Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

Infraestrutura para o Desenvolvimento de Aplicações Baseadas em Localização e Orientadas a Domínios

Lorena Fernandes Maia

Campina Grande, Paraíba, Brasil

Novembro – 2011

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Infraestrutura para o Desenvolvimento de Aplicações Baseadas em Localização e Orientadas a Domínios

Lorena Fernandes Maia

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Engenharia de Software

Orientadores:
Angelo Perkusich
Hyggo Oliveira de Almeida

Campina Grande, Paraíba, Brasil

M217i Maia, Lorena Fernandes
Infraestrutura para o desenvolvimento de aplicacoes baseadas em localizacao e orientadas a dominios / Lorena Fernandes Maia. - Campina Grande, 2011.
62 f. : il.

Dissertacao (Mestrado em Ciencia da Computacao) - Universidade Federal de Campina Grande, Centro de Engenharia Eletrica e Informatica.

1. Computacao Pervasiva 2. Desenvolvimento de Aplicacoes Moveis 3. Arquitetura Baseada em Plugins 4. Dissertacao I. Perkusich, Angelo, Dr. II. Almeida, Hyggo Oliveira de, Dr. III. Universidade Federal de Campina Grande - Campina Grande (PB) IV. Título

CDU 004.416.3(043)

"INFRAESTRUTURA PARA O DESENVOLVIMENTO DE APLICAÇÕES BASEADAS EM LOCALIZAÇÃO E ORIENTADAS A DOMÍNIOS"

LORENA FERNANDES MAIA

DISSERTAÇÃO APROVADA EM 28.11.2011

ANGELO PERKUSICH, D.Sc
Orientador(a)

HYGGO OLIVEIRA DE ALMEIDA, D.Sc
Orientador(a)

KYLLER COSTA GORGÔNIO, Dr.
Examinador(a)

LEANDRO DIAS DA SILVA, D.Sc
Examinador(a)

CAMPINA GRANDE - PB

Resumo

Os avanços observados nos dispositivos portáteis, nas tecnologias de comunicação de rede sem fio e nos sistemas de posicionamento, tem motivado uma das áreas mais promissoras de serviços móveis: sistemas baseados em localização (LBS). Esses sistemas fornecem serviços de informação para o usuário de acordo com sua posição geográfica. Exemplos de serviços desse tipo incluem exibição de mapas da região, navegação terrestre, condições climáticas locais, situação de tráfego, guia turístico, entre outros.

As aplicações e sistemas cientes de localização existentes, em sua maioria, disponibilizam serviços genéricos, de pouco interesse para o usuário, com soluções para uma determinada área ou domínio. Entretanto, diferentes tipos de informações baseadas em localização, provenientes de áreas distintas podem ser relevantes ao usuário, a exemplo do conjunto de serviços citados anteriormente.

Considerando esses aspectos, neste trabalho é proposta uma infraestrutura para o desenvolvimento de aplicações baseadas em localização fornecendo serviços personalizados sobre múltiplos domínios. A validação do trabalho foi realizada partir da implementação de um estudo de caso para o domínio específico de clima, demonstrando o suporte oferecido pela infraestrutura para o desenvolvimento de aplicações envolvendo diferentes domínios.

Abstract

The advancements in portable devices, wireless communication and positioning technologies have enabled one of the most promising mobile services: location-based systems (LBS), which provide information to mobile users according to their geographic locations. Typical examples of such services include area maps, navegation, local weather, traffic condition, and tour guide, etc.

Existing location-aware applications and systems typically provide services for a specific application domain. However, different types of location-based information from distinct domains could be relevant to the user context such as the set of services mentioned above.

Taking these aspects into account, this paper presents an infrastructure to develop location-aware applications providing personalized services about several domains. This issue have been addressed by developing a scalable and extensible architecture, and providing an API to access services. A case of estudy for the weather domain was implemented demonstrating the infrastructure support to applications development involving several domains.

Agradecimentos

A toda minha família, em especial Maria das Graças, Glória de Lourdes e Maria Fernandes, pelo empenho na minha formação, por todo apoio prestado, pela compreensão incomensurável e pelo amor incondicional.

Aos meus amigos Danilo Freire, Larissa Maia, Luiz Paulo e Matheus Gaudencio, pela força e estímulo; Bruno Moura, Ademar Netto e Taísa Felix, pelas boas lembranças.

A Gabriela Almeida, Braúlio Almeida, Romeryto Lira, Maurílio Silva e Kyller Gorgônio, pelo questionamento incessante: "Defende quando?".

Aos colegas de curso: Mariana Romão, Magno Jefferson, Lucas Vieira, Daniel Bruno, Paulo Rômulo, Leonardo Sampaio, Gustavo Soares e Marco Rosner, pelas conversas e pelos momentos de descontração.

Aos antigos alunos do laboratório: Diego Bezerra, Marcos Fábio, Thiago Santos, Felipe Pontes e Glauber Ferreira, pelas colaborações.

Aos meus orientadores, pela credibilidade, pelas oportunidades oferecidas, pela paciência e pelos ensinamentos.

A Tales Gurjão, Daniel Albert, Danilo Freitas, Yuri Farias, Ed Rodolfo, Francisco Thiago e ao professor Tiago Massoni, por compreenderem minha ausência nas atividades do laboratório.

As secretárias da COPIN, Vera Oliveira, Rebeka Lemos, e, carinhosamente, Aninha Sauvé, pela presteza e atenção.

As pessoas que fazem a CAPES e o Laboratório de Sistemas Embarcados e Computação Pervasiva (Embedded), pelo apoio financeiro e pela infraestrutura disponibilizada durante o período da pesquisa, respectivamente.

Enfim, a todos que contribuíram de alguma forma para a realização deste trabalho.

Conteúdo

Lista de Símbolos

3GPP - *3rd Generation Partnership Project*

DNS - *Domain Name System*

GPS - *Global Positioning System*

GSM - *Groupe Spécial Mobile*

GUI - *Graphical User Interface*

INMET - *Instituto Nacional de Meteorologia*

IP - *Internet Protocol*

KML - *Keyhole Markup Language*

LBS - *Location-based Systems*

MD5 - *Message-Digest algorithm 5*

NWS - *National Weather Service*

RFC - *Request for Comments*

SNMP - *Simple Network Management Protocol*

SOAP - *Simple Object Access Protocol*

TCP - *Transmission Control Protocol*

UDP - *User Datagram Protocol*

UMTS - *Universal Mobile Telecommunications System*

URL - *Uniform Resource Locator*

WSDL - *Web Service Definition Language*

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A evolução dos sistemas de comunicação móvel aliada à disponibilização de aparelhos portáteis integrados com recursos como GPS (*Global Positioning System*), Bluetooth, Wi-Fi, dentre outros, tem propiciado o desenvolvimento de sistemas baseados em localização (LBS - *Location Based Systems*). Como o próprio nome sugere, esse conceito caracteriza aplicações que incorporam localização geográfica com a noção de serviços.

Apesar de se tratar de um termo discutido há alguns anos, não existe uma definição ou até mesmo uma terminologia comum. Um motivo desse dilema deve-se ao fato de as características dos serviços oferecidos terem sido determinadas por diferentes comunidades, especialmente o setor de telecomunicações e a área de computação ubíqua. A GSM *Association*[1] define LBS como sistemas que utilizam a localização para agregar valor a um serviço oferecido. Já o 3GPP (*3rd Generation Partnership Project*)[2] faz distinção entre LBS e serviços de localização. O primeiro trata-se de um provedor de serviços que utiliza a informação de localização do terminal, e o último dedica-se à obtenção e propagação da localização do dispositivo [52].

No meio acadêmico a definição mais aceita é a de que LBS é um subconjunto de serviços cientes de contexto [32]. Em computação pervasiva, esses tipos de serviços adaptam seu comportamento para refletir o ambiente físico em que um usuário está inserido, sendo este obtido por meio de seu dispositivo móvel [53]. Assim, localização geográfica pode ser entendida como um tipo especial de informação de contexto. Entretanto, não existe uma dis-

[1] Consórcio de 600 operadoras de redes GSM.

[2] Federação internacional que visa prover a especificação para GSM e UMTS.

tinção nítida entre LBS e serviços cientes de contexto. Em muitos casos as informações de contexto relevantes para o serviço estão associadas à localização do objeto, como por exemplo, dados de temperatura ou poluição. Consequentemente, a localização deve ser obtida antes de disponibilizar outros dados de contexto.

O ramo de aplicação desses serviços engloba áreas como propaganda, jogos, saúde, trânsito, turismo, entre muitos outros, indo desde um simples compartilhamento de localização, como o Google Latitude[3], a aplicações mais específicas, como o AndWellness[4], esta última mantém informação sobre hábitos alimentares dos usuários auxiliando no controle de dietas.

Estima-se que essas aplicações sejam utilizadas por milhões de usuários, os quais se autenticam e realizam atualizações de seus estados. A infraestrutura utilizada deve considerar um crescente número de acessos simultâneos e uma massiva quantidade de dados a fim de disponibilizar serviços de forma escalável. Geralmente essa infraestrutura envolve uma coleção de módulos, cada um responsável pela manipulação de informações diferentes que podem abranger várias máquinas em muitas instalações físicas espalhadas em várias localidades.

1.1 Problemática

Sistemas baseados em localização, em sua maioria, possuem entidades operacionais caracterizadas pelas seguintes funções: o usuário, o dispositivo, a operadora de telefonia, o provedor de localização e o de conteúdo. Estas cooperam durante a execução de um LBS solicitando ou disponibilizando sub-serviços. Cada entidade mantém uma infraestrutura técnica que vai desde simples dispositivos móveis (usuários) a servidores (LBS, provedores de dados, de localização) em grande escala envolvendo complexas redes de celulares (operadoras de telefonia). A interação entre estas acontece mediante protocolos e serviços de conectividade oferecidos por diferentes redes de comunicação. Como pode-se observar, a construção de um LBS envolve grandes desafios, desde a elaboração à implantação do sistema.

Além da complexidade intrínseca de desenvolver um sistema LBS, poucas são as soluções centradas no usuário. A partir de um *Web Service* [29] é possível requisitar dados de

[3] www.google.com/laútitude

[4] http://andwellness.cens.ucla.edu/CI2/index.php

clima de serviços como Yahoo! Weather[5], Weather Channel[6], National Weather Service[7], entre outros. Entretanto a informação provida não considera as preferências do solicitante podendo não ser relevante aos interesses do mesmo. Muitas vezes aquela é muito geral e não corresponde às expectativas. Quando o usuário requisita a temperatura de um local específico como uma rua ou avenida, o serviço retorna a temperatura média da cidade ou até mesmo do estado, por exemplo.

Uma solução plausível para esse problema é incrementar o número de estações meteorológicas. Isso já acontece em países da Europa onde alguns cidadãos possuem pequenas estações climáticas em casa. Apesar de simplórios, os dados fornecidos por estas podem aumentar a precisão da informação disponibilizada. Estações mantidas por órgãos governamentais também podem ser usadas como fonte de informação meteorológica.

O cenário mencionado acima está relacionado a serviços de clima, entretanto diferentes tipos de serviços podem fornecer informação relevante ao usuário, como cultura, trânsito, comércio, entre outros, caracterizando-se como domínios de aplicação. Cada serviço possui provedores distintos que, por sua vez, fornecem dados, também, distintos. A diferença dos dados não se restringe apenas a domínios diferentes, inclui também os diferentes provedores de dados de um mesmo domínio, os quais disponibilizam informação de maneiras diferenciadas. Utilizando o domínio de condições climáticas como exemplo: as estações meteorológicas são os provedores de dados e cada uma os manipula da maneira que julgar conveniente. Essa diferença é mais perceptível na mudança de países onde existe alteração de unidades e tecnologias de coletas de dados, por exemplo. O manuseio dos dados do INMET (Instituto Nacional de Meteorologia[8]) aqui no Brasil difere do manuseio do NWS (National Weather Service[9]) nos Estados Unidos da América.

Dados os aspectos acima discutidos anteriormente, como construir uma infraestrutura a fim de disponibilizar informações de domínios diferentes através de serviços baseados em localização considerando os aspectos de construção desse tipo de sistema e o perfil do usuário?

[5]http://weather.yahoo.com/
[6]http://www.weather.com/
[7]http://www.nws.noaa.gov/
[8]http://www.inmet.gov.br/
[9]http://www.nws.noaa.gov/

Algumas soluções existentes para o desenvolvimento de LBS abordam alguns dos aspectos deste problema. Os *frameworks* desenvolvidos em [10] e [27] proveem informações de navegação considerando e não considerando, respectivamente, as preferências e perfil do usuário, porém leva em consideração apenas um domínio. As soluções propostas em [59] e [56] disponibilizam informações de clima desprezando os interesses do usuário. Porém, não foram encontrados trabalhos na literatura que abordem múltiplos domínios de aplicação ou estrutura que possibilite simples extensão da solução.

1.2 Objetivo

Neste trabalho tem-se como objetivo desenvolver uma infraestrutura para a construção de aplicações baseadas em localização que disponibilizem informações personalizadas para o usuário, a partir de dados fornecidos colaborativamente pelos diferentes provedores de vários domínios. A infraestrutura elaborada, denominada Voilà, deve ser: escalável, para permitir uma grande quantidade de acessos simultâneos em escala mundial; extensível, para permitir a integração de novos serviços e provedores de informações em diversos domínios; e simples, facilitando o desenvolvimento de novas aplicações e serviços.

Para alcançar o objetivo principal, os seguintes objetivos específicos são considerados:

1. Definição uma arquitetura para aplicações LBS orientadas a domínio;
2. Definição uma abordagem para promover a escalabilidade da arquitetura;
3. Definição uma API (*Application Programming Interface*) simples para o desenvolvimento de novas aplicações utilizando a infraestrutura;
4. Desenvolvimento de uma aplicação como estudo de caso utilizando a infraestrutura proposta a fim de validá-la.

Para garantir a extensibilidade da infraestrutura, utiliza-se uma arquitetura cliente-servidor baseada em *plug-ins*, que são responsáveis pela gerencia e manipulação dos dados dos diversos domínios de aplicação. Caso se torne necessária a inclusão de um novo tipo de provedor de dados, esse pode ser adicionado sem grandes alterações na arquitetura bastando

a integração com o *plug-in* do domínio associado. Da mesma forma, novos domínios podem ser adicionados por meio da integração de novos *plug-ins*.

A escalabilidade da arquitetura do sistema foi alcançada utilizando-se aspectos de sistemas distribuídos. Foram definidos módulos independentes, capazes de funcionar em máquinas distintas, os quais se comunicam através de *web-services* e/ou de *sockets* TCP, possibilitando o paralelismo de atendimento de requisições dos usuários. Com o intuito de não sobrecarregar o sistema, é realizado um balanceamento de distribuição dessas requisições.

Por fim, utilizando a infraestrutura, uma aplicação LBS foi desenvolvida para um dispositivo compatível com a plataforma Android que utiliza dados do domínio de clima para prover serviços e informações para o usuário.

1.3 Relevância

As aplicações LBSs estão tornando-se mais complexas. Estas enredam funcionalidades e informações de vários campos diferentes em uma mesma interface. Como é o caso do aplicativo Nokia Sports Traker[10], que possibilita a geração de rotas, a visualização e integração de fotos de locais da trilha realizada, e ainda, o compartilhamento das mesmas com amigos. Nesse exemplo observa-se claramente a presença de dois domínios diferentes, rotas e mídia. Considerando ainda outro exemplo, pode-se citar um aplicativo de tráfego de veículos disponibilizando informações sobre congestionamento, vias interditadas, etc. Neste caso, o domínio de rotas também pode ser encontrado, além do domínio de tráfego de automóveis. Com tantas possibilidades o desenvolvedor dessas aplicações precisa se preocupar em prover uma arquitetura que consiga lidar com uma numerosa quantidade de usuários e de volume de dados, específicos de cada domínio envolvido. Ou seja, além de ter de entender com clareza o tipo de dado a ser tratado em cada domínio, o desenvolvedor deve preocupar-se com o balanceamento de carga, comunicação entre os módulos, tolerância a falhas, entre outros aspectos do sistema, para então estruturar um mecanismo de extração da informação para o domínio em questão.

Desta forma, faz-se necessário o desenvolvimento de um mecanismo que possibilite a expansão do sistema para outros domínios de forma escalável, abstraindo vários detalhes

[10]http://sportstracker.nokia.com/nts/main/index.do

envolvidos no tratamento dos dados obtidos, deste modo, tornando o desenvolvimento de sistemas LBSs mais robusto, simples e transparente.

Por fim, o trabalho está inserido no contexto do projeto PerComp, do Laboratório de Sistemas Embarcados e Computação Pervasiva[11] da UFCG, servindo como base para a proposição de novas abordagens de LBS e contribuindo com o grupo de pesquisa da instituição.

1.4 Organização

Essa dissertação segue estruturada da seguinte forma:

- No **Capítulo 2** são descritos os conceitos envolvidos no desenvolvimento desse trabalho. Inicialmente são explicitados aspectos da computação pervasiva e suas implicações. Em seguida, sistemas baseados em localização são exemplificados a fim de permitir que o leitor compreenda com clareza a área na qual esse trabalho está inserido. Por fim, são detalhadas arquiteturas baseadas em *plug-ins*.

- No **Capítulo 3** são descritas as abordagens relacionadas com o trabalho proposto, discutindo as limitações de cada uma.

- No **Capítulo 4** é apresentada uma visão geral da infraestrutura do sistema, destacando a interação entre os módulos. Em seguida, os aspectos relacionados a arquitetura do servidor e da aplicação cliente móvel são explicados em detalhes.

- No **Capítulo 5**, é apresentado o desenvolvimento de um estudo de caso para o domínio de clima, denominado *Weather*, onde cada componente arquitetural, tanto do servidor como da aplicação cliente são discutidos.

- No **Capítulo 6** são apresentadas as considerações finais e os trabalhos futuros, destacando a contribuição desse trabalho.

[11] http://www.embeddedlab.org/

Capítulo 2

Fundamentação teórica

Neste capítulo são descritos os principais conceitos envolvidos na elaboração deste trabalho. Inicialmente, uma descrição do paradigma de computação pervasiva e suas características são apresentadas, seguida de uma explicação mais detalhada de sistemas baseados em localização e seus princípios. Por último, um embasamento teórico sobre arquitetura de software baseada em *plug-ins* é apresentado.

2.1 Computação Pervasiva

O mundo está cada vez mais povoado de aparelhos digitais projetados para automatizar nossas atividades, enriquecer as interações sociais, entre outros benefícios. Os atuais dispositivos, a exemplo dos *smartphones*, possuem GPS, câmeras, microfones, diferentes tipos de sensores (e.g. acelerômetro, giroscópio, magnetômetros) integrados, possibilitando a determinação da localização e de outras informações referentes às imediações do usuário. Especula-se que alguns fabricantes estão considerando, ainda, a inclusão de sensores para detectar outras características do meio, como temperatura e pressão atmosférica [35]. Esses aparelhos também possuem tecnologias de comunicação, como Bluetooth e Wi-Fi, que possibilitam o descobrimento de infraestrutura computacional e serviços disponíveis. A maioria dos usuários desse tipo de dispositivo conectam-se a Internet de modo a transferir dados multimídia, sejam eles textos alfanuméricos, vídeo ou áudio, em quase todo lugar e a qualquer hora (embora restrições de quantidade de tráfego possam ser aplicadas).

A proliferação de dispositivos como *notebooks*, *netbooks*, *tablets*, os já citados *smartpho-*

nes, entre outros; assim como, os avanços recentes em redes de comunicações, telecomunicações e tecnologia da informação, tem estimulado o desenvolvimento de aplicações pervasivas. A perspectiva de "um usuário, um sistema" tem cedido lugar para sistemas heterogêneos de larga-escala envolvendo muitos dispositivos e muitos usuários, os quais colaboram em diferentes escalas de espaço e de tempo [35]. Com este cenário, a computação pervasiva tem se expandido e passou a estar mais presente no cotidiano das pessoas.

"*The goal is to achieve the most effective kind of technology, that which is essentially invisible to the user.*" [58]. A clássica declaração de Mark Weiser sintetiza o conceito de computação pervasiva ou ubíqua: a possibilidade do usuário poder acessar informações em qualquer lugar, a qualquer hora, por meio de qualquer dispositivo, de forma transparente e intuitiva [33, 36, 46, 57]. A computação pervasiva tem sido uma área de pesquisa emergente e representa um paradigma revolucionário para os modelos computacionais [23], o qual tem alterado a relação humano-máquina, convertendo-a de uma interação centrada na máquina para em uma interação centrada no usuário [48].

Um sistema de computação pervasiva é aquele que intercala informações entre os mundos físico e digital com o objetivo de fornecer assistência, ou a informação de que o usuário precisa na realização de suas atividades, de forma proativa e conveniente [8]. Para a proatividade ser efetiva é crucial que sistemas desse tipo analisem a intenção do usuário [51]. Conhecer o ambiente em que ele está inserido, traçar seu perfil, entre outras; são algumas das maneiras de prever e determinar quais ações do sistema serão mais adequadas. Essas informações fazem com que o sistema mantenha uma interação mais natural com usuários, indo além do tradicional sistema de interação isolado dos computadores pessoais [50].

A dificuldade encontra-se em como desenvolver aplicações que se adaptam continuamente ao meio e permaneçam em funcionamento, considerando que os usuários se deslocam ou mudam de dispositivos com uma certa periodicidade [22]. Uma infraestrutura desenvolvida para ambientes pervasivos deve suportar ciência de contexto, mobilidade, adaptação, interoperabilidade e escalabilidade [25]. Nas subseções seguintes, os desafios envolvidos no cumprimento dos principais requisitos são comentados.

2.1.1 Interoperabilidade

Atualmente os desenvolvedores de aplicativos usam uma grande variedade de modelos de programação, linguagens e ambientes de desenvolvimento. Acredita-se que essa heterogeneidade deve continuar no futuro, especialmente porque a gama de possibilidades de utilização das tecnologias de computação se expande. Assim, a infraestrutura para computação pervasiva deve suportar diversos tipos de componentes de software, considerando-se que a mesma terá de integrar esses componentes em composições capazes de interagir e cooperar para realizar tarefas comuns [25].

Aplicações em ambientes de computação pervasiva, provavelmente, terão de adaptar-se a novas tarefas e situações com o passar do tempo. Para atender a esse requisito, essas aplicações deverão ser formadas a partir de componentes de software disponíveis dinamicamente. Diante deste fato, a infraestrutura necessitará dispor de interoperabilidade dinâmica no nível do componente, que supere a heterogeneidade do ambiente e dos próprios componentes. A principal função dos componentes será a de adquirir conhecimento da interface e do comportamento dos outros de forma dinâmica, a fim de aprender a interagir com os componentes até então desconhecidos [25].

2.1.2 Escalabilidade

Futuros ambientes de computação pervasiva enfrentarão uma proliferação de usuários, aplicativos, dispositivos, e meios de comunicações interagindo em uma escala nunca antes estabelecida. O número de dispositivos conectados e a intensidade das interações humano-máquina acompanha o crescimento da inteligência do ambiente pervasivo [16].

O desenvolvimento tradicional demanda uma diferente elaboração da aplicação para cada novo dispositivo. Mesmo que uma empresa possa gerar novas aplicações tão rápido quanto a adição de novos dispositivos (escrevendo a lógica da aplicação apenas uma vez independentemente do dispositivo) esta teria um significativo trabalho para resolver o problema da escalabilidade das aplicações [50].

Além disso, as aplicações são normalmente distribuídas e instaladas separadamente para cada classe de dispositivo e família de processadores. O aumento do número de dispositivos torna impraticável a distribuição e instalação de aplicativos para cada classe existente,

especialmente em uma ampla área geográfica [50].

Para resolver o problema de escalabilidade, é necessário desenvolver software que considera a abundância de usuários, interações, componentes e dispositivos, evitando soluções centralizadas e gargalos. Em outras palavras, é essencial a construção de uma plataforma de software tolerante a falhas e com componentes distribuídos [25].

2.1.3 Ciência de Contexto

A invisibilidade citada por Weiser pode ser em parte alcançada pela redução de entradas fornecidas pelo usuário, sendo estas substituídas pelo conhecimento do contexto. Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma entidade. Uma entidade é uma pessoa, lugar ou objeto, considerada relevante para a interação entre o usuário e a aplicação, incluindo o próprio usuário e o aplicativo [33].

Os componentes de software cientes de contexto exploram informações como: estado emocional do usuário; quais pessoas estão nas proximidades; atividades em que o usuário está envolvido; localização; hora do dia; e condições meteorológicas [3]. O conhecimento do contexto também se faz necessário para permitir a adaptação às mudanças no ambiente, como presença/ausência de banda-larga e entrada/saída de dispositivos, situações que podem ser provocadas pela mobilidade [25].

Informações de contexto podem se originar a partir de uma ampla variedade de fontes, com considerável heterogeneidade em termos de qualidade e persistência. A percepção do contexto é em geral, altamente dinâmica e propensa a ruídos e erros de detecção. Informações fornecidas pelo usuário (histórico de interação com a aplicação ou o perfil) também compõem informações de contexto e são inicialmente muito confiáveis, mas ao longo do tempo podem perder sua importância caso o usuário não as atualize [24].

A infraestrutura para computação pervasiva deve suportar a ciência de contexto, facilitando a coleta de informações de sensores ou outra fonte; realizando interpretação de dados; divulgando informações de contexto às partes interessadas de forma escalável e oportuna; e fornecendo modelos de programação de aplicações sensíveis ao contexto [25].

2.2 Sistemas Baseados em Localização

Sensoriamento e ciência de localização têm sido um tópico relevante em computação pervasiva por quase duas décadas. Dispositivos de baixo custo contendo GPS permitem navegação computadorizada e muitos usuários se agradam de atividades associadas, como *geocaching*[1]. Redes celulares de alta velocidade combinadas com ciência de localização possibilitam a criação de versões móveis de aplicativos populares, como o de busca local de produtos. A maioria das plataformas de dispositivos móveis disponibilizam recursos de localização de maneira funcional, e, assim, desenvolvedores podem integrá-los em uma grande variedade de aplicações [18].

Serviços Baseados em Localização podem ser definidos como quaisquer serviços com valor agregado oferecidos em um ambiente com conectividade sem fio que analisam a informação de localização de um terminal móvel [52]. A OGC (*Open Geospatial Consortium*) [37] caracteriza como sendo um serviço que utiliza informação geográfica para fornecer serviços a um usuário móvel ou a uma aplicação que explora a posição do dispositivo móvel. No meio acadêmico, esses tipos de sistema são geralmente considerados um subconjunto especial de serviços cientes de contexto [32]. Esses serviços obtêm a localização do usuário usando métodos de posicionamento e incorporam informação adicional com conteúdo relevante para o cliente móvel [45]. Pode-se classificar sistemas baseados em localização, de acordo com o tipo de negócio e pela perspectiva do usuário, como: emergência, navegação, propaganda, entretenimento rastreamento e pagamento [21, 61].

O conceito de localização pode ser entendido como um lugar específico no mundo real, sendo este determinado por meio de um nome de uma rua, ou por coordenadas geográficas. Kuepper [32] classifica localização nas seguintes subcategorias:

- Localização descritiva - está sempre relacionada aos objetos geográficos naturais como territórios, montanhas, lagos, ou aos objetos geográficos feitos pelo homem como cidades, países, estradas, edifícios, e salas dentro de um edifício. Estas estruturas são referenciadas pelas descrições, isto é, nomes, identificadores, ou números, de onde esta categoria de localização derivou seu nome.

[1] Versão tecnológica do passatempo de caça ao tesouro. Utiliza-se dados de GPS guiar o usuário para o local onde está escondido o objeto.

- Localização espacial - representa um único ponto no espaço Euclidiano. É expressa geralmente por meio de duas ou três coordenadas dimensionais, que são dadas como um vetor de números, cada uma delas fixando a posição em uma dimensão.

- Localização por rede - consulta a topologia de uma rede de comunicações, por exemplo, Internet ou sistemas celulares como GSM. Isto é possível pelo uso de endereços de rede que contêm a informação de roteamento, em combinação com serviços do diretório, para traçar números, identificadores, ou nomes.

A seguir alguns dos métodos de posicionamento mais utilizados para se obter a localização geográfica do usuário, são comentados.

2.2.1 Sistemas de posicionamento

Dentre as categorias classificadas por Kuepper, citadas anteriormente, as mais utilizadas pelos dispositivos são: a localização espacial, a exemplo do GPS; e a por rede (e.g. Cell-ID, Wi-Fi); sendo a descritiva mais comum nos diálogos diários entre os seres humanos. Há ainda uma distinção entre sistemas que funcionam em ambientes abertos (*indoor*) e/ou fechados (*outdoor*). O funcionamento do GPS, Cell-ID e da localização via Wi-Fi são analisados nas subseções que se seguem.

GPS

O sistema de posicionamento global (GPS) foi projetado para fornecer a posição instantânea bem como a velocidade de um ponto sobre a superfície da Terra ou próximo a ela, num referencial tridimensional. Este sistema é baseado em satélites que enviam mensagens específicas que são interpretadas pelo receptor GPS.

O princípio que norteia o funcionamento do GPS é o conceito de tempo de chegada (*time of arrival*). O receptor mede o tempo de viagem de um código pseudo-aleatório enviado de um satélite GPS até ele (na prática, cerca de 1 segundo). Este intervalo de tempo é então multiplicado pela velocidade de propagação do sinal obtendo-se a distância emissor-receptor. Através do tempo de propagação do sinal emitido por uma constelação de emissores (24 satélites e três *backups* [26]), em posição conhecida, o receptor pode determinar a sua posição [19].

Este sistema é capaz de servir um número ilimitado de receptores e fornecer informação de localização com uma precisão que varia de 1 a 10 metros, entretanto, o GPS funciona apenas em ambientes abertos e consome bastante bateria do dispositivo.

Cell-ID

Cell-ID é uma técnica de localização física de telefones celulares, sendo um dos primeiros métodos adotados com essa finalidade. A infraestrutura e os meios de acesso a essa tecnologia são fornecidos pela rede de telefonia móvel celular.

Um operadora de rede de telecomunicações possui centenas de estações rádio base que, conectadas compõem a rede como um todo. Cada estação base é uma célula, que provê cobertura a um espaço físico. Estas células possuem tamanho variado de acordo com a densidade de estações base instaladas na área. Um telefone celular se conecta à rede através da célula informando sua identificação e onde está localizado, como ilustrado na Figura 2.1. A diferença no tamanho da célula afeta a precisão da localização, já que a localização retornada é a da estação base, e o dispositivo pode estar em qualquer lugar dentro do limite desta [19].

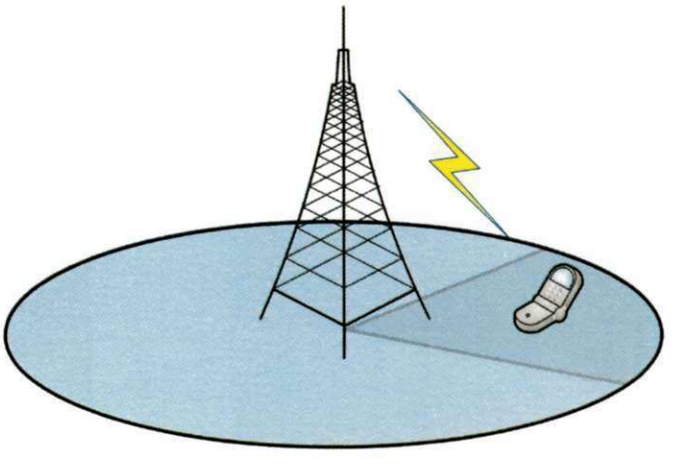


Figura 2.1: Representação do funcionamento do Cell-ID.

O deslocamento do dispositivo acarreta em mudança de célula onde a escolha da estação rádio base é realizada de acordo com a proximidade do aparelho, ou seja, através da monitoração da qualidade do sinal nas diferentes estações.

Apesar da rede sempre conhecer a localização da célula onde se encontra o aparelho,

pouca precisão é obtida com o uso dessa tecnologia, normalmente o erro associado em centros urbanos é de cerca de 500m e na zona rual pode chegar a 15km. Entretanto, esse modelo apresenta baixo custo de implantação, visto não serem necessários nenhum equipamento especial nem modificações no aparelho e na rede; apresenta respostas rápidas, pois cálculos não são realizados na obtenção da localização desejada; opera em ambientes fechados; e funciona na maioria dos aparelhos celulares.

Wi-Fi

Wi-Fi (*Wireless Fidelity*) é um termo que identifica redes e dispositivos que implementam a especificação IEEE 802.11 para redes sem fio [1]. Uma rede Wi-Fi estruturada é composta por dispositivos que se comunicam por sinais de rádio-frequência dentre os quais um ou mais são pontos de acesso (PA). Pontos de acesso são dispositivos que, por um lado, conectam-se à rede cabeada e, por outro, comunicam-se com os outros dispositivos Wi-Fi, servindo como ponte para que tais dispositivos acessem a rede [42].

Essa infraestrutura pode ser utilizada por um sistema de localização que analisa os sinais das redes para inferir coordenadas de posição. A mesma se caracteriza pela utilização de uma estratégia de rastreamento de usuários em ambientes fechados usando a informação de intensidade do sinal de rádio (RSSI[2]) provida pela rede. Essa informação pode ser facilmente medida interrogando o ponto de acesso que é usado para fornecer cobertura de rede sem fio [4].

É possível, utilizando RSSI, identificar um local em um espaço bidimensional a partir de três pontos de acesso. Na prática, as características físicas de uma construção, como paredes, elevadores, e mobiliário, bem como a atividade humana, causam flutuações do sinal, adicionando ruído significativo para medições de RSSIs. Nesses casos, abordagens estatísticas são usadas para estimar a localização [38].

O processo de localização é feito em fases [11, 20, 28, 60]. Na primeira, é construído um banco de dados que armazena a informação das RSSIs e das frequências de presença dos sinais provenientes dos diversos PAs que se encontram no ambiente, informação esta que é obtida por um dispositivo móvel que realiza observações em diferentes localizações físicas do ambiente. Esse banco de dados costuma ser chamado de mapa de RSSI. Na segunda fase,

[2] *Radio Signal Strength Information*

esse mapa é então usado como referência para determinar a qual das posições gravadas o conjunto de valores de RSSI que um dispositivo observa num determinado momento deve corresponder, possibilitando assim a estimativa de localização.

2.3 Arquitetura de Software Baseada em *Plug-ins*

A computação está em constante evolução e a cada dia surgem novas tecnologias e recursos. Portanto, a maioria dos software sofrerão alguma forma de progresso ao longo de seu período de atividade, tanto para ajustar mudanças de requisitos e adição de novas funcionalidades, como para corrigir *bugs* e problemas à medida que forem descobertos. Tradicionalmente, a realização de atualizações, correções ou reconfiguração de sistemas de software demandam recompilação do código fonte ou suspensão e reinicialização do mesmo. Este é um cenário desagradável para sistemas com alto índice de disponibilidade, pois o desligamento temporário do serviço implica em altos custos e riscos para o negócio. Para outros sistemas, onde a disponibilidade não é um fator crítico, interromper a execução de uma fração do software a fim de realizar uma atualização é simplesmente inconveniente [13].

Um aspecto importante na elaboração de software é a capacidade de lidar com a evolução de sistemas em resposta às mudanças de requisitos não previstas no momento inicial do projeto [14]. A habilidade de responder rapidamente a essas mudanças resulta na cobrança de elaboração de aplicações extensíveis, flexíveis e adaptáveis.

Os aspectos enunciados anteriormente podem ser alcançados através de diferentes abstrações e técnicas de engenharia de software, algumas delas são: arquiteturas baseadas em componentes, em *plug-ins*, entre outras [14]. A primeira se caracteriza por favorecer o reuso de código pela decomposição de seus elementos em partes minimamente dependentes, os quais possuem uma interface de comunicação bem definida. A arquitetura baseada em *plug-ins* é atrativa para os desenvolvedores pelo foco dado à modularização de funcionalidades. Qualquer um pode rapidamente personalizar aplicações combinando *plug-ins* existentes, ou escrevendo novos, na ausência de funções desejadas. Uma importante diferença entre arquiteturas baseadas em *plug-ins* e arquiteturas baseadas em componentes é que *plug-ins* são opcionais ao invés de imprescindíveis [14]. A figura 2.2 ilustra a estrutura de arquiteturas baseadas em *plug-ins*.

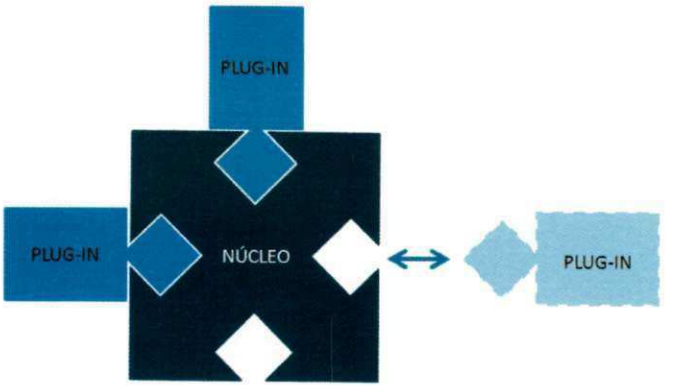


Figura 2.2: Visão geral de arquiteturas de *software* baseadas em *plug-in*.

Arquiteturas baseadas em *plug-ins* tornou-se uma abordagem comum para obter extensibilidade. *Browsers* modernos como Firefox[3] e Konqueror[4] possuem um mecanismo de carregamento de *plug-ins*, os quais permitem ao *browser* exibir novos tipos de conteúdos e gerenciar as preferências do usuário. Porém estes não são apenas componentes extras para aplicativos. Atualmente, existem software desenvolvidos inteiramente usando essa abstração, como o Eclipse[5] e o Jedit[6].

Plug-ins são blocos de software dinamicamente conectados através de interfaces e mecanismos de extensão e, em sua maioria, não são compilados dentro da aplicação a que servem. Estes implementam funções que são reconhecidas e ativadas de acordo com a necessidade. Não são programas autônomos; sua execução é gerenciada por um mecanismo de carregamento [12]. Este mecanismo compõe o núcleo do sistema que possui o mínimo de funcionalidades que a aplicação necessita para executar. Portanto, aplicações baseadas em *plug-ins* podem executar independentemente de possuírem alguma extensão instalada [33]. O preço a ser pago por tamanha flexibilidade é o suporte propiciado pelo núcleo, aos seguintes requisitos básicos: (i) descobrimento, carregamento e execução do código do *plug-in* apropriado; (ii) desassociação de *plug-ins*; (iii) persistência dos *plug-ins* instalados com suas respectivas funções; e (iv) gerenciamento das dependências associadas ao *plug-in*.

Esta técnica é usada para guiar projetos de software que possuem as seguintes exigências:

[3]http://br.mozdev.org/
[4]http://www.konqueror.org/
[5]http://www.eclipse.org/
[6]http://www.jedit.org/

simples adição de funcionalidades ao sistema; decomposição em módulos, de maneira que apenas partes indispensáveis para resolver uma situação peculiar sejam usadas; atualização sem necessidade de parar e reiniciar o serviço; e incorporação de extensões desenvolvidas por terceiros [14].

Capítulo 3

Trabalhos relacionados

Neste capítulo são apresentados os principais trabalhos relacionados à infraestrutura apresentada neste trabalho para desenvolvimento de aplicações baseadas em localização extensível a múltiplos domínios.

3.1 LAISYC

O sistema de informação baseado em localização, denominado LAISYC [6], disponibiliza serviços personalizados em tempo-real baseados na localização atual do usuário e no seu histórico de viagens. Este sistema suporta aplicações distribuídas inteligentes, concentrando-se no desempenho do cliente móvel a fim de minimizar o consumo de energia do dispositivo quanto à utilização dos recursos disponibilizados, como GPS e Wi-Fi. A estrutura foi desenvolvida para a plataforma Java ME e segue as especificações estabelecidas pela mesma. Além disso, a solução utiliza protocolos padrões de rede o que possibilita o desenvolvimento de software por terceiros.

A solução é baseada em uma arquitetura cliente/servidor e a comunicação entre as camadas é feita por meio do protocolo HTTP (quando as informações não estão relacionados à localização) ou UDP (no caso das informações transportadas estarem relacionadas à localização). A aplicação cliente possui uma arquitetura baseada em componentes divididos em duas categorias: gerenciamento de sistemas de posicionamento e de comunicação. O primeiro é responsável por: obter a localização do dispositivo; combinar informações provenientes das diferentes tecnologias para estimar a posição atual do dispositivo, caso os dados de localiza-

ção do sistema de posicionamento principal não estejam disponíveis; e ajustar a frequência de recálculos da posição para evitar trabalho desnecessário e, consequentemente, desperdício da energia da bateria. O segundo componente encapsula e criptografa a localização obtida; e gerencia o módulo de sessão, que mantém dados de conexão com o servidor.

O servidor também é dividido em dois componentes: gerenciamento de comunicação e análise de dados de localização. Assim como na aplicação cliente, o componente de gerenciamento de comunicação manipula dados da sessão aberta entre cliente e servidor. Já o último componente, como o próprio nome sugere, utiliza a localização atual do usuário, assim como o histórico de viagens para prever o caminho que ele pode seguir em um futuro próximo.

A infraestrutura desenvolvida permite a criação de aplicações que necessitem ou usufruam de rotas traçadas, como por exemplo um assistente virtual para viagens. O diferencial deste trabalho em comparação com o LAISYC é que o último não possibilita a adição de domínios de aplicações diferentes ao sistema e não disponibiliza uma API para implementação de novas aplicações.

3.2 *Framework* para desenvolver aplicações baseadas em localização distribuídas

O *framework* apresentado em [31] é uma solução aberta para criação de aplicações baseadas em localização. A infraestrutura provida oferece suporte para visualização de lugares e caminhos nas redondezas da localização do usuário. Entenda-se por lugar, as posições geográficas de interesse; e caminho, um conjunto de lugares identificados pelos mesmos metadados, podendo ser uma rua, instruções de como chegar de um ponto a outro, um passeio de bicicleta, etc. Nessa solução o usuário é capaz de adicionar novos lugares e caminhos bastando, para isso, informar um metadado associado àquela localização.

A arquitetura criada para disponibilizar esses serviços possui três módulos: o cliente móvel, o servidor de aplicação e o banco de dados contendo informações necessárias a aplicação. A comunicação entre os módulos é feita utilizando-se o conceito de Web-services[1]. A aplicação móvel obtém a posição geográfica do dispositivo, encapsula em um envelope

[1]Sistema de software projetado para suportar a interoperabilidade na interação máquina-máquina em uma rede de comunicação

SOAP[2], o qual é processado pelo servidor de aplicações retornando os resultados obtidos. O servidor de aplicação se comunica com o banco de dados a fim de obter informações da localização.

As principais diferenças entre este *framework* e a infraestrutura desenvolvida neste trabalho, são: (i) as informações possíveis de serem disponibilizadas não podem ser generalizadas para qualquer domínio pois se baseiam no conceito de lugares, assim o *framework* pode ser visto como um servidor de pontos-de-interesse (POI[3]); (ii) a informação não é personalizada, na realidade não existe uma entidade que represente um usuário, ou seja, João e José recebem as mesmas informações, bastando estarem na mesma posição geográfica (informações abrangentes); e (iii) o *framework* não oferece um mecanismo simples de adição de novos servidores de aplicação.

3.3 Sistema para entrega de propaganda

O sistema de entrega de propagandas baseado em localização apresentado em [10], trata-se de uma infraestrutura para permitir o direcionamento de propagandas para os usuários mais adequados considerando a posição geográfica em que eles se encontram. O sistema provê métodos de adicionar usuários e anunciantes, sendo necessária a identificação destes antes de utilizar qualquer funcionalidade. Nesse processo, o usuário especifica suas áreas de interesse e sua agenda. A localização, encontrada usando-se GPS, é combinada com aquelas informações a fim de determinar as propagandas com maior probabilidade de interessar ao usuário. Após concluída essa inferência, um SMS é enviado ao dispositivo contendo um lembrete das atividades presentes na agenda cadastrada acompanhado da propaganda relevante.

A infraestrutura desenvolvida possui uma arquitetura cliente/servidor simples, composta por uma base de dados com as informações dos usuários e dos anunciantes, um servidor de aplicação e uma interface WEB para acesso e atualização dessas informações.

Apesar de analisar o perfil do usuário para oferecer uma propaganda, esta é uma solução para uma área de aplicação específica e não é extensível para outros domínios. Além disso,

[2] *Simple Object Access Protocol* é um protocolo para troca de informações estruturadas em uma plataforma descentralizada e distribuída

[3] *Point of Interest* - conceito utilizado para definir pontos geográficos de utilidade pública, e.g. hospitais, escolas, restaurantes, etc.

os autores não discorreram sobre o aspecto de escalabilidade do sistema desenvolvido.

3.4 Discussão sobre Trabalhos Relacionados

Apesar das soluções enunciadas utilizarem informações de localização, estas são pouco flexíveis, relacionando-se a um campo de aplicação específico. Além disso, em algumas soluções o conhecimento de quem acessa o sistema é um fator desprezível, mesmo sendo uma informação de contexto importante. Diversos outros trabalhos foram encontrados, mas não foram considerados tão importantes quanto os citados anteriormente. A maioria deles é focada em um domínio de aplicação específico [2, 5, 7, 15, 27, 39, 40, 47] ou voltado apenas à aquisição de dados de localização [9, 30, 43, 44, 54, 55]. Diante desse quadro, não há uma abordagem isolada que forneça as seguintes funcionalidades: (i) suporte a múltiplos domínios; (ii) disponibilização de informação relevante baseada na localização do usuário; e (iii) API de adição de novas aplicações relacionadas ao domínio.

Na Tabela 3.1 são sumarizadas as funcionalidades apresentadas por cada uma das soluções analisadas anteriormente.

Requisitos/Trabalhos	Suporte à múltiplos domínios	Informação personalizada	API
LAISYC [6]	Não	Sim	Não
Framework para desenvolver aplicações baseadas em localização distribuídas [31]	Não	Não	Sim
Sistema para entrega de propaganda [10]	Não	Sim	Não

Tabela 3.1: Análise dos trabalhos relacionados

Capítulo 4

Voilà

4.1 Visão Geral

A infraestrutura desenvolvida neste trabalho disponibiliza informações relevantes ao usuário, com base em sua localização geográfica, provenientes de múltiplos domínios. Entenda-se por domínio uma área de aplicação específica, como turismo, condições meteorológicas, comércio eletrônico, publicidade, entre outras.

Em alto nível, esta infraestrutura possui uma arquitetura cliente/servidor composta pelas seguintes entidades: aplicação cliente móvel, servidor e provedores de dados, como ilustrado na Figura 4.1.

A interação com o sistema começa quando o usuário requisita informações a respeito de uma dada localização. O servidor recebe esta solicitação (através da Internet), recupera dados da localização enviada e atualiza o histórico de requisições daquele usuário. A resposta é, então, formatada de acordo com as preferências do solicitante e exibida no dispositivo.

Através de dispositivos móveis, os usuários fornecem as localizações das quais desejam obter informações. Isso é provido a partir de uma interface gráfica onde a localização é apontada em um mapa. Caso a informação se relacione à posição geográfica atual do usuário, esta pode ser obtida utilizando diferentes sistemas de posicionamento. Além disso, localizações específicas podem ser adicionadas aos "Favoritos", bastando indicar a localidade cadastrada para obter informações sobre a mesma.

Usufruindo do histórico e do perfil do usuário, o sistema pode prever requisições futuras, antecipando a ação do mesmo e aprimorando a relevância da resposta. Por exemplo, caso

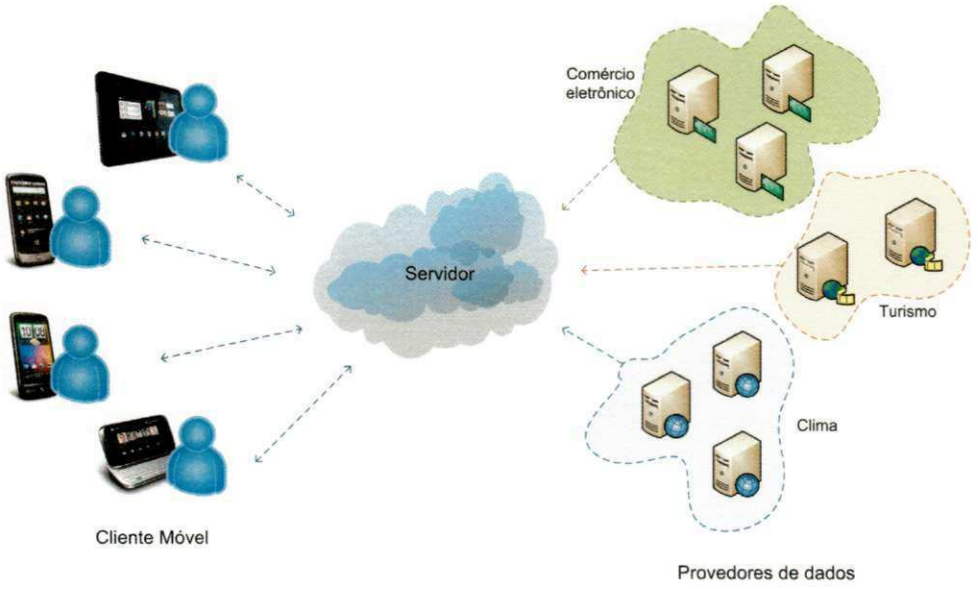


Figura 4.1: Visão geral da arquitetura da infraestrutura.

o usuário solicite, com uma certa frequência, informações de João Pessoa, toda sexta-feira em um determinado horário, o sistema, proativamente, pode recuperar dados para aquela localidade e enviar ao cliente móvel, sem que para isso o usuário interaja com a aplicação.

Com base no ciclo de interação descrito acima, o cliente móvel é responsável por fornecer dados do usuário, armazenados no servidor; e consumir informações através do acesso aos domínios disponíveis.

Os provedores de dados, como o próprio nome sugere, são responsáveis por disponibilizar dados de domínios específicos ao servidor, associados a uma determinada área de cobertura geográfica. No domínio de meteorologia, um exemplo desses dados seria a temperatura atual de uma localização específica. Nesse caso, existem várias organizações que fornecem dados climáticos através da Internet, para diferentes regiões do mundo, como o INMET (Brasil), e NWS (EUA), entre outros. Além dessas organizações, é possível receber dados de estações meteorológicas pessoais. Qualquer fonte capaz de disponibilizar dados para o servidor é considerada um provedor de dados. O mesmo conceito serve para outros domínios: provedores de dados de cultura, turismo, tráfego, etc.

O servidor, por sua vez, exerce o papel de gerenciar, de forma transparente, a interação entre o cliente móvel e os provedores de dados, recebendo dados do último e os enviando ao primeiro. É ele que cria, armazena e manipula o perfil dos usuários, utilizados para otimizar

o desempenho na recuperação de informações, além de, antecipar futuras solicitações, como a exemplificada anteriormente. Do ponto de vista do usuário, a existência do provedor de dados é transparente, ou seja, o servidor e o provedor formam um módulo só.

Considerando os aspectos acima mencionados, a infraestrutura mantém um conjunto de "nuvens" de diferentes domínios e um banco de dados distribuído com informações dos usuários. A arquitetura do servidor e os aspectos envolvidos na obtenção da escalabilidade do sistema são discutidos na seção seguinte, seguido da explicação da arquitetura do cliente móvel.

4.2 Servidor

Em uma visão simplificada, o sistema é formado por vários servidores, ou nuvens. Cada um deles armazena um tipo específico de dados, distribuídos de acordo com a posição geográfica. Por exemplo, um servidor implantado no Brasil armazena dados relacionados ao Brasil e/ou regiões próximas. Isso significa dizer que servidores instalados em diferentes localidades possuem dados diferentes. A mesma ideia é usada no acesso a esses dados, ou seja, caso a requisição de um usuário seja oriunda de algum lugar do Brasil, esta sempre será atendida por um servidor localizado no Brasil.

O principal tipo de servidor do sistema é o Núcleo. Este é responsável por armazenar dados dos usuários, incluindo informações pessoais, localidades adicionadas aos Favoritos e dados de autenticação, como *username* e senha. Considerando a arquitetura baseada em *plug-ins*, o Núcleo representa o mecanismo de acoplamento de *plug-ins*. Os outros tipos de servidores, os quais implementam funcionalidades e armazenam dados relacionados a um domínio específico, são implantados como *plug-ins*. Os últimos possuem dependência com o Núcleo no acesso a informações do usuário que são disponibilizadas através de uma API baseada em *Web Services*[1].

Como mencionado anteriormente, os dados são distribuídos dentre os servidores baseados na localização geográfica. O papel de distribuir e acessar esses dados é exercido pelo DNS (*Domain Name Server*) e pelos Servidores de Diretório. O DNS é responsável por es-

[1] Programas modulares, independentes e auto-descritivos, que podem ser descobertos e requisitados através da Internet ou de uma intranet corporativa.

colher o Servidor de Diretório apropriado para processar a requisição originada da aplicação cliente baseado na localização do solicitante, a qual é obtida por meio do endereço IP. Nesse sentido, o Servidor de Diretório é o mais próximo do usuário.

O *gateway* da nuvem completa a lista de elementos arquiteturais. No contexto deste trabalho, uma nuvem é um *cluster* de servidores de um mesmo tipo, com as mesmas funcionalidades, possuindo dados replicados e sincronizados. Os *gateways* da nuvem são responsáveis por processar e redirecionar requisições para um servidor específico dentro desta. A seleção deste servidor considera balanceamento de carga e tolerância a falhas, evitando a sobrecarga e provendo suporte a *failover*[2]. Estes são elementos opcionais e dependem da necessidade de replicação de servidores.

Na Figura 4.2 ilustra-se a comunicação dos componentes arquiteturais citados anteriormente, sem *plug-ins* de domínio. Na subseção seguinte os módulos que compõem a arquitetura do servidor são detalhados. As nuvens específicas de domínios, implementadas por meio de *plug-ins*, são comentadas no próximo capítulo através do estudo de caso *Weather*.

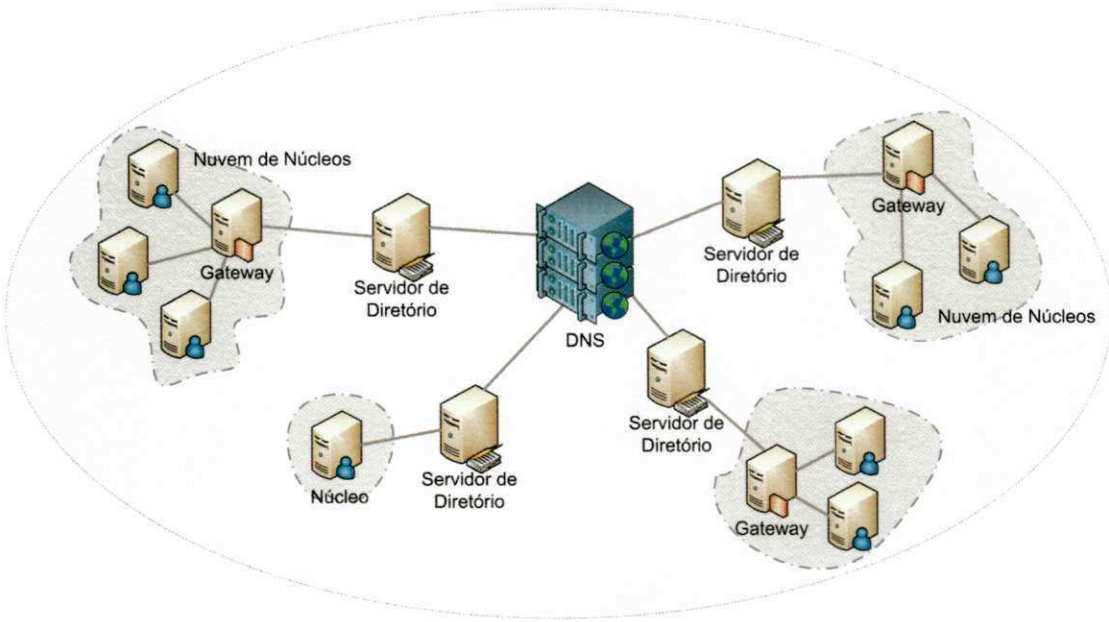


Figura 4.2: Visão da arquitetura sem *plug-ins* de domínio acoplados.

[2]O processo no qual uma máquina assume os serviços de outra.

4.2.1 Módulos

Domain Name Server

O DNS fornece o primeiro nível de balanceamento de carga e tolerância a falha do sistema. Este funciona baseado em três parâmetros: o endereço IP da aplicação cliente, o endereço de acesso a infraestrutura e a lista de endereços IP dos Servidores de Diretório disponíveis. Este módulo foi desenvolvido usando o PowerDNS[3], o qual é compatível com a especificação RFC 1035 [41] que rege a implementação de servidores DNS.

Quando a aplicação cliente envia uma requisição, o DNS determina a URL[4] recebida a um endereço IP de um Servidor de Diretório específico, consultando a localização através da base de dados do GeoIP[5], como ilustrado no fluxograma da Figura 4.3. Assim, a aplicação cliente se conecta com o Servidor de Diretório mais próximo, no que concerne a distância geográfica, e, consequentemente, com a nuvem mais próxima.

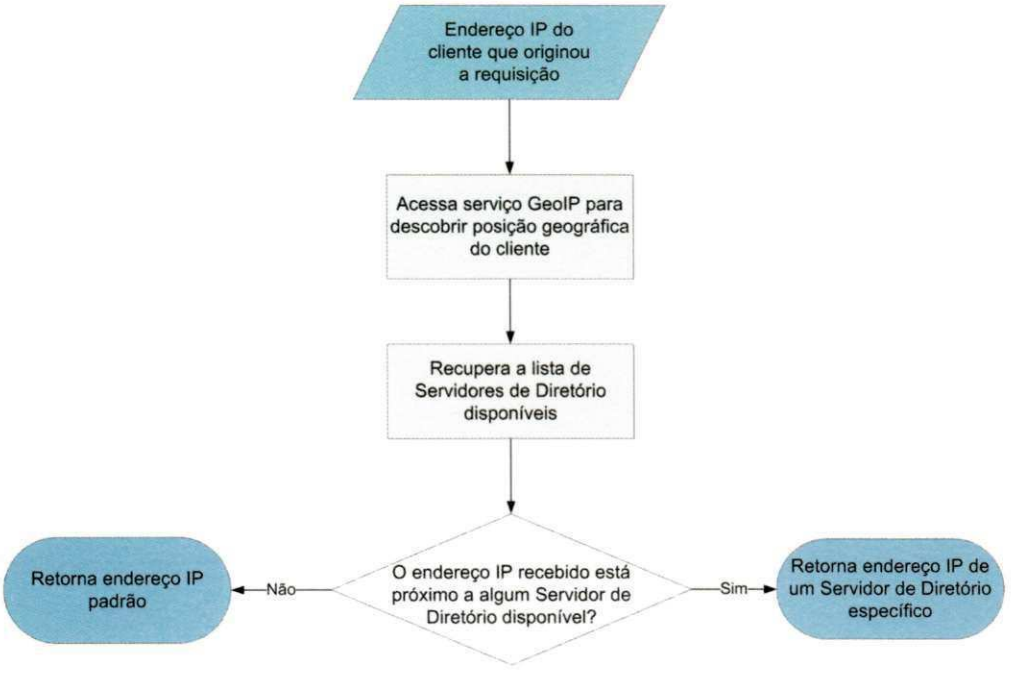


Figura 4.3: Fluxograma de obtenção do endereço IP apropriado.

[3] http://www.powerdns.com/

[4] *Uniform Resource Locator* - endereço de um recurso disponível em uma rede

[5] GeoIP fornece uma maneira não-invasiva para determinar em tempo real informações geográficas por meio da Internet. Este pode indicar qual país, região, cidade, código postal, e latitude/longitude através do endereço IP usado para acessar a rede

Com o intuito de determinar se um Servidor de Diretório está online ou offline, o DNS utiliza um *daemon* o qual atualiza constantemente a lista de endereços IP dos servidores, mantendo apenas aqueles que estão disponíveis. Esse processo é esboçado na Figura 4.4. Esta lista é armazenada em um banco de dados e usada pelo DNS quando este recebe uma solicitação. Ademais, neste mesmo banco de dados é guardada a localização geográfica dos Servidores de Diretório, usada para determinar o endereço IP que será retornado a aplicação cliente baseado no resultado da consulta ao serviço de GeoIP.

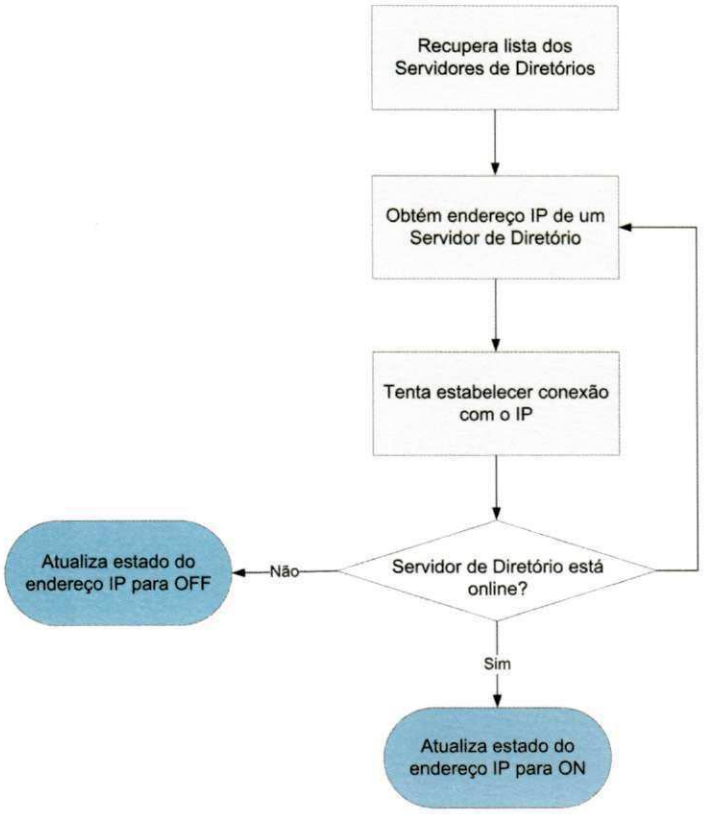


Figura 4.4: Fluxograma de manutenção dos Servidores de Diretório disponíveis.

Servidores de Diretório

Um Servidor de Diretório é responsável por enviar a requisição recebida para uma nuvem, de acordo com os parâmetros da solicitação. A requisição é examinada a fim de identificar o *host* para o qual esta será encaminhada. Considerando que cada tipo de nuvem possui uma palavra de identificação específica, o caminho da requisição é analisada em busca desse descritor. Desta forma, solicitações com a palavra "*core*" são redirecionadas para uma nu-

vem de Núcleo, com a palavra "*turism*", para uma nuvem no domínio de turismo, e assim sucessivamente.

Este serviço possui uma tabela que mapeia caminhos para *hosts*, a fim de possibilitar o funcionamento com diferentes domínios. Para cada requisição, este procura em sua tabela pelo dado caminho e redireciona-a para o *host* apropriado, como ilustrado na Figura 4.5. Os Servidores de Diretório usados na solução foram implementados através de um servidor Apache *Web* como um *proxy* reverso usando *mod proxy*.

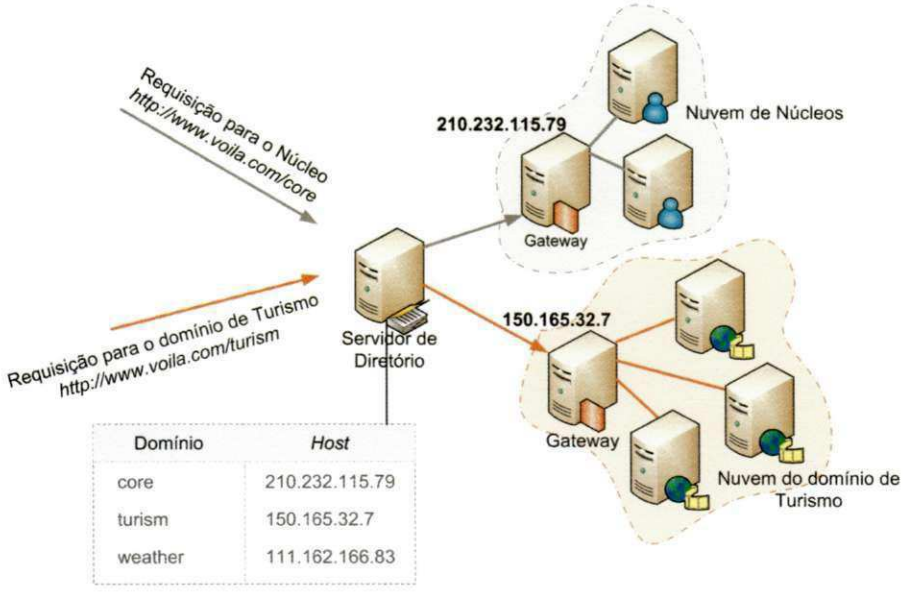


Figura 4.5: Visão geral da arquitetura da infraestrutura.

Cada um destes elementos deve registrar apenas uma instância de um determinado tipo de servidor, ou seja, este pode manter vários domínios diferentes, mas apenas um "ponto de entrada" para cada um deles. Essa decisão foi tomada com base no fato de que este não possui caráter de balanceamento de carga, é apenas o encarregado de escolher um domínio de acordo com a requisição do usuário.

Núcleo

O Núcleo constitui o principal servidor do sistema, este armazena dados independentes de domínio os quais incluem o perfil do usuário, localidades de interesse (adicionados aos Favoritos) e informações de autenticação. Uma nuvem de Núcleos possui um ou mais servidores replicados a fim de promover tolerância a falhas.

O banco de dados do servidor é distribuído e cada nuvem possui dados de usuários diferentes. A decisão de em qual nuvem os dados de um determinado usuário vão ser armazenados é baseado no somatório do código *hash*[6] do *username* por ele fornecido, utilizando, mais especificamente, o algoritmo de criptografia MD5[7], a fim de evitar uma massiva quantidade de dados em apenas uma nuvem.

As funcionalidades do Núcleo estão disponíveis através de três tipos de *Web Services*, discutidos a seguir:

- Público - serviços disponíveis para a aplicação cliente móvel.
- Interno - serviços fornecidos para entidades do lado servidor. A principal operação realizada é a autenticação do usuário a cada requisição recebida por um servidor da infraestrutura.
- Privado - serviços para uso exclusivo na comunicação entre servidores do mesmo tipo. Apesar de possuir as mesmas funcionalidades dos outros dois tipos, este serviço executa localmente (não encaminha a operação para outros servidores) evitando *loops* de requisições no sistema.

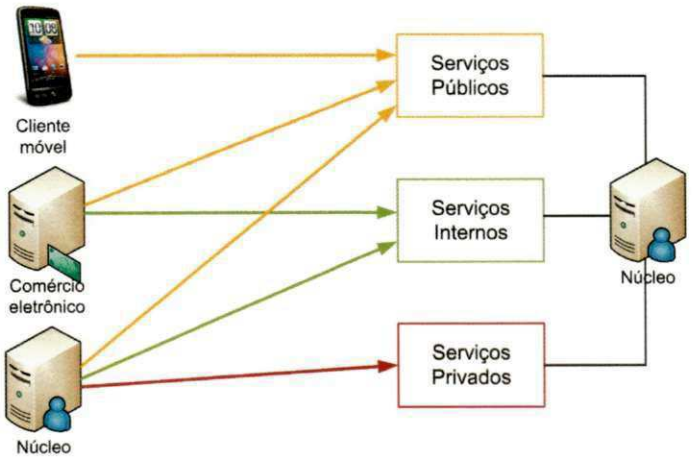


Figura 4.6: Tipos de serviços disponibilizados pela infraestrutura.

[6]Uma espécie de assinatura ou impressão digital que representa o conteúdo de um fluxo de dados.

[7]*Message-Digest algorithm 5* é uma função de criptografia *hash* de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc. Especificada na RFC 1321 [49], o MD5 tem sido empregado em uma grande variedade de aplicações de segurança, verificação de integridade de dados e *logins*.

As permissões de acesso aos diferentes tipos de serviços estão ilustradas na Figura 4.6. Os serviços públicos estão disponíveis para todos os elementos da infraestrutura, sendo solicitados, principalmente, pela aplicação cliente. Já os serviços internos e privados podem ser acessados por servidores de qualquer tipo e por servidores do mesmo tipo, respectivamente. Por exemplo, caso o servidor que deseja-se obter informações seja o Núcleo, um servidor do domínio de comércio eletrônico pode acessar qualquer serviço interno, porém não os privados os quais são exclusivos para servidores do tipo Núcleo.

O núcleo disponibiliza os *Web Services* usando o protocolo SOAP[8], possuindo dois WSDL[9] públicos principais: um para autenticação e outro para manipulação dos Favoritos. O primeiro permite a identificação do usuário através do *login* no sistema, e saída por meio do *logout*. O último possibilita a realização das operações de adicionar, remover ou atualizar uma localidade nos Favoritos. Os serviços disponibilizados pelo Núcleo são sumarizados na Tabela 4.1.

Visibilidade do serviço	Serviço	Descrição
Público	*login*	Verifica o *username* e a senha do usuário e cria um identificador de sessão (*cookie*)
	logout	Remove o identificador criado no *login*
	getBookmarks	Recupera as localidades dos "Favoritos"
	createBookmark	Adiciona uma nova localização aos "Favoritos"
	updateBookmark	Atualiza informações de uma localização contida nos "Favoritos"
	removeBookmark	Remove uma localização contida nos "Favoritos"
Interno	*authenticate*	Autentica o usuário através do *cookie*

Tabela 4.1: Serviços disponibilizados pelo Núcleo da infraestrutura.

[8]Procolo projetado para invocar aplicações remotas através de RPC (*Remote Procedure Calls*) ou trocas de mensagens, em um ambiente independente de plataforma e de linguagem de programação.

[9]Linguagem baseada em XML utilizada para descrever *Web Services*.

Gateway da Nuvem

Os *gateways* da nuvem são incumbidos de processar e enviar requisições recebidas dos Servidores de Diretório para um servidor específico dentro da nuvem. Este pode ser usado tanto em nuvens de Núcleos, quanto naquelas de domínios específicos, a medida que balanceamento de carga e tolerância a falhas se façam necessários.

O processo de seleção de um servidor é baseado na quantidade de requisições sendo processadas a fim de evitar sobrecarga dos servidores e reduzir a perda de desempenho nas respostas. Além disso, este implementa um mecanismo de *failover* com o objetivo de tornar as falhas de requisições transparentes à aplicação cliente, ou seja, caso um servidor esteja *offline* e não possa atender a solicitação, o próprio *gateway* se encarrega de redirecioná-la a outro disponível, como ilustrado na Figura 4.7.

Quando um *Gateway* recebe uma solicitação, observa o estado dos servidores na nuvem, identificando aqueles que estão indisponíveis e encaminhando-a a um *online*. Este acompanha periodicamente as condições de disponibilidade dos elementos da nuvem, estabelecendo conexões TCP na porta 80, para detectar falha em servidores. Além disso, o sistema utiliza o protocolo SNMP[10] para verificar o uso do processador e da memória de um determinado servidor, promovendo balanceamento de carga.

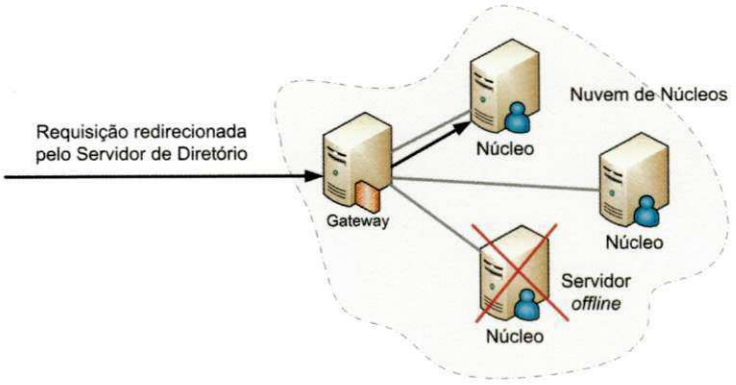


Figura 4.7: Mecanismo de tolerância a falhas.

[10] *Simple Network Management Protocol* - protocolo de gerência típica de redes UDP, que possibilita administrar o desempenho da rede, encontrar e resolver seus eventuais problemas, e fornecer informações para o planejamento de sua expansão.

4.2.2 Escalabilidade

Escalabilidade é uma propriedade que o sistema possui de permanecer efetivo mesmo quando há um aumento significativo do número de recursos e de usuários [17]. Para alcançar essa propriedade, deve-se considerar a abundância de dispositivos, de interações e de componentes, evitando soluções centralizadas. Por esse motivo, optou-se por utilizar uma arquitetura distribuída com nuvens de servidores replicados, possibilitando balanceamento de carga e tolerância a falhas.

Escalabilidade, no sentindo mais amplo, é um problema crítico em computação pervasiva. Nesse sentindo, é essencial reduzir interações com recursos e entidades distantes geograficamente. Essa preocupação é o que norteia o conceito de escalabilidade localizada [51], por mais que isso entre em discordância com a atual orientação de transparência da rede, na qual o acesso a recursos locais e remotos é realizado com operações idênticas independente da sua localização física. Apesar de um usuário gerar requisições a recursos distantes com informações relevantes para ele, a preponderância de interações será local. Desta forma, deve-se considerar a posição física de recursos e priorizar interações locais em contraposição as distantes. O servidor DNS desenvolvido neste trabalho utiliza esse conceito a fim de diminuir o tempo de resposta.

O balanceamento de carga, como comentado anteriormente, pode ser observado: (i) no momento que o DNS recebe uma requisição e redireciona para a nuvem mais próxima; (ii) na atribuição de processamento da solicitação a um dos servidores da nuvem realizada pelo *gateway*; e (iii) na alocação dos dados do usuário baseado no seu *username*.

A natureza dos dados em sistemas baseados em localização é distribuída, pois estes são diretamente relacionados com a posição geográfica. Ademais, considerando a grande quantidade de dados, não é possível manter um único servidor ou *cluster*. Além disso, os dados são constantemente atualizados, tornando inviável replicá-los entre servidores distribuídos mundialmente. O uso de bancos de dados distribuídos tem sido aplicado com sucesso em sistemas de larga-escala na Internet, como eBay[11] e Amazon[12], dada a consolidação dessa solução. Este trabalho segue a mesma ideia.

[11] http://www.ebay.com/

[12] http://www.amazon.com/

4.3 Aplicação Cliente

A aplicação cliente móvel é composta por módulos que possibilitam o acesso direto a infraestrutura elaborada, através de uma interface gráfica baseada em mapas com serviços de navegação. Por meio desta aplicação os usuários requisitam informações de um determinado domínio, especificam suas preferências e cadastram localidades aos "Favoritos". A arquitetura do cliente é baseada em *plug-ins*, a fim de possibilitar a adição de novos domínios, seguindo a mesma ideia do servidor, e está ilustrada na Figura 4.8. Na subseção seguinte, cada módulo é detalhado.

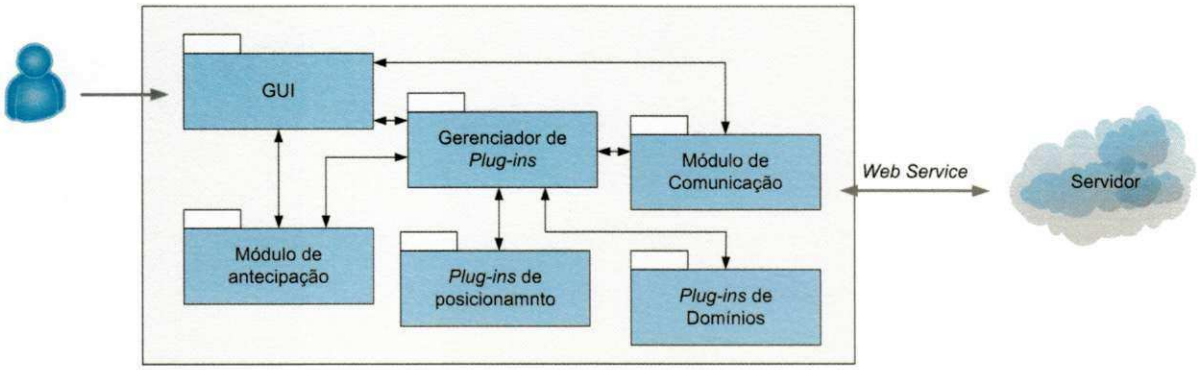


Figura 4.8: Arquitetura da aplicação cliente.

4.3.1 Módulos

Interface Gráfica

O módulo da GUI é composto pelos elementos de interface gráfica, os quais permitem a interação amigável do usuário com as funcionalidades da aplicação cliente. Esse módulo contem todas as classes das janelas e diálogos, incluindo menu para atualização das preferências do usuário para um determinado domínio, botões de adição de localidades aos "Favoritos", e de obtenção de informação. Além disso, por meio do Gerenciador de *plug-ins*, este módulo é responsável por manipular os *plug-ins*, que podem ser habilitados e desabilitados de acordo com a conveniência do usuário, exibindo elementos de interface.

O fluxo da aplicação é mantido pela GUI, a qual interage com o módulo de Comunicação quando há necessidade de solicitação de serviços providos pela infraestrutura do servidor,

como por exemplo no momento de autenticação do usuário. Ademais, com o intuito de exibir a posição atual do dispositivo, este módulo requisita a obtenção da localização geográfica aos *plug-ins* de posicionamento, por meio do gerenciador destes.

Este módulo também é responsável por desenhar e manter os elementos do mapa. O que significa dizer que todas as ações realizadas pelo usuário sobre o mapa, como *zoom* e *panning*, são manipuladas pela GUI. Além disso, este módulo exibe e os componentes do serviço de navegação, requisitando em uma certa periodicidade de tempo, a localização atual do usuário, a fim de prover informação de direção e orientação.

A principal tela da interface gráfica é definida em camadas exibidas sobre o mapa. Cada item diferente, é desenhado em uma camada diferente. Por exemplo, o ícone representando a localização do usuário compõe uma camada, os ícones dos favoritos, outra camada, e assim sucessivamente. Ou seja, cada domínio, possui sua camada sobre o mapa, na qual os dados relacionados são mostrados.

Módulo de Comunicação

O módulo de comunicação é a interface entre a aplicação cliente e a infraestrutura do servidor, sendo o responsável por controlar as requisições aos serviços disponibilizados pelo sistema. Este módulo fornece um mecanismo de acesso para as solicitações independentes e dependentes de domínio, por meio do Gerenciador de *plug-ins*.

Informações necessárias para criação das requisições, como a URL da infraestrutura, nome dos WSDL, entre outras, são mantidas em um arquivo de configuração editável pelo usuário. Essas informações são acessadas no momento da autenticação do usuário, guardadas na memória, e usadas a cada nova requisição. Esse mesmo processo é utilizado para os *plug-ins* de domínio, que possuem seu próprio arquivo de configuração, acessíveis através do módulo gerenciador.

Gerenciador de *plug-ins*

O gerenciador de *plug-ins* é composto por um mecanismo de carregamento automático de *plug-ins* de domínio e de posicionamento. Responsável por oferecer uma interface de acesso e comunicação dos *plug-ins* pelos outros módulos da aplicação cliente, este módulo é executado na inicialização da aplicação, descobrindo e registrando os componentes compatíveis

com uma interface definida previamente. Nesse caso, o registro permite que, através da interface gráfica, o usuário pode ativar ou desativar um determinado *plug-in*. Os dois tipos de extensão especificados na arquitetura são comentados a seguir:

- *Plug-ins* de domínio: são extensões que fornecem informações dos diferentes domínios disponíveis na infraestrutura. Para cada um deles, desenvolve-se um *plug-in* associado, criando tanto uma nova camada sobre o mapa, quanto menus de atualização das preferências do usuário. A nova camada exibe as informações obtidas através da comunicação com o servidor do domínio.

- *Plug-ins* de posicionamento: são extensões que obtêm a localização geográfica do dispositivo usando diferentes sistemas de posicionamento. Como comentado na Seção 2.2.1, existem muitas formas de descobrir a posição de um dispositivo, e a arquitetura deve ser flexível no que tange a utilização desses sistemas. A determinação de qual *plug-in* usar é baseada em uma ordem de prioridade atribuída a cada um, assim como na disponibilidade do recurso.

Módulo de antecipação

O módulo de antecipação é o responsável por adquirir informações dos domínios de forma proativa. Para cada domínio existente, este obtêm e mantém em um repositório local, as listas com as regras de associação correspondentes, geradas a partir do histórico de requisições do usuário, logo após o processo de autenticação.

Este módulo possui um mecanismo de antecipação, que requisita proativamente informações à infraestrutura do servidor, obedecendo as condições das regras de associação obtidas para cada domínio. Com o intuito de não interromper a interação do usuário com a aplicação, este mecanismo é executado em segundo plano de forma paralela.

4.4 Conclusão

Neste capítulo foi apresentada a visão geral do projeto da Infraestrutura Escalável para o Desenvolvimento de Aplicações Baseadas em Localização e Orientadas a Domínios, denominada Voilà. As entidades envolvidas (cliente móvel, servidor, provedor de dados) na

estrutura e os papéis desempenhados pelos mesmos foram apresentados como "caixas pretas", de modo a esclarecer como todo o sistema funciona. Em seguida, o lado servidor e a aplicação cliente foram esmiuçados em módulos e explicados detalhadamente. A prova de conceito e a validação da infraestrutura foram realizados por meio de um estudo de caso para o domínio de clima, denominado *Weather*, o qual é discutido no capítulo 5.

Capítulo 5

Estudo de caso

Neste capítulo é apresentado o desenvolvimento de um estudo de caso para validar o suporte da infraestrutura à criação de aplicações para sistemas baseados em localização com múltiplos domínios, denominado *Weather*. Inicialmente, é realizada uma breve descrição do estudo de caso, seguida de uma explicação dos elementos criados no servidor e na aplicação cliente, a fim de demonstrar o uso da infraestrutura elaborada.

5.1 Descrição do estudo de caso

Weather é uma implementação no domínio de meteorologia usando a infraestrutura descrita no capítulo anterior, que possui como objetivo disponibilizar informações climáticas (e.g. temperatura, umidade, pressão atmosférica) a respeito de uma determinada localização geográfica a um usuário móvel. Um cenário de aplicação da solução desenvolvida é apresentado a seguir:

Juliana, cansada de uma longa semana de trabalho, está se programando para realizar uma viagem no final do expediente da sexta-feira. Decidida a sair da rotina, ela chega em casa para arrumar sua mochila e fica em dúvida sobre o que colocar pois não sabe se está chovendo ou não no litoral. Para agilizar a partida, pega seu smartphone e informa a posição geográfica através de um mapa para uma aplicação instalada. A aplicação exibe no dispositivo dados das condições meteorológicas do ponto informado e Juliana descobre que o céu está aberto por lá, optando por levar roupas mais leves. Antes de sair, ela resolve checar como está a visibilidade na estrada, informando a localização da rodovia que dá

acesso a praia a qual deseja ir. Juliana, então, decide ir dirigindo até o hotel onde pretende descansar pois verifica que não irá chover e que a visibilidade está excelente.

Uma visão geral do percurso que as requisições de Juliana fazem antes dos resultados serem exibidos no dispositivo é explanado na Figura 5.1. Primeiramente, a solicitação (1) chega ao DNS, o qual retorna o endereço IP do Servidor de Diretório (2) mais próximo a Juliana. Este último determina que nuvem da infraestrutura é responsável por processar a requisição (4), encaminhando para um servidor do domínio de clima. O servidor, por sua vez, busca por informações climáticas para aquela localização nas fontes de dados disponíveis (6) e, caso encontre, retorna-as (7) formatadas de acordo com as preferências de Juliana. Toda comunicação e processamento realizados para atender a requisição é totalmente transparente para Juliana.

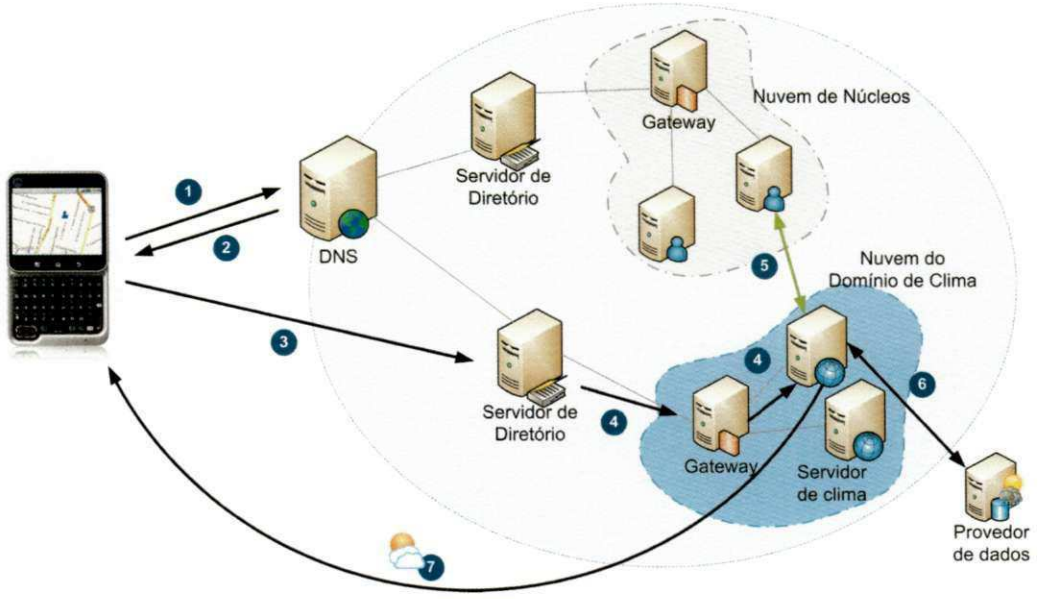


Figura 5.1: Arquitetura do servidor do domínio de clima com seus módulos.

Como pode-se perceber, é necessário dar suporte a esses serviços tanto do lado cliente, como do servidor. Logo, o desenvolvimento desse estudo de caso, foi divido em duas fases: criação da nuvem e dos serviços do domínio no servidor e adição dos *plug-ins* à aplicação cliente. Nas seções seguintes detalhes da implementação de ambos aspectos são apresentados.

5.2 Servidor

O domínio de clima foi adicionado a infraestrutura como uma nuvem, a qual caracteriza um *plug-in*. Com o objetivo de tornar pública a existência desse novo domínio para infraestrutura, registrou-se no Servidor de Diretório o ponto de entrada para o caminho contendo a palavra *weather*, através de uma interface *Web* de administração. Assim, esta nuvem pode usufruir dos mecanismos de balanceamento de carga, de tolerância a falhas, e escalabilidade da infraestrutura.

O desenvolvimento dos serviços para esse domínio foi auxiliado por um mecanismo de carregamento dinâmico de *Web Services* desenvolvido na linguagem de programação Python utilizando o framework Twisted[1] e a biblioteca soaplib[2].

A arquitetura do servidor do domínio de clima é composta por módulos responsáveis por obter informações de condições meteorológicas e gerenciar as requisições do usuário, como ilustrado na Figura 5.2. As requisições encaminhadas diretamente pelo Servidor de Diretório ou pelo *Gateway* da nuvem, são recebidas pelo módulo de Informação, o qual atua como gerenciador de requisições, selecionando o Provedor de Dados adequado, recuperando e retornando dados climáticos para uma determinada posição geográfica. O gerenciador de Provedores, por sua vez, oferece interfaces, baseadas em *Web Service*, para atualização e registro de fontes de informações de clima. Antes de processar as requisições, o usuário é autenticado, através do *cookie*, e identificado pelo módulo do Perfil do Usuário, que mantém dados específicos relacionados ao domínio, como tipo de informações e unidades preferenciais dos usuários. As solicitações atendidas são, então, gravadas no histórico de requisições, utilizado pelo módulo de Conhecimento na busca e definição de padrões, a fim de proativamente disponibilizar essas informações ao usuário. Além disso, com o intuito de reduzir o número de interações dentro do sistema, cada servidor mantém uma estrutura de dados replicada que lista informações de quais nuvens possuem dados climáticos para uma determinada área geográfica. Os módulos mencionados anteriormente são discutidos detalhadamente nas subseções seguintes.

[1] http://twistedmatrix.com/

[2] http://pypi.python.org/pypi/soaplib/

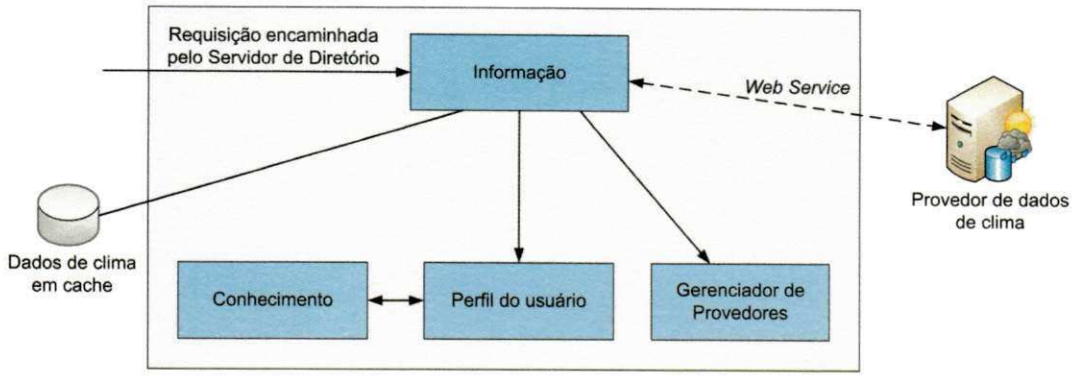


Figura 5.2: Arquitetura do servidor do domínio de clima com seus módulos.

5.2.1 Módulos

Módulo de Informação

O Módulo de Informação é o principal componente da arquitetura do servidor deste domínio, sendo responsável por processar as requisições do usuário no que tange a dados climáticos. Esse módulo mantém um sistema de *cache* local com informações recuperadas dos provedores de dados com a finalidade de reduzir o tempo de resposta de uma solicitação para uma mesma localidade.

Considerando que todos os usuários do sistema estão armazenados em servidores do tipo Núcleo, e que esse cria o identificador de sessão (*cookie*), usado para autenticar e identificar o usuário, no momento do *login*, as requisições originadas da aplicação cliente solicitam o serviço interno de autenticação ao núcleo apropriado, como ilustrado no passo 5 da Figura 5.1. Essa é a dependência que todos os outros tipos de servidores possuem com o Núcleo da arquitetura, pois é necessário saber quem é o usuário no provimento de informações relevantes.

A primeira ação realizada para processar uma requisição de informações climáticas é a determinação da nuvem mais apropriada para atendê-la, que nem sempre será a que está mais próxima do usuário. Assim, para cada item existente na lista replicada em todas as nuvens do domínio, contendo a latitude e longitude de referência das mesmas, é aplicada a equação de Harvesine[3] com base na localização recebida. Caso o resultado da fórmula

[3]Equação utilizada em sistemas de navegação que calcula a distância entre dois pontos do globo terrestre, a partir de dados da latitude e longitude.

remeta ao próprio servidor, este realiza uma busca na *cache* local por dados climáticos para aquela localidade. Se houver a ocorrência, a informação é retornada, se não, esta é solicitada ao provedor de dados, armazenada na *cache* e então encaminhada para a aplicação, de acordo com as preferências do usuário. Porém, caso o resultado do algoritmo remeta a outra nuvem, uma solicitação ao serviço privado de obtenção de dados climáticos é enviada e o mesmo processo, descrito na busca local, é realizado. O fluxograma da Figura 5.3 ilustra esse processo.

Em ambas situações, de processamento local ou remoto, caso o provedor de dados atribuído de fornecer informações para a localização estiver indisponível, este módulo solicita ao gerenciador de Provedores outras fontes que proveem cobertura à coordenada geográfica. Se existir um provedor para o ponto recebido, o processo de requisição é realizado. Caso contrário, o módulo de Informação notifica que não existe dados disponíveis para aquela localização.

Os serviços fornecidos por este módulo não se reduzem apenas a obtenção de dados climáticos. O módulo de Informação atua como um ponto de entrada no sistema, disponibilizando uma interface de acesso a todas as informações providas para o domínio de clima, como por exemplo a recuperação das preferências do usuário para exibição na aplicação cliente, entre outros. A Tabela 5.1 sumariza os serviços públicos oferecidos por este módulo através de *Web Service*. Este domínio possui apenas um serviço privado, o "*getWeatherLocal*", encarregado de buscar dados climáticos localmente.

Serviço	Descrição
getWeather	Obtêm informações a respeito de uma dada localização geográfica
updatePreferences	Atualiza as preferências do usuário
retrievePreferences	Recupera as preferências do usuário
retrieveTriggers	Recupera os padrões identificados na mineração das requisições

Tabela 5.1: Serviços públicos disponibilizados pelo para o domínio de clima.

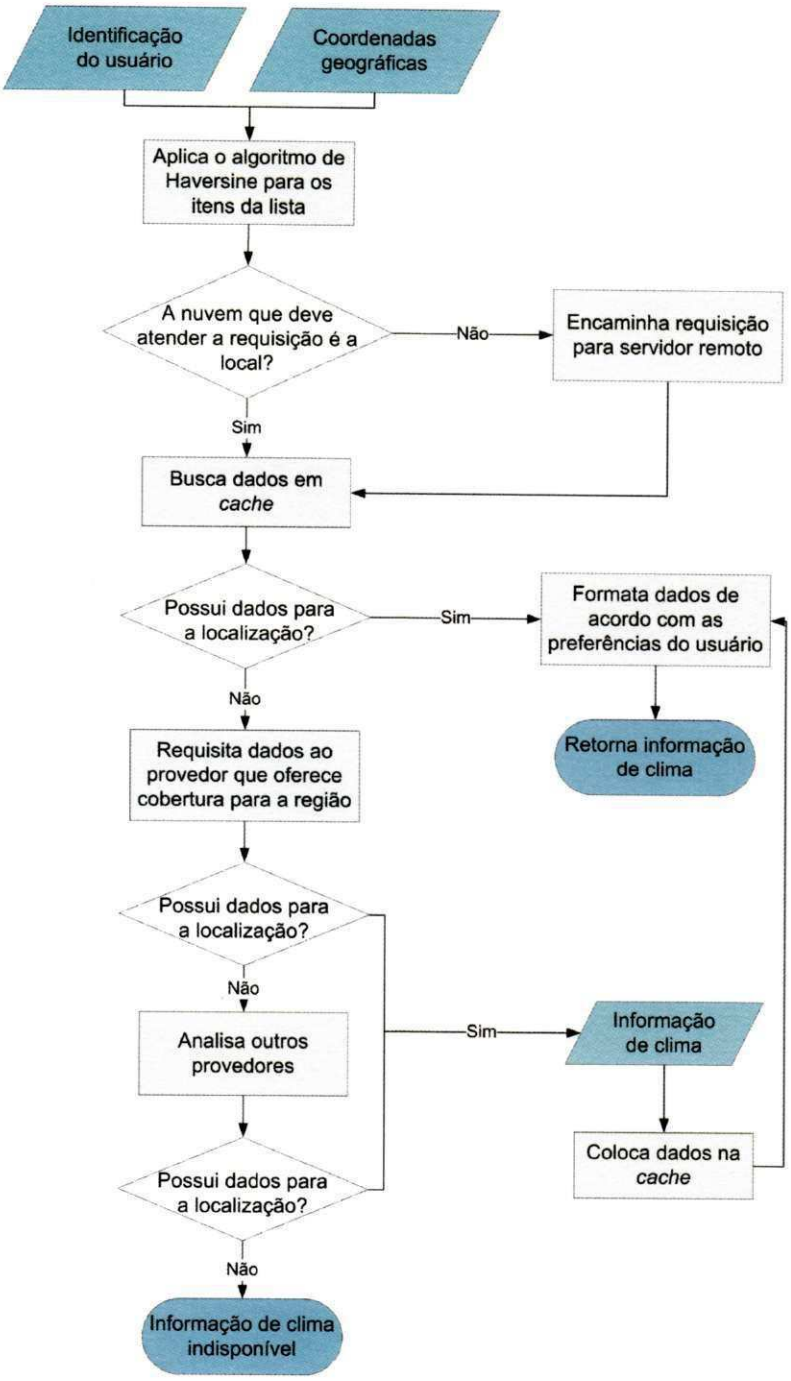


Figura 5.3: Fluxo do processo da requisição recebida no domínio de clima.

Gerenciador de Provedores

O gerenciador de Provedores é o módulo responsável por manipular as fontes de dados disponíveis na nuvem. Assim, cada nuvem possui um número de provedores atrelado a ela, os quais fornecem informações a respeito de uma região geográfica. Este é um módulo simples que disponibiliza serviços de adição, remoção e atualização de fontes de dados.

Módulo do Perfil do Usuário

Como o próprio nome sugere, o módulo do Perfil do Usuário é o responsável por manter informações a respeito de quais tipos de dados, com suas respectivas unidades, o usuário deseja receber, compondo as preferências do mesmo. Além disso, este módulo fornece uma interface para acessar e manipular o histórico de requisições dos usuários. A Figura 5.4 ilustra em detalhes a associação entre este componente e os módulos de Informação e de Conhecimento.

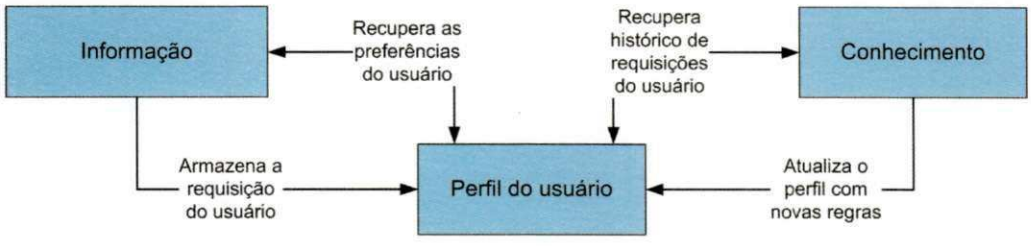


Figura 5.4: Relações existentes entre os módulos.

Antes de retornar o resultado das requisições para a aplicação cliente, o módulo de Informação recupera as preferências do usuário através do seu identificador, a fim de formatar a informação com os interesses do usuário. Ademais, para cada solicitação de dados climáticos, o módulo de Informação armazena os parâmetros da requisição no banco de dados por meio deste módulo. Esses parâmetros podem ser as coordenadas geográficas de interesse e o momento de realização da requisição. O módulo de Conhecimento recupera esses dados com a finalidade de descobrir novas informações sobre o usuário. Por exemplo, pode se identificar requisições recorrentes para uma determinada posição geográfica. Baseado nesse descoberta, o perfil do usuário é atualizado por meio da definição de tal localização como ponto de interesse.

O módulo do Perfil do Usuário não oferece serviços públicos, assim o acesso e a atualização das preferências dos usuários é realizada por meio dos *Web Services* disponibilizados pelo módulo de Informação, a exemplo do *updatePreferences* e *retrievePreferences*.

Módulo de Conhecimento

O módulo de Conhecimento é responsável por analisar o histórico de requisições do usuário, gerando regras de associação ao perfil do usuário. O uso de técnicas de descoberta de conhecimento permite a identificação de padrões e a previsão de futuras ações do usuário no sistema.

As regras de associação são produzidas através de um processo de mineração de dados com as informações obtidas do perfil do usuário. O processo de mineração de dados é realizado de forma assíncrona, executado em uma *thread* separada, sempre que o usuário realiza uma nova requisição. As regras de associação passam, então, a compor o perfil do usuário e são usadas como gatilhos[4], promovendo recuperação proativa de informação.

Por exemplo, o processo de mineração de dados pode identificar o seguinte padrão: "Sábado, 09:00, Praia de Copacabana, Rio de Janeiro, Brasil". Baseado nesse padrão, o módulo de Conhecimento cria uma regra de associação que é implementada como um gatilho no banco de dados. Nesse sentido, este pode antecipar a ação do usuário, recuperando informação sobre a localização 5 minutos antes do tempo descrito na regra. A lista de gatilhos para um determinado usuário é recuperado pela aplicação cliente móvel durante o processo de *login* no sistema. Logo, se a aplicação estiver em execução e o dispositivo conectado ao sistema a aplicação pode antecipar a requisição do usuário e mostrar a informação sem interação direta deste.

Se a conectividade do sistema só estiver disponível às 09:00, o usuário pode fazer a requisição interagindo explicitamente com a aplicação cliente. Nesse cenário, não existe proatividade na perspectiva do usuário, pois este tem que requisitar a informação. No entanto, como o módulo de Conhecimento antecipa a requisição, o resultado já estará armazenado na *cache* o que reduz o tempo de resposta da solicitação, melhorando o desempenho.

Os sistemas de gerenciamento de banco de dados, em sua maioria, suportam funciona-

[4]Também chamados de *trigger*, este é um tipo especial de procedimento armazenado em um banco de dados que é executado sempre que uma condição/evento ocorrer.

lidades de mineração de dados, fato que diminuiu a complexidade de implementação dessa funcionalidade. Porém, discussões com especialistas da área são necessárias para definir tipos de padrões válidos para esse domínio.

5.2.2 Provedores de dados

O provedor é qualquer fonte de dados climáticos acessado pelo servidor, independentemente da tecnologia de comunicação, isso inclui serviços baseados na Internet (INMET, NWS, etc.), estações pessoais, servições governamentais, entre outros.

Essas fontes não seguem um padrão específico para descrever e fornecer acesso aos dados meteorológicos. Assim, a fim de permitir que um servidor processe dados de diferentes fontes, é necessário traduzir o dado recuperado para uma linguagem comum. Da mesma forma, os parâmetros de requisição devem ser formatados diferentes de acordo com a fonte. Sendo esse processo de "tradução" implementado no servidor, este deve ser atualizado sempre que novos provedores de dados se tornem disponíveis.

Considerando os aspectos comentados anteriormente, foi definida uma entidade intermediária, com a finalidade de manter a escalabilidade da arquitetura do sistema. Essa entidade, denominada *Wrapper*, é responsável por recuperar dados de um provedor específico e entregar o resultado ao servidor, em outras palavras, ela atua como um *Adapter*[5]. As requisições *Web Services* advindas de um servidor do domínio de clima são convertidas de acordo com o provedor e encaminhadas para obtenção dos dados. Quando o provedor retorna a informação, essa entidade faz o caminho inverso, convertendo o resultado para o formato entendido pelo servidor. O processo é ilustrado na Figura 5.5.

O acesso a esta entidade é realizado por meio de *Web Services*, permitindo que sua implementação seja realizada utilizando diferentes linguagens de programação e que a comunicação com os provedores seja feita usando diferentes tecnologias de comunicação. Além disso, o servidor se torna muito flexível, permitindo adicionar, remover e atualizar informações sobre provedores. Uma vez que todos os *Wrappers* seguem o mesmo protocolo, mudanças nos provedores afetam apenas o respectivo componente, não surtindo impacto no servidor.

[5]Padrão de projeto usado para permitir que classes com interfaces incompatíveis trabalhem em conjunto

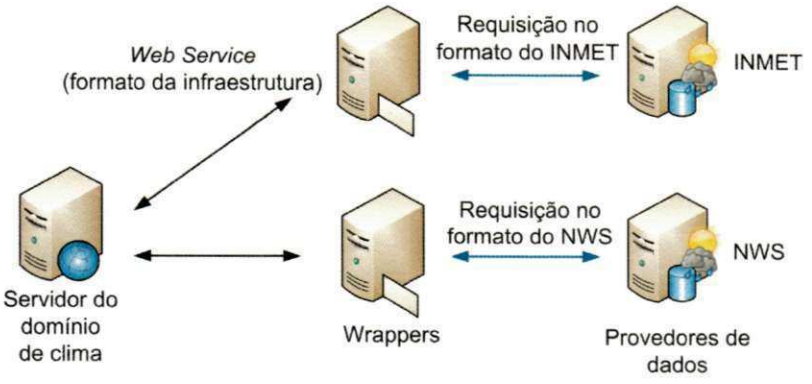


Figura 5.5: Comunicação entre os provedores de dados.

5.3 Aplicação cliente

A aplicação cliente foi desenvolvida para os *smartphones* da plataforma Android versão 2.1, com o auxílio da API do Google Maps, utilizado nos aspectos relacionados a exibição e manipulação de dados sobre o mapa. Além disso, utilizou-se a biblioteca ksoap2[6] para realizar a comunicação com os WSDL dos serviços do servidor. O mecanismo de carregamento de *plug-ins* definido na arquitetura da Seção 4.3 foi implementado através do componente do Android denominado *Service*, que permite a execução de operações em segundo plano, sem elementos de interface gráfica. O *plug-in* de domínio de clima desenvolvido é um serviço Android que contempla a interface definida na arquitetura do cliente, disponibilizando informações sobre a conexão e acesso aos *Web Services* definidos pelo servidor.

As funcionalidades desenvolvidas são listadas na seção seguinte. Por último, os cenários de interação com o sistema e suas respectivas telas são apresentados.

5.3.1 Funcionalidades

Gerenciamento do acesso do usuário

É necessário que o usuário se identifique, a fim de permitir acesso aos serviços e informações personalizadas. Assim, essa funcionalidade provê um mecanismo de autenticação, onde o usuário informa seu *username* e senha, previamente, cadastrados na infraestrutura. De pose desses dados, a aplicação realiza uma requisição ao serviço de *login* oferecido pelo Núcleo.

[6]http://code.google.com/p/ksoap2-android/

Quando o usuário desejar sair do sistema, o serviço de *logout* é solicitado, o qual remove o *cookie* criado no momento da autenticação.

Obtenção da posição geográfica do usuário

Essa funcionalidade é responsável por obter a localização geográfica do usuário através dos sistemas de posicionamento existentes. Para isso, é utilizada a API de localização padrão da plataforma Android, fornecendo simples acesso ao GPS e à infraestrutura *Wi-Fi* conectada. Com isso em mente, foram desenvolvidos dois *plug-ins* de localização, um para o GPS (com prioridade maior) e outro para o *Wi-Fi*. A localização atual do cliente auxilia no desenho da fração do mapa onde o usuário se encontra, com o intuito de orientá-lo na visualização e guiá-lo por uma rota definida. Essa funcionalidade não acessa serviços da infraestrutura, pois é um processamento local, usando os recursos disponibilizados pelo dispositivo.

Geração de Rotas

Essa funcionalidade fornece um mecanismo de geração de rotas a partir de um ponto origem a um ponto de destino indicados no mapa. Esse mecanismo foi implementado através de requisições HTTP ao *Google Driving Directions*[7], o qual retorna um arquivo KML[8] de acordo com os parâmetros enviados na solicitação. Além de traçar uma rota entre os pontos, este informa orientações de navegação de acordo com o movimento do usuário. O uso desse *gadget* foi motivado pela inexistência de uma API gratuita para obtenção de rotas. Ademais, essa funcionalidade permite que informações de domínio sejam requisitados para os pontos da rota, exibindo-as à medida que o usuário se movimenta.

Exibição e manipulação dos favoritos

Essa funcionalidade permite a recuperação e manipulação da lista de posições geográficas cadastradas pelo usuário. As localidades de interesse são requisitadas ao servidor e exibidas no mapa da aplicação, possibilitando que o usuário visualize e atualize informações sobre estas de forma amigável. Além disso, novas localidades podem ser adicionadas pela indicação

[7] http://maps.google.com/help/maps/gadgets/directions/

[8] Keyhole Markup Language é uma notação baseada em XML para expressar informações geográficas e de visualização de mapas, como navegação terrestre.

do ponto geográfico no mapa. As atualizações nos Favoritos são sincronizadas com o sistema, possibilitando que o usuário acesse as localidades cadastradas por outros dispositivos executando a aplicação.

Os serviços da API acessados por esta funcionalidade são *getBookmarks*, *createBookmark*, *updateBookmark* e *removeBookmark*, para obter a lista de favoritos, criar uma nova localidade, atualizar e remover uma localidade existente, respectivamente.

Habilitação de um determinado *plug-in*

Essa funcionalidade permite a ativação e desativação de um determinado *plug-in* de domínio. A desativação implica na não exibição das informações a respeito daquele domínio, ou seja, apenas os dados dos *plug-ins* habilitados são apresentados pela interface. A lista dos *plug-ins* habilitados é armazenada no repositório local e pode ser atualizado a qualquer momento pelo usuário.

Atualização das preferências do usuário

Essa funcionalidade é responsável por manipular as preferências do usuário, incluindo as unidades dos dados (e.g. temperatura em graus Celsius) e a lista com os tipos de informações de interesse (e.g. umidade, temperatura, visibilidade, etc.). As atualizações realizadas através da interface gráfica são armazenadas no repositório local do usuário e comunicadas ao servidor, a fim de possibilitar a conversão das unidades e a formatação dos dados obtidos dos provedores. Essa funcionalidade acessa os serviços *updatePreferences* e *retrievePreferences*, para atualizar e recuperar as preferências do usuário, respectivamente.

Recuperação de informações relacionadas a clima

Essa é a principal funcionalidade e tem como objetivo adquirir informações sobre o domínio, especificamente, o de clima. Através da interface, o usuário indica sobre qual posição geográfica este deseja obter dados climáticos. A aplicação se encarrega de enviar a solicitação para infraestrutura, retornando a informação de acordo com as preferências do usuário. O mesmo cenário é utilizado para outros domínios. Caso a aplicação possua mais de *plug-in* de domínio habilitado, uma requisição é enviada para cada tipo de informação e exibida no dispositivo. No domínio de clima, o serviço acessado por essa funcionalidade é o *getWeather*.

Antecipação do provimento de informação

Essa funcionalidade disponibiliza informações climáticas proativamente por meio da recuperação da lista de gatilhos, obtida logo após a autenticação, gerados pelo servidor a partir das ocorrências de requisições do usuário. Um mecanismo de requisição foi implementado e é executado assincronamente, com intuito de solicitar informações de acordo com a condição estabelecida no gatilho. Os serviços acessados por essa funcionalidade são o *retrieveTriggers* e o *getWeather*, que recuperam a lista de gatilhos e obtêm dados climáticos, respectivamente.

5.3.2 Interação

A primeira tela da aplicação contem elementos de interface, onde o usuário pode informar seus dados para autenticação. Com base nos dados fornecidos, é enviada uma requisição de *login* ao servidor, cujo resultado redireciona o fluxo para outra tela. Caso o *login* seja realizado com sucesso, obtêm-se a posição geográfica atual do usuário através do GPS ou *Wi-Fi*, de acordo com a disponibilidade dos recursos, exibindo os dados sobre o mapa, como ilustrado na Figura 5.6(a). Paralelamente, a aplicação envia uma solicitação para recuperação da lista das localidades adicionadas aos "Favoritos", a qual é exibida de acordo com a Figura 5.6(b).

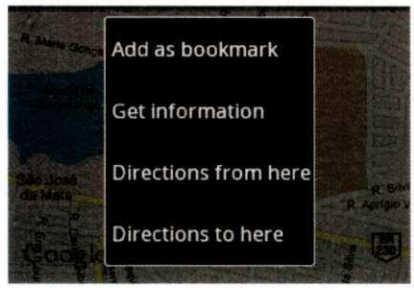
(a) Exibição da localização atual do usuário

(b) Exibição dos Favoritos

Figura 5.6: Telas com informações do usuário independentes de domínio.

O usuário pode adicionar novos itens aos Favoritos, assim como requisitar informação a respeito de uma localização geográfica por meio de um toque longo sobre o ponto de interesse. Uma janela com a lista de opções disponíveis é exibida em *popup*, como a apresentada na Figura 5.7(a). A seleção da alternativa "Add as bookmark" exibe outra janela (ilustrada

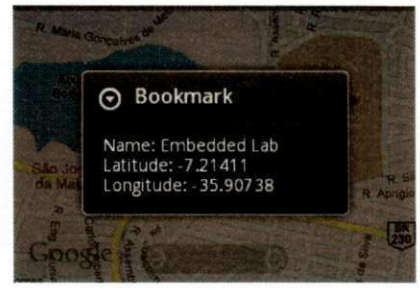
na Figura 5.7(b)) para informar o nome da localidade indicada e adicionar aos Favoritos, enquanto que a alternativa "Get Information" obtêm informação dos domínios existentes para aquele ponto, e as opções de "Directions from here" e "Directions to here", por sua vez, marcam o ponto inicial e final, respectivamente, de uma rota.



(a) Opções disponíveis para o ponto de interesse

(b) Adição de um item aos Favoritos

(c) Exibição do item criado



(d) Informação a respeito da localidade adicionada

Figura 5.7: Telas de adição e visualização das localidades existentes nos Favoritos.

Depois de realizada a solicitação de adição de um novo item aos Favoritos, o ícone é desenhado sobre o ponto no mapa, como ilustrado na Figura 5.7(c). A Figura 5.7(d) exibe a janela contendo dados (nome do Favorito, latitude e longitude) a respeito da localidade adicionada, quando o usuário clica em cima do ícone.

O mecanismo de geração de rotas é ilustrado pelas telas da Figura 5.8. A Figura 5.8(a) exibe o ícone indicando a marcação do ponto de origem da rota, realizada pelo usuário a partir do toque longo sobre o ponto. Na Figura 5.8(b), o mesmo processo da Figura 5.8(a), é elucidado para o ponto de destino. Depois de indicado o início e fim, a rota entre os pontos é requisitada e traçada sobre o mapa, como ilustrado na Figura 5.8(c).

A Figura 5.9 ilustra as telas de exibição da informação sobre condições meteorológicas obtida pela opção "Get Information", como comentado anteriormente. Na Figura 5.9(a), da

(a) Marcação da origem

(b) Marcação do destino

(c) Rota traçada entre os pontos

Figura 5.8: Telas com geração de rotas.

mesma forma que acontece com os Favoritos, o usuário pode visualizar um ícone desenhado sobre o mapa, representando a situação climática da localização indicada. Já a Figura 5.9(b), ilustra a janela com os dados recebidos do servidor sobre o clima naquela região, que é mostrado quando clica-se sobre o respectivo ícone.

(a) Exibição do ícone do clima sob a localização solicitada

(b) Informações climáticas obtidas para a localização

Figura 5.9: Telas com informações do domínio de clima.

5.4 Conclusão

Neste capítulo foi apresentado um mecanismo de validação da infraestrutura desenvolvida expondo o desenvolvimento de um estudo de caso para o domínio de clima. O objetivo desse estudo de caso foi demonstrar como a infraestrutura pode ser utilizada na criação de domínios diferentes sem grande alteração arquitetural, bastando implementar um *plug-in* para o mesmo, tanto no servidor como no cliente.

Capítulo 6

Considerações Finais

A criação de sistemas baseados em localização tem conquistado grande espaço, tanto na indústria como na academia, motivada pelo crescente aumento de dispositivos móveis com recursos capazes de adquirir a posição geográfica do aparelho e pelos avanços em redes de comunicação. Vários são os modelos e *frameworks* elaborados para assistir o desenvolvimento de aplicações desse tipo de sistema.

Em sua maioria, essas soluções fornecem mecanismos de obter e combinar dados de localização, a fim de determinar com maior precisão a posição geográfica do usuário. Outras soluções encontradas se limitam a prover informações para áreas de aplicação (no contexto desse trabalho, chamados de domínios) específicas (e.g. turismo).

Neste trabalho foi apresentada uma infraestrutura escalável para o desenvolvimento de aplicações baseadas em localização e orientadas a domínios. Esta infraestrutura foi elaborada em uma arquitetura distribuída com mecanismo de extensão através de *plug-ins*, possibilitando a simples adição de novos domínios. Além disso, foi desenvolvida uma API utilizando *Web Services* que fornece acesso aos serviços disponibilizados pela infraestrutura, como a adição de uma posição geográfica aos favoritos e recuperação de informações personalizadas baseadas na localização dado um domínio específico.

Um estudo de caso para o domínio de clima foi construído utilizando os componentes definidos na infraestrutura (Seção 4.2 e 4.3) e a API disponibilizada, com o intuito de validar a solução. Este estudo de caso, denominado *Weather*, foi constituído da criação de um *plug-in* para a aplicação cliente móvel e de outro acoplado ao servidor, a fim de fornecer informações de condições meteorológicas para o usuário.

Os aspectos envolvidos na obtenção da escalabilidade do sistema foram comentados, com explicações das principais decisões tomadas no que tange a arquitetura da infraestrutura. A interoperabilidade foi alcançada através do uso de *Web Services*, que caracterizam como serviços independentes de plataforma e linguagem de programação, utilizados na comunicação entre os elementos da infraestrutura.

6.1 Contribuições

As principais contribuições deste trabalho são enumeradas a seguir:

- Elaboração de uma infraestrutura que disponibiliza serviços relevantes ao usuário utilizando informações de localização;
- Especificação de um mecanismo de adição de novos domínios, cada um fornecendo serviços diferenciados;
- Disponibilização de uma API baseada em *Web Services* para acesso aos serviços, a qual viabiliza a independência de linguagem ou plataforma, proporcionando interoperabilidade à infraestrutura.

6.2 Trabalhos Futuros

A escalabilidade em computação pervasiva é uma preocupação condizente com o aumento de dispositivos e usuários utilizando o sistema. Neste trabalho, as questões utilizadas para alcançar esse requisito foram a implantação de banco de dados distribuídos, servidor descentralizado composto por vários nós, entre outras, porém um estudo do impacto de adição de um novo domínio ou de novos *clusters* não foi avaliado. Assim, faz-se necessária uma análise da capacidade do sistema permanecer disponível e eficiente, frente a um aumento significativo do número de acessos e do volume de informações disponíveis.

Nesse sentido, vários tipos de testes de desempenho podem ser aplicados para avaliar o comportamento do sistema com o acréscimo gradativo de novos domínios e com crescimento no número de usuário, identificando situações indesejáveis. Os principais testes desse tipo são: teste de *stress*, que permite a determinação da quantidade de acessos que o sistema

suporta; e teste de carga, por meio do qual é possível estipular o tempo de resposta do sistema, auxiliando na identificação e eliminação de gargalos. Considerando a independência de plataforma da solução apresentada neste trabalho, é ainda interessante a realização de testes de escalabilidade envolvendo máquinas com hardware e software diversificados.

No que tange a aplicação cliente móvel, o avanço na capacidade de armazenamento e processamento dos dispositivos móveis, aliado ao conjunto de recursos presentes nesses, culmina em uma maior precaução no que diz respeito ao consumo de bateria. Levando-se em consideração a perspectiva de otimização do uso da bateria do dispositivo, pretende-se avaliar a utilização dos recursos pela aplicação cliente móvel a fim de evitar excessos e minimizar o dispêndio de energia [34], a exemplo do trabalho desenvolvido em [6]. Isso pode ser atingido pelo uso de protocolos de rede mais leves (e.g. UDP), utilização de uma base de mapas local, criação de um algoritmo de geração de rotas *offline*, dentre outros.

Bibliografia

[1] *IEEE Standard for Information technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Dezembro 2007.

[2] Lauri Aalto, Nicklas Göthlin, Jani Korhonen, and Timo Ojala. Bluetooth and WAP push based location-aware mobile advertising system. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, MobiSys '04, pages 49–58, New York, NY, USA, 2004. ACM.

[3] Frank Adelstein, Sandeep KS Gupta, Golden Richard III, and Loren Schwiebert. *Fundamentals of Mobile and Pervasive Computing*. McGraw-Hill, 1st edition, 2004.

[4] Johnson I. Agbinya, Oya Sevimli, and Sam Reisenfeld, editors. *Advances in Broadband Communication and Networks*. River Publishers, 2008.

[5] Arnon Amir, Alon Efrat, Jussi Myllymaki, Lingeshwaran Palaniappan, and Kevin Wampler. Buddy tracking - efficient proximity detection among mobile friends. *Pervasive Mob. Comput.*, 3:489–511, Outubro 2007.

[6] Sean Barbeau, Rafael Perez, Miguel Labrador, Alfredo Perez, Philip Winters, and Nevine Georggi. A location-aware framework for intelligent real-time mobile applications. *IEEE Pervasive Computing*, 10:58–67, Julho 2011.

[7] Ugo Barchetti, Alberto Bucciero, Tania A. De Benedittis, Francesca Macchia, Luca Mainetti, and Antonio Tamborino. MoWeT: A Configurable Framework to Support Ubiquitous Location-Aware Applications. In *Proceedings of the 2009 Symposia and*

Workshops on Ubiquitous, Autonomic and Trusted Computing, UIC-ATC '09, pages 75–82, Washington, DC, USA, 2009. IEEE Computer Society.

[8] Bo Begole. *Ubiquitous Computing for Business*. FT Press, 1st edition, 2011.

[9] Paolo Bellavista, Antonio Corradi, and Carlo Giannelli. The PoSIM middleware for translucent and context-aware integrated management of heterogeneous positioning systems. *Comput. Commun.*, 31:1078–1090, Abril 2008.

[10] R Benita and G Vejey Subash. Design and Implementation of Location Based Ad Delivery System. *International Journal on Recent Trends in Engineering and Technology*, 5:86–89, 2011.

[11] Mary Berna, Brennan Sellner, Brad Lisien, Sebastian Thrun, Geoffrey Gordon, and Frank Pfenning. A learning algorithm for localizing people based on wireless signal strength that uses labeled and unlabeled data. In *Proceedings of the 18th international joint conference on Artificial intelligence*, pages 1427–1428, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[12] Dorian Birsan. On Plug-ins and Extensible Architectures. *Queue*, 3:40–46, March 2005.

[13] Robert Chatley, Susan Eisenbach, and Jeff Magee. Modelling a Framework for Plugins. In *Proceedings of the SAVCBS 2003 Workshop*, pages 49–57, 2003.

[14] Robert Chatley, Susan Eisenbach, and Jeff Magee. MagicBeans: a Platform for Deploying Plugin Components. In Wolfgang Emmerich and Alexander Wolf, editors, *Component Deployment*, volume 3083 of *Lecture Notes in Computer Science*, pages 97–112. Springer Berlin / Heidelberg, 2004.

[15] O-Hoon Choi, Jung-Eun Lim, and Doo-Kwon Baik. A Design of Location Information Management System in Positioning Systems. In *Proceedings of the 2007 International Conference on Convergence Information Technology*, ICCIT '07, pages 114–120, Washington, DC, USA, 2007. IEEE Computer Society.

[16] Cristiano Andre da Costa. *Continuum: A Context-aware Service-based Software Infrastructure for Ubiquitous Computing*. Tese de Doutorado em Ciência da Computação, Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brasil, 2008.

[17] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. *Distributed Systems: Concepts and Design*. Addison Wesley, 5th edition, 2011.

[18] Anind Dey, Jeffrey Hightower, Eyal de Lara, and Nigel Davies. Location-based services. *IEEE Pervasive Computing*, 9:11–12, 2010.

[19] T. D'Roza and G. Bilchev. An Overview of Location-Based Services. *BT Technology Journal*, 21:20–27, Janeiro 2003.

[20] Eiman Elnahrawy, Xiaoyan Li, and Richard P. Martin. Using Area-Based Presentations and Metrics for Localization Systems in Wireless LANs. In *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, LCN '04, pages 650–657, Washington, DC, USA, 2004. IEEE Computer Society.

[21] George M. Giaglis, Panos Kourouthanassis, and Argirios Tsamakos. *Towards a classification framework for mobile location services*, pages 67–85. IGI Publishing, Hershey, PA, USA, 2003.

[22] Robert Grimm, Janet Davis, Eric Lemar, Adam Macbeth, Steven Swanson, Thomas Anderson, Brian Bershad, Gaetano Borriello, Steven Gribble, and David Wetherall. System support for pervasive applications. *ACM Trans. Comput. Syst.*, 22:421–486, Novembro 2004.

[23] Aboul-Ella Hassanien, Jemal H. Abawajy, Ajith Abraham, and Hani Hagras, editors. *Pervasive Computing: Innovations in Intelligent Multimedia and Applications*. Springer, New York, 2009.

[24] Karen Henricksen and Jadwiga Indulska. A Software Engineering Framework for Context-Aware Pervasive Computing. In *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, PERCOM '04, pages 77–86, Washington, DC, USA, 2004. IEEE Computer Society.

[25] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Infrastructure for Pervasive Computing: Challenges. In *Workshop on Pervasive Computing INFORMATIK 01, Viena*, pages 214–222, 2001.

[26] Jeffrey Hightower and Gaetano Borriello. Location Systems for Ubiquitous Computing. *Computer*, 34:57–66, Agosto 2001.

[27] Yoshihide Hosokawa, Naohisa Takahashi, and Hiroyasu Taga. A System Architecture for Seamless Navigation. In *Proceedings of the 24th International Conference on Distributed Computing Systems Workshops - W7: EC (ICDCSW'04) - Volume 7*, ICDCSW '04, pages 498–504, Washington, DC, USA, 2004. IEEE Computer Society.

[28] Andrew Howard, Sajid Siddiqi, and Gaurav S. Sukhatme. An Experimental Study of Localization Using Wireless Ethernet. In *in 4th International Conference on Field and Service Robotics*, Julho 2003.

[29] Andrew Jagoe. *Mobile Location Services: The Definitive Guide*. Pearson Education, Dezembro 2002.

[30] A. Katasonov and M. Sakkinen. Content quality in location-based services: a case study. *International Conference on Pervasive Services*, 0:461–464, 2005.

[31] Andrej Krevl and Mojca Ciglarič. A framework for developing distributed location based applications. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, pages 263–263, Washington, DC, USA, 2006. IEEE Computer Society.

[32] Axel Küpper. *Location-Based Services: Fundamentals and Operation*. Wiley, Outubro 2005.

[33] Emerson Loureiro, Glauber Ferreira, Hyggo Almeida, and Angelo Perkusich. Pervasive computing: What is it anyway? In *Ubiquitous and Pervasive Knowledge and Learning Management: Semantics, Social Networking and New Media to their full potential*, volume 4, chapter 1, pages 9–36. Idea Group Inc, Hershey, PA, USA, 2007.

[34] Saulo Oliveira Dornelles Luiz. Gerenciamento de Energia em Sistemas Embarcados. Dissertação de Mestrado em Engenharia Elétrica, Universidade Federal de Campina Grande (UFCG), Campina Grande, Paraíba, Brasil, 2008.

[35] Paul Lukowicz, Tanzeem Choudhury, and Hans Gellersen. Beyond Context Awareness. *IEEE Pervasive Computing*, 10:15–17, 2011.

[36] Kalle Lyytinen and Youngjin Yoo. Introduction. *Communications of the ACM*, 45:62–65, Dezembro 2002.

[37] Marwa Mabrouk. OpenGIS Location Services (OpenLS): Core Services, 2004.

[38] David Madigan, Wen-Hua Ju, P. Krishnan, A. S. Krishnakumar, and Ivan Zorych. Location estimation in wireless networks: a Bayesian approach. In *Statistica Sinica*, volume 16, pages 495–522. 2006.

[39] Atsushi Maruyama, Naoki Shibata, Yoshihiro Murata, Keiichi Yasumoto, and Minoru Ito. P-Tour: A Personal Navigation System for Tourism. In *Proc. of 11th World Congress on ITS*, pages 18–21, 2004.

[40] Katina Michael, Andrew McNamee, and MG Michael. The Emerging Ethics of Humancentric GPS Tracking and Monitoring. In *Proceedings of the International Conference on Mobile Business*, pages 34–44, Washington, DC, USA, 2006. IEEE Computer Society.

[41] P. Mockapetris. *RFC 1035 Domain Names - Implementation and Specification*. Internet Engineering Task Force, Novembro 1987.

[42] André Iasi Moura. WBLS: um sistema de localização de dispositivos móveis em redes Wi-Fi. Dissertação de Mestrado em Engenharia Elétrica, Escola Politécnica da Universidade de São Paulo, São Paulo, 2007.

[43] J. Nord, K. Synnes, and P. Parnes. An Architecture for Location Aware Applications. In *Proceedings of the 35th Annual Hawaii International Conference on System Sciences (HICSS'02)-Volume 9 - Volume 9*, HICSS '02, pages 293–298, Washington, DC, USA, 2002. IEEE Computer Society.

[44] Jeni Paay and Jesper Kjeldskov. Understanding the user experience of location-based services: five principles of perceptual organisation applied. *Journal of Location Based Services*, 2:267–286, Dezembro 2008.

[45] Krassie Petrova and Bin Wang. Location-based services deployment and demand: a roadmap model. *Ubiquitous Electronic Commerce Systems*, 11:5–29, Janeiro 2011.

[46] Stefan Poslad. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. Wiley Publishing, 1st edition, 2009.

[47] Dimitrios Raptis, Nikolaos Tselios, and Nikolaos Avouris. Context-based design of mobile applications for museums: a survey of existing practices. In *Proceedings of the 7th international conference on Human computer interaction with mobile devices & services*, MobileHCI '05, pages 153–160, New York, NY, USA, 2005. ACM.

[48] Weizheng Ren, Zhongliang Deng, Lianming Xu, and Yujia Zhu. Research of architecture for digital campus LBS in Pervasive Computing Environment. In *Proceedings of the Third International Conference on Pervasive Computing and Applications*, ICPCA 2008, pages 473–478. IEEE Computer Society, 2008.

[49] Ronald L. Rivest. *The MD5 Message-Digest Algorithm*, Abril 1992.

[50] Debashis Saha and Amitava Mukherjee. Pervasive Computing: A Paradigm for the 21st Century. volume 36, pages 25–31, Los Alamitos, CA, USA, Março 2003. IEEE Computer Society.

[51] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8:10–17, 2001.

[52] Jochen Schiller and Agnes Voisard. *Location-Based Services*. Elsevier Science and Technology, Dezembro 2004.

[53] Albrecht Schmidt, Michael Beigl, and Hans w. Gellersen. There is more to context than location. *Computers and Graphics*, 23:893–901, 1998.

[54] Manos Spanoudakis, Aggelos Batistakis, Ioannis Priggouris, Anastasios Ioannidis, Stathes Hadjiefthymiades, and Lazaros Merakos. Extensible platform for location based services provisioning. In *Proceedings of the Fourth international conference on Web information systems engineering workshops*, WISEW'03, pages 72–79, Washington, DC, USA, 2003. IEEE Computer Society.

[55] Graeme Stevenson, Juan Ye, Simon Dobson, and Paddy Nixon. LOC8: A Location Model and Extensible Framework for Programming with Location. *IEEE Pervasive Computing*, 9:28–37, Janeiro 2010.

[56] B. Pröll W. Schwinger, Ch. Grün and W. Retschitzegger. A Light-Weight Framework for Location-Based Services. *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, pages 206–210, 2005.

[57] Roy Want and Trevor Pering. System challenges for ubiquitous & pervasive computing. In *Proceedings of the 27th international conference on Software engineering*, ICSE '05, pages 9–14, New York, NY, USA, 2005. ACM.

[58] Mark Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36:75–84, Julho 1993.

[59] Shiow-Yang Wu and Kun-Ta Wu. Effective location based services with dynamic data management in mobile environments. *Wireless Networks*, 12(3):369–381, 2006.

[60] Z. Xiang, S. Song, J. Chen, H. Wang, J. Huang, and X. Gao. A wireless LAN-based indoor positioning technology. *IBM Journal of Research and Development*, 48:617–626, Setembro 2004.

[61] Vasileios Zeimpekis, George M. Giaglis, and George Lekakos. A taxonomy of indoor and outdoor positioning techniques for mobile location services. *ACM SIGecom Exchanges - Mobile commerce*, 3:19–27, Dezembro 2002.