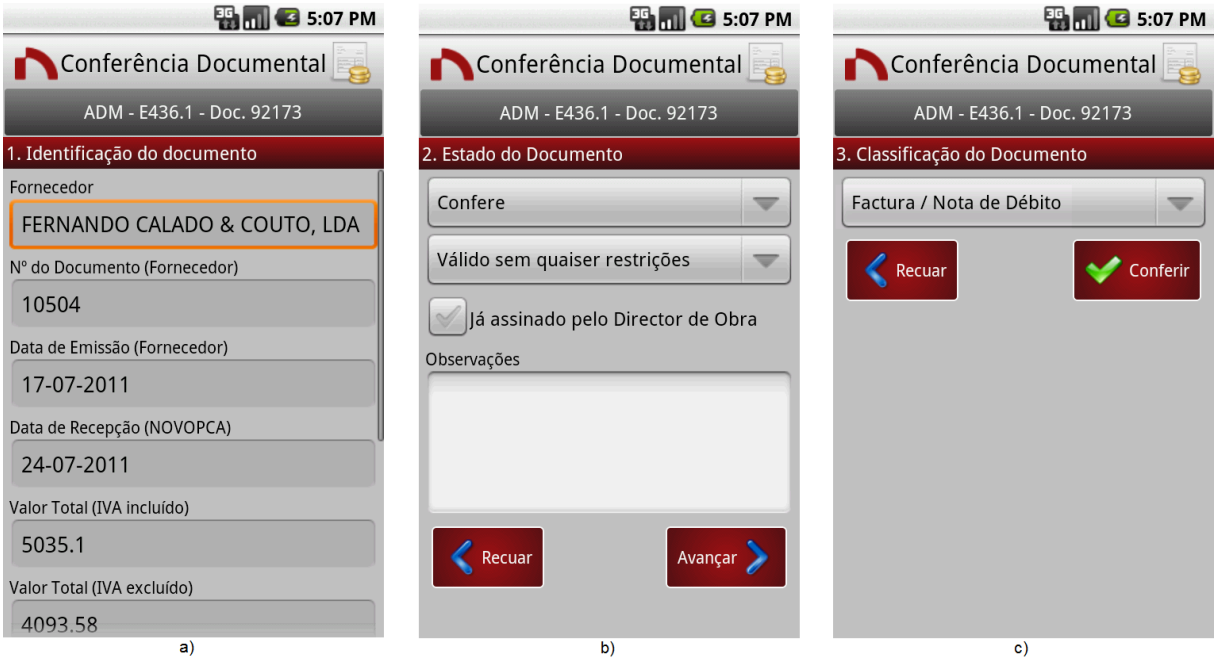

Figura 37 – Conferência de documento – a) identificação b) estado c) classificação

A validação da conferência por parte dos directores de obra engloba todos os passos existentes na conferência pelo administrativo mais um quarto passo de informação adicional. A diferença está no facto de nos três primeiros passos o utilizador não ter possibilidade de

introduzir ou alterar nenhum dos dados, que foram preenchidos anteriormente pelo administrativo. Nesta *activity* também foi utilizado o componente ViewFlipper.

No primeiro passo (Figura 38 a)) o utilizador poderá visualizar quem conferiu o documento e quando este foi conferido, assim como a restante informação identificativa do documento. No segundo (Figura 38 b)) e terceiro (Figura 38 c)) passo são apresentadas as informações seleccionadas e introduzidas anteriormente pelo administrativo. No último passo (Figura 38 d)), informação adicional, é dada ao utilizador a possibilidade deste inserir informação que ache relevante para a validação do documento. Aqui o utilizador deverá de validar ou rejeitar o documento, seleccionando o respectivo botão, ou retornar para trás na navegação, para rever as informações associadas ao documento.

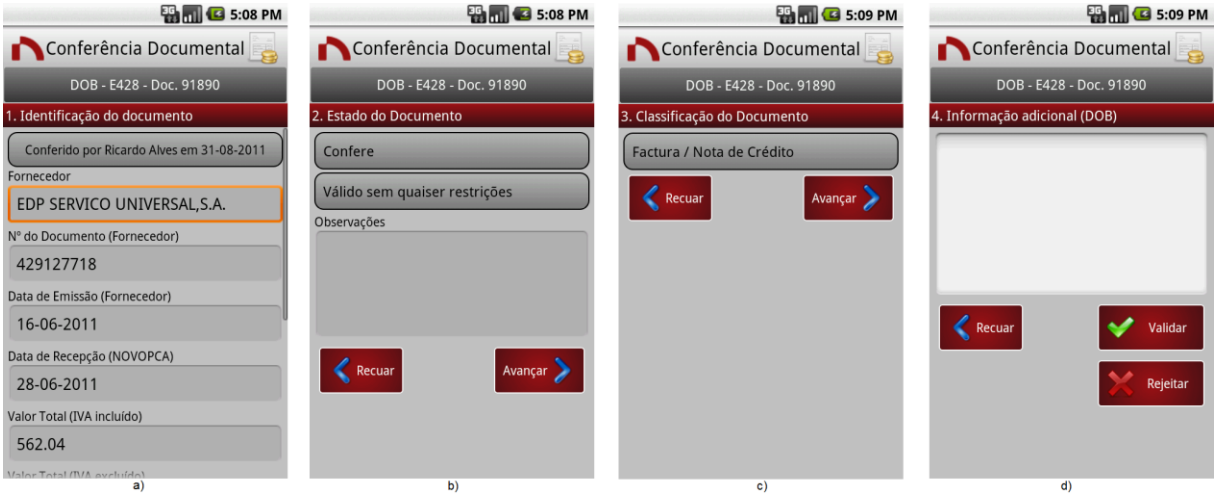

Figura 38 – Validação de Documento – a) identificação b) estado c) classificação c) informação adicional

6.3.3 Fornecedores

Uma das principais necessidades dos trabalhadores das empresas de construção é a obtenção constante de informações relativas a fornecedores de produtos ou serviços. A coordenação de uma grande obra é um processo muito delicado e complexo, existindo a necessidade constante de comunicação entre os colaboradores em obra e os fornecedores.

O módulo apresentado neste capítulo tem como principal objectivo proporcionar aos utilizadores uma forma intuitiva e rápida de obtenção de dados relativos a fornecedores. Deverá, portanto, existir um mecanismo de pesquisa de fornecedores que permitirá aos utilizadores encontrarem e consultarem rapidamente informação relativa ao fornecedor que procuram. A consulta de um fornecedor engloba várias informações a este associadas, nomeadamente: dados gerais, contactos, facturação, materiais e serviços, documentos e recursos humanos. Sempre que possível, o módulo deverá permitir uma interacção com informação relevante, por exemplo, sempre que seja apresentado um número de telefone deverá existir a possibilidade de ligar para esse número.

As funcionalidades disponíveis aos utilizadores estão representadas no diagrama de caso de uso presente na Figura 39.

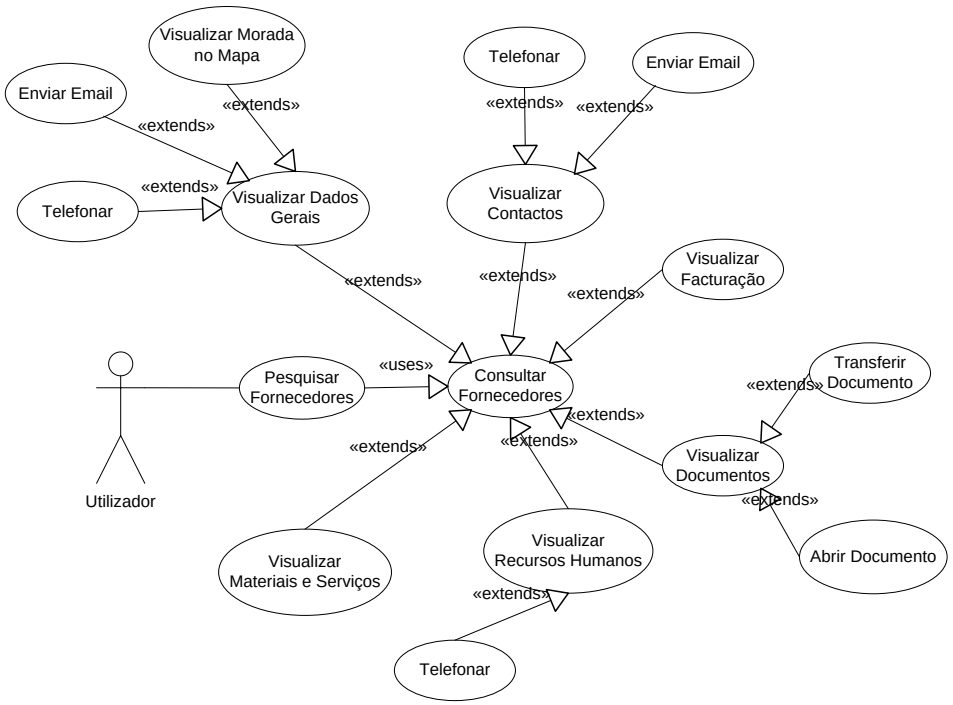

Figura 39 – Caso de uso do módulo de Fornecedores

As informações relativas a fornecedores são alteradas esporadicamente e por isso é aqui presente um caso onde deverá ser implementado o sistema proposto no capítulo 5.3.1. Sendo um módulo de consulta de dados, não é gerado qualquer tipo de informação que seja necessária enviar para o servidor.

6.3.3.1 Composição

O módulo de fornecedores é construído por vários componentes, sendo a mais abrangente a interface de utilizador que é composta por um número alargado de *activities*. Na Figura 40 está representada toda a composição deste módulo e respectivas interacções entre componentes.

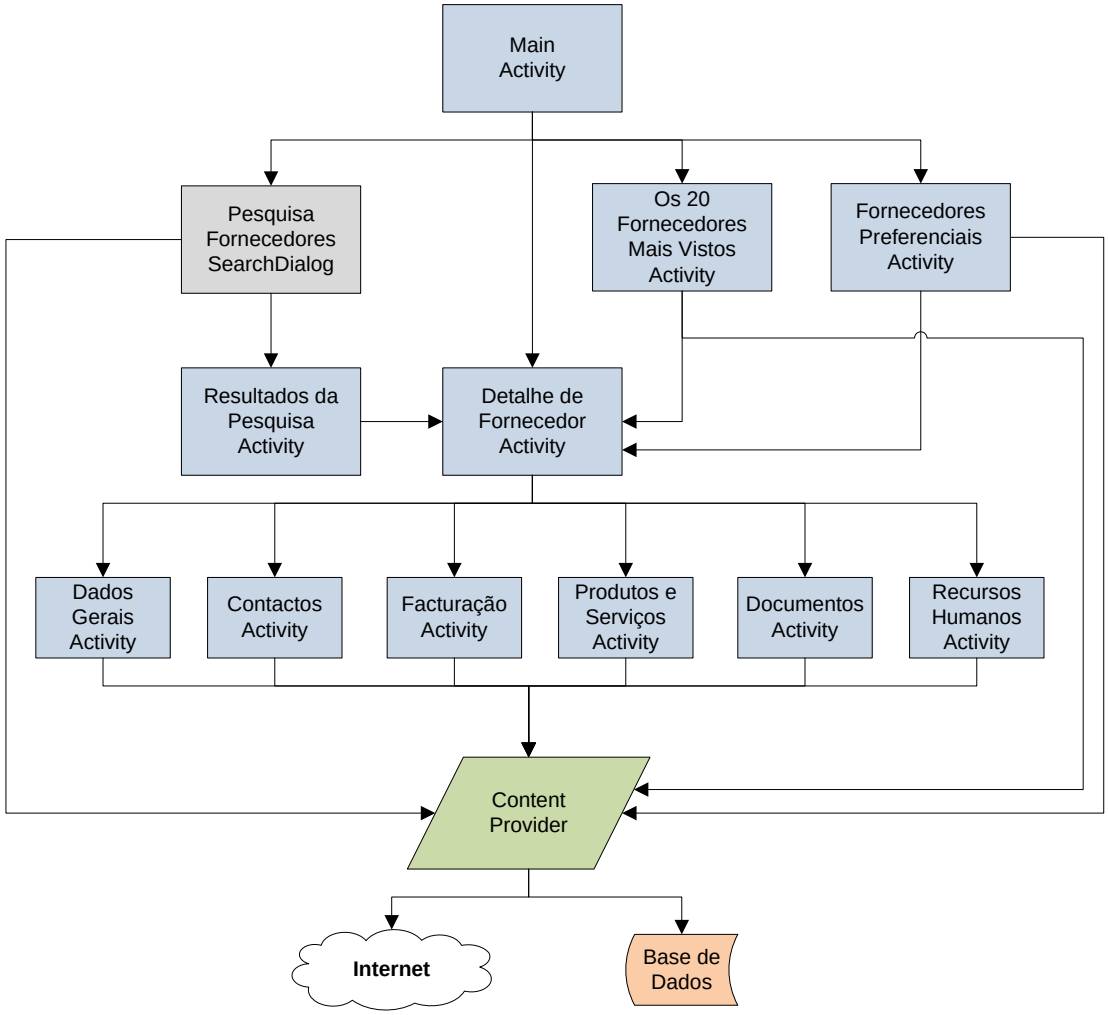

Figura 40 – Composição do módulo de Fornecedores

A interface de utilizador é composta por um total de onze *activities* (que serão abordadas com mais detalhe no próximo capítulo) e por um componente de pesquisa fornecido pelo Android, denominado *SearchDialog*. O *SearchDialog* é um componente da interface de utilizador que é controlado pelo sistema Android. Quando activado pelo utilizador, este componente apresenta uma interface de pesquisa sobreposta sobre a *activity* que esteja visível [Google Inc, 2011a]. É possível estender o seu comportamento adicionando, por exemplo sugestões de pesquisa com base nos dados que estão a ser digitados pelo utilizador ou com base em pesquisas anteriormente realizadas. Ao ser realizada uma pesquisa, o *SearchDialog* envia um *Intent* para uma determinada *activity,* com todos os caracteres introduzidos pelo utilizador denominado *query.* A *activity* que recebe o *query* (que é neste módulo a *Resultados da Pesquisa*) será então responsável pela criação dos resultados obtidos com base na pesquisa efectuada pelo utilizador.

O *Content Provider* tem a função de disponibilizar informação à interface de utilizador; este componente irá implementar o sistema de economia de tráfego na recepção de informação descrito no capítulo 5.3.1.

6.3.3.2 Interface de utilizador

O módulo de fornecedores é constituído, ao nível da interface de utilizador, por um total de onze *activities.* A *activity* inicial (Figura 41 a)) é o ponto de partida para a obtenção de informações de fornecedores. O utilizador poderá aqui seleccionar se pretende efectuar pesquisa, visualizar o último fornecedor consultado, visualizar uma listagem dos fornecedores preferenciais ou uma listagem dos vinte fornecedores mais vistos globalmente.

A pesquisa por fornecedores (Figura 41 b)) é efectuada, como já referido, através da componente *SearchDialog*. Aqui o utilizador deverá de introduzir o nome a pesquisar ou seleccionar um dos nomes previamente pesquisados. Concluído este processo, é apresentado ao utilizador uma lista de resultados obtidos (Figura 41 c)).

As *activities* que apresentam os fornecedores preferenciais (Figura 41 d)) e os vinte fornecedores mais vistos são muito semelhantes à *activity* que apresenta os resultados de pesquisa. No entanto a apresentação dos fornecedores preferenciais tem uma diferença em relação às restantes: aqui são apresentados resultados agrupados em números de vinte elementos, sendo dada a possibilidade ao utilizador deste obter os próximos vinte registos, até ao ponto em que não existem mais resultados para apresentar.

Figura 41 – Fornecedores - a) ecrã inicial b) pesquisa c) resultados da pesquisa d) fornecedores preferenciais

Ao ser seleccionado um fornecedor, de uma das listagens mencionadas, é apresentada a *activity* de detalhe de fornecedor (Figura 42 a)). Neste ecrã é apresentado ao utilizador a indicação se o fornecedor é um fornecedor preferencial, apto ou inapto, assim como é dada

a possibilidade deste visualizar as informações relativas a este fornecedor e que são apresentadas noutras *activities.*

Os dados gerais (Figura 42 b)) de um fornecedor contemplam várias informações, desde número de contribuinte até à respectiva morada. O utilizador poderá aqui executar algumas operações sobre os dados, como por exemplo telefonar ao fornecedor ou visualizar no Google Maps a morada do fornecedor.

Uma das principais funcionalidades para os colaboradores da empresa é a obtenção dos contactos associados a fornecedores. A *activity* de contactos (Figura 42 c)) apresenta uma listagem dos contactos existentes e permite que o utilizador telefone ou envie um e-mail ao contacto que pretenda.

A facturação do fornecedor é apresentada em forma de gráfico, como representado na Figura 42 d). A criação do gráfico foi realizada através da utilização de uma livraria, denominada *androidplot*, que permite a criação de gráficos dinâmicos ou estáticos dentro de aplicações para o sistema Android [AndroidPlot, 2011]. A sua estrutura, apesar de um pouco complexa, é muito poderosa e permite a construção de gráficos de diversos tipos.

Figura 42 – Fornecedores – a) detalhe de fornecedor b) dados gerais c) contactos d) facturação

O ecrã de produtos e serviços (Figura 43 a)) apresenta uma listagem em árvore de todos os produtos e serviços fornecidos pelo fornecedor, meramente de consulta.

Os documentos associados a um fornecedor podem ser visualizados na *activity* de documentos (Figura 43 b)). Ao ser seleccionado um documento é iniciada a sua transferência e respectivo armazenamento no *SD-Card* do dispositivo, sendo posteriormente invocado a sua visualização através de qualquer aplicação existente no sistema que a suporte. Caso o documento já tenha sido transferido anteriormente e exista no *SD-Card*, apenas é invocado o pedido de visualização do documento da seguinte forma (para os casos em que a extensão do documento seja *.pdf*):

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.fromFile(file));
intent.setType("application/pdf");
DocumentsActivity.this.startActivity(intent);
```

A transferência do documento é realizada através da utilização da classe AsyncTask, já referida anteriormente.

Por último existe a *activity* de recursos humanos (Figura 43 c)), onde são listados todos os colaboradores que podem ser subcontratados ao fornecedor. Pouca informação existe sobre estes colaboradores, sendo que normalmente apenas existe a indicação do nome e da categoria profissional deste. Para os casos em que exista indicação de número de telefone, a *activity* permite que o utilizador realize uma chamada para o colaborador.

Figura 43 – Fornecedores – a) produtos e serviços b) documentos c) recursos humanos

6.3.4 Presenças em Obra

No âmbito de obra, é necessário que o encarregado desta realize o controlo sobre as presenças e ausências dos colaboradores subcontratados temporariamente a outras organizações. Sem a existência de um processo informático para o efeito este controlo pode por vezes ser muito moroso. O responsável pela subcontratação de trabalhadores (que está normalmente situado na sede) tem toda a informação relativa aos colaboradores que foram subcontratados para as obras, enviando-a todos os dias para o estaleiro da obra normalmente através de correio electrónico. O encarregado terá então de imprimir o documento e utiliza-lo para efectuar o registo de presenças.

O módulo de presenças em obra tem como principal objectivo agilizar este processo. A sua interface deverá ser o mais simplista possível para que o utilizador consiga realizar o controlo de presenças de uma forma bastante rápida.

O módulo deverá permitir que o utilizador seleccione a obra onde se encontra e que, dentro do âmbito da obra seleccionada, realize o controlo de presenças. A Figura 44 representa as funcionalidades disponibilizadas ao utilizador através de um diagrama de caso de uso.

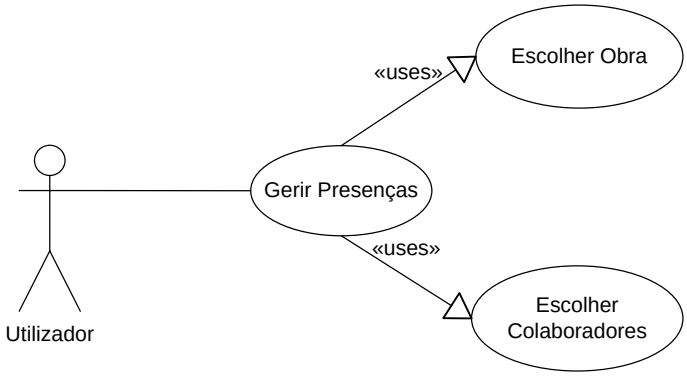

Figura 44 – Caso de uso do módulo de Presenças em Obra

A informação utilizada por este módulo sofre várias alterações diariamente, pelo que deverá estar sempre actualizada. Os registos de presenças serão enviadas para o servidor para futura análise e utilização nos cálculos de custos.

6.3.4.1 Composição

Um dos objectivos para o desenvolvimento deste módulo, como já referido, é a criação de uma interface de utilizador que seja o mais simples possível. A sua composição utiliza por essa razão o mínimo de componentes indispensáveis à realização do processo de controlo de presenças, denominados: *activities, content provider, network receiver* e base de dados local.

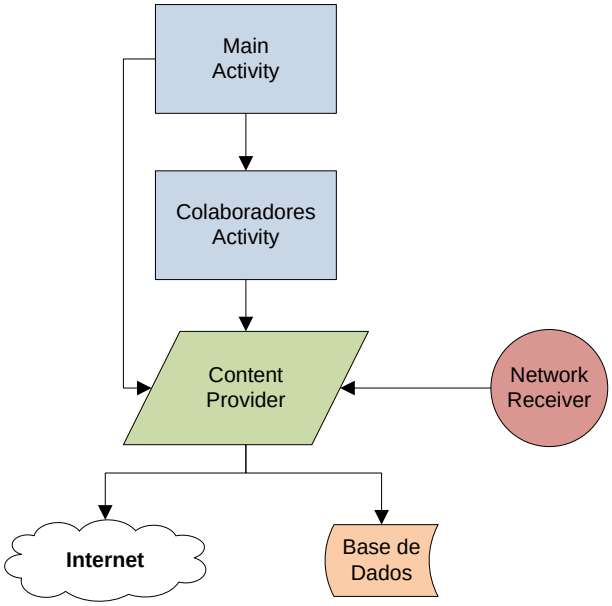

Figura 45 – Composição do módulo de Presenças em Obra

As *activities* que englobam a interface de utilizador são apenas duas: a *activity* principal de selecção de obras e a *activity* com uma listagem de colaboradores. Estas duas *activities* serão apresentadas com mais pormenor no próximo capítulo (6.3.4.2).

O *Content Provider* é responsável pela comunicação entre a interface de utilizador e o servidor ao nível da recepção de dados. Quando o utilizador indica que um determinado colaborador está presente na obra, este registo é inserido na base de dados local para futuro envio para o servidor. O envio de dados implementa o sistema apresentado no capítulo 5.3.2 com o auxílio de um *Network Receiver*.

O *Network Receiver*, que implementa as funcionalidades de um *broadcast receiver*, recebe as notificações enviadas pelo Android quando existem alterações às interfaces de conexão à Internet. Caso a notificação recepcionada indique que está disponível uma ligação WiFi o *Network Receiver* obtém, através do *Content Provider*, todos os registos armazenados temporariamente e envia-os para o servidor.

6.3.4.2 Interface de utilizador

O presente módulo é constituído, ao nível da interface de utilizador, por duas *activities*: a *activity* principal (Figura 46 a)) que apresenta uma lista de obras em curso e a *activity* de colaboradores (Figura 46 b) e c)) que apresenta uma listagem dos colaboradores alocados à obra seleccionada.

Figura 46 – Presenças em Obra - a) lista de obras b) lista de colaboradores c) lista de colaboradores com menu visível

A lista de obras utiliza um tipo de *activity* especial denominada *ListActivity* que alberga um objecto do tipo *ListView* que pode ser vinculado a diferentes tipos de fontes de dados. Uma *ListView* é uma *view* que apresenta itens numa lista vertical; na nossa *activity* a fonte de dados ao qual esta se encontra vinculada é do género ListAdapter. Este componente faz a

ponte entre a *ListView* e os dados que esta irá apresentar. Através desta lista o utilizador pode seleccionar qual a obra para o qual quer realizar a gestão de presenças.

A lista de colaboradores é obtida através da utilização de uma *GridView*, que não é mais do que uma *ViewGroup* (ou agrupamentos de *views*) que exibe um conjunto de itens numa grelha bidimensional com funcionalidades de *scroll*. A ligação entre a GridView e os itens que irá abranger é realizada através de um *BaseAdapter*. Cada item é composto por um botão com a fotografia de um colaborador como imagem central. A representação da presença ou não do colaborador na obra é realizada através de um mecanismo de cores. Ao seleccionar um item, o utilizador faz com que o fundo do botão deste mude imediatamente de cor e passe a indicar se o colaborador seleccionado se encontra na obra (cor verde) ou não (cor vermelha).

Pressionando o botão físico "*menu"* existente no dispositivo é apresentado ao utilizador duas hipóteses: inverter a selecção e actualizar lista. Ao inverter a selecção, todos os colaboradores marcados como presentes passaram a ficar como ausentes e vice-versa. A actualização da lista invoca a obtenção da lista de colaboradores novamente do servidor.

6.4 Testes

A realização de testes ocorreu depois de terminada toda a implementação do projecto apresentado nesta dissertação.

Os testes ao nível de utilizador iniciaram-se com a instalação da aplicação NovopcApp num dispositivo Android. A aplicação foi executada e realizaram-se todas as operações por esta disponibilizada de forma a verificar o seu correcto funcionamento. Mesmo quando o dispositivo se encontrava sem ligação à internet e não existia qualquer informação armazenada temporariamente a aplicação executou sem problemas, indicando que não existia uma ligação disponível à internet. A gestão de módulos realizou-se sem problemas, sendo por vezes detectado que a actualização da listagem de módulos, depois de instalar um módulo, demorava uns segundos a actualizar e a colocar esse mesmo módulo como instalado.

Os testes ao nível de utilizador realizados aos módulos não apresentaram problemas significativos, sendo que todas as operações foram realizadas com sucesso.

A aplicação *middleware* foi testada com a realização de múltiplos pedidos simultâneos realizados por dois clientes em simultâneo. Todos os pedidos efectuados foram respondidos com sucesso, o que demonstrou que o grau de fiabilidade do servidor é muito razoável, sendo por vezes, e compreensivelmente, demorado na realização das respostas.

Foi também testado uma característica muito importante em todas as aplicações que tenham interacção com os utilizadores: os tempos de resposta. Para a realização deste teste, foi utilizada a classe *TimingLogger,* já referida neste documento, em algumas das *activities* que integram a interface de utilizador. Obtidos os resultados e realizada a média analítica dos tempos de resposta, obtemos um valor final de 2,01 segundos. O valor apesar de não ser exagerado é um pouco alto e pode por vezes diminuir a experiência do utilizador. Com um valor médio de tempo de resposta superior a 1 segundo, todas as *activities* que obtêm dados quer seja do servidor ou da base de dados devem apresentar ao utilizador um componente que dê a entender que a aplicação esteja ocupada a obter dados [Nielsen, 1993]. Em todas as *activities* desenvolvidas é apresentado ao utilizador um *ProgressDialog* (Figura 47) enquanto a aplicação recebe ou envia dados para o servidor ou para a base de dados.

Figura 47 – Exemplo de um ProgressDialog

Os testes realizados mostraram que a plataforma Novopca Mobile comportou-se como esperado.

7 Conclusões e trabalho futuro

Neste capítulo serão apresentadas as conclusões obtidas através dos estudos, escolhas e implementações realizadas. Serão ainda discutidas as vantagens e desvantagens da plataforma Novopca Mobile assim como o trabalho futuro passível de realização.

O principal objectivo do projecto aqui documentado era a criação de uma solução móvel que permitisse o aumento de eficiência de vários processos de trabalho da empresa Novopca. O desenvolvimento de uma solução deste género obrigou ao estudo, análise e escolha de várias soluções.

O início deste projecto foi composto pelo levantamento do estado da arte de temas relevantes à sua execução e pelo estudo de qual o sistema operativo móvel que melhor se enquadrava com os objectivos pretendidos. O estudo realizado levou à escolha do Android como sendo o sistema operativo que mais garantias dava ao desenvolvimento deste projecto como por exemplo garantias de crescimento futuro da sua utilização. No final do projecto, posso afirmar que a escolha foi a mais correcta, pois foi possível desenvolver todos os objectivos propostos. O Android mostrou-se ser uma plataforma já madura, que em conjugação com o Eclipse IDE permite o desenvolvimento de aplicações de forma facilitada e intuitiva. As ferramentas contidas no ADT do Android mostraram-se muito úteis, tanto no desenvolvimento como na realização de testes de performance.

A troca de informação entre a plataforma móvel e o servidor, obrigatória à realização deste projecto, levou ao estudo de uma das formas mais conhecidas e documentadas de troca de informação entre maquias, os *Web Services*. Os *Web Services* são constituídos por dois tipos, os *Big* e os *RESTful Web Services*. O estudo realizado a estes dois tipos de serviços Web mostrou que os *Big Web Services*, apesar de amplamente utilizados hoje em dia, não são a forma mais adequada à comunicação entre dispositivos móveis e outros sistemas. Os serviços Web baseados em REST, por sua vez, mostraram ser mais leves e mais simples de implementar; a sua arquitectura aberta permite que os dados sejam transmitidos em qualquer formato. Como tal, foram estudados os dois principais tipos de representação de informação: o XML e o JSON. Os resultados mostram que o JSON ganha vantagem sob o XML, tanto ao nível de tempos de processamento como ao nível do tamanho da informação. Foi a conjugação entre REST e JSON o método de comunicação escolhido para o desenvolvimento do projecto. A escolha efectuada ao nível da comunicação mostrou-se também acertada, pois os tempos de comunicação da plataforma móvel com o servidor são muito bons levando a que os utilizadores tenham uma boa experiencia com a aplicação. REST tem porém a desvantagem de não definir a forma como a segurança na comunicação

deverá de ser implementada, ao contrário dos *Big Web Services* que tem já especificações a este nível.

A arquitectura proposta foi a ideal no contexto deste projecto. Como os colaboradores das empresas de construção estão normalmente dispersos por vários locais do nosso país (por vezes até do mundo), é condição obrigatória que a arquitectura apresentada fosse uma arquitectura modular, permitindo que os utilizadores obtenham novos módulos de trabalho em qualquer parte do mundo.

Os sistemas de economia de tráfego utilizados mostraram também ser um grande aditivo ao projecto. Para além de permitirem a poupança de tráfego dos dispositivos, estas permitem também a poupança da bateria dos equipamentos pela não utilização da internet assim como uma maior fluidez da navegação dentro plataforma.

Em termos de implementação o projecto correu como planeado, não tendo existido grandes dificuldades de execução, para além da típica habituação inicial à plataforma Android. Os módulos desenvolvidos mostram o potencial da plataforma e abrem novas portas para futuros módulos a desenvolver.

Os testes realizados ao projecto desenvolvido mostraram que o sistema tem um funcionamento muito razoável. Os pedidos ao servidor foram sempre respondidos e as aplicações nunca mostraram problemas de funcionamento dignos de registo.

Futuramente a plataforma Novopca Mobile deverá de englobar cada vez mais módulos, tornando-se assim cada vez mais uma ferramenta única de trabalho. Os módulos já desenvolvidos podem também ser alvo de correcções e de melhorias.

8 Referências Bibliográficas

[Apple Inc, 2011]

Apple – iPhone 4. Disponível em http://www.apple.com/iphone/. Acedido a 15 de Agosto de 2011.

[AICCOPN, 2011]

AICCOPN quer construção como motor da economia. Disponível em http://www.aiccopn.pt/news.php?news_id=739. Acedido a 20 de Fevereiro de 2011.

[Baker, 2005]

Baker A. *Symbian Device - The OS Evolution*. 2005. Disponível em http://www.i-symbian.com/wp-content/uploads/2009/11/Symbian_Evolution.pdf. Acedido a 20 de Fevereiro de 2011.

[Bowden et al., 2006]

Bowden, S., Dorr, A., Thorpe, T., Anumba, C. (2006). *Mobile ICT support for construction process improvement*. In: Automation in Construction, Volume 15, Setembro de 2006.

[Bray, 2004]

WS-Pagecount. Disponível em http://www.tbray.org/ongoing/When/200x/2004/09/21/WS-Research. Acedido a 8 de Abril de 2011.

[Businessweek, 2005]

Google Buys Android for its Mobile Arsenal. Disponível em http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm. Acedido a 20 de Fevereiro de 2011.

[Comer, 2000]

Comer, D. (2000), *Internetworking with TCP/IP*, 4ª ed.: Addison Wesley.

[Deibert et al. 2008]

Deibert, S., Heinzl, A. and Rothlauf, F. (2008). *The Impact Logic of Mobile Technology Usage on Job Production*. In: Proceedings of the Fourteenth Americas Conference on Information Systems, 2008, Toronto, ON, Canada

[Deibert et al., 2009]

Deibert, S., Hemmer, E. and Heinzl, A. (2009). *Mobile Technology in the Construction Industry – the Impact on Business Processes in Job Production.* In: Proceedings of the Fifteenth Americas Conference on Information Systems, 2009, San Francisco, California

[Destak, 2011]

Sector da construção na pior crise de há memória. Disponível em http://www.destak.pt/artigo/84807. Acedido a 20 de Fevereiro de 2011.

[Eckerson, 1995]

Eckerson, W. W. (1995), *Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications*. Open Information Systems 10

[Fielding, 2000]

Fielding, R. (2000), *Architectural Styles and the Design of Network-based Software Architectures*. Tese de Doutoramento, University of California.

[Forsberg, 2009]

Forsberg, D. (2009), *RESTful Security*, In: W2SP'09, Web 2.0 Security and Privacy, Maio de 2009, Oakland, USA.

[Gargenta, 2011]

Gargenta, M. (2011), *Learning Android.* 1º ed.: O'Reilly Media.

[Gartner Inc, 2010]

Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009; Market Remained Flat in 2009. Disponível em http://www.gartner.com/it/page.jsp?id=1306513. Acedido a 20 de Fevereiro de 2011.

[Gartner Inc, 2011]

Gartner Says Sales of Mobile Devices in Second Quarter of 2011 Grew 16.5 Percent Year-on-Year; Smartphone Sales Grew 74 Percent. Disponível em http://www.gartner.com/it/page.jsp?id=1764714. Acedido a 20 de Agosto de 2011.

[Google Inc, 2011a]

Android developers. Disponível em http://developer.android.com/. Acedido a 15 de Março de 2011.

[Google Inc, 2011b]

Nexus S. Disponível em http://www.google.pt/nexus/. Acedido a 22 de Agosto de 2011.

[Hodin, 2011]

Hodin, K. (2011) *Android Architecture*. Disponível em https://plus.google.com/117430033927577602978/posts/3T8uM3WgoXY. Acedido a 15 de Março de 2011.

[HTC, 2011]

HTC HD7 Apresentação de Produto. Disponível em http://www.htc.com/pt/smartphones/htc-hd7/. Acedido em 12 de Setembro de 2011.

[Jonas, 2011]

Mobile Construction Software. Disponível em http://www.jonas-construction.com/products/mobile. Acedido a 15 de Janeiro de 2011.

[JSON, 2011]

JSON. Disponível em http://www.json.org/. Acedido a 18 de Abril de 2011.

[kSoap, 2011]

A lightweight and efficient SOAP library for the Android platform. Disponível em http://code.google.com/p/ksoap2-android/. Acedido a 20 de Junho de 2011.

[Lascelles and Flint, 2006]

Lascelles, F. and Flint, A. (2006) *WS Security Performance*. Disponível em http://websphere.sys-con.com/node/204424. Acedido a 8 de Abril de 2011.

[Macworld, 2007]

Apple unveils iPhone. Disponível em http://www.macworld.com/article/54769/2007/01/iphone.html. Acedido a 20 de Fevereiro de 2011.

[Metodista, 2011]

A comunicação é imprescindível. Disponível em http://www.metodista.br/jornal-metodista/77/opiniao. Acedido a 20 de Agosto de 2011.

[Microsoft Inc, 2011a]

Windows Phone 7. Disponível em http://www.microsoft.com/windowsphone/pt-pt/. Acedido a 10 de Setembro de 2011.

[Microsoft Inc, 2011b]

Metro Design Language of Windows Phone 7. Disponível em http://www.microsoft.com/design/toolbox/tutorials/windows-phone-7/metro/. Acedido a 10 de Setembro de 2011.

[Nielsen, 1993]

Nielsen, J. (1993), *Usability Engineering*. 1ª ed. São Francisco.

[Nokia Inc, 2011]

Symbian Belle. Disponível em http://europe.nokia.com/symbian-belle. Acedido a 3 de Setembro de 2011.

[Oasis, 2011]

OASIS Web Services Security (WSS) TC. Disponível em http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss. Acedido a 16 de Junho de 2011.

[Open Handset Alliance, 2011]

Open Handset Alliance. Disponível em http://www.openhandsetalliance.com/. Acedido a 20 de Fevereiro de 2011.

[Oracle, 2011]

Why StAX?. Disponível em http://download.oracle.com/docs/cd/E17802_01/webservices/webservices/docs/1.6/tutorial/doc/SJSXP2.html. Acedido a 16 de Abril de 2011.

[Pautasso et al., 2008]

Pautasso, C., Zimmermann, O., Leymann, F. (2008), *RESTful Web Services vs "Big" Web Services: Making the Right Architectural Decision*. In: Proceeding WWW '08, Proceeding of the 17th international conference on World Wide Web, ACM New York, NY, USA 2008.

[Penta, 2011]

Mobile Construction Software from Penta - Field Service Software, Field Reporting Software. Disponível em http://www.penta.com/mobilesolutions.html. Acedido a 13 de Fevereiro de 2011.

[RFC2617, 1999]

HTTP Authentication: Basic and Digest Access Authentication. Disponível em http://www.ietf.org/rfc/rfc2617.txt. Acedido a 20 de Abril de 2011.

[Richardson and Ruby, 2007]

Richardson, L. and Ruby, S. (2007), *RESTful Web Services*. 1ª ed.: O'Reilly Media.

[Satyanarayanan, 1996]

Satyanarayanan, M. (1996). *Fundamental challenges in mobile computing*. In: Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing. Philadelphia, United States. Maio 23 a 26 de 1996.

[Schneidawind, 1992]

Schneidawind, J. (1992), *Big Blue unveiling*. USA Today.

[Sharkey, 2009]

Sharkey, J. (2009) *Coding for Life--Battery Life, That Is*. Disponível em http://61.153.44.88/google/io/2009/W_0300_CodingforLife-BatteryLifeThatIs.pdf. Acedido a 15 de Junho de 2011.

[Stockholm, 2011]

History - Stockholm Smartphone. Disponível em http://www.stockholmsmartphone.org/history/. Acedido a 20 de Fevereiro de 2011.

[TIOBE Inc, 2011]

TIOBE Programming Community Index for September 2011. Disponível em http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html. Acedido a 16 de Agosto de 2011.

[Vilkko et al., 2008]

Vilkko, T.; Kallonen, T.; Ikonen, J. (2008). *Mobile fieldwork solution for the construction industry*. In: Software, Telecommunications and Computer Networks, 2008. Setembro de 2008.

[W3C, 2008]

Extensible Markup Language (XML) 1.0 (Fifth Edition) - Origin and Goals. Disponível em http://www.w3.org/TR/REC-xml/#sec-origin-goals. Acedido a 15 de Abril de 2011.

[W3C, 2011a]

Web Services Glossary. Disponível em http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211/. Acedido a 16 de Junho de 2011.

[W3C, 2011b]

Extensible Markup Language (XML). Disponível em http://www.w3.org/XML/. Acedido a 15 de Abril de 2011.

[Wikipédia, 2011a]

Mobile operating system. Disponível em http://en.wikipedia.org/wiki/Mobile_operating_system. Acedido a 20 de Fevereiro de 2011.

[Wikipédia, 2011b]

Web service. Disponível em http://pt.wikipedia.org/wiki/Web_service. Acedido a 16 de Junho de 2011.