Um Framework para Carga Dinâmica e Transição Suave de Mapas Contextuais de Bases Heterogênicas*

Danilo Inácio de Souza Resende†, Heitor Menezes de O. Pereira, Ricardo C. Antunes da Rocha‡

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brasil

{daniloresende,ricardo}@inf.ufg.br, heitormzs@gmail.com

Palavras-chave: Mapas contextuais; Contexto; Frameworks; Aplicativos baseados em mapas.

1. Introdução

A popularização dos servicos baseados em localização têm produzido uma série de aplicações em rastreamento de veículos, pacotes e pessoas, identificação de rotas, além de promover o uso de localização em aplicações de presença móvel e de redes sociais. De fato, há um interesse evidente no desenvolvimento de aplicações baseadas em mapas e sensíveis a localização e, na pesquisa em computação ubíqua, que faça um uso dos mecanismos de computação sensível ao contexto para torná-las flexíveis, dinâmicas e adaptativas [Endler et al. 2010].

Existem diversas empresas e instituições que disponibilizam bases de mapas, tais como NASA, Microsoft, Yahoo e Google. Associado a cada um desses provedores, tipicamente existe um framework para desenvolvimento de aplicações que permitem consultar e integrar seus mapas a aplicações de propósito específico. O Google Maps[1] é um exemplo de framework, disponível em plataformas como Android, iPhone OS e navegadores web[2]. Esses frameworks são fortemente acoplados à uma única base de mapas e com uma semântica de localização bem definida, sendo tipicamente, coordenadas geoespaciais baseadas em latitude e longitude. Embora esses frameworks e sua semântica de localização sejam adequadas à aplicações de propósito geral, eles limitam a sua aplicação em cenários de escopo geográfico restrito. Por exemplo, diversos museus, como o British Museum, oferecem um mapas aos visitantes baseado em salas de visitação, que mantém obras diferentes organizadas por diferentes critérios. Uma aplicação, apenas no escopo de localização daquele museu, deve ser capaz de utilizar outro tipo de inferência de localização, onde as informações para determinar a localização de um determinado dispositivo são dadas a partir de pontos de referências próprios do mapa e seguindo uma semântica própria. Como os frameworks de propósito geral não permitem tratar as especificidades de tais ambientes, aplicações sensíveis a localização precisam ser construídas do zero.

*Revisado pelo orientador
†Aluno de PIVIC, Instituto de Informática, UFG
‡Orientador PIVIC
[1]http://code.google.com/apis/maps/
[2]Plataformas que executam javascript.

Além disso, elas não são flexíveis por não permitirem interagir com outras fontes de mapa em um contexto mais amplo.

Uma aplicação que faça uso desses mapas deve ser capaz de utilizar outro tipo de inferência de localização, onde as informações para determinar a localização de um determinado dispositivo são dadas a partir de pontos de referências próprios do mapa. Nessas aplicações, a própria interface com o usuário deve levar em consideração os pontos de referência. Este tipo de localização é chamada localização simbólica [Pradhan 2000].

O objetivo desse trabalho é implementar o conceito de *mapas contextuais*, proposto em [de Oliveira Pereira and da Rocha 2010], por meio de um framework para desenvolvimento de aplicativos na plataforma Android que permita carregar e exibir mapas de múltiplas fontes, e não apenas da base de mapas do Google Maps. Durante o carregamento de um novo mapa, também serão carregados as informações dos seus pontos de referência próprios, capazes de inferir uma localização.

Este artigo é uma versão estendida do artigo[Resende et al. 2011] e está organizado da seguinte forma. A seção 2 descreve um cenário para uso do conceito de mapas contextuais e elabora os principais requisitos do *framework*. A seção 3 apresenta a arquitetura de implementação do cenário proposto, e define a responsabilidade do *framework* e a sua interação com outros elementos do cenário, como um middleware sensível ao contexto e os serviços de mapas. As seções 4 e 5 descrevem a arquitetura interna do *framework* e a sua implementação na plataforma Android, respectivamente. A seção 6 avalia o *framework* proposto de acordo com três estudos de caso, enquanto que a seção 7 compara-o com outros trabalhos encontrados na literatura. Por fim, a seção 8 apresenta as conclusões deste trabalho e futuras direções de pesquisa.

2. Cenário

No cenário de referência deste trabalho, uma aplicação móvel de localização permite a um usuário se localizar dentro do campus da UFG, onde ocorre um evento aberto à comunidade chamado “Espaço das Profissões”. Este evento promove diversas atividades onde a comunidade interna e externa à universidade pode conhecer melhor cada curso de graduação.

Para chegar ao campus, um usuário utiliza os serviços providos pelo Google Maps, incluindo exibição de mapas e determinação de caminhos entre dois pontos. Entretanto, ao chegar na universidade, o serviço Google Maps deixou de ter utilidade pois não provia informações específicas do evento que estava ocorrendo, como local e horário das palestras.

Considere agora que a própria universidade possui um serviço de mapas interno à universidade que oferece mapas e primitivas de navegação especificamente relacionadas ao evento em questão. Como este serviço de mapas, um usuário poderia pesquisar qual estacionamento está mais próximo do estande do curso de bacharelado em Ciência da Computação.

Embora os dois serviços de mapas pudessem ser providos por aplicações distintas, é

especialmente inconveniente ao usuário ter que trocar de aplicação para executar uma tarefa similar mas que aplicada a um contexto diferente (escopo de atuação do serviço de mapas da universidade). De fato, a mesma integração que existe entre os vários tipos de mapas providos pelo Google Maps poderia também ser oferecida com os mapas oferecidos pela universidade. O desafio, neste caso, é que o serviço Google Maps e correspondentes componentes de desenvolvimento de aplicações, são estaticamente ligados à base de provedores de mapas do Google e à sua própria semântica de mapas.

Dessa maneira, este cenário sugere a necessidade de um *framework* para aplicações baseadas em mapas, que permita aos provedores de mapas oferecer componentes que permitam a integração dos seus mapas em aplicações e a troca suave de uma base de mapas para outra, seja por interação do usuário, seja por ação da aplicação. O *framework* deve ainda ser integrado com um middleware sensível ao contexto que permita o tratamento de mapas como contexto, disparar descoberta e carregamento de mapas contextuais, de acordo com a localização, assim como o controle da informação de localização. Com o *framework*, deseja-se evitar a integração explícita e estática de uma aplicação com diversas bases de dados e inter-relacionamento entre os mapas providos por cada base.

3. Arquitetura

A figura 1 ilustra a arquitetura de um sistema sensível ao contexto que faz uso do *framework* de mapas contextuais. Nesta arquitetura, quatro elementos interagem entre si: (i) a UI da aplicação, (ii) o *framework* de carregamento de mapas contextuais, (iii) um middleware sensível ao contexto e (iv) os serviços ou provedores de mapas.

Um serviço de mapas ou provedor de mapas é um servidor capaz de responder a requisições por mapas, tipicamente limitado a um certo escopo físico. Para permitir a transição suave entre os mapas de bases heterogêneas, o *framework* assume como premissa que todos os mapas providos pelos serviços devem possuir um sistema de coordenadas em comum. No protótipo implementado, assumimos coordenadas geográficas baseadas na tupla $(latitude, longitude, altitude)$, utilizada em diversos serviços de localização como GPS e bases de mapas como Google Maps. Portanto, todos os serviços de mapas devem oferecer primitivas de recuperação de mapas que permitam, no mínimo, responder a requisições baseadas neste sistema de coordenadas.

O middleware sensível ao contexto oferece serviços assíncronos para obtenção de informações contextuais relevantes para interação com os serviços de mapas. Em especial, o *framework* interage com o middleware para obter atualizações na localização de entidades exibidas nos mapas ou do próprio dispositivo, além da descoberta de serviços de mapas contextuais. Para descobrir serviços de mapas associados a uma localização, o *framework* registra no middleware sensível ao contexto o interesse por receber notificações contextuais de provedores de mapas no atual escopo em que a aplicação está executando. *Escopo de mapa* é uma abstração

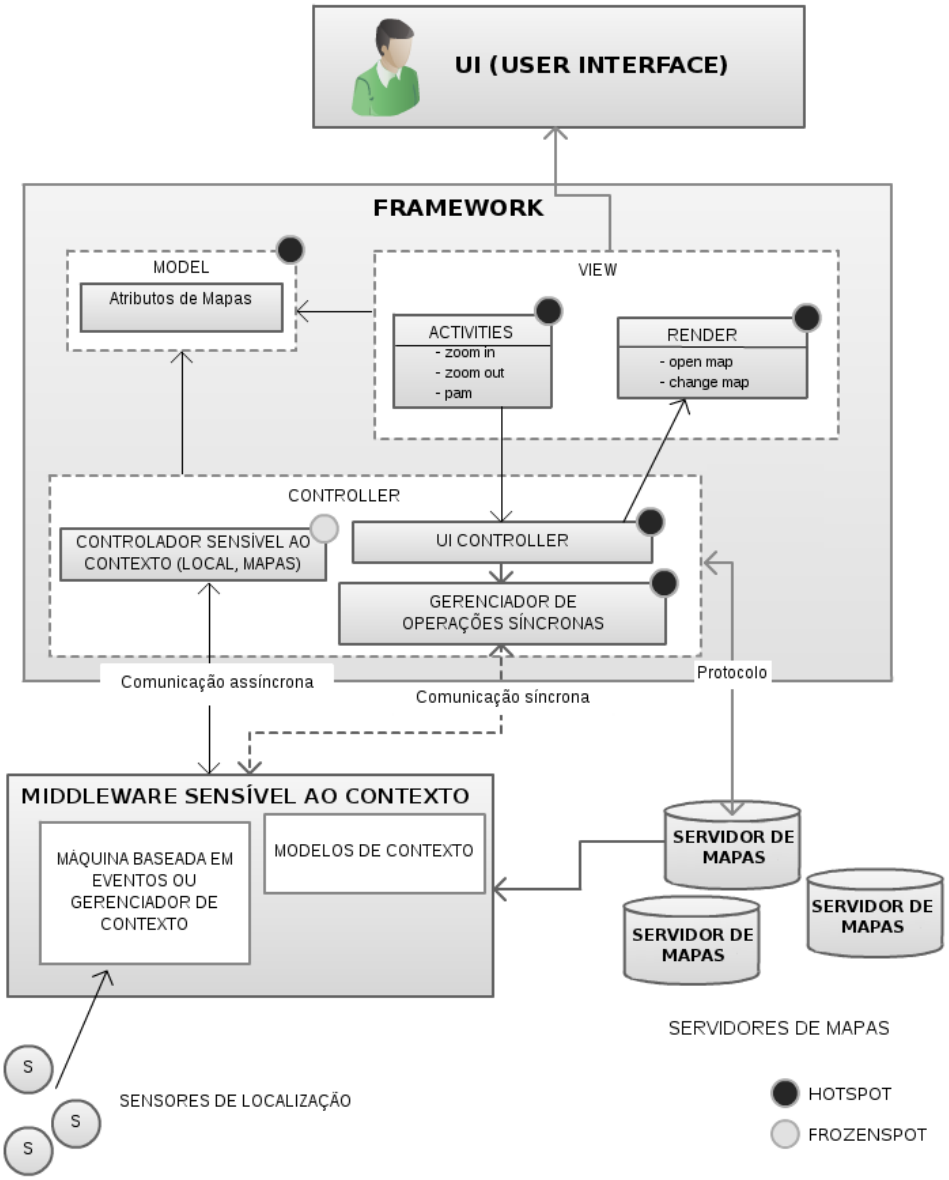

Figura 1. Arquitetura do *Framework*

do domínio que um provedor de mapas é capaz de cobrir. Um escopo de mapa pode ser uma certa área física ou lógica (como a universidade ou museu), uma rede ou qualquer outro espaço que possa ser modelado por contexto. No protótipo desenvolvido, o escopo adotado foi o de rede, o que significa que uma troca da rede do dispositivo pode disparar uma troca do provedor de mapas.

O *framework* permite a interação transparente da UI do usuário e do middleware sensível ao contexto com um serviço de mapas. Uma instância do *framework* implementa a interação com um serviço de mapas em particular. O *framework* é independente da implementação do middleware, bastando que este ofereça uma interface assíncrona de comunicação (publish/subscribe).

4. Framework para Carregamento de Mapas Contextuais

O objetivo do framework é possibilitar o desenvolvimento de serviços de localização que utilizem diferentes provedores de mapas e diferentes semânticas, baseada no contexto do usuário. Neste cenário, as responsabilidades do *framework* de mapas contextuais são: oferecer os componentes de interface de navegação nos mapas, renderizar os mapas obtidos na UI do usuário, implementar o protocolo de comunicação com os serviços de mapas e interagir com o middleware sensível ao contexto com o objetivo de obter notificações relativas à descobertas de mapas e atualização da localização do usuário. Dentre essas responsabilidades, tipicamente a renderização de mapas e o protocolo com os serviços de mapas representam os *hotspots* do *framework*, ou seja, devem ser implementados em cada instância particular do *framework*.

4.1. Arquitetura do Framework

A arquitetura do *framework* segue o padrão MVC (Model-View-Controller), com o qual é possível isolar as suas funcionalidades referentes à interface com usuário (navegação em mapas e renderização de mapas), de comunicação com o middleware sensível ao contexto e com os servidores de mapas. A figura 1 detalha a arquitetura interna do *framework* e seus componentes `Model`, `View` e `Controller`, assim como a interação entre eles.

O componente `Model` é responsável por modelar em objetos os mapas que são manipulados pelo *framework*. Estes atributos serão específicos para cada instância do *Framework*. O componente `View` trata a renderização de um mapa e a resposta aos comandos da UI, como operações de navegação e *zoom in/out*. E o componente `Controller` possui os elementos responsáveis pela comunicação com o middleware sensível ao contexto e com os serviços de mapas. O protocolo de comunicação com os servidores pode ser específico para cada instância do *framework* e define também as regras de transferência de dados. Ele é implementado no *Gerenciador de Comunicação Síncrona.*

4.1.1. Componente `Controller`

O `Controller` é responsável por interagir com o middleware sensível ao contexto, registrando o interesse em receber notificações de situações contextuais que interferem no(s) mapa(s) exibido pela aplicação. Há no mínimo duas informações contextuais que podem interferir no funcionamento da aplicação: a localização e o mapa associado à localização. Do ponto de vista do funcionamento do *framework*, não há diferença entre obter a localização a partir de uma interação com a UI ou de uma notificação recebida por um middleware. Em um cenário estático, eventos de interface do usuário disparam a troca do mapa e/ou execução de operações com a base de mapas. O controlador registra junto ao middleware o interesse em receber notificações de tipos de mapas para uma determinada localização. O *framework* utiliza as notificações para atualizar no componente `View` a interface que permite ao usuário selecionar um mapa ou outro para exibição. Portanto, uma requisição a novo mapa pode ser disparada de duas formas:

- *Por interação do usuário* (eventos de UI): Componente `View` atualiza na interface os tipos de mapas disponíveis para a localização coberta pela aplicação, ele então pode selecionar qual tipo de mapa atende melhor aos seus requisitos. Neste caso, o middleware recebe o registro pelo interesse em todos os tipos de mapas que se aplicam a uma localidade.
- *Por mudanças no contexto*: a aplicação baseada no *framework* recebe as notificações da mudança de contexto, e verifica a existência ou alteração dos mapas existentes e decide exibir ou não um dos mapas disponíveis.

4.1.2. Componente `View`

O componente `View` é responsável pela apresentação gráfica da interface com o usuário, incluindo construir uma representação gráficas dos mapas recuperados do serviço de mapas. Cada componente `View` é associada a um controlador apresentado apenas como um atributo do tipo `Controller`. Este atributo é responsável por traduzir as operações de navegação (movimentação pelos mapas) ou detalhamento (operação de zoom) requeridas, capazes de trabalhar com semânticas diversas de localização, isto é, capaz de manipular diferentes valores obtidos por diversas bases de mapas.

Definida a semântica utilizada por um serviço de mapas, é necessário que esta possua referência ao sistema de coordenadas geográficas, dessa forma é possível efetuar uma troca suave entre os mapas contextuais de interesse.

Considerando serviços baseados em mapas, é necessário definir o grau de detalhamento do mapa. No caso do Google Maps, por exemplo, essa informação é descrita como *zoom* do mapa. Para tratamento dessas informações o *framework* implementa métodos para manipular o nível de zoom dos mapas contextuais.

4.1.3. Componente `Controller`

O componente `Controller` é responsável pelas operações internas de controle dos mapas e externas de comunicação com outros componentes do sistema. As primitivas de localização, como existência e interseção, são definidas para cada semântica utilizada pelos mapas contextuais a partir do serviço oferecido. Dessa maneira, cada instância do *framework* define qual o tratamento para cada interação com o mapa. Tipicamente, o componente `Controller` traduz as operações de navegação do `View`, mencionadas anteriormente, em requisições à respectiva base de mapas. Por exemplo, uma instância do *framework* pode implementar os protocolos de comunicação utilizando diretamente *socket* ou utilizando HTTP.

5. Implementação

O *framework* de carregamento de mapas contextuais foi implementado na plataforma Android e testado tanto no emulador da plataforma como em dispositivos reais. O middleware sensível ao contexto foi implementado como um serviço Android e, portanto, independente do *framework*. Neste protótipo, utilizamos um proxy para um futuro serviço de contexto, implementado em outro trabalho e aproveitando as interfaces propostas em [da Rocha 2009]. Os componentes `Model` e `Controller` são classes tradicionais integradas à aplicação, enquanto que o componente `View` é implementado como uma *Activity*[3] Android. Devido à ausência de espaço, este artigo não discutirá as interfaces implementadas em cada componente.

Com o objetivo de facilitar a integração do *framework* com a implementação da API do Google Maps, a implementação da UI e do componente `View` adotou algumas das convenções dessa API. Desta forma, o conceito de camadas de mapas é implementado na forma de um objeto `Overlay` Android, fornecido pela API mencionada, assim como as operações comuns de navegação em mapas, como `setCenter(GeoPoint)`, `panEast`, `panNorth`, `setZoom(int)`, `zoomIn()` e `zoomOut()`.

Todas as mensagens internas trocadas entre os componentes e indicadas na figura 1 são implementadas como invocação direta de métodos de objetos, exceto as interações da UI com o componente `View`, que são implementadas como *Intents* Android.

A figura 2 ilustra a execução de uma aplicação com alguns pontos de referência, que também podem ser sobrepostos aos mapas. Quando a aplicação inicia um componente `View`, ela também inicializa o serviço de descoberta por meio do controlador sensível ao contexto, com o qual identifica novos serviços de mapas presentes na sua localização atual. Com essa informação, o componente `View` exibe na interface as camadas que representam cada tipo de mapa disponível, tipicamente associado a um diferente serviço de mapas. A figura 3 ilustra a sobreposição entre dois tipos de mapas e um conjunto de pontos de referência.

6. Avaliação

A avaliação foi feita a partir do desenvolvimento de três instâncias do *framework*. Todas as instâncias criadas usaram o Google Maps como provedor principal de mapas. As seguintes instâncias foram desenvolvidas: Instância Google Maps, onde as funções básicas do *framework* como navegação e zoom foram testadas, validando o modelo do *framework*; Instância Provedor de mapas simples bitmapeado, que responde a consultas por mapas simbólicos retornando uma imagem bitmap que será sobreposta aos mapas da base principal (Google Maps); E por último a Instância Provedor de Mapas Simbólicos, que responde consultas elaboradas de locais simbólicos com objetos de sobreposição aos mapas do Google Maps.

Os testes foram feitos em dois ambientes, a maquina virtual Dalvik (*Dalvik VM*) e o

[3]Componente de interface gráfica da plataforma Android

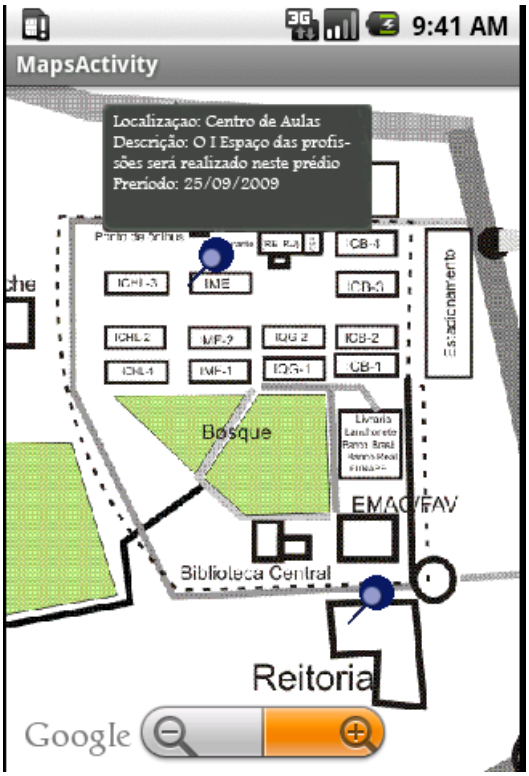

Figura 2. Exemplo de tela de uma aplicação exibindo mapas simbólicos

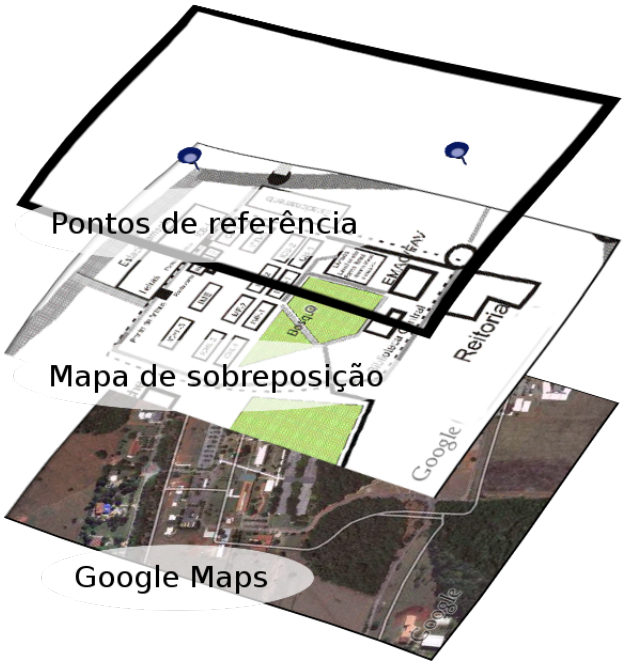

Figura 3. Exemplo de camadas

dispositivo móvel *Samsung Galaxy 5 I5500*, ambos executando a plataforma Android versão 2.1.

6.1. Instância Google Maps

O Google Maps fornece uma API nativa para manipulação de seus mapas no Android, estabelecendo seu próprio protocolo de comunicação e método de renderização. Essa API foi usada para validar a estrutura de integração do *framework*, onde os mapas providos pelo Google Maps são usados como base da aplicação. Nessa instância não há uma base de mapas simbólicos de semântica geral, portanto não há sobreposição de mapas.

6.2. Instância Provedor de Mapas simples bitmapeado

A construção de um serviço de mapas é iniciada pela definição dos dados, valores e atributos utilizados e como estes são relacionados. Para o estudo de caso foram identificados dois principais atributos, a imagem estática da região e as coordenadas geográficas que delimitam a região referente ao mapa, relacionados por meio de um objeto que possua pelo menos esses dois atributos.

As informações de mapas contextuais foram associadas a um objeto serializável definido no `Model`, capaz de ser transmitido por meio de um canal de comunicação síncrona via socket. Este objeto possui um valor inteiro referente ao nível de zoom adequado, duas informações de posicionamento geográfico representando os pontos superior esquerdo e o inferior direito

capazes de inferir a região representada pelo mapa contextual, por fim, dois valores textuais, o primeiro representando o nome do mapa e o segundo é a URL (Uniform Resource Locator) onde é encontrada a imagem do mapa contextual.

O servidor de mapas contextuais ou a base de mapas consiste em uma aplicação Java capaz de disponibilizar informações de mapas contextuais através de comunicação via socket, a partir de uma busca por posicionamento geográfico ou pelo nome do mapa. A consulta é feita em uma lista de mapas, onde todos os elementos são percorridos para o casamento com a consulta. Devido a quantidade relativamente pequena de mapas, este método não causa impacto no desempenho do servidor.

O tratamento dado aos mapas obtidos pela base estática será de sobreposição aos mapas do Google Maps. Para isso é usado o próprio `MapView` disponibilizado pelo Google Maps. Este elemento consiste no principal elemento visual que será carregado no dispositivo para exibir as informações de mapas, e consistirá na visualização padrão da aplicação capaz de interagir com o usuário por meio de navegação e alteração do grau de detalhamento (*zoom in/out*) do mapa exibido.

A aplicação se comunica com servidor quando for solicitado uma busca por mapas contextuais, recebendo uma lista de objetos com as informações dos mapas requisitados. Os mapas de sobreposição serão adquiridos através da URL dos objetos recebidos, que serão usados para requisitar a imagem do mapa no endereço especificado. E por fim a `View` atualiza a tela com a imagem obtida sobrepondo-a ao mapa do Google Maps fazendo ajustes necessário de zoom e posicionamento do centro do display.

6.3. Instância Provedor de Mapas simbólicos

Nessa instância a aplicação é capaz de realizar consultas na base de mapas usando localização simbólica [Hu and Lee 2004], o que torna a busca mais intuitiva para o usuário. Assim consultas do tipo ``ufg/alfiteatro'' podem ser realizadas. Também foi usado o formato *KML*[4] para modelar os dados dos mapas de sobreposição.

6.3.1. Consultas

O servidor de mapas simbólicos é capaz de responder por dois tipos de consultas: por localização geográfica, ou por localização simbólica. No primeiro caso o cliente encaminha ao servidor uma consulta contendo um par de coordenadas geográficas, que será usada para verificar na base de mapas quais mapas possuem informações sobre aquela localização, por exemplo ``geo:latitude,longitude''. Para processar essa consulta foi utilizado a biblioteca JTS[VIVID Solutions] que auxilia a busca por pontos em regiões.

No segundo caso a requisição será no formato de hierarquia, onde um sequência

[4]http://code.google.com/intl/pt-BR/apis/kml/documentation/

de palavras representam um nome simbólico[Durr and Rothermel 2003], por exemplo ``loc:br/go/ufg/inf''. Para esse tipo de consulta o servidor de mapas organiza as referências dos mapas em uma estrutura de árvore, onde cada nó define um mapa e o nível da árvore determina a especialização do local.

Em ambas as consultas o servidor de mapas retorna para o cliente uma lista de mapas que casaram com a consulta. Caso nenhum mapa seja encontrado nos dois tipos de consultas, a aplicação notifica ao usuário que a consulta não obteve mapas.

6.3.2. Implementação

Componente Model O componente `Model` nesta instância é compostos por um parser de arquivos *XML*, pois todos os atributos referentes aos mapas são descritos no modelo *KML*. Tal modelo facilita a construção do servidor, pois os mapas de sobreposição podem ser criados usando a aplicação *GoogleEarth*, onde através de uma interface o usuário é capaz de criar de maneira intuitiva um mapa de sobreposição e gravar essas informações em um arquivo *KML* que pode ser cadastrado no servidor de mapas simbólicos.

Componente Controller O componente `Controller` constrói a consulta para ser encaminhada ao servidor com base na entrada do usuário. A partir de string de consulta o `Controller` realiza a comunicação enviando a consulta e aguardo pelo objeto de resposta. Após receber os mapas como resposta do servidor de mapas, o `Controller` interpreta os dados percorrendo o arquivo *KML* recebido e extraindo as informações (parser) para que o componente `View` seja capaz de fazer a sobreposição do mapa.

Componente View O componente `View` exibe ao usuário o mapa provido pela base de dados principal e através dos dados obtidos pelo `Controller`, ele realiza o download da imagem através do atributo "icon/href" descrito no *KML* que representa uma URL para o download da imagem. Após obter a imagem, o componente `View` realiza a sobreposição do mapa contextual na aplicação usando as coordenadas descritas pelo `Model`.

7. Trabalhos Relacionados

Existem diversos *frameworks* e componentes para desenvolvimento de aplicações baseadas em mapas, principalmente para *Web*, onde essas aplicações são fortemente disseminadas. Citando dois desses *frameworks* que possibilitam o desenvolvimento para dispositivos móveis, que utilizam da J2ME (*Java 2 Platform, Micro Edition*), ou que são baseadas na plataforma Android temos, o Mobile Maps e a Google Maps API.

O Mobile Maps [Ericsson Labs] é um *framework* desenvolvido pela *Ericsson Labs*, que provê uma API para criação de aplicações baseadas em mapas para dispositivos móveis. A partir

dos mapas obtidos do *TeleAtlas*[5] ou do *OpenStreetMaps*[6], este *framework* oferece uma API que implementa a interação com o usuário, sendo possível a navegação pelo mapa, operações de zoom, como também a criação de camadas (`Layer`), possibilitando a adição de elementos gráfico sobre o mapa.

O Google Maps API [Google Inc: Android Developers] é o mais popular *framework* para desenvolvimento de aplicativos baseados em mapas, tanto para *Web*, quanto para dispositivos móveis. Esta API oferece um objeto gráfico chamado de `MapView` responsável pela visualização de mapas providos pelo provedor do Google Maps através do protocolo HTTP. Utilizando localização baseada em IP ou *cellID*, a API infere o posicionamento geográfico do dispositivo por um serviço oferecido no servidor de mapas do Google.

Embora intensivamente utilizados, tais APIs são fortemente acopladas a um único serviço de mapas, assim como semântica de localização, impossibilitando a incorporação de outras fontes de mapas em uma aplicação que faça uso de localização simbólica. Em outras palavras, uma aplicação que faça uso de um desses serviços ou APIs fica fortemente dependente dos mapas descritos nas respectivas bases e, portanto, não é capaz de lidar com mapas de escopo limitado ou propósito específico.

8. Conclusões e Trabalhos Futuros

Este artigo apresentou um *framework* implementado em Android para o desenvolvimento de aplicações baseadas em mapas contextuais. Como próximos passos dessa pesquisa, integraremos o *framework* com um middleware sensível ao contexto baseado em CEP/esper e avaliaremos a instância de mapa contextual baseada em localização simbólica com o uso de sensores cricket de localização. Outro trabalho de pesquisa em andamento, é a aplicação de R-OSGi [Rellermeyer et al. 2007] para o carregamento dos serviços de mapas no *framework*.

Referências

da Rocha, R. C. A. (2009). *Context Management for Distributed and Dynamic Context-Aware Computing*. PhD thesis, Departamento de Informática, Pontificia Universidade Catolica do Rio de Janeiro, Rio de Janeiro.

de Oliveira Pereira, H. M. and da Rocha, R. C. A. (2010). Localizacao, mapas e contexto: Um framework para desenvolvimento de aplicações baseadas em mapas contextuais. In *II Simpósio Brasileiro de Computacao Ubiqua e Pervasiva (SBCUP)*.

Durr, F. and Rothermel, K. (2003). On a location model for fine-grained geocast. In Dey, A., Schmidt, A., and McCarthy, J., editors, *UbiComp 2003: Ubiquitous Computing*, volume 2864 of *Lecture Notes in Computer Science*, pages 18–35. Springer Berlin / Heidelberg.

[5]http://www.teleatlas.com/index.htm
[6]http://www.openstreetmap.org/

Endler, M., Malcher, M., Aquino, J. F., Fonseca, H., and Valeriano, A. (2010). *Handbook of Research on Mobile Software Engineering: Design, Implementation and Emergent Applications*, chapter Developing Map-based and Location-aware Collaborative Applications for Mobile Users. IGI Global.

Ericsson Labs. Mobile maps. Disponível na Internet em: <https://labs.ericsson.com/apis/mobile-maps/>, acesso em Abril, 2011.

Google Inc: Android Developers. Android developers: Location and maps. Disponível na Internet em: <http://developer.android.com/guide/topics/location/index.html>, acesso em Abril, 2011.

Hu, H. and Lee, D.-L. (2004). Semantic location modeling for location navigation in mobile environment. *Mobile Data Management, IEEE International Conference on*, 0:52.

Pradhan, S. (2000). Semantic location. *Personal and Ubiquitous Computing*, 4(4):213–216.

Rellermeyer, J. S., Alonso, G., and Roscoe, T. (2007). R-osgi: distributed applications through software modularization. In *Proceedings of the 8th ACM/IFIP/USENIX international conference on Middleware*, MIDDLEWARE2007, pages 1–20, Berlin, Heidelberg. Springer-Verlag.

Resende, D. I., da Rocha, R. C. A., and Pereira, H. M. (2011). Framework para carregamento dinâmico e transição suave entre mapas contextuais. In *III Simpósio Brasileiro de Computacao Ubiqua e Pervasiva (SBCUP)*.

VIVID Solutions. Java Topology Suite. `http://www.vividsolutions.com/jts/jtshome.htm`. Último acesso em 11 de abril de 2011.