Universidade Federal do Rio Grande do Sul
Instituto de Informática
Ciência da Computação

Trabalho de Conclusão II

Modelagem e prototipação de uma aplicação LBS utilizando a plataforma Android.

Orientando: Ricardo da Silva Floriano Machado
Orientador: Prof. Doutor Cláudio Fernando Resin Geyer

Porto Alegre
2010

Ricardo da Silva Floriano Machado

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação, pelo Curso de Ciência da Computação da Universidade Federal do Rio Grande do Sul.

Orientador:
Prof. Doutor Cláudio Fernando Resin Geyer

Porto Alegre
2010

Dedico este trabalho a todos aqueles
que contribuíram para sua realização.

Agradecimentos

Tudo aquilo que construímos, que apresente um considerável grau de complexidade, tanto em nível intelectual quanto prático, certamente contém a influência das pessoas que nos cercam. Este é o caso deste trabalho de conclusão de curso, e portanto aqui gostaria de deixar meu agradecimento sincero, pela prestativa contribuição mais direta, que algumas pessoas tiveram na elaboração do mesmo:

Ao professor Cláudio F. R. Geyer, pela inspiração para o tema do trabalho e pela orientação durante o desenvolvimento do mesmo.

Ao colega e amigo, Eduardo Bobsin Machado, por compartilhar o conhecimento nos níveis de arquitetura de sistemas, as preocupações necessárias, e pela paciência no trabalho.

Aos colegas e amigos Gustavo Jotz, Miguel Horlle e Daniel Tamiosso, por serem grandes desenvolvedores e terem me dado a oportunidade de me melhorar como programador, e ter sido capaz de colocar em prática este trabalho, ao Daniel um agradecimento especial pela dica da utilização da tecnologia REST.

À Bárbara Zaffari Cavedon, pelo amor, pela paciência e pela revisão de várias partes do texto final.

“I'll tell you this —

No eternal reward will forgive us now for wasting the dawn.”.

(Jim Morrison)

Resumo

Com a evolução das tecnologias relacionadas com a obtenção de dados de localização geográfica, o aumento de recursos disponíveis em dispositivos móveis e o forte crescimento do mercado destes aparelhos, serviços baseados na localização cada vez mais ganham espaço tanto em nível de pesquisa como de desenvolvimento de aplicações. Entendendo que este tipo de serviço agregaria qualidade ao nível de vida de uma população, se oferecido pelos órgãos públicos, este trabalho foi concebido com o objetivo de construir um modelo de arquitetura e desenvolver um protótipo de uma aplicação que trabalhe com estes serviços, ou seja, serviços voltados para o universo da mobilidade. Para isso, outro objetivo deste trabalho foi o de utilizar para o desenvolvimento desta aplicação, a plataforma para aparelhos móveis, Android, do Google, trabalhando alguns dos conceitos que esta tecnologia trás consigo. A aplicação VaDeOnibus, desenvolvida neste trabalho, consiste em uma consulta de linhas de ônibus que o usuário pode utilizar para ir do ponto onde se encontra, até um local de destino especificado. O prototótipo foi implementado utilizando o emulador de aparelho celular contido no kit de desenvolvimento do Android. Os testes realizados com a aplicação mostraram que o modelo proposto fornece resultados corretos para as consultas efetuadas. Portanto, com o desenvolvimento deste trabalho, foi possível constatar que, utilizando tecnologias apropriadas, é possível construir um software capaz de fornecer um serviço de valor para o usuário, de maneira que este não veja a complexidade embutida na aplicação. Além do fato de ter sido todo desenvolvido dentro do paradigma do software livre, não seria impeditivo de ser implantado pelas órgãos públicos, como um serviço de acesso livre pela população.

Palavras-chave: serviços baseados na localização. Android. Celular. Tecnologias móveis. Mobilidade.

Lista de figuras

Sumário

1. Introdução

1.1 - Motivação

Não é de hoje, nem há pouco tempo atrás, que a noção de mobilidade existe entre aqueles que trabalham, utilizam ou simplesmente consomem itens de tecnologia. Computadores deixaram de ser estações de trabalho fixas em uma mesa, para passarem a acompahar o usuário, como é o caso dos notebooks, onde quer que estes estejam. Porém foi somente com o advento dos celulares, que a tecnologia móvel passou a evoluir a passos largos. E esta evolução se deve muito ao surgimento de um mercado forte, ligado a estes dispositivos, muito mais do que em relação aos notebooks.

Com uma demanda de uma quantidade cada vez maior de usuários, muitos já acostumados com o uso de tecnologia e tantos outros novos neste meio, as tecnologias móveis deram um salto evolutivo muito grande recentemente. Aparelhos celulares deixaram de ser simplesmente itens voltados à telefonia, para se tornarem fornecedores de acesso a muitos tipos de informação.

Foi com o incremento tecnológico dos aparelhos móveis, como PDAs e celulares, que surgiu uma categoria de serviços que visam tirar vantagens da natureza destes dispositivos. Com os sistemas de localização globais acompanhando a tendência de evolução das tecnologias referidas acima, serviços que utilizem os dados da posição atual do aparelho ou do indivíduo para ofereçcer resultados de valor à vida dos usuários, ganham força nos ambientes de pesquisa e desenvolvimento de sistemas de informação.

Os chamados Serviços Baseados na Localização constituem o objeto principal de estudo deste trabalho, e a compreensão de que uma coletividade urbana (como a população de uma cidade por exemplo, a qual poderia obter vantagens ou facilidades, através da disponibilização de tais tipos de serviços pelos órgãos responsáveis pelo setor público) fundamenta aquilo que pode ser considerada a motivação para elaboração deste trabalho.

1.2 - Objetivo

A evolução, mencionada acima, das tecnologias móveis, foi tanto em nível dos dispositivos e suas capacidades, como em nível de software que vêm sendo desenvolvidos para usufruir da crescente gama de recursos nestes aparelhos. Outro foco de estudo deste

trabalho trata exatamente de uma destas novas tecnologias em nível de software, a plataforma para aparelhos celulares Android, um conjunto de recursos para desenvolvimento de aplicações móveis do Google.

Como será visto no capítulo dedicado ao assunto, a plataforma Android é implementada usando a linguagem de programação Java, e se encaixa no contexto das tecnologias de software livre, mais um aspecto que vem de encontro à motivação deste trabalho. Pois, sistemas desenvolvidos dentro da filosofia do software livre, costumam apresentar menores custos se comaprados com o universo das aplicações proprietárias, e é interesse deste trabalho que o software proposto aqui, seja voltado para o setor público, onde questões relacionadas a custos, são sempre de grande importância.

Portanto, o objetivo deste trabalho consiste na elaboração de um modelo e na implementação de um protótipo de uma aplicação que ofereça serviços baseados na localização, desenvolvida utilizando a plataforma Android.

Como será visto no capítulo referente ao modelo da aplicação, o serviço idealizado neste trabalho é uma consulta online de linhas de ônibus que o usuário pode utilizar para ir de sua posição atual, detectada pelo dispositivo móvel, até uma localização de destino, informada por ele à aplicação. A idéia é desenvolver esta aplicação para celular, onde o usuário seria apresentado a uma tela para informar o endereço de destino e efetuar a consulta, obtendo como resultado uma tela com a listagem das possíveis linhas e itinerários que ele poderia utilizar para se dirigir ao local de interesse. Por último, ao selecionar uma das linhas retornadas na listagem, seria apresentado um mapa ao usuário, mostrando sua posição atual, e a localização das paradas próximas onde ele pode pegar o ônibus.

1.3 - Organização do texto

A seguir será apresentada a organização da estrutura deste trabalho, visando oferecer uma breve descrição do assunto tratado em cada capítulo que compõe esta estrutura. A idéia básica que guiou a organização dos assuntos do texto foi simplesmente, abordar inicialmente os pontos mais teóricos, passando gradualmente aos itens mais práticos do desenvolvimento da proposta de aplicação que é o objetivo deste trabalho.

No capítulo 2, serão tratados, de maneira sucinta porém abrangente, os *location-based services* (LBS). Serão apresentadas as definições existentes para o que estes vêm a ser. Também serão mostradas as áreas de aplicações, existentes e potenciais, comumente associadas ao uso destes serviços. Serão descritas tecnologias que sustentam

tecnologicamente este tipo de serviço, como é o caso do sistema de posicionamento global, mais conhecido como GPS. Serão vistas outras características dos serviços baseados na localização, como as categorias em que podem ser divididos.

A plataforma Android é o tema do capítulo 3, onde poderá ser visto o que constitue esta tecnologia, a motivação por trás do desenvolvimento desta opção de API para a construção de aplicações móveis. Serão apresentadas características estruturais da plataforma, como ela se organiza para oferecer os recursos que disponibiliza. Uma breve comparação com outra tecnologia voltada para mobilidade, de grande abrangência no mercado, também pode ser encontrada neste capítulo. Ao final desta parte serão vistos aspectos que ilustram o modo como Android aborda os *location-based services* .

A parte principal do trabalho está contida no capítulo 4, o qual descreve a modelagem da aplicação, elucidando aspectos de arquitetura e escolhas que levaram à organização do software. Seguindo neste capítulo, são apresentados pontos de maior importância da implementação do protótipo que concretizam o modelo desenvolvido como proposta do trabalho. Por fim, são mostrados os resultados dos testes executados com o protótipo, utilizando dados pseudo-fictícios, afim de validar a corretude da solução modelada.

Por fim é colocado o capítulo de conclusão do texto, onde as impressões do autor são apresentadas. Uma análise final sobre o quanto o objetivo do trabalho foi alcançado, sobre o processo de desenvolvimento da aplicação VaDeOnibus, em nível teórico e prático. Também, nesta parte final são comentadas algumas potencialidades visualizadas para futuras versões do software desenvolvido.

2. Serviços baseados na localização

Neste capítulo serão abordados os serviços baseados na localização, com a intenção de apresentar a compreensão de um dos temas fundamentais deste trabalho. Na primeira parte serão vistas algumas definições existentes para os LBS, assim como o contexto que propiciou o surgimento desta tecnologia. Na segunda seção são trabalhadas as características de algumas áreas de aplicação de serviços desta natureza. Por fim, são colocadas as técnicas de localização utilizadas como alternativa ao GPS, que se servem das tecnologias celulares.

2.1 - O que são Location-Based Services (LBS)

O interesse na pesquisa e desenvolvimento de aplicações que façam uso de Serviços Baseados na Localização aumentou consideravelmente. Isto se deve aos recentes avanços da comunicação para dispositivos móveis e, além disto, ao potencial de mercado consumidor para tal serviço.

Porém ainda não existe uma definição consensual para o termo *Location-Based Services*, e isto vem do fato de que é uma tecnologia existente em áreas com enfoques e terminologias distintas, como são as áreas das telecomunicações e da computação.

Por exemplo, como apresentado em [KUPPER05], a GSM Association define LBS como serviços que usam a localização de um 'alvo' para adicionar valor ao serviço, onde este 'alvo' seria a entidade a ser localizada. Esta definição abstrata levanta a questão do que seria este valor adicional gerado pelo uso de serviços baseados na localização. A apresentação da localização de um alvo em um mapa ou a ativação automática de um serviço disparada pela aproximação a uma determinada localização são exemplos, colocados pela GSM Association, do que viria a ser este valor adicional obtido pelo uso de informação filtrada (no caso, a localização de um aparelho celular).

Outra entidade, a 3GPP, apresenta uma distinção interessante entre LBS e location services, que tb pode ser encontrada em [KUPPER05]. Ela coloca que estes últimos lidam exclusivamente com a obtenção da localização de pessoas ou objetos alvos e com a possibilidade de tornar disponível a atores externos o resultado desta localização. Um serviço de localização não implica o processamento do dado da localização no sentido de filtrar ou selecionar informações relacionadas a esta, possui como função somente a geração e distribuição do dado referente à localização em si. Portanto, location services podem ser

vistos como parte integrante de um serviço LBS ou um subserviço, uma vez que sem um serviço de localização, uma aplicação LBS deveria requisitar ao usuário a entrada da localização.

LBS podem ser também vistos como um subconjunto de um outro grupo de serviços, os chamados context-aware services. Estes, por sua vez, são definidos como sendo serviços que adaptam automaticamente seu comportamento, com relação a um ou mais parâmetros relacionados ao contexto em que o dispositivo alvo se encontra. Em [KUPPER05] é apresentada uma estrutura de context-aware services.

Outro ponto importante de se notar, quando se procura apresentar uma definição do que são *location-based services*, é a divisão destes em reativos e pró-ativos. Como apresentado por [KUPPER05] basicamente um LBS reativo é sempre ativado de forma explícita pelo usuário, funcionando dentro do modelo requisição/resposta e apresentando, por isso, um comportamento síncrono entre o serviço e o usuário. Aplicações LBS pró-ativas são aquelas que não são ativadas explicitamente pelo usuário, funcionando, assim, de maneira assíncrona.

O foco principal das modernas definições dos serviços baseados na localização reside na integração entre a obtenção da posição/localização do dispositivo móvel e o uso de informações que sejam pertinentes ao usuário, afim de gerar aplicações que forneçam serviços com maior valor para este.

2.2 – Áreas de aplicação

Existem diversos setores onde os serviços baseados na localização podem ser utilizados e desenvolvidos. Comumente são apresentados divididos em duas grandes áreas, como pode ser visto em [KUPPER05]. A primeira consiste no grupo formado pelos serviços de iniciativas públicas, onde aparecem os serviços ligados a setores governamentais (aqui também incluído o setor militar). Neste grupo, pode-se caracterizar, o foco está na redução de custos para o Estado e na melhoria da qualidade dos serviços oferecidos pelos órgãos de governo à população.

Na segunda área de atuação dos *location-based services* são agrupados os serviços de iniciativas comerciais, onde a principal motivação de investimento é obter um ganho na receita, através do aumento do tempo de uso/consumo do usuário, venda de informação de localização para terceiros e do fornecimento de serviços focados nas necessidades especiais dos usuários.

Como apresentado em [SCHILLER04], considera-se o GPS (Global Positioning System) a tecnologia precursora para obtenção de dados de localização, utilizando para isso a triangulação de satélites para obter uma posição em coordenadas geográficas de um ponto na Terra. Este sistema foi inicialmente desenvolvido e custeado pelo Departamento de Defesa dos Estados Unidos, para utilização militar. Porém, por volta da década de 80, sua utilização foi liberada para empresas ou instituições não militares nem ligadas ao governo, que apresentassem interesse em fomentar a pesquisa e o desenvolvimento no setor de tecnologia de satélites.

Com a possibilidade do uso do GPS por diversas instituições, passou a ocorrer a criação efetiva de serviços que se valessem da localização de um dispositivo para fornecer e agregar valor ao usuário ou ao negócio. A partir disto, foi possível o surgimento dos serviços baseados na localização.

Conforme mencionado, pode-se dividir os serviços baseados na localização em dois grandes grupos: um formado pelo setor da iniciativa pública e outro formado pelo setor da iniciativa privada, ou setor comercial.

Com a melhoria das tecnologias ligadas aos sistemas de comunicação e o consequente aumento da utilização e desenvolvimento de aplicações para estes sistemas - como pode ser visto hoje com o uso da Internet - entidades governamentais e instituições públicas vislumbraram a possibilidade que se apresentava. Oportunidade esta de utilizar sistemas ou aplicações como serviços à população, que fizessem uso destas novas tecnologias e potencialidades trazidas com elas, especialmente a utilização da localização do dispositivo do usuário.

Após apresentar este cenário [SCHILLER04] trás o exemplo do que ocorreu por volta do final dos anos 90, nos Estados Unidos, quando a Federal Communications Comission (FCC) (*explicação do que é*), expediu um mandato estabelecendo um prazo para que as operadoras de telefonia móvel se adaptassem ao que a FCC pretendia lançar como novo sistemas de chamadas 911 (chamadas de emergência nos EUA).

[SCHILLER04] apresenta ainda que estudos demonstraram que muitas das chamadas de emergência eram provenientes de telefones celulares e, em uma grande quantidade destas, havia dificuldades, por parte dos usuários, de informarem sua localização. Desta forma, a FCC estabeleceu a normativa de que as operadoras deveriam sempre obter a localização de onde a chamada provinha. E, em casos de emergência, estas deveriam conceder tal informação para as autoridades como parte do chamado.

Outro exemplo de utilização de *location-based services* por parte da iniciativa pública,

como pode ser visto em [KUPPER05], são os novos projetos para cobrança de pedágios nas estradas. Pelo fato de praças de pedágio tornarem-se gargalos de tráfego em determinadas épocas do ano, a cobrança automática apresenta-se como um grande atrativo para os controladores do sistema de pedágios.

Aqui algumas alternativas aparecem, como, por exemplo, utilizar um dispositivo acoplado ao veículo, operando com GPS e acessando algum sistema remoto a fim de passar as informações necessárias para efetuar o pagamento corretamente. Outra forma é usar estratégias que possibilitem o uso do aparelho celular de algum dos passageiros do veículo. Neste caso, se apresenta a oportunidade de formar uma parceria entre governo e operadoras, para que a cobrança da taxa possa ser incluída na conta telefônica do usuário que realizou o pagamento.

As oportunidades de parceria entre iniciativa pública e privada (como as operadoras de telefonia, por exemplo) podem ser consideradas práticas costumeiras. Em outras palavras, as demandas da iniciativa pública podem gerar oportunidades de negócio para a iniciativa privada no que diz respeito ao desenvolvimento de aplicações e sistemas.

Ao que se refere à tecnologia e ao uso das novidades eletrônicas, os fatores que impulsionaram a iniciativa privada a investir em sistemas que utilizem LBS são basicamente os mesmos; porém, o principal fator de interesse do setor comercial é, evidentemente, o aumento de receita com a oferta de maior número de serviços aos usuários.

O modelo de aplicação mais básico que utiliza LBS é aquele que consiste em uma consulta de algum tipo de informação ligada à posição do dispositivo móvel ou do usuário. Este tipo de aplicação pode ser esquematizado da seguinte maneira: 1) uma parte inicial, em que a posição atual é passada diretamente pelo usuário ou obtida através de um fornecedor de localização, como pelo uso do sistema GPS, por exemplo; 2) a consulta propriamente dita, que representa a aplicação em si, pois esta tem o objetivo de consultar algo para o usuário e, neste caso, podem ser inúmeros itens de interesse (como lojas, restaurantes, cinemas, etc.); e 3) a apresentação do resultado da consulta para o usuário.

Gerenciamento de frota e logística de entregas é outro ramo em que sistemas podem ser beneficiados com o uso de LBS, e que também pode ser encontrado em [KUPPER05] como exemplo de cenário de atuação deste tipo de serviço. Ao obter a localização dos veículos da frota, o controle central pode tomar decisões com relação a alterações de rotas e volume de fluxo, por exemplo, possibilitando um ganho sobre os custos oriundos de uma frota com um funcionamento mais estático. O mesmo raciocínio se aplica à logística de entregas, em que se supera o modelo coleta-no-ponto-A-entrega-no-ponto-B por um modelo

mais dinâmico, podendo-se otimizar coletas e entregas durante as movimentações dos veículos.

Um novo ramo potencialmente atrativo para os negócios é o chamado mobile marketing, o qual [KUPPER05] trás como nada mais do que a interação entre as agências de serviços e seus visados clientes através dos dispositivos móveis, especialmente aparelhos celulares. Este contato é geralmente realizado através de SMS, MMS ou WAP e possibilita a este tipo de marketing atingir um público mais selecionado, partindo de informações como áreas de interesse presentes em um perfil de cada usuário.

Com a utilização de LBS em aplicações de mobile marketing, o usuário pode receber propagandas não só de produtos de interesse, mas de produtos que podem ser encontrados em estabelecimentos próximos à localização do usuário, ou receber mensagens de marketing somente ligadas à localização do cliente. Isto levanta um aspecto muito importante deste tipo de abordagem. O sucesso do mobile marketing está fortemente ligado à questão da privacidade dos usuários, pois se os clientes se sentirem incomodados pelas campanhas que chegam pelos seus celulares, por exemplo, a abordagem não obterá muito sucesso. Além deste ponto de vista relativo ao desempenho das campanhas de marketing, há também a questão relativa ao direito à privacidade do usuário, sendo de suma importância o consentimento de quem possivelmente utilizaria o serviço de uma aplicação como esta.

Estes são alguns exemplos de contextos ou áreas em que o uso de *location-based services* pode gerar melhorias nos serviços oferecidos. Tanto para a iniciativa pública quanto para a iniciativa privada existe uma enorme gama de possibilidades de ação. Muitos estudos estão sendo realizados visando entender melhor as potencialidades do uso de LBS em sistemas, com diversos objetivos, dependendo dos interesses embutidos.

A questão da privacidade é um ponto de importante estudo, pois deverá ser fundamental para o sucesso deste tipo de serviço. Mas como foi dito no início deste capítulo, os serviços baseados na localização se apresentam como uma tecnologia muito interessante para aplicações futuras, uma vez que o mundo cada vez mais se torna consumidor e usuário de dispositivos e de tecnologias móveis.

2.3 – LBS utilizando tecnologias celulares

Além de utilizar o GPS como sistema para obtenção da posição geográfica do aparelho, os serviços baseados na localização, podem se valer de outra estratégia para descobrir esta informação, que seja, a infra-estrutura da rede das operadoras de telefonia

celular e ou a triangulação do sinal entre antenas de celulares.

Segundo consta no artigo sobre Mobile Phone Tracking (Monitoramento de Telefones Móveis) no site da Wikipedia, posicionamento móvel (Mobile Positioning) “é a tecnologia utilizada pelas companhias de telecomunicações para determinar onde um celular, e portanto, muito provavelemnte seu usuário, se encontra em dado momento”.

No caso da tecnologia GSM (Global System for Mobile Communications), este artigo coloca que para efetuar a localização de um aparelho celular, é utilizada multilateração pela potência do sinal próximo a antenas de celular. Multilateração é o processo de localização de um objeto através do cálculo preciso da diferença de tempo de chegada do sinal do aparelho até três ou mais antenas próximas.

Ainda segundo o artigo referido acima, os serviços baseados na localização utilizando as tecnologias celulares, podem ser divididos em: baseados na rede (Network Based), baseados no aparelho (Handset Based) e híbridos. Como aparece no artigo “técnicas baseadas-na-rede utilizam a infraestrutura da rede do provedor de serviço, para identificar a localização do aparelho celular. A vantagem desta abordagem é que pode ser implementada não-instrusivamente, sem afetar os aparelhos.” Também é salientado no texto do artigo, que a precisão de tal técnica, é relacionada com a densidade de antenas na região onde um usuário utiliza seu aparelho celular, implicando com isto, que a precisão é maior em áreas urbanas.

A técnica baseada-no-aparelho utiliza identificação da célula mais próxima do aparelho, através da comparação da força do sinal do dispositivo em relação a esta célula a qual ele se encontra na área, e as antenas das células vizinhas. Porém, é colocado no artigo que uma desvantagem desta técnica é o fato de ser necessária a instalação de um software cliente no dispositivo.

A terceira abordagem, chamada de híbrida pelo artigo, é aquela onde seriam utilizados elementos das duas abordagens mencionadas, network-based e handset-based, além do uso de GPS para optenção da posição no aparelho se o dispositivo conter o hardware relacionado. A estratégia híbrida trás como vantagens mais precisão, porém trás as desvantagens de ambas as abordagens que a constituem.

No caso da plataforma Android, utilizar GPS ou a infraestrutura das tecnologias celulares é indiferente no nível de desenvolvimento, pois a API para localização disponível no Android, trata ambas as abordagens como sendo provedores de posição. Inclusive, como será visto mais para frente neste trabalho, a API fornece maneiras de se construir aplicações que trabalhem com LBS, levando em conta o uso de ambas as estratégias, dependendo das circunstâncias. Podendo ser escolhida o provedor mais conveniente para a aplicação, levando

em conta alguns critérios importantes, como por exemplo, custo relacionado ao consumo de energia do aparelho, força do sinal de GPS, quantidade de antenas encontradas na região, provedor ligado ou desligado no dispositivo e também pode ser dada a opção de seleção do que utilizar para obter a localização, pelo próprio usuário

2.4 – Considerações finais

Neste capítulo foram vistos aspectos do que se constituem os *location-based services*, como definições e áreas de aplicação. O intuito foi tentar elucidar algumas características deste assunto, pois a aplicação desenvolvida neste trabalho tem como objetivo oferecer este tipo de serviço. Com certeza não foram abodados todos os aspectos dos LBS, pois é um campo grande em opções ainda mais com os avanços das tecnologias que propiciam seu desenvolvimento. A seguir no próximo capítulo serão vistas as características principais da plataforma Android, outro tema fundamental do trabalho, pois foi a tecnologia escolhida para desenvolver o cliente da aplicação desenvolvida.

3. Plataforma Android

Uma descrição de algumas características da platforma Android constitue o conteúdo deste capítulo. O objetivo é o de apresentar o suficiente da plataforma para que possam ser compreendidos melhor alguns elementos utilizados na modelagem da aplicação que constitui o objeto deste trabalho. As duas principais razões para utilizar a tecnologia Android neste trabalho são dois aspectos importantes da plataforma, uma é o fato de utilizar a linguagem de programação Java no nível de aplicação e a outra é por ser toda desenvolvida dentro da mentalidade de código-aberto.

3.1 - Introdução[1]

Até o momento presente, a maior parte dos dispositivos móveis, especialmente os aparelhos celulares, ainda são desenvolvidos utilizando softwares proprietários, tanto em nível de sistema operacional como diversos outros programas fundamentais para o funcionamento e operação do aparelho. Esta tendência pode ser justificada por questões tecnológicas, como performance e desempenho, uma vez que os recursos disponíveis em dispositivos deste porte são um dos maiores obstáculos para o desenvolvimento de aplicações. Também por fatores econômicos, e talvez de maneira mais decisiva, pois os aparelhos e os softwares neles contidos aparecem como produtos das empresas fabricantes do ramo.

Esta preferência por software proprietário, altamente vinculado com o hardware do aparelho, é a responsável pela existência de uma barreira para o desenvolvimento de aplicações por entidades não ligadas às empresas fabricantes. A plataforma Android vem exatamente no caminho oposto, uma vez que busca seguir na direção do desenvolvimento inserido na mentalidade do código aberto, possibilitando a construção de aplicações para dispositivos móveis em um nível mais colaborativo, aumentando potencialmente a gama de aplicações e projetos para o tipo de contexto que a plataforma se propõem, pois não trabalha com as restrições existentes em plataformas proprietárias.

Mais recentemente, com o advento de aparelhos móveis mais sofisticados, contendo maior riqueza de serviços e tecnologias - como GPS, Wi-Fi e Bluetooth, por exemplo - novos sistemas operacionais foram desenvolvidos a fim de possibilitar o uso destes serviços e

1 Esta seção foi toda elaborada baseada nas introduções de [MURPHY09], [HASHIMI09], [ABLESON08], [ROGERS09], [MEIER09] e do site Android Developers

facilitar a criação de aplicações por terceiros. Como por exemplo, Symbian (Nokia), iOS (iPhone), BlackBerry OS, Windows Mobile e Android. Porém ainda assim se faz necessária a utilização de APIs proprietárias, principalmente para acesso ao hardware, mantendo, efetivamente, a barreira que existe para a geração de softwares de autoria não vinculada com a empresa fabricante.

O uso de APIs proprietárias para acesso ao hardware do dispositivo implicava baixa portabilidade das aplicações. Java Micro Edition (JME) surgiu como uma alternativa para o desenvolvimento de software para aparelhos celulares e endereçava de maneira interessante esta questão da portabilidade, uma característica natural da linguagem Java. Porém, esta capacidade foi obtida ao preço de restrições de acesso ao hardware do dispositivo, o que, de uma certa maneira, fez com que o principal obstáculo ao desenvolvimento massivo por terceiros, para aparelhos celulares, se mantivesse.

O Android é uma plataforma voltada e desenvolvida dentro da filosofia de software livre, portanto, é a maior alternativa a todos os sistemas proprietários para celular. Sua arquitetura pode ser apresentada como sendo uma pilha de softwares de código aberto, contendo desde o sistema operacional, middleware, programas utilitários, até bibliotecas API para desenvolvimento de aplicações.

A plataforma foi concebida na linguagem Java; desta forma, já trabalha com a idéia de portabilidade de aplicações entre diferentes hardwares e está disponível gratuitamente e abertamente através de um kit de desenvolvimento de software (SDK). Como a linguagem utilizada pelo Android é Java, é esperado que as aplicações necessitem de uma máquina virtual para rodarem, porém a plataforma não utiliza uma máquina virtual Java (JVM) padrão. O Android utiliza uma máquina virtual otimizada para dispositivos portáteis que apresentam um potencial restrito em nível de recursos, chamada Dalvik VM.

3.2 – Dalvik Virtual Machine[2]

Performance e desempenho ainda são uma forte preocupação e, ao mesmo tempo, um grande obstáculo para o desenvolvimento de aplicações em dispositivos portáteis, devido ao fato de que ainda hoje, mesmo crescendo bastante em nível de hardware, os aparelhos celulares ainda operam com restrita capacidade de recursos, como memória e processador, se comparados a computadores desktop ou até mesmo notebooks. Porém, o kit de desenvolvimento do Android é composto por uma grande quantidade de APIs e bibliotecas

2 Seção baseada em [MEIER09] Cap.1 Introducing. - Delving into the Dalvik VM

para construção de aplicações em Java, sem que estas sejam otimizadas para dispositivos com estas características restritivas relacionadas a performance.

Tendo estas questões, relacionadas a performance dentro do contexto dos compactos dispositivos móveis, como determinantes para o bom funcionamento da plataforma neste nicho de aparelhos, o Google, ao desenvolver o projeto do Android com o intuito de tornar mais econômico o uso dos recursos não abundantes deste tipo de dispositivos, fez uma ampla revisão da máquina virtual Java para obter uma otimização que contemplasse estes pontos. O produto deste trabalho é a Dalvik Virtual Machine.

Dentro daquilo que foi otimizado, pode-se destacadar dois itens:

- uma ferramenta utilizada pela máquina virtual combina diversos arquivos .class gerados, em um único arquivo .dex. Informações duplicadas, como atributos strings, utilizados por mais de uma classe, são condensados em um só, diminuindo a exigência de espaço.
- a máquina virtual Dalvik é baseada em registradores ao invés da pilha, diminuindo o número de instruções em nível de montagem. Outro detalhe importante com relação à Dalvik VM é que não é utilizado bytecode Java, e sim bytecode próprio.

3.3 – Comparação entre Android e JME[3]

Como foi colocado anteriomente, o projeto da plataforma Android foi concebido com foco na questão do desenvolvimento de uma arquitetura baseada em código aberto e com o objetivo de elaborar um conjunto completo de programas para utilização e desenvolvimento de aplicações para dispositivos móveis. Desta forma, torna-se uma alternativa às tecnologias proprietárias existentes atualmente, utilizando Java para construção da plataforma.

Este fato suscita uma comparação entre a plataforma Android e a JME, pois a segunda é a solução para dispositivos móveis oferecida pela família de tecnologias Java e se propunha exatamente ao que se propõe o Android.

De acordo com Hashimi & Komatineni (2009), cabe comparação aos seguintes quesitos:

– *Configurações para Múltiplos Dispositivos*: Java ME considera duas classes de micro dispositivos e oferece soluções padronizadas e distintas para ambos. São as

3 Seção baseada em [MEIER09] Cap.1 Introducing. - Comparing Android and Java ME

configuracões CDC (Connected Device Configuration) e CLDC (Connected Limited Device Configuration), que diferem entre si pelo grau de restrição de recursos dos dispositivos para a qual cada uma foi desenvolvida para atender. O Android, por outro lado, é aplicado somente para uma classe, não sendo capaz de ser utilizado em dispositivos de muito baixo nível, pelo fato de as restrições serem muito grandes.

- *Facilidade de entendimento*: Como o Android aparelha-se na direção de somente uma classe de dispositivos móveis, se torna mais fácil de compreendê-la do que Java ME. Esta contém múltiplos modelos de interface com o usuário (UI) para cada configuracão, dependendo do que o dispositivo suporta: MIDlets, Xlets, AWT e Swing.
- *Responsividade*: A Máquina Virtual Dalvik é mais otimizada e mais responsiva se comparada à JVM padrão. Porém, a comparação correta seria com a Kilobyte Virtual Machine (KVM), que é a otimização utilizada pelo Java ME; esta apresenta maior desempenho em dispositivos de baixo nível, que possuem muito menos memória.
- *Compatibilidade com Java*: Devido ao uso da Dalvik VM, Android roda bytecode .dex ao invés de bytecode Java. Em nível de desenvolvimento isto não deve ser um grande problema enquanto Java for compilado em classes padrão Java. Somente interpretação em tempo de execução do bytecode Java não será possível, no Android, pois um arquivo .dex é uma compactação de diversos arquivos .class.
- *Suporte a JSE*: CDC é a principal configuração oferecida pelo Java ME, mas é uma otimização de Java SE, o que resulta numa diminuição de recursos em relação a plataforma JSE. Já o Android, como é voltado para um único grupo de dispositivos e trabalha diretamente com todas as APIs Java padrão, oferece maior suporte a Java SE.

3.4 - Overview da Arquitetura da Plataforma Android[4]

Como já foi mencionado anteriomente, a plataforma Android está disponível na forma de um conjunto completo de softwares desde o sistema operacional, incluindo também as

4 Seção baseada em [MEIER09] Cap.1 Introducing. - Understanding the Android Software Stack; [ABLESON08] Chapter 1: Targeting Android - 1..2 Stacking up Android

bibliotecas para uso de serviços e recursos, gerenciadores de componentes, até aplicações para utilização e operação do aparelho. Na realidade se não considerarmos que a plataforma foi desenvolvida visando o restrito contexto portátil, pode-se afirmar que a arquitetura da plataforma se apresenta como um completo ambiente de computação, desvinculado da magnitude da estrutura onde será implantado . É comum apresentar a arquitetura do Android estruturada como uma pilha de softwares, em que os serviços são oferecidos pelas camadas mais baixas para as camadas mais altas.

Na base desta arquitetura, ou na base da pilha, o Android utiliza um kernel Linux versão 2.6, que faz a abstração do nível de hardware para as camadas acima. É responsável por questões como:

- controle de drivers do dispositivo, como Wi-Fi e Bluetooth, por exemplo;
- acesso aos recursos do aparelho, como GPS, câmera e telas sensíveis ao toque;
- gerenciamento de energia;
- gerenciamento de processos, memória e sistema de arquivos.

Disposto sobre este kernel Linux fica um conjunto de bibliotecas de sistema. Estas disponibilizam uma série de recursos que viabilizam serviços para as aplicações usufruírem. Algumas destas bibliotecas são: OpenGL, para gráficos 3D; WebKit, que é uma biblioteca responsável por fornecer suporte a browser (é a mesma biblioteca que dá suporte ao Google Chrome e ao Safari da Apple); FreeType, responsável pelo suporte a fontes; Secure Sockets Layer (SSL) e SQLite, que é um banco de dados relacional de código aberto e independente, e Media, que são bibliotecas baseadas na PacketVideo's OpenCore, responsáveis pela gravação e reprodução de áudio e vídeo. Uma biblioteca chamada Surface Manager controla acesso ao sistema de apresentação visual, e suporta 2D e 3D.

O acesso a estas bibliotecas é obtido através da máquina virtual Dalvik. Como já foi dito neste capítulo, é uma máquina virtual otimizada para aparelhos com restrição de recursos, como é o caso dos dispositivos portáteis, mas que é compatível com as APIs contidas na edição Java SE. As principais APIs Java no Android incluem telefonia, localização, UI, provedores de conteúdo (dados) e gerenciadores de pacotes (instalação, por exemplo). Aplicações finais, como aplicações de usuário, são construídas sobre estas APIs.

3.5 - Componentes de Aplicações Android[5]

Na arquitetura da plataforma Android, cada aplicação é executada sobre um processo (gerenciado pelo kernel linux) distinto e utilizando uma exclusiva instância da máquina virtual Dalvik. Assim, gera um alto nível de isolamento de código entre as aplicações. Porém, isto não é uma característica obrigatória, mas sim, um padrão, existindo a possibilidade de, em situações onde se apresente a necessidade, compartilhar um processo entre diversas aplicações.

Uma aplicação Android é formada por componentes disponibilizados pelo pacote de desenvolvimento. Cada um destes componentes possui um papel dentro do modelo de aplicações da plataforma e, na construção de um software Android, devem ser arranjados de forma que seu relacionamento ofereça a solução para o problema que a aplicação será utilizada para resolver. Os principais componentes de uma aplicação Android são os quatro apresentados a seguir.

- *Activities:* correspondem a um item de interface visual da aplicação que faz o papel de relacionamento com o usuário seja requisitando ou apresentando dados ou simplesmente compondo a tela, como, por exemplo, um elemento gráfico. Podem ser instanciadas pelo usuário, quando este executa um programa, por outras Activities, ou, ainda, por serviços rodando no sistema. Uma aplicação pode ser composta por apenas uma ou por um conjunto. Se for o caso de conter diversas Activities, uma delas é designada como sendo a inicial e cada uma inicia a seguinte, quando for necessário trafegar entre elas.
- *Services:* não possuem interface visual para o usuário, pois são (exatamente como o nome diz) serviços utilizados por outros componentes. Normalmente executam em background, por tempo indeterminado. O exemplo mais comum do uso de um Service é uma aplicação para rodar arquivos mp3 no celular, pois após selecionar as músicas que se quer ouvir, a aplicação tem que continuar rodando, apesar de não estar ativa, liberando o uso de outros recursos do aparelho. Neste caso a aplicação utiliza Activities como interface com o usuário para que se possa selecionar uma lista de músicas, reproduzi-las, pará-las, etc. Quando se quer utilizar o celular para outras atividades, a aplicação inicia um Service que mantém a música e a lista em execução, porém, em background. Services expõem uma

5 Seção baseada em [ROGERS09] Cap 1.Getting to Know Android - Components of an Android Application; [MURPHY09] Cap. 1 The Big Picture - What Androids Are Made Of e no site Android Developers

interface para que outros componentes possam se conectar a eles, a fim de interagir com o que estes oferecem. Os Services utilizados por uma aplicação rodam na thread principal do programa, afim de não bloquear outras aplicações ou o aparelho como um todo.

- *Broadcast Receivers:* têm por função receber e reagir a anúncios ou requisições oriundas de qualquer parte do sistema. Estes anúncios podem ser originados do sistema operacional, como, por exemplo, um aviso de que a bateria está acabando; ou podem vir de outras aplicações como um download finalizado, uma foto capturada, etc. Podem, também, iniciar outras aplicações. São componentes que possibilitam a aplicações se tornarem sensíveis a eventos que o aparelho ou outras aplicações possam disparar.
- *Content Providers:* são componentes utilizados pelas aplicações para compartilhamento de dados. Basicamente um Content Provider é uma aplicação que fornece acesso a um conjunto de dados armazenados em qualquer forma disponível no dispositivo.

Outro componente fundamental dentro da organização da plataforma Android são os chamados *Intents.* Estes são utilizados para comunicação entre os demais componentes. São como mensagens entre eles, requisitando o uso de algum recurso. Dentro estes a instanciação de alguma Activity, a conexão com algum Service rodando em background, o acesso aos dados de um determinado Content Provider. Tais requisições são capturadas pelos Broadcast Receivers.

O modelo baseado em componentes permite uma característica muito interessante das aplicações no Android. Qualquer aplicação pode fazer uso de partes de outras aplicações, não precisando muitas vezes possuir ela mesma aquela parte contida em si. Ou seja, se já existe algum recurso de interface em alguma aplicação instalada no sistema, como uma barra de rolagem (por exemplo) e uma nova aplicação pretende fazer uso do mesmo recurso em sua interface visual, ela não precisará conter no código dela uma barra de rolagem; ela simplesmente usa a que já existe. Isto é outra solução dentro da preocupação em melhorar o desenpenho e otimizar o uso de memória e armazenamento num dispositivo portátil. Decorrente deste modelo, uma aplicação Android não possui um único ponto de entrada, mas sim, uma série de componentes que podem ser inicializados quando for preciso.

3.6 - *Location-based Services* no Android[6]

Location-based Services na plataforma Android estão disponíveis através da utilização dos recursos oferecidos pelas APIs encontradas nos pacotes android.location e com.google.android.maps. O primeiro, contendo os meios para obtenção da localização do aparelho e, o segundo, oferecendo meio para controle e apresentação de mapas. Aplicações desenvolvidas para dispositivos operando com Android, que tenham funcionalidades de LBS, deverão fazer um uso combinado das classes existentes nestes dois pacotes.

Uma das funcionalidades centrais básicas de uma aplicação que objetiva oferecer serviços baseados na localicação é a obtenção ou descoberta da posição geográfica atual do aparelho. É no pacote android.location que ficam as classes responsáveis por tonar isto possível. O serviço LocationManager, que pode ser apresentado como a principal classe do pacote de localização, fornece duas funcionalidades importantes: os mecanismos para obtenção da localicação geográfica do dispositivo e a capacidade de disparar alertas quando o aparelho se afasta ou se aproxima de uma determinada posição.

Para uma aplicação fazer uso de LBS é necessário obter acesso a um serviço LocationManager, informando a tecnologia ou a maneira pela qual este serviço irá obter a localicação do aparelho. Esta maneira é definida a partir de uma classe chamada LocationProvider, que nada mais é do que a tecnologia que será utilizada no processo de determinação da posição geográfica. Existem dois grandes tipos de LocationProviders: GPS e NETWORK. O primeiro obviamente utilizando o Sistema de Posicionamento Global e o segundo, fazendo uso do sistema de triangulação de antenas ou de redes Wi-Fi.

Aqui aparece uma das grandes vantagens do uso de LBS na plataforma Android, que é a possibildade de escolher a maneira pela qual será obtida a localização, podendo ser determinada por parâmetros importantes para a aplicação, como custo, consumo de energia e precisão.

Aplicações que trabalham fazendo proveito de *location-based services* utilizam coordenadas geográficas para expressar a localização do aparelho; porém, o mundo real não é composto somente de coordenadas geográficas, mas, mais comumente, uma localização é dada por um endereço urbano. Daí surge a necessidade de conversão entre os dois tipos de dados: coordenadas geográficas e endereços reais. No pacote de localização do Android, este serviço de tradução entre o tipo da informação de localização é função da classe Geocoder.

6 Seção baseada em [MURPHY09] Cap. 33; [ABLESON08] Cap. 11; [ROGERS09] Cap. 9; [MEIER09] Cap. 7 e no site Android Developers

Além do pacote de localização, quando se pretende trabalhar com LBS na plataforma Android, outro importante recurso é o pacote que fornece acesso aos serviços do Google Maps. Este pacote (com.google.android.maps) possibilita os meios para, por exemplo, apresentar mapas na tela, controlar a interação do usuário com o mapa e apresentar dados sobre estes. A classe central para utilizar os recursos de mapas é a MapView. É esta que oferece os meios para gerenciar uma aplicação de apresentação de mapas. Mas para poder utilizar a classe MapView obtendo acesso aos recursos do Google Maps é necessário obter uma chave disponibilizada pelo Google.

3.7 – Considerações finais

Efetivamente não foram abordados todos os aspectos da plataforma Android neste capítulo, até mesmo por que não se fazia necessário devido ao escopo do trabalho. O que se procurou trabalhar, foram alguns temas fundamentais para compreensão desta tecnologia, em um nível mais estrutural, e características relacionadas com o tema deste trabalho, e que serão importantes para o melhor entendimento de partes do modelo que será proposto no capítulo seguinte. Portanto, após tratar de uma maneira mais teórica os dois principais temas que fundamentam o objetivo deste trabalho, pode-se apresentar a proposta do modelo da aplicação LBS VaDeonibus, assunto do próximo capítulo.

4. Modelo da Aplicação

Este capítulo contém a apresentação da modelagem proposta para a aplicação VaDeOnibus, para consultas de linhas de ônibus. Após ter tratado os dois principais assuntos relacionados a idéia por trás desta aplicação, os serviços baseados na localização e a plataforma Android, tem-se condições de detalhar o modelo construído para resolver o probelma da consulta referida acima, a partir de um aparelho celular.

Na primeira parte será colocado o cenário que pode ser considerado como o problema a ser resolvido pelo software a ser desenvolvido. Em seguida inicia-se o detalhamento da modelagem propriamente dita. A apresentação do modelo foi estruturada da seguinte maneira, inicialmente é mostrado uma visão mais panorâmica da arquitetura concebida, procurando mostrar como o programa funcionaria como um todo, a partir daí serão mostrados com maiores detalhes cada uma das principais partes que constituem o sistema, nas quais o modelo pode ser dividido.

4.1 - Introdução[1]

De maneira geral o processo de desenvolvimento de software pode ser dividido em quatro grandes etapas, em primeiro lugar é necessário fazer a identificação de um problema a ser resolvido, aquilo que será o objetivo da aplicação resultante. Em um segundo momento, conhecendo o problema que se quer solucionar, inicia-se a fase de análise do problema e planejamento da solução, levantando os requisitos exigidos e as restrições impostas, afim de gerar um modelo da aplicação a partir do modelo pode-se iniciar a implementação da solução, o que constitue a terceira etapa do processo de desenvolvimento e por fim a validação da solução através da execução da implementação, através de testes sobre um protótipo.

4.2 - Identificação do Problema

Como já foi explicado anteriormente, este trabalho consiste em um estudo sobre os serviços baseados na localização (*LBS – Location-Based Services*), através do desenvolvimento de uma aplicação que possua como funcionalidade, um serviço dessa natureza. Portanto, levando em conta esta orientação fundamental do trabalho, o *problema*

1 Seção baseada no Cap. 1 de [ALBIN03]

escolhido para ser resolvido, pelo processo de desenvolvimento a ser descrito a seguir, necessariamente deveria apresentar características relacionadas com o uso da localização geográfica para a produção de informação relevante para o usuário.

Partindo desta premissa, e adicionando, aquilo que podemos colocar como uma restrição ao projeto da aplicação, ou seja, o fato de que este trabalho foca no grupo dos serviços relacionados ao setor público, o problema visualizado e escolhido para ser o objeto da aplicação foi o seguinte: como uma pessoa, utilizando seu celular, ou outro dispositivo móvel, poderia obter a informação de quais linhas de ônibus a levariam da posição onde se encontra em determinado momento, até um destino desejado, além da informação da localização das paradas mais próximas de sua posição, onde ela poderia pegar as linhas.

4.3 - Modelagem da Solução e Arquitetura da Aplicação

4.3.1 – Arquiterura de software[2]

Como em muitas áreas da computação, quando se trata de apresentar uma definição de algum conceito que ainda possui um grande volume de pesquisa associada, surgem várias versões, na maior parte convergentes nas idéias fundamentais, mas com variações decorrentes da importância que cada autor coloca para determinados detalhes.

No ramo da arquitetura de software não é diferente, mas é claramente visto em diversas definições observadas, que existe um conunto de conceitos em concordância que podem ser vistos como os alicerces da definição do que vem a ser arquitetura em sistemas de informação.

A idéia básica comum nas definições é a de que arquitetura de software consiste em uma abstração dos elementos que compõem o sistema, dos relacionamentos entre eles, e com entidades externas, na fase de operação do sistema.

Como elementos principais a maior parte das definições apontam duas entidades, os componentes e os conectores. Componentes são a abstração de um conjunto de funcionalidades que podem ser encapsuladas e que assumem algum tipo específico de responsabilidade dentro do sistema. E conectores são os elementos responsáveis pela coordenação, colaboração e comunicação entre os diversos componentes.

Na elaboração da arquitetura de um sistema de computação, um dos objetivos é a obtenção de um modelo do software, para validação da solução idealizada para o

2 Seção baseada no Cap. 1 de [ALBIN03]

problema identificado. Neste processo se define quais componentes farão parte do modelo, quais características podem ser encapsuladas em um componente afim de que este possa assumir determiandas responsabidilidades funcionais dentro do modelo. Outra definição importante diz respeito a maneira como estes componentes são relacionados, portanto, que conectores utilizar, visando atender aos requisitos e obedecer às restrições identificadas para a solução do problema.

Com o desenvolvimento da área de arquitetura de software, um conceito importante que surgiu foi o de padrões de arquitetura. A noção de que diversas soluções para diferentes problemas, apresentavam as mesmas características, no que diz respeito ao modelo desenvolvido para resolver os problemas, levaram à elaboração de padrões de organização de sistemas, ou seja, padrões que uma arquitetura poderia utilizar.

4.3.2 - Arquitetura geral da aplicação

A partir da descrição apresentada na seção 4.2, do problema identificado na primeira etapa do desenvolvimento da aplicação, podemos descrever de maneira bem ampla e resumida a funcionalidade principal do software VaDeOnibus com o auxílio do diagrama de casos de uso a seguir.

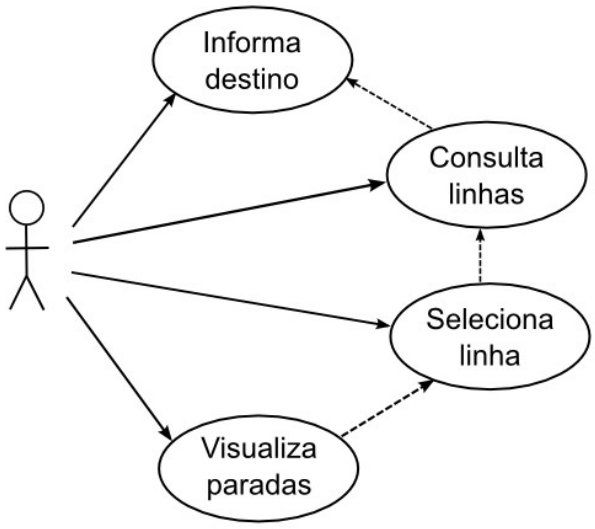

Figura 1: Caso de uso da aplicação VaDeOnibus

No diagrama acima está sendo apresentado, da uma maneira superficial, no que consiste o software desenvolvido para solucionar o problema identificado. Como pode ser visto no diagrama de casos de uso, o objetivo principal da aplicação é o de informar ao

usuário o conjunto de linhas de ônibus capazes de transportá-lo até um endereço destino informado, e, ao selecionar uma linha retornada da consulta, informar a localização das paradas onde ele pode pegar o ônibus.

A partir da observação dos casos de uso que constituem a funcionalidade da aplicação proposta, mesmo sem entrar nos detalhes que fazem parte da solução, podemos destacar algumas características que se apresentam como requisitos do projeto e que terão influência na escolha e organização da arquitetura do software.

Por exemplo, fica evidente que, se o programa consiste em uma consulta de linhas de ônibus, será necessária uma forma de armazenar estas informações e as que mais forem necessárias, a fim de fornecer algum resultado ao usuário. Portanto, uma pré-concepção que se pode chegar é a de que a aplicação fará uso de um banco de dados, para, de alguma forma, oferecer os dados das empresas, linhas, paradas e itinerários existentes no contexto no qual o usuário faz a consulta.

Porém, cabe lembrar que este trabalho trata de um software que oferece um serviço baseado na localização, e que está inserido no contexto das aplicações para dispositivos móveis. Assim, uma das primeiras restrições a se levar em conta no projeto da aplicação é a reduzida capacidade de memória e a exigência do bom gerenciamento da mesma, características desta classe de dispositivos, a fim de manter certos padrões de performance.

Considerando-se que existe o requisito funcional da necessidade do uso de um banco de dados e a restrição relacionada à reduzida capacidade de armazenamento de dados dos dispositivos móveis, chegou-se à conclusão de que uma abordagem adequada para a solução deste contexto seria pensar o banco de dados como sendo um recurso a ser acessado por algum tipo de serviço remoto. Com isso, coloca-se o dispositivo móvel na posição de um cliente que se comunica com esta base de dados, a fim de executar as consultas do usuário.

Chegando a esta decisão de projeto, obtem-se uma definição primordial da aplicação, a separação entre cliente e base de dados. Ligando estes, há um canal de comunicação remota. O meio de comunicação entre este cliente e a base de dados remota possui forte influência na organização do modelo e, consequentemente, na definição da arquitetura do software.

O modelo de organização de software que logo vem à mente, levando em conta as características apresentadas acima, é o clássico padrão de arquitetura de software *cliente-servidor*, que pode ser visto na Figura 2, mostrando a idéia de uma aplicação móvel.

Figura 2: Modelo cliente-servidor

Várias características da solução desenvolvida no software deste trabalho, tanto em nível de restrições quanto de requisitos, são contempladas pelas características do modelo cliente-servidor. Um exemplo é um servidor que pode ser acessado simultaneamente por diversos clientes, como seria o caso de um serviço oferecido pelo sistema de uma prefeitura municipal acessado pelas pessoas que se encontram dentro dos limites da cidade.

Um outro ponto interessante é a questão do controle da segurança dos dados contidos em um servidor, facilitado pelo fato de as informações ficarem centralizadas em um elemento do modelo. Outra vantagem desta arquitetura refere-se ao processo de manutenção da base de dados, como, por exemplo, a atualização das informações das linhas e itinerários. Caso a base estivesse contida no dispositivo móvel, ou seja, no cliente, qualquer atualização de dados seria mais complicada, pois ficararia sob responsabilidade do usuário.

Em sistemas baseados no modelo cliente-servidor, uma preocupação é sempre a de sobrecarga da rede e, portanto, da disponibilidade do serviço. Porém, como o sistema apresentado aqui consiste em uma consulta, sem requisições de alterações de dados por parte dos clientes, o tráfego é bem mais reduzido, e o tipo de transações mais simples e menos intrusivas.

Se faz necessário apresentar uma outra maneira de olhar para a arquitetura, levando-se em conta as características gerais descritas até aqui, o que pode ser colocado como uma outra visão do modelo da aplicação deste trabalho. Tal visão é trazida por [FIELDING00], em sua dissertação ao abordar o tema das arquiteturas de software baseados em rede. Ele traz uma distinção entre este tipo de arquitetura em relaçao à arquitetura de sistemas distribuídos. Defende a idéia de que, em contraste com os sistemas distribuídos, “os sistemas baseados em rede são aqueles capazes de operar através de uma rede, mas não necessariamente em um estilo que seja transparente para o usuário” (página da dissertação).

4.3.3 – Arquitetura do cliente

A porção cliente da aplicação para consulta de linhas de ônibus, proposta neste trabalho, tem papel fundamental dentro do contexto dos *location-based services*, pois está nela a responsabilidade pela obtenção dos dados das posições geográficas de origem e destino da busca.

Cumprindo com outras duas motivações deste trabalho, o cliente foi concebido como uma aplicação a ser utilizada em um aparelho celular, rodando a plataforma Android. Portanto, como será visto, o modelo do cliente se encaixa nos moldes dos padrões propostos pela arquitetura desta plataforma.

Então, de acordo com estes padrões, a aplicação cliente está organizada em uma pilha formada por três entidades, que, de fato, são consideradas componentes da plataforma Android, denominadas Activities, tendo cada uma responsabilidade dentro do projeto do cliente.

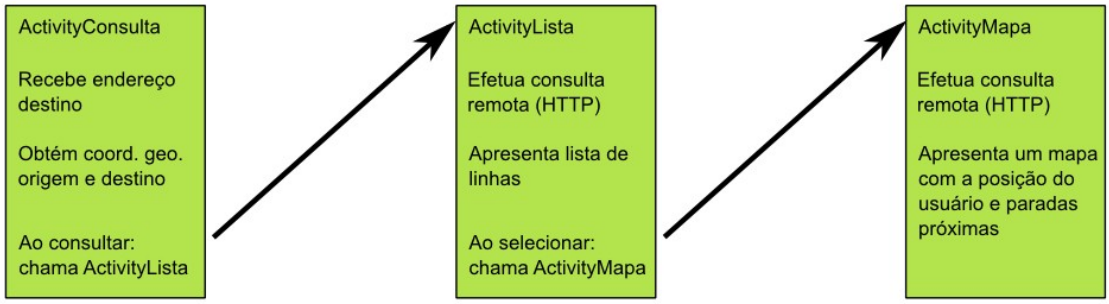

Figura 3: Organização do cliente

Na figura acima é apresentada a estrutura da organização da aplicação cliente, mostrando seus componentes e suas principais ações. Como foi descrito anteriomente, serviços baseados na localização têm por característica principal o fornecimento de algum resultado relevante ao usuário, utilizando, para isto, a informação da localização geográfica do mesmo.

Na aplicação VaDeOnibus, a informação final para o usuário - as linhas que levam da origem ao destino - virá da consulta ao banco de dados remoto. Já a funcionalidade relacionada com a localização do usuário foi colocada como atribuição do cliente. Devido a caraterística móvel do cliente é compreensível deixar como responsabilidade dele a obtenção dos dados da localização.

O primeiro componente do cliente é a Activity de consulta, onde o usuário poderá digitar o endereço para onde pretende ir e disparar a consulta, chamando a Activity de listagem. De fato, ao efetuar uma consulta, antes da chamada para a Activity de listagem ocorrem as definições das posições de origem e destino para poder passar estes dados como parâmetro desta chamada. Assim, será apresentado ao usuário o resultado da consulta, ou seja, a listagem com as linhas encontradas consideradas hábeis a levá-lo da origem ao destino. A Figura 4 mostra a tela correspondente à Activity de consulta:

Figura 4: Tela de consulta

Como foi colocado acima, ao clicar no botão “consultar” da Activity inicial do cliente, ocorre a chamada da Activity na qual será apresentada a listagem das linhas, sendo passados os dados das localizações de origem e destino obtidos pela Activity de consulta. Estes dados constituem-se nos parâmetros da consulta, pois na fase de inicialização da Activity de listagem, para montar a lista de linhas, é executada a consulta destas via web. Um exemplo da tela de listagem pode ser visto na Figura 5, abaixo:

Figura 5: Tela de listagem

Com relação à passagem de dados, uma idéia bem semelhante foi usada para a terceira Activity, em que será apresentado um mapa com a posição onde o usuário fez a consulta e as paradas de ônibus onde ele pode pegar a linha selecionada na lista. Neste momento, além da informação de origem e destino, são passados também os dados da linha e itinerário selecionados na listagem destas.

Portanto, na ação de selecionar um item da lista de linhas é disparada a terceira Activity, esta onde será apresentado um mapa ao usuário. Como no caso do componente da listagem, é na inicialização da Activity mapa que será executada a consulta para buscar a lista de paradas próximas da localização de origem do usuário, para serem visualizadas no mapa. Um caso mostrando a posição atual do cliente e paradas nas proximidades aparece na Figura 6, apresentada a seguir:

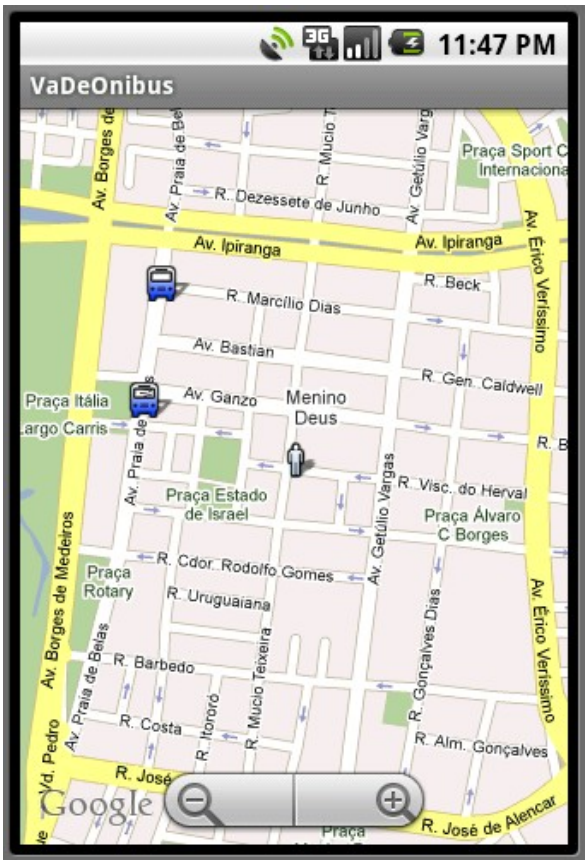

Figura 6: Tela de mapa

4.3.4 – Arquitetura do serviço de comunicação

Como foi colocado anteriormente, a aplicação proposta neste trabalho está organizada de forma a ser possível afirmar que seu modelo fundamental de arquitetura é o cliente-servidor. Neste modelo, um ponto de grande importância é o canal de comunicação entre o cliente e o servidor, tanto em nível da tecnologia utilizada pelo canal, quanto dos serviços disponibilizados.

Nesta seção, serão apresentadas as características do *link* entre o cliente e o servidor na aplicação VaDeOnibus, principalmente os serviços que são oferecidos pelo canal de comunicação, e como estes contribuem para encaixar a aplicação no escopo dos *location-based services*. Como a idéia do trabalho consiste em uma aplicação rodando em um aparelho móvel que faz uma consulta a uma base de dados remota, a tecnologia do meio de comunicação do modelo da aplicação terá que ser compatível com o universo wireless. De fato, o meio de comunicação utilizado no software aqui proposto é a *internet*, pois ela comporta os requisitos exigidos pelo tipo de serviço que é o cerne da aplicação VaDeOnibus.

Como está sendo apresentado até aqui, o software desenvolvido neste trabalho pode ser dividido em três partes: um cliente responsável por obter os dados relativos às posições geográficas da posição em que o usuário se encontra e do destino desejado, um serviço de consulta de linhas de ônibus, e uma base de dados com as informações sobre linhas, paradas e itinerários; compondo, assim, uma aplicação que oferece serviços baseados na localização.

Portanto, partindo do ponto de vista de que a aplicação se constitui em um serviço de consulta de linhas de ônibus, é possível caracterizar o modelo da aplicação dentro das arquiteturas orientadas a serviços ou *SOA – Service-Oriented Architecture*, que podem ser vistas como especialização do modelo cliente-servidor.

De acordo com o verbete da Wikipedia para Service-oriented architecture, SOA pode ser vista como uma proposta de organização de componentes que integram um modelo de software. Esta proposta parte do ponto de vista de que um software pode ser constituído por um conjunto de serviços que encapsulam, ou representam, as diversas regras de negócios existentes nele. Tal modelo propõe que estes serviços sejam auto-suficientes em sua definição e fracamente acoplados em suas funcionalidades. Além disso, pressupõe que, para uma arquitetura funcionar corretamente, o ponto chave é a relação entre estes. Desta forma, apresenta a noção de descrição de serviços, que é a maneira como SOA tenta possibilitar a relação entre estes, oferecendo uma estrutura que contém informações relevantes sobre as características destes.

Portanto, a relação entre os serviços, que acaba de ser apresentada como um elemento chave de uma arquitetura SOA, se dá com base na ciência de uns serviços com relação aos outros, a partir das descrições disponibilizadas. Esta descrição é formada pelo nome do serviço, endereço onde pode ser encontrado, parâmetros esperados e tipo do resultado retornado. A idéia, por fim, é a geração de uma aplicação como um conjunto de serviços reusáveis, desacoplados e distribuídos, sendo esta útlima a grande característica deste tipo de arquitetura. E foi após o advento das tecnologias de *web services* que SOA aumentou bastante sua abrangência.

As tecnologias de webservices possibilitaram o desenvolvimento de sistemas construídos de acordo com o modelo preconizado em SOA. Um exemplo é a abordagem proposta por Roy Fielding em sua dissertação [FIELDING00]: uma modelagem de sistemas que se vale fortemente do uso da internet, mais precisamente da world wide web, como meio onde são disponibilizados serviços, que ele denominou de Transferência de Estado Representacional (*REST - Representational State Transfer*). Mas, apesar de ser um modelo de

arquitetura estruturado nos conceitos da SOA, a arquitertura REST possui em suas características fundamentais alguns pontos que diferem bastante desta e trazem em si algumas vantagens.

A principal idéia proposta por Fielding, com a arquitetura REST, é a abstração da noção de que tudo aquilo que serviços oferecidos por sistemas distribuídos disponibilizam, pode ser visto como recursos acessíveis a partir de um identificador único. Outro conceito fundamental desta proposta é a utilização da world wide web como ambiente para disponibilização de serviços por parte de aplicações distribuídas. Para isso, mesmo que o modelo trazido por REST não defina o uso de um protocolo de aplicação específico, HTTP (Hypertext Transfer Protocol) é o mais utilizado de fato.

O protocolo HTTP serve aos propósitos da arquitetura REST, como será visto a seguir, mas uma observação interessante é o fato de o protocolo possuir uma característica básica, que é ser síncrono operando nos moldes de requisição e resposta, isto vai de encontro às características do modelo mais abrangente apresentado como a base da arquitetura da aplicação desenvolvida neste trabalho, ou seja, o modelo cliente-servidor. O uso do protocolo HTTP como protocolo de aplicação de uma arquitetura REST fornece a infraestrutura ideal para a proposta trazida por Fielding e, inclusive, ajuda a definir as cinco principais características de sua dissertação.

A questão de acesso a um recurso, utilizando um identificador único, compõe uma destas cinco características e é abordada como a questão de endereçamento. Fielding propõe o uso de URIs (Uniform Resource Identifier), através dos quais um recurso do sistema deve ser acessível. Toda requisição HTTP realizada deve ser composta pelo URI do recurso desejado. Utilizando um URI único para cada serviço disponível no sistema possibilita-se que todos os recursos sejam acessíveis via *hyperlinks*.

Outro ponto importante dentro da arquitetura REST é a utilização de uma interface restrita às operações que fazem parte do protocolo de aplicação. O objetivo disto é simplificar a lógica das comunicações entre pares e componentes do modelo de sistemas distribuídos. Com uma interface simplificada, mais enxuta, a complexidade das regras de negócio são deslocadas para os serviços, ou o que eles encapsulam. O conjunto sucinto de operações do protocolo HTTP servem bem muitos modelos de aplicações distribuídas, e, no caso da aplicação VaDeOnibus, basicamente é utilizando o método GET.

Uma característica que inclusive está colocada no nome da arquitetura proposta por Fielding é a idéia de que o mesmo recurso pode existir em mais de uma representação; ou seja, que um mesmo recurso pode ser disponibilizado de várias formas,

adequando às necessidades dos possíveis clientes que venham a acessar o serviço. Este é outro ponto que o protocolo HTTP fornece a infraestrutura adequada para implementação do conceito, pois com os tipos MIME (Multipurpose Internet Mail Extension) é possível disponibilizar um recurso nos formatos HTML, JSON e XML, dependendo do cliente que acessa o serviço.

Outros dois pontos da arquitetura REST são a comunicação sem contexto e o conceito de usar hipermídia como ferramenta-motor para alterar o estado da aplicação. O primeiro consiste no fato de, no modelo proposto, o servidor não manter informações de sessão do cliente, simplesmente tratando requisições a serviços que disponibilizam recursos. Esta é mais uma característica também básica do protocolo HTTP, diminuindo, assim, o overhead de informações trafegando na rede. Já o segundo ponto é a utilização de hiperlinks sendo retornados como informação e utilizados para alterar o estado da aplicação.

Toda esta apresentação da arquitetura proposta por Fielding foi colocada aqui, pois o modelo REST é o modelo utilizado na aplicação VaDeOnibus. Um diagrama mostrando a arquitetura REST simplificada na aplicação pode ser visto na Figura 7:

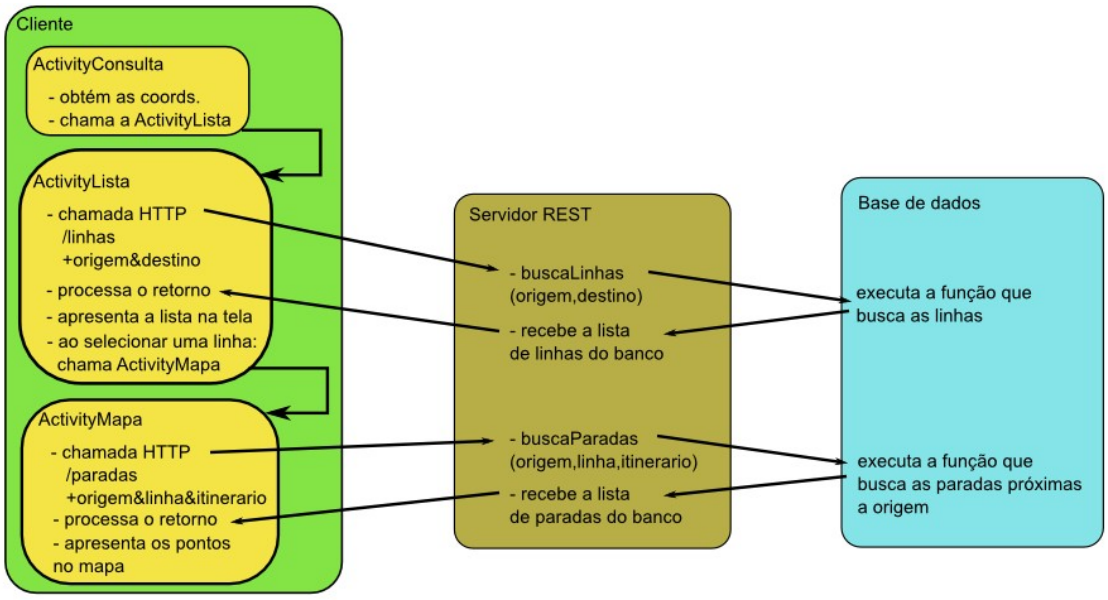

Figura 7: arquitetura REST da aplicação

Como pode ser visto, existem duas consultas disponibilizadas como serviços e ambas requerem como parâmetros os dados das localizações de origem e destino obtidas pela aplicação cliente. A primeira é uma consulta que retorna uma lista contendo as linhas de ônibus que o usuário pode usar para ir da sua posição atual até uma localização desejada,

informando o nome da linha, a empresa responsável por esta e o nome ou descrição do itinerário. A outra consulta é derivada desta primeira, pois, além das posições geográficas, espera como parâmetro também, o identificador da linha selecionada pelo usuário e o nome ou descrição do itinerário, a fim de retornar uma lista com as paradas existentes ao longo deste itinerário e que se encontram dentro de um raio de proximidade da posição do usuário.

4.3.5 – Modelo do banco de dados

Após apresentar o modelo do cliente e do canal de comunicação que fazem parte da aplicação VaDeOnibus, nesta seção serão colocados os pontos relacionados com a base de dados da aplicação.

Como está sendo abordada, a construção do modelo do software objeto deste trabalho se baseia na separação das partes, utilizando como critério as responsabidilidades que cada elemento possuirá no contexto geral para fornecimento do serviço proposto.

Seguindo esta abordagem, a responsabilidade sobre as operações e armazenamento dos dados relativos às linhas de ônibus, aos itinerários e às paradas foi colocado no contexto da base de dados. O modelo de dados desenvolvido para solucionar a questão da representação do universo das linhas, itinerários e paradas de ônibus, será apresentado em seguida, mas, antes, é importante salientar que uma característica chave da base de dados é o fato de ela disponibilizar o acesso às informações via funções.

O objetivo de um banco de dados, através de sua organização, é o de modelar uma abstração da realidade, representar um contexto do mundo real. No caso da aplicação VaDeOnibus, o contexto que precisa ser modelado pela base é o que contém o conjunto de linhas de ônibus, as empresas, as paradas por onde as linhas passam e os itinerários existentes.

Decorrente do fato de a aplicação VaDeOnibus procurar oferecer um serviço baseado na localização, de alguma forma a base deve ser capaz de armazenar dados relativos a posições geográficas, pois o algoritmo de busca irá usar este tipo de informação para encontrar as linhas que contemplem a necessidade da consulta efetuada pelo usuário.

Apesar da existência de sistemas de bancos de dados especialmente construídos para o trabalho com dados de geoprocessamento, como é o caso dos sistemas de informação geográfica (SIG ou *GIS - Geographic Information System)*[3], a base de dados desta aplicação, mesmo tendo a função de armazenar dados geográficos, não foi elaborada dentro

3 WIKIPEDIA. Geographic information system

destes moldes. A principal razão por trás da opção de não utilizar estes sistemas especializados em geoprocessamento é porque, no software de consulta de linhas de ônibus proposto aqui, as coordenadas geográficas foram representadas em graus decimais, e não em graus, minutos e segundos.

A estrutura do banco de dados da aplicação VaDeOnibus foi construída usando o modelo entidade-relacionamento, constituindo-se, assim, em uma base de dados do tipo relacional. Abaixo a Figura 8 e o modelo de tabelas e relacionamentos do banco de dados:

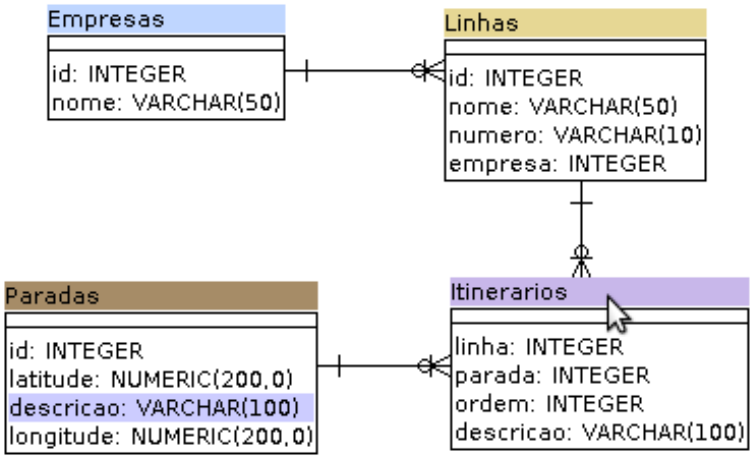

Figura 8: Modelo do banco de dados

Como pode ser visto no diagrama do modelo acima, o banco de dados é bastante simples, pois mapeia essencialmente aquilo que é necessário para disponibilizar resultados relevantes para a consulta de linhas de ônibus que passem em duas posições geográficas, em um determinado sentido.

A entidade vdo_empresa é a mais simples, pois serve somente como cadastro dos nomes das empresas que oferecem serviço de linhas de ônibus, e foi colocada no modelo para trazer mais veracidade aos resultados retornados pela consulta, não fazendo diferença na busca em si.

As tabelas vdo_linhas e vdo_paradas e o relacionamento entre elas formam a tríade fundamental para o algoritmo de busca das linhas desejadas pelo usuário. Vdo_linhas é a entidade que representa a informação principal da lista retornada por uma consulta e deverá conter as informações de nome, número e empresa a qual pertence. Para o funcionamento do algoritmo de identificação das linhas, os dados de posições geográficas são fundamentais e esta informação foi colocada na tabela vdo_paradas na base de dados. Para relacionar linhas e paradas por onde estas passam, foi criado o relacionamento vdo_itinerarios, que, além desta

ligação entre as duas principais entidades do modelo, também possui dados para definir o sentido das rotas que as linhas executam.

Como foi colocado antes, a base de dados da aplicação VaDeOnibus faz uso de funções para acesso aos dados, e as razões para utilização deste modelo serão abordadas a seguir. Primeiro cabe apresentar o que fazem estas funções disponibilizadas pelo banco de dados. O conjunto de funções existentes na base de dados da aplicação deste trabalho implementa o algoritmo de busca das linhas de ônibus, que é o cerne do software sendo proposto.

O algoritmo para consulta das linhas que levem o usuário até o destino desejado foi elaborado da seguinte maneira: inicialmente são recebidas as coordenadas geográficas de origem e destino como parâmetros. De posse dessas informações, o passo seguinte é descobrir o conjunto de paradas de ônibus que se encontram dentro de um raio de proximidade de cada uma das coordenadas passadas como parâmetros do algoritmo, tanto para origem quanto destino. Para encontrar estas paradas dentro de um raio determinado, foi utilizada a fórmula de Haversine[4], uma equação muito usada para navegação, que efetua o cálculo da distância entre dois pontos na superfície de uma esfera.

Após descobrir as paradas que ficam próximas das coordenadas relacionadas à consulta, o algoritmo descobre os itinerários que passam por estas paradas, levando em conta o sentido desejado. Através do levantamento dos itinerários, o algoritmo retorna como resultado a lista de linhas que cumprem estas rotas.

Pelo fato de o algoritmo de consulta das linhas estar localizado em um conjunto de funções do banco de dados, pode-se afirmar que a responsabidilidade deste elemento dentro do modelo geral da aplicação é de fornecer a lista de linhas desejada. Pois, desta maneira, pode ser compreendido que a base encapsula tanto os dados quanto o acesso a eles, ou seja, além de conter as informações sobre linhas e paradas, é o próprio banco que possui a inteligência para obter o resultado das consultas por linhas.

Este encapsulamento de dados e acesso a eles, por parte do banco, propicia uma interface mais concisa para ligação com o banco, sendo uma das razões para a opção de trabalhar com funções. A interface com a base de dados torna-se concisa, pois o que quer que estabeleça uma conexão com o banco a fim de efetuar uma consulta, não necessita possuir conhecimento da estrutura interna da base, bastando conhecer a função, seus parâmetros e o tipo do resultado retornado por ela.

O modelo do banco de dados concebido desta maneira procura seguir os

4WIKIPEDIA. Fórmula de Haversine

princípios das arquiteturas orientadas a serviços, utilizadas neste trabalho, ou seja, construindo a base de armazenamento como um elemento independente e buscando um forte desacoplamento de qualquer outra parte da aplicação.

5 - Implementação do Protótipo

Nesta seção serão abordados detalhes da implementação de alguns dos pontos de maior importância da aplicação VaDeOnibus. Uma vez que a modelagem do software foi realizada sobre a divisão entre o cliente móvel, o serviço de consulta via web e a base de dados, a mesma abordagem será utilizada para a apresentação da implementação. Portanto, primeiramente serão abordados detalhes do cliente, como por exemplo a obtenção dos dados relativos à posição do aparelho ou usuário e como o cliente faz a conexão para chamar o serviço de busca das linhas. Em seguida será mostrado como foi construída a aplicação que implementa a arquitetura REST e disponibiliza o serviço de consulta de linhas. Por fim serão mostradas as funções que o banco de dados oferece para efetuar as buscas das informações requisitadas.

5.1 – Obtenção dos dados de localização

Como foi colocado anteriormente neste trabalho, a parte da aplicação, que faz o papel do cliente, foi desenvolvida utilizando a plataforma Android. Uma das razões foi justamente a simplicidade oferecida por algumas APIs desta tecnologia, para trabalhar com recursos de localização e mapas.

A principal responsabidilidade atribuída à porção cliente foi a da obtenção das coordenadas de origem do usuário e de destino desejado, tanto no formato de latitude e longitude como de endereços urbanos, e a implementação desta função foi desenvolvida através do conjunto de classes oferecidas pelo pacote *android.location*, que faz parte da API disponibilizada pela plataforma Android.

A Activity que recebe como entrada o endereço desejado pelo usuário e dispara a consulta das linhas, foi implementada pela classe VDOConsulta.java, a porção mais significativa do conjunto de métodos que constituem esta classe, é aquela responsável pela obtenção dos dados das coordenadas geográficas de origem e destino. O trecho de código contido na Figura 9 mostra o método *obtendoLocalizacaoOrigem()*:

```
88  private VDOLocation obtendoLocalizacaoOrigem(){
89
90    LocationManager locationManager =
91      (LocationManager)getSystemService(Context.LOCATION_SERVICE);
92    LocationProvider locationProvider =
93      locationManager.getProvider(LocationManager.GPS_PROVIDER);
94
95    Location currentLocation =
96      locationManager.getLastKnownLocation(locationProvider.getName());
97
98    VDOLocation localizacaoOrigem = new VDOLocation();
99
100   if (currentLocation != null){
101     localizacaoOrigem.setLatitude(currentLocation.getLatitude());
102     localizacaoOrigem.setLongitude(currentLocation.getLongitude());
103   }
104
105   Geocoder geocoder = new Geocoder(this, Locale.getDefault());
106   try {
107     List<Address> addresses =
108     geocoder.getFromLocation(localizacaoOrigem.getLatitude(),
109                              localizacaoOrigem.getLongitude(), 1);
110     if (addresses.size()>0){
111        localizacaoOrigem.setAddress(addresses.get(0).getAddressLine(0));
112     }
113   } catch (IOException e) {
114     e.printStackTrace();
115   }
116
117   return localizacaoOrigem;
118 }
119
```

Figura 9: Código fonte para obtenção da localização de origem

O método *obtendoLocalizacaoOrigem()* é chamado assim que o usuário clica no botão "Consular" na tela inicial e é um bom exemplo do uso da api para localização do Android. Como pode ser visto, nas linhas 90-91 é obtida uma instância da classe LocationManager; esta classe é quem disponibiliza acesso aos serviços de localização do sistema. A seguir, nas linhas 92-93 através de um método do locationManager, é obtida uma instância de um LocationProvider, uma classe que encapsula um serviço de localização, como GPS ou Rede De Antenas.

Existem diversas maneiras de se obter os provedores de localização, além desta utilizada, mais direta, buscando um provedor por sua identificação textual no sistema, é possível ainda obter uma lista de provedores ativos no dispositivo, possibilitando uma escolha pela aplicação ou até mesmo pelo usuário. Outra maneira que pode ser usada é buscar um provedor usando filtros de busca, visando obter a melhor opção, seja por custo, seja por qualidade de sinal. Enfim, a plataforma fornece várias maneiras para se obter os locationProviders.

Nas linhas 95-96 são buscados os dados da posição obtida na última leitura do provedor de localização. De posse deste dados, é criada uma instância da classe

(VDOLocation.java) criada para a aplicação, que terá a função de encapsular as informações relacionadas a estes dados de localização, como latitude, longitude e endereço urbano da posição. Se o provedor de localização estiver ativo e retornar dados, estas informações, latitude e longitude (em graus decimais) são colocadas na instância da VDOLocation, como pode ser visto nas linhas 100-103. Por fim, das linhas 105 a 115 é utilizada a classe Geocoder, que faz o mapeamento entre coordenadas geográficas e endereço urbano, para armazenar, se possível, esta informação também na instância da classe VDOLocation.

```
120 private VDOLocation obtendoLocalizacaoDestino(){
121 
122     Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
123     VDOLocation localizacaoDestino = new VDOLocation();
124     String endereco = edTxtEndereco.getText().toString();
125     localizacaoDestino.setAddress(endereco);
126     try {
127         List<Address> addresses = geoCoder.getFromLocationName(endereco, 5);
128         localizacaoDestino.setLatitude(addresses.get(0).getLatitude());
129         localizacaoDestino.setLongitude(addresses.get(0).getLongitude());
130     } catch (IOException e) {
131         e.printStackTrace();
132     }
133     return localizacaoDestino;
134 }
```

Figura 10: Código fonte para obtenção da localização de destino

Acima, na Figura 10, pode ser vista a implementação da obtenção dos dados geográficos do destino que o usuário entrou no campo da tela da Actvity de consulta. Na linha 124 o conteúdo do campo existente na tela da activity é armazenado para ser colocado no objeto que irá guardar as informações da posição de destino. Logo após, através do uso da classe Geocoder, mencionada acima, é possível chegar nos dados de latitude e longitude do endereço de interesse do usuário, para também poder salvar estes dados no objeto VDOlocation retornado por este método.

Como pode ser visto, ambos os métodos para obtenção das posições de origem e destino que irão fazer parte da consulta de linhas de ônibus, tem como tipo de retorno a classe VDOLocation, que como foi colocado acima, tem como propósito encapsular as informações que caracterizam uma localização. Partindo da premissa de que latitudes e longitudes não são informações em um formato simples de assimilação por parte de um usuário comum, foi embutida na classe VDOLocation uma outra maneira de representar a posição que ela encapsula, o endereço urbano da coordenada.

Com isso, se fez necessário utilizar alguma estratégia de mapeamento entre os dois tipos de representação de uma localização. Como pode ser visto nas listagens acima contendo

os códigos-fonte dos métodos de obtenção das posições, a solução foi utilizar a classe Geocoder, contida no pacote android.location. Esta classe tem a capacidade de efetuar o mapeamento, nos dois sentidos, entre coordenadas geográficas (lat/lon) e endereços urbanos. É interessante observar que, o serviço de fato, de geocoding, não é implementado pela classe, e portanto deve ser fornecido para uso da mesma. No caso da plataforma Android, este serviço de suporte a geocoding é disponibilizado pela API do Google Maps, que é utilizada pela aplicação VaDeOnibus.

5.2 – REST: serviço de consulta das linhas

No capítulo referente à modelagem da aplicação proposta por este trabalho, foi mostrado que o serviço de consulta às linhas de ônibus foi construído sobre a arquitetura cliente-servidor, mais precisamente sobre a arquitetura REST proposta por [FIELDING00]. No protótipo da aplicação VaDeOnibus, a camada de serviço REST foi desenvolvida utilizando os recursos fornecidos pelo projeto Jersey. Este projeto consiste em uma implementação da especificação JSR-311: JAX-RS, esta sendo a API da linguagem de programação Java para RESTful Webservices. A seguir serão mostrados pontos chave da implementação deste serviço na aplicação.

A Api JAX-RS utiliza anotações da linguagem Java para simplificar o desenvolvimento de web services RESTful. Arquivos de classes Java devem conter anotações para definir recursos e as ações que podem ser executadas sobre estes recursos. Estas Anotações Jersey irão, em tempo de execução, gerar as classes e artefatos para o recurso, e então o conjunto destes serão organizados em um arquivo de aplicação web (web application archive - WAR). Os recursos são expostos aos clientes ao disponibilizar o arquivo WAR em um servidor de aplicação ou servidor web.

```
14 @Path("/linhas")
15 public class LinhasResourceServer {
16
17     @GET
18     @Produces({MediaType.APPLICATION_XML})
19     public List<Linha> buscaLinhas(@QueryParam("latOrigem")String latOrigem,
20                                    @QueryParam("lonOrigem")String lonOrigem,
21                                    @QueryParam("latDestino")String latDestino,
22                                    @QueryParam("lonDestino")String lonDestino){
23
24
25         LinhasDAO linhaDAO = new LinhasDAO();
26
27         List<Linha> linhas =
28         linhaDAO.consultaLinhas(latOrigem, lonOrigem, latDestino, lonDestino);
29
30         return linhas;
31     }
32
33 }
```

Figura 11: Código fonte do serviço REST

Na Figura 11 é apresentado o código da classe que disponibiliza a lista de linhas para um cliente que faça uma reguisição http para o servidor web onde a aplicação se encontra. Basicamente é uma simples classe Java, que chama outra, para buscar uma lista de objetos Java. O detalhe fundamental, como foi mencionado acima, é o uso das anotações *@Path*, *@GET*, *@QueryParam* e *@Produces*.

Anotando a classe com *@Path*, estamos dizendo para o servidor web, o local onde se encontram as operações sobre o recurso que esta classe representa. No caso do protótipo, *@Path("/linhas")* informa para o servidor, que requisições HTTP, que usem uma URI para este caminho, irão ter acesso aos serviços oferecidos pela classe. Estes serviços são na realidade métodos anotados com as operações HTTP. Na linha 17 da listagem acima, vemos que o método que busca as linhas, está anotado como *@GET*, isso faz com que um request HTTP GET por uma URI *http//host/VaDeOnibus/linhas* irá gerar a execução do nétodo buscaLinhas.

Outras duas anotações também colodadas no método se referem, primeiro, à representação do resultado retornado pelo método, e em segundo aos parâmetros que aguardados. De fato, após a consulta na base de dados que ocorre na execução deste método buscaLinhas() o mesmo retorna uma lista de objetos Linha, porém para melhor desacoplamento, seguindo os padrões sugeridos pelas arquiteturas SOA, para o request HTTP, o método é orientado a representar seu dado de retorno utilizando XML. Isto é determinado pelo uso da anotação *@Produces({MediaType.APPLICATION_XML})*. Por fim a anotação *@QueryParam* diz ao método que ele receberá seus parametros via URI. Abaixo a Figura 12 mostra a listagem do cliente que faz a requisição HTTP para buscar este recurso:

```
170 private List<VD0Linha> buscaLinhas(){
171
172     HttpHost host = new HttpHost("192.168.0.107", 8080, "http");
173     String parametros = "?lat0rigem="+vdoLocalizacao0rigem.getLatitude()
174                         +"&lon0rigem="+vdoLocalizacao0rigem.getLongitude()
175                         +"&latDestino="+vdoLocalizacaoDestino.getLatitude()
176                         +"&lonDestino="+vdoLocalizacaoDestino.getLongitude();
177
178     HttpClient client = new DefaultHttpClient();
179     String enderecoUrl = "/VaDe0nibusServer0.1/rest/linhas"+parametros;
180     System.out.println(enderecoUrl);
181     HttpGet get = new HttpGet(enderecoUrl);
182
183     ArrayList<VD0Linha> linhasEncontradas = new ArrayList<VD0Linha>();
184
185     try{
186         DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
187         DocumentBuilder builder = factory.newDocumentBuilder();
188         HttpEntity entity = client.execute(host, get).getEntity();
189
190         linhasEncontradas = extraiRestXmlToVD0LinhaList(builder, entity);
191
192     }catch(Exception e){
193         e.printStackTrace();
194     }
195
196     return linhasEncontradas;
197 }
```

Figura 12: Código fonte do cliente HTTP do serviço REST

Uma vez que a camada de serviço, disponibiliza o acesso aos recursos, ou às linhas de ônibus, através do protocolo HTTP, qualquer cliente que seja capaz de enviar uma requisição GET utilizando o mesmo protocolo, pode obter a listagem de itinerários. No caso do protótipo foi bastante simples a implementação desta requisição, pois a plataforma Android contém a API apache.http, do projeto Jakarta Apache, que oferece uma série de componentes que possibilitam o acesso a recursos através do protocolo HTTP.

Na linha 172 da listagem acima, é preparado um objeto que representa o host para o qual será enviada a requisição, e nas linhas 173 e 179 é montado o restante da URI de acesso à aplicação que contém o serviço, inclusive com a passagem dos parâmetros necessários para a consulta. Com estas duas informações, host e endereço da aplicação no servidor web, só resta criar um objeto HttpClient para efetuar a consulta através de uma requisição ao serviço REST, como pode ser visto na linha 188. Nesta mesma linha, o objeto retornado pela consulta HTTP, é definido, ou pelo menos sua forma, pela anotação descrita acima *@Produces*, no método que executa a busca das linhas. Neste caso, um documento XML, que será devidamente processado, utilizando recursos oferecidos pela linguagem Java, e que irá ser tranfosmado em uma listagem de linhas para serem apresentadas na tela da aplicação.

5.3 – Busca das linhas pela função na base de dados

Na seção sobre a modelagem da aplicação, foi bem explicado a distribuição de responsabilidades das partes que compõem o sistema VaDeOnibus, de como o banco de dados foi construído encapsulando tanto dados quanto a lógica para obtenção das listagens que a aplicação apresenta para o usuário, a partir destes dados. A camada comentada acima, que implementa o serviço REST para acesso via web, obtém a lista de linhas da consulta do cliente, a partir de uma conexão com o banco de dados e execução da função *busca_linhas()*. Abaixo a Figura 13 mostra a listagem desta função:

```
4
5  CREATE OR REPLACE FUNCTION busca_linhas(lat1 numeric, long1 numeric,
6                                          lat2 numeric, long2 numeric)
7    RETURNS SETOF record AS
8  $BODY$
9  DECLARE
10     rec record;
11 BEGIN
12     FOR rec IN (
13
14         select distinct origem.vdo_lin_id, origem.vdo_lin_numero,
15                         origem.vdo_lin_nome, origem.vdo_iti_descricao
16         from (select * from busca_linhas_origem(lat1, long1) as (/**/))
17              as origem,
18              (select * from busca_linhas_destino(lat2, long2) as (/**/))
19              as destino
20         where origem.vdo_lin_id = destino.vdo_lin_id
21           and origem.vdo_iti_ordem < destino.vdo_iti_ordem
22           and origem.vdo_iti_descricao = destino.vdo_iti_descricao
23
24               ) LOOP
25        return next rec;
26     end loop;
27 END;
28 $BODY$
29   LANGUAGE 'plpgsql' VOLATILE
30   COST 100
31   ROWS 1000;
32 ALTER FUNCTION busca_linhas(numeric, numeric, numeric, numeric)
33 OWNER TO postgres;
34
```

Figura 13: Código fonte da função que faz a busca de linhas

Como pode ser visto, a consulta propriamente dita utiliza recursos de PL/SQL, do banco PostgreSQL. Resumidamente o que esta função faz é, buscar informações de linhas e seus itinerários, que passem por paradas próximas das posições passadas como parâmetro para a funçao. De posse dessas informações a query filtra os resultados que preenchem as condições de sentido da origem para o destino. Nas linhas 16 e 18 aparecem as chamadas para as funções que buscam as informações das linhas por proximidade de paradas na região

das coordenadas geográficas que formam a consulta. Das linhas 20 a 22 é feita a filtragem dos resultados obtidos por estas funções de proximidade.

Um ponto fundamental do algoritmo de busca das linhas de ônibus é o cáculo de proximidade das paradas em relação às localizações passadas como parâmetros para a consulta. Este cálculo consiste na realidade, na definição da distância entre dois pontos geográficos. Portanto, para este ponto do algoritmo de busca das linhas foi desenvolvida uma função de cáculo de proximidade entre dois pontos geográficos, implementada utilizando a fórmula de Haversine. Esta fórmula é uma importante equação usada em navegação, fornecendo distâncias entre dois pontos de uma esfera a partir de suas latitudes e longitudes. O fato de a fórmula utilizar funções trigonométricas para encontrar a distância entre duas coordenadas geográficas foi o que definiu a necessidade de utilizar funções em PL/SQL para implementar o algoritmo de consulta das linhas de ônibus neste trabalho.

```
1  CREATE OR REPLACE FUNCTION paradas_proximas(lat numeric, lon numeric)
2    RETURNS SETOF record AS
3  $BODY$
4  DECLARE
5      _distance numeric;
6      _parada vdo_paradas%ROWTYPE;
7  BEGIN
8
9      FOR _parada IN select * from vdo_paradas LOOP
10         select *
11         into _distance
12         from distance(lat, lon,
13                       _parada.vdo_par_latitude, _parada.vdo_par_longitude);
14         if _distance < 1000 then
15            return next _parada;
16         end if;
17     end loop;
18
19 END;
20 $BODY$
21   LANGUAGE 'plpgsql' VOLATILE
22   COST 100
23   ROWS 1000;
24 ALTER FUNCTION paradas_proximas(numeric, numeric) OWNER TO postgres;
```

Figura 14: Código fonte da função que busca paradas póximas de uma posição

Na Figura 14, podemos ver a função *paradas-proximas(lat,long)*, utilizada pelas funções chamadas nas linhas 16 e 18 da listagem apresentada pela Figura 13, que recebe os dados de uma localização, por exemplo, da origem ou do destino do usuário, e busca no banco, todas as paradas que se encontram dentro um raio definido de distância, desta posição. Para o protótipo, foi definido um raio fixo, mas esta informação é possível de ser colocada como opção do usuário.

5.4 - Validação da Implementação – Testes do protótipo

Para a implementação do protótipo do software VaDeOnibus, foi utilizado o SDK fornecido pelo Google, para o desenvolvimento de aplicações na plataforma Android. Foi no emulador de aparelhos celulares (que rodam o sistema operacional Android) distribuído como parte deste kit, que foi executado o protótipo da aplicação VaDeOnibus e aqui serão apresentados os resultados dos testes elaborados para validar o algoritmo de consulta de linhas. Como foi utilizado um software emulador, é importante salientar que os dados da localização atual do aparelho tiveram que ser injetados no emulador. Informações de teste foram criadas para a execução das consultas e foram cadastrados no banco de dados algumas linhas de ônibus, assim como algumas empresas, paradas e alguns itinerários. Os casos de teste foram organizados em três grupos, divididos pela posição de origem do momento da consulta, uma localização relativa a uma residência, outra localização representando um local de trabalho e outra localização referindo a posição do Instituto de Informática no campus do vale da Ufrgs. A seguir são apresentados os resultados obtidos.

5.4.1 – Posição de origem: Casa

Os testes apresentados a seguir simulam casos onde o aparelho, ou o usuário, se encontra na posição localizada na Rua Visconde do Herval, 437, bairro Menino Deus, na cidade de Porto Alegre (localização escolhida por ser o endereço da casa do autor deste trabalho), e que será identificada como Residência. As coordenadas geográficas desta localização são 30° 3' 11.46" S de latitude e 51° 13' 28.01" O de longitude, porém, na aplicação VaDeOnibus e para usar no emulador da plataforma Android, onde o teste foi executado, estas coordenadas foram convertidas em graus decimais, portanto a latitude usada foi -30.053183333333333 e a longitude -51.224447222222224.

Dentro do conjunto de dados usados para validar a implementação, a lista de linhas que passam na região próxima da posição onde o usuário se encontra neste caso são mostradas na Figura 15:

	id integer	numero character varying	nome character varying
1	1	T1D	T1 Direta
2	2	T1	Transversal 1
3	5	T5	Transversal 5
4	6	T7	Nilo Praia de Belas
5	11	177	Menino Deus
6	12	149	Icarai
7	13	178	Praia de Belas
8	14	375	Agronomia

Figura 15: Linhas que passam próximas a posição Residência

No primeiro teste, foi feita a consulta passando como destino o endereço da Empresa Human Mobile (empresa onde o autor trabalha), localizada na rua Félix da Cunha, 836, bairro Moinhos de Vento, também na cidade de Porto Alegre, como pode ser visto na Figura 16, onde é mostrada a execução da consulta na aplicação:

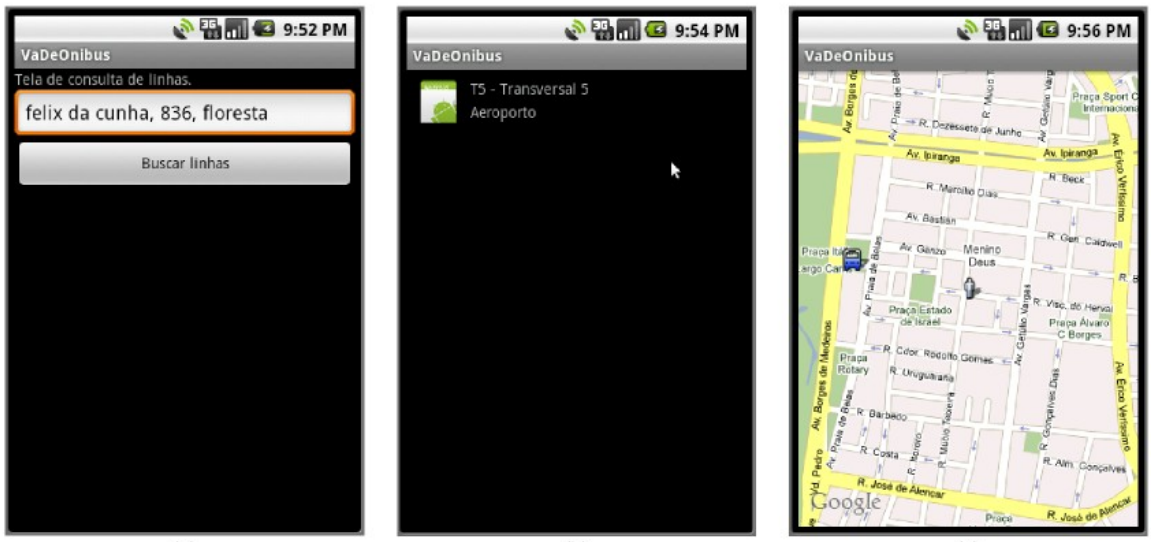

Figura 16: Caso de teste Residência - Empresa

Como pode ser visto na imagem acima, na primeira tela (a), é informado o endereço desejado, a seguir, na imagem (b) , é apresentada a lista de linhas detectadas como viáveis de levar o usuário ao destino de interesse, e de fato, é a única linha que liga diretamente os dois pontos na realidade, cruzamentos de linhas são possíveis, mas no protótipo da aplicação, a capacidade de cruzar linhas não foi modelada. Na imagem (c), é

apresentado um mapa onde podem ser identificadas as posições do usuário e da, ou no caso de mais, das paradas onde ele pode pegar a linha selecionada na listagem da tela anterior.

Na imagem a seguir (Figura 17), é mostrado a simulação da consulta usando como destino o endereço do Campus do Vale da UFRGS:

Figura 17: Caso de teste Residência - Universidade

Neste último teste usando como origem a posição Residência, o objetivo era verificar o comportamento da aplicação no caso de, efetivamente serem encontradas várias linhas, como pode ser visto nas Figuras 18 e 19, a seguir, onde é realizado o teste buscando linhas que levem até uma localização no centro da cidade, especificamente neste exmplo, foi colocada a praça da Alfãndega em Porto Alegre. É interessante observar as telas de mapa, para verificar que dependendo da linha selecionada, a aplicação está corretamente mostrando as posições das paradas relacionadas, como pode ser visto na imagem (c) onde aparecem as paradas por onde passa a linha Agronomia Centro e Bairro, na imagem (d) onde podem ser vistas as paradas das linhas Menino Deus e Icaraí e na imagem (e) que apresenta as paradas para a linha Praia de Belas Centro.

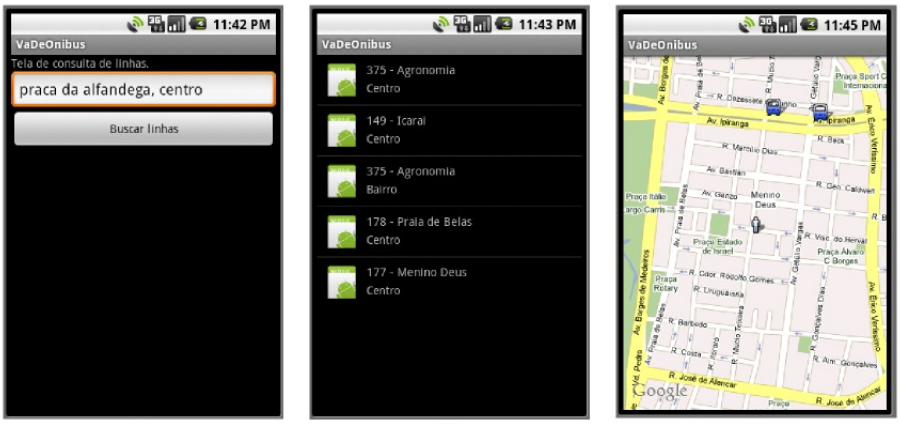

Figura 18: Caso de teste Residência - Centro, selecionando a linha Agronomia

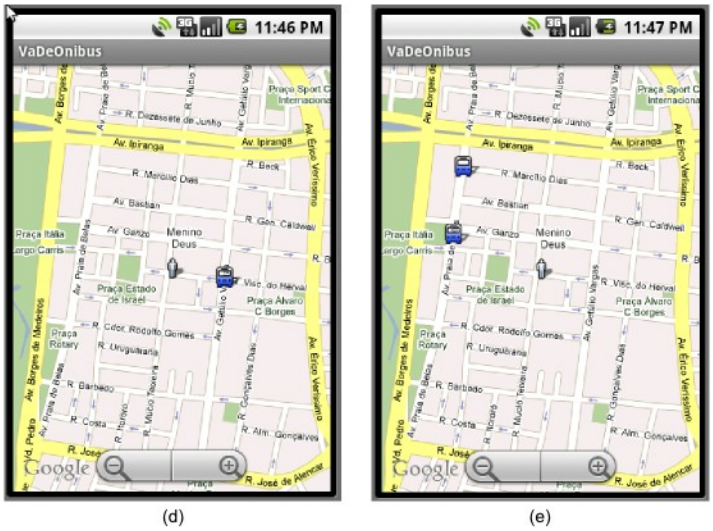

Figura 19: Caso de teste Residência – Centro, selecionandoas linhas 177, 149 e 178

5.4.2 – Posição de origem: Empresa (Human Mobile)

A segunda posição usada como posição atual, ou seja, posição de origem para os testes, é a localização da empresa Human Mobile, já apresentada acima, no caso onde foi usada como endereço de destino, que se encontra a 30° 1' 12.95" S de latitude e 51° 12' 6.62" O de longitude. Esta posição será denominada de Empresa, e na Figura 20, a seguir, pode ser visto a execução da consulta, visando descobrir as linhas que levam o usuário da Empresa até a Residência:

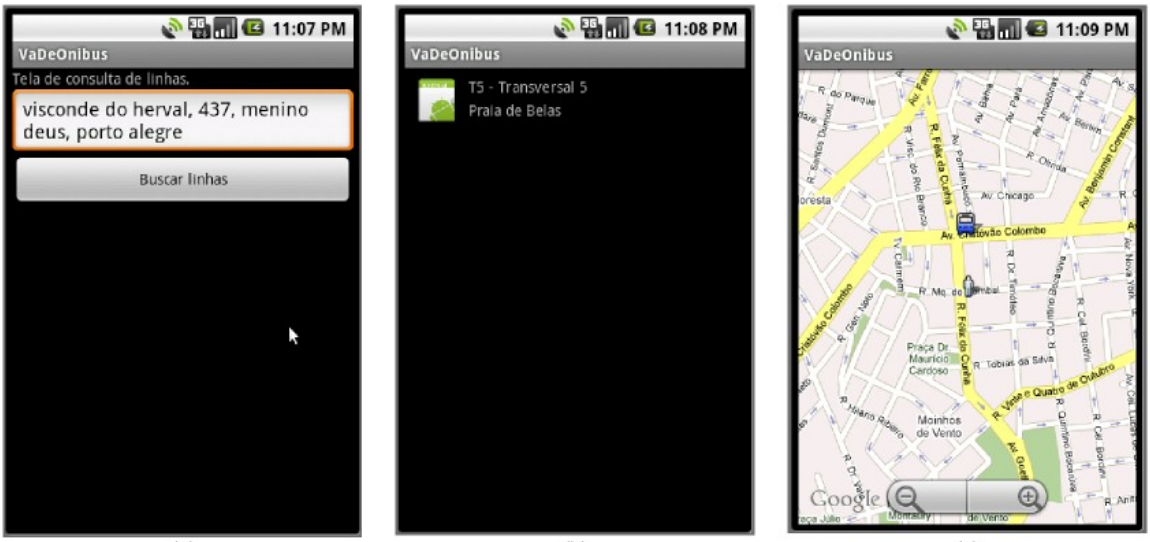

Figura 20: Caso de teste Empresa - Residência

A listagem abaixo, na Figura 21, mostra o conjunto de linhas de ônibus que passam próximas a localização Empresa, usadas paras os testes:

	id integer	numero character varying	nome character varying
1	5	T5	Transversal 5
2	7	T8	Campus Farrapos

Figura 21: Linhas que passam próximas a posição Empresa

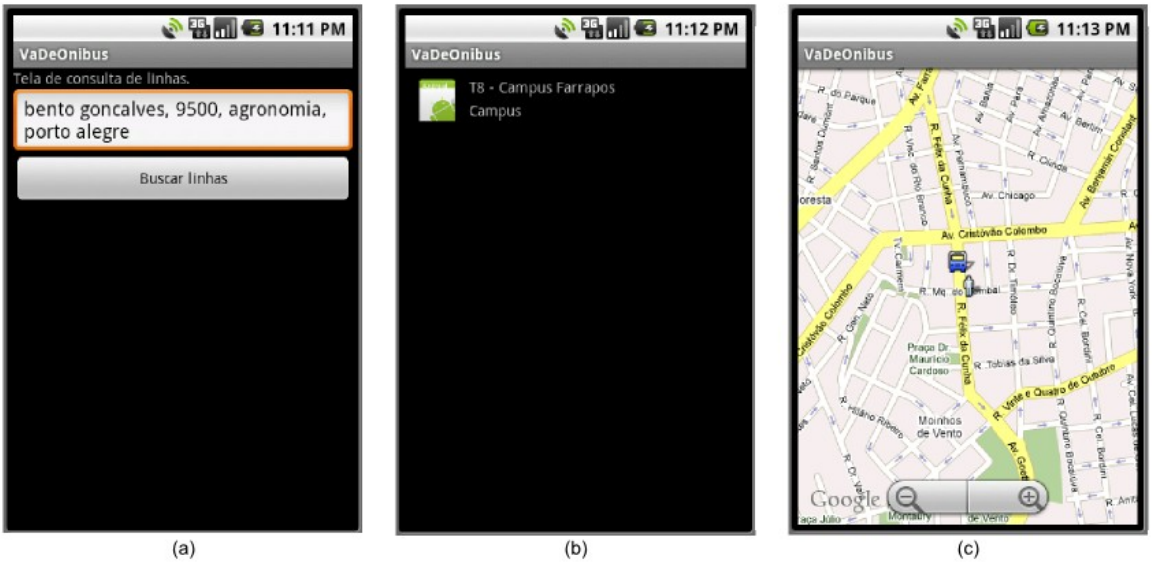

Figura 22: Caso de teste Empresa - Universidade

A Figura 22, acima, apresenta outro teste usando a localzação de origem Empresa, mas desta vez, consulta linhas que possam ser usadas para ir até a posição Universidade, para exemplificar aqui esta posição, foi usado o endereço do Campus do Vale, da Ufrgs, na Agronomia.

5.4.3 – Posição de origem: Ufrgs – Campus do Vale

No último conjunto de testes, a posição usada como origem, foi a localização do Campus do Vale da UFRGS, mais precisamente avenida Bento Gonçalvez, 9550, Agronomia, a 30° 4' 5.51" S de latitude e a 51° 7' 13.61" O de longitude, que irá ser referida aqui como Universidade. Nas Figuras 23 e 24 são apresentados, respectivamente, os casos de consultas com destino na localização Empresa, Residência e uma posição no centro da cidade:

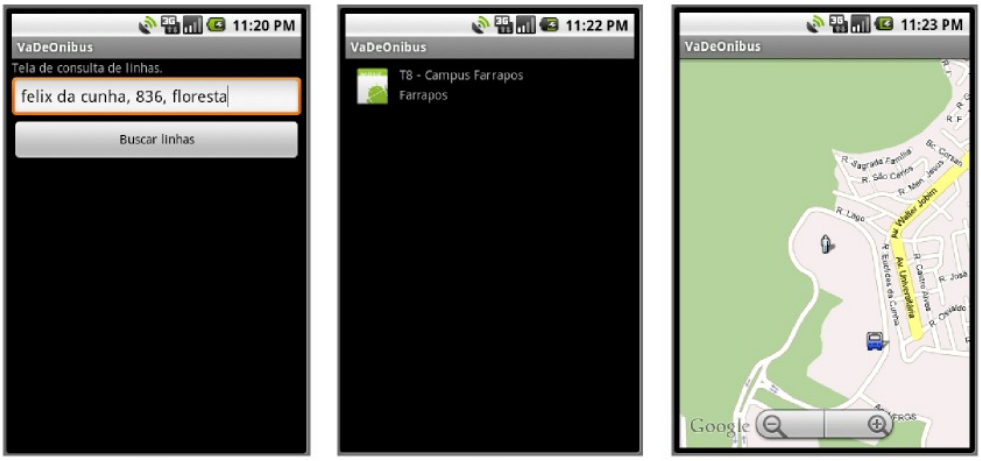

Figura 23: Caso de teste Universidade - Empresa

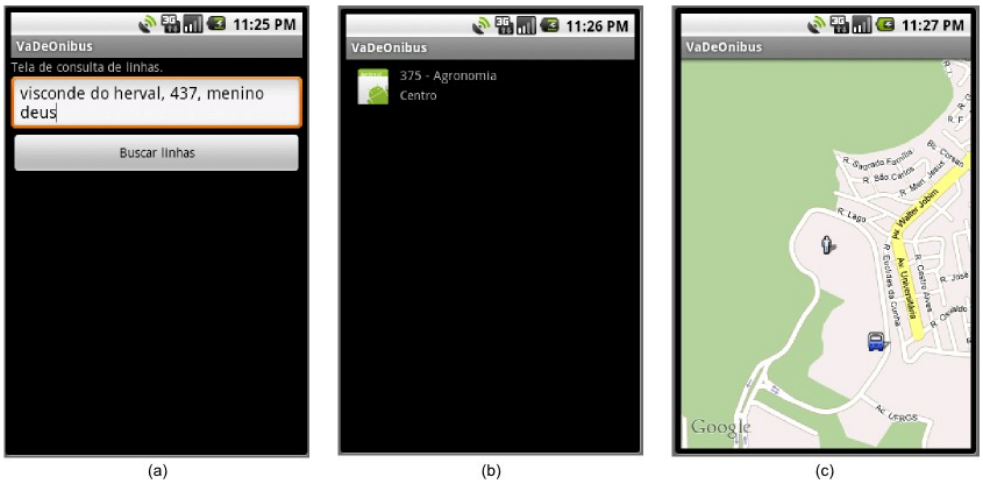

Figura 24: Caso de teste Universidade - Residência

6. Conclusão

6.1 – Análise geral do trabalho

Antes de fazer as colocações sobre os resultados obtidos neste trabalho, com relação ao objetivo proposto, vale a pena, retomar a motivação inicial para escolha do assunto aqui tratado. Como já foi mencionado anteriormente, vivemos em um mundo onde a tecnologia, principalmente, a parte ligada à tecnologia da informação, ruma a passos largos para o universo da mobilidade. Cada vez mais, as pessoas, ou de uma maneira mais precisa, os usuários de dispositivos eletrônicos, querem o máximo de informação e serviços onde quer que estejam.

Também foi visto que dentro desta tendência surgiram os serviços baseados na localização, que se constituem em uma abordagem que se vale da idéia de usar a mobilidade de maneira a oferecer informação que agregue valor ao usuário. A partir da compreensão de que a tecnologia deve servir para trazer melhoria de vida às pessoas, um dos fatores de maior importância dentro da motivação deste trabalho, é a idéia de que este tipo de serviço seria de grande utilidade se oferecido por órgãos públicos, ocasionando um aumento do nível de interação do cidadão com sua comunidade. Pois, por exemplo, uma pessoa não seria obrigada a se direcionar até um estabelecimento para obter acesso a algum serviço que uma prefeitura forneça; utilizando serviços baseados na localização, onde quer que esta pessoa se encontre ela está conectada a rede de serviços disponibilizados pela administração de sua cidade.

A partir desta motivação, o objetivo traçado para este trabalho, se tornou o de elaborar a modelagem de uma aplicação que fornecesse este tipo de serviço, assim como o desenvolvimento de um protótipo que implementasse o modelo construído.

Uma vez que o modelo foi implementado através de um protótipo, a maneira mais simples de verificar o quanto a modelagem de um serviço desta natureza foi bem sucedida, seria através de casos de teste sobre este protótipo, que correspondessem a situações verossímeis de consulta. De fato, esta foi a estratégia utilizada neste trabalho, e para isso, foram usados dados reais, não exaustivos, mas suficientes para demonstrar se o funcionamento do algoritmo da consulta da aplicação foi satisfatoriamente correto.

No final do capítulo que aborda a modelagem da aplicação que busca as linhas de ônibus, foram apresentados alguns testes realizados com o protótipo, utilizando o conjunto de dados inseridos no banco de dados. Todos os testes apresentaram resultados corretos, mas é

importante salientar que, como foi usado um emulador de um aparelho celular, alguns fatores possivelmente existentes em um aparelho real, não puderam ser bem testados, como diferenças entre o uso de GPS ou uso de triangulação por antenas, ou força do sinal Wi-Fi. De qualquer maneira, os testes mostraram que o algoritmo desenvolvido para encontrar as linhas corretas para o usuário chegar no destino desejado, funciona corretamente.

Outro objetivo deste trabalho, foi o de usar a plataforma Android como a tecnologia na qual a parte móvel da aplicação foi elaborada. Esta escolha foi motivada inicialmente por questões técnicas, de verificar o quanto, e como, esta plataforma oferece recursos para o trabalho com informações obtidas através da obtenção da localização do dispositivo. Porém ao desenvolver o modelo do software proposto, utilizando esta tecnologia, surgiram interessantes questões em nível de arquitetura, como estilos, filosofias de modelagem, que não tinham sido visualizadas previamente. Questões interessantes e relevantes de como organizar os componentes de uma aplicação voltada para o universo móvel, ou inseridas dentro do conjunto das soluções que buscam oferecer serviços aos usuários.

Um exemplo disso, foi a utilização de elementos característicos das arquiteturas orientadas a serviços, conhecidas como SOA e já tratadas no respectivo capítulo. A partir dos estudos para construção do modelo, chegou-se ao conhecimento de uma implementação dos princípios trazidos por SOA, ou seja, o estilo de arquitetura RESTful. Estes princípios foram aplicados no decorrer do desenvolvimento da implementação. De acordo com estes princípios, é considerado que, em um sistema que pretenda oferecer serviços de maneira remota, ou melhor, através de uma rede como a internet, estes serviços devem ser tratados como recursos, que devem ser identificados inequivocamente, que sejam desacoplados o suficiente para não criar dependências que prendam desnecessariamente componentes independentes do sistema.

Toda essa descoberta de que a aplicação aqui proposta, possuía estas características, foi decorrente da intenção de usar a plataforma Android no cliente do sistema, na aplicação que iria rodar no aparelho celular. Pois esta plataforma utiliza estas abordagens para acesso a recursos remotos, recursos aqui compreendidos como serviços.

Um detalhe interessante do resultado da elaboração do modelo, foi que o processamento de informações relevantes para o funcionamento do algoritmo de consulta das linhas de ônibus, ficou dividido uma parte no cliente e outra na base de dados utilizada pelo sistema. Nenhum processamento ocorre na camada, por assim dizer, de serviço; esta somente transita a requisição e o resultado da consulta, o que em uma aplicação móvel, que se vale e depende das condições da rede disponível, tem grande importância na parformance final do

software.

6.2 – Possibilidades futuras

O objetivo prático deste trabalho foi o de desenvolver um protótipo que implementasse o modelo construído, para uma aplicação que oferecesse serviços baseados na localização. O serviço idealizado foi o de consultas de linhas de ônibus capazes de levar o usuário da posição onde se encontra até um endereço informado como destino, simplesmente isso. Porém, no decorrer do desenvolvimento do protótipo e por consequência do maior conhecimento das tecnologias utilizdas para a construção do mesmo, potencialidades foram identificadas.

Por exemplo, no protótipo, quando uma linha é selecionada na lista retornada pela consulta, é apresentado ao usuário um mapa mostrando sua posição, identificada pelo aparelho, e as paradas, próximas a esta posição, onde ele pode pegar a linha escolhida. Poderia ser oferecida ao usuário, a opção de qual mapa deseja visualizar, se o mapa com as paradas próximas a origem, ou o mapa que mostra as próximas ao destino, pois ele pode saber onde pegar o ônibus, mas pode não saber onde irá descer do mesmo. Ou ainda, poderia mostrar somente um mapa, como faz no protótipo, porém mostrando as paradas próximas a origem e as paradas próximas ao destino, além do traçado do itinerário da linha selecionada; a API fornecida para trabalhar com mapas no Android, oferece estas funcionalidades.

Outra funcionalidade, oferecida pela API que a plataforma Android disponibiliza para trabalhar com localização, é o recurso de aviso de proximidade de um ponto pré-estabelecido, portanto, sendo possível apresentar um aviso ao usuário, de que ele está se aproximando da parada onde deve descer do ônibus.

Outra funcionalidade que não foi inserida no modelo da aplicação, mas que foi possível vislumbrar após o desenvolvimento do mesmo, decorrente do maior conhecimento das arquiteturas e das tecnologias utilizadas, seriam outras maneiras de informar o destino desejado, ou sua localização, para a consulta. No protótipo, o destino é informado através da entrada de um endereço urbano, mas as vezes o usuário não tem conhecimento desta informação, um turista por exemplo. Seria interessante se a aplicação oferecesse condições para que o usuário informasse o destino através de uma lista de locais, ou informasse apenas uma palavra chave que identificasse o lugar, como 'prefeitura', 'hospital' por exemplo. Ou ainda que pudesse informar o nome de alguma pessoa, de algum contato, e a aplicação resolvesse o endereço de destino.

A maneira como a plataforma Android trabalha com acesso a recursos na internet, propicia a implementação destas outras formas de informar um destino desejado. O compartilhamento de informações em redes sociais por exemplo, a aplicação cliente, poderia fazer uma consulta a alguma dessas redes, onde se encontram dados de contatos, de locais, e resolveria a posição de destino mais facilmente e até mesmo de maneira transparente para o usuário.

Portanto, um software construído em partes que trabalham em conjunto, ou seja, componentes independentes que se relacionam afim de alcançar um objetivo, dificilmente pode ser considerado acabado, pois a escalabilidade é uma característica destes modelos. Mas, a partir da observação dos testes executados com o protótipo desenvolvido, foi possível verificar que o algoritmo básico idealizado para este estudo de serviços baseados na localização, foi bem sucedido no seu objetivo, fornecendo resultados corretos para os dados usados.

Ficou claro que apesar de o objetivo deste trabalho ter sido alcançado, a aplicação apresentou potencialidades que candidatam ela para servir a outros trabalhos que desenvolvam os mesmos estudos, relacionados aos serviços baseados na localização, e voltados para o contexto dos dispositivos móveis.

7. Bibliografia

7.1 - Livros

MURPHY, Mark L. Beginning Android. Apress, 2009.

HASHIMI, Sayed Y. and KOMATINENI, Satya. Pro Android. Apress, 2009.

ABLESON, W.Frank and COLLINS, Charlie and SEN, Robi. Unlocking Android: A developer's Guide. Manning, 2008.

ROGERS, Rick and LOMBARDO, John and MEDNIEKS, Zigurd and MEIKE, Blake. Android Application Development. O'Reilly, 2009.

MEIER, Reto. Professional Android Application Development. Wiley, 2009.

SCHILLER, Jochen and VOISARD, Agnes. Location-Based Services. Elsevier, 2004.

KUPPER, Axel. Location-based Services: Fundamentals and Operation. John Wiley & Sons Ltd, 2005.

ALBIN, Stephen T.. The Art of Software Architecture: Design Methods and Techniques. John Wiley & Sons, 2003

BURK, Bill. RESTful Java with JAX-RS. O'Reilly, 2010

FIELDING, Roy Thomas, 2000. Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine, USA.

7.2 - Sites

Android Developers. Disponível em:<http://developer.android.com/index.html>. Acesso em: Primeiro Semestre 2010

Nazmul. Developerlife - tutorial: Android ListView and custom adapter. Disponível em: <http://developerlife.com/tutorials/?p=327>. Acesso em: Março 2010

WEIMENGLEE. mobiForge - Using Google Maps in Android Disponível em: <http://mobiforge.com/developing/story/using-google-maps-android>. Acesso em: Março 2010

WIKIPEDIA. Representational State Transfer. Disponível em: <http://en.wikipedia.org/wiki/Representational_State_Transfer>. Acesso em: Abril 2010

INFOQ. InfoQ Explores: REST. Disponível em: <http://www.infoq.com/minibooks emag-03-2010-rest>. Acesso em: Primeiro Semestre 2010

Jersey Home. Disponível em: <https://jersey.dev.java.net/>. Acesso em: Abril 2010

FREEMAN, MARK. Calling a REST web service from Android. Disponível em: <http://breaking-catch22.com/?p=127>. Acesso em: Abril 2010

ROMEN. Romen's Techno-Babble Blog: Consuming Web Services from Android. Disponível em: <http://romenlaw.blogspot.com/2008/08/consuming-web-services-from-android.html>. Acesso em: Abril 2010

NARESH. Naresh's Blog: Accessing query parameters using @QueryParam. Disponível em: <http://blogs.sun.com/naresh/entry/accessing_query_parameters_ using_queryparam>. Acesso em: Abril 2010

WIKIPEDIA. Fórmula de Haversine. Disponível em: <http://pt.wikipedia.org/wiki/F %C3%B3rmula_de_Haversine>. Acesso em: Março/Abril 2010.

CODE CODEX. Calculate Distance Between Two Points on a Globe. Disponível em: <http://www.codecodex.com/wiki/Calculate_Distance_Between_Two_Points_on_a_ Globe>. Acesso em: Março/Abril 2010.

WIKIPEDIA. Service-oriented architecture. Disponível em: <http://en.wikipedia.org/ wiki/Service-oriented_architecture>. Acesso em: Abril 2010

FIELDING, ROY THOMAS. Representational State Transfer (REST). Disponível em: <http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: Março/Abril 2010

WIKIPEDIA. Mobile phone tracking. Disponível em: <http://en.wikipedia.org/wiki/ Mobile_phone_tracking>. Acesso em: Junho 2010

WIKIPEDIA. Geographic information system. Disponível em: <http://en.wikipedia.org/ wiki/Geographic_information_system>. Acesso em: Junho 2010.