UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

LEANDRO NATAL CORAL

PROVISÃO DE INTEROPERABILIDADE NO ACESSO A SERVIÇOS DE DIRETÓRIO EM UM LBS UTILIZANDO PADRÕES ABERTOS E WEB SERVICES

CRICIÚMA, DEZEMBRO DE 2010

LEANDRO NATAL CORAL

PROVISÃO DE INTEROPERABILIDADE NO ACESSO A SERVIÇOS DE DIRETÓRIO EM UM LBS UTILIZANDO PADRÕES ABERTOS E WEB SERVICES

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Prof. MEng. Evânio Ramos Nicoleit

CRICIÚMA, DEZEMBRO DE 2010

LEANDRO NATAL CORAL

Provisão de Interoperabilidade no Acesso a Serviços de Diretório em um LBS Utilizando Padrões Abertos e Web Services

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. MEng. Evânio Ramos Nicoleit (UNESC)
Orientador

Prof. Esp. Fabrício Giordani (UNESC)

Esp. Fabrizio Colombo Machado (TI - UNESC)

Dedico este trabalho a meus pais, por me apoiarem em todas as conquistas da minha vida.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelo dom da vida e capacidade de construir o conhecimento.

Aos meus pais Evandro e Sonia, que me incentivaram a continuar na busca pelo conhecimento. Pelo modo como construíram minha educação, me ensinando a lutar pelos meus sonhos. Aos meus irmãos Danillo e Carolini, pelo companheirismo em todas as circunstâncias.

A minha namorada Katiane, por muitas vezes aturar o meu cansaço, me apoiando nos momentos de desânimo, e caminhado junto a mim ao longo desta jornada. Por tudo que representas, te amo!

Aos meus colegas de curso, pelos anos de convívio e amizades verdadeiras.

A todos os professores e coordenadores do curso de Ciência da Computação, pelo apoio e dedicação, em especial ao meu professor e orientador Evânio pelo incentivo, dedicação e conhecimento.

“Sucesso é acordar de manhã – não importa quem você seja, onde você esteja, se é velho ou se é jovem – e sair da cama porque existem coisas importantes que você adora fazer, nas quais você acredita, e em que você é bom. Algo que é maior que você, que você quase não aguenta esperar para fazer hoje.”

Whit Hobbs

RESUMO

Serviços Baseados em Localização (LBS) são serviços que exploram a posição geográfica de Dispositivos Móveis (DM), fornecendo informações úteis aos seus usuários. Dentre os possíveis serviços existentes em LBS está o Serviço de Diretório, que fornece a localização de Pontos de Interesse específicos, ou mais próximo de determinada localização. Esta pode ser a localização de Veículos de Transporte Rodoviário de Cargas, obtidas por DM dotados de tecnologias de posicionamento, conhecidos como rastreadores veiculares. LBS requerem cada vez mais interoperabilidade, pois podem estar disponíveis em diferentes plataformas, sistemas operacionais e infraestruturas de rede. Uma das formas de garantir interoperabilidade na comunicação entre LBS e aplicações clientes é por meio do uso de padrões abertos. O uso de Web Services (WS) garante interoperabilidade no acesso aos serviços. A combinação de WS com padrões abertos fornece uma arquitetura eficiente no acesso a LBS. Este trabalho descreve o desenvolvimento de um protótipo de LBS, especificamente um Serviço de Diretório, com ênfase na obtenção da posição de Veículos de Transporte Rodoviários de Cargas, utilizando WS e padrões abertos para prover interoperabilidade. O padrão adotado para o desenvolvimento do protótipo foi o OpenLS, por ser baseado em XML e prover soluções para acesso aos principais núcleos de LBS, incluindo Serviços de Diretório. A provisão de interoperabilidade se dá na troca de mensagens SOAP com o WS, contendo documentos XML formatados de acordo com a especificação OpenLS. O sistema provê a localização de Pontos de Interesse tanto a partir de suas propriedades, quanto a partir da proximidade com a posição geográfica de DM, como a dos rastreadores veiculares.

Palavras-Chave: LBS; Serviços de Diretório, Dispositivos Móveis; OpenLS, Web Services.

ABSTRACT

Location Based Services (LBS) are services that explore the geographical position of Mobile Devices (MD), providing useful information to their users. Among the possible LBS services available is the Directory Service, which provides the location of specific Points of Interest, or nearest a location. This may be the Road Freight Transport Vehicles location, obtained by MD equipped with positioning technologies, known as vehicle trackers. LBS increasingly require interoperability, because they might be available on different platforms, operating systems and network infrastructures. One way to ensure interoperability in communication between LBS and client applications is through the use of open standards. Using Web Services (WS) ensures interoperability in the services access. The combination of WS and open standards provides an efficient architecture for LBS access. This paper describes the development of a LBS prototype, specifically a Directory Service, with emphasis on obtaining the Road Freight Transport Vehicle position, using Web Services and open standards to provide interoperability. The standard adopted for the prototype development was OpenLS, because it is XML based and it provides solutions to access the main LBS cores, including Directory Services. The interoperability provision occurs in the exchange of SOAP messages with WS, containing XML documents formatted according to the OpenLS specification. The system provides the location of Points of Interest both from their properties, as from the proximity to the geographical position of MD, such as the vehicular trackers.

Keywords: LBS; Directory Services, Mobile Devices; OpenLS, Web Services.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

ADT	Tipos Abstratos de Dados
A-GPS	GPS Assistido
API	Interfaces de Programação de Aplicações
Cell-ID	Identificação da Célula
DAO	Objeto de Acesso a Dados
DM	Dispositivos Móveis
DTD	Definição de Tipo de Documento
ERB	Estações Rádio-Base
FTP	Protocolo de Transferência de Arquivos
GIS	Sistemas de Informações Geográficas
GML	Linguagem de Marcação Geográfica
GPS	Sistema de Posicionamento Global
HB	Soluções Baseadas em Terminais
HTTP	Protocolo de Transferência de Hipertexto
IDE	Ambiente de Desenvolvimento Integrado
JAR	Java Archive
JAXB	Java Architecture for XML Binding
JAX-WS	Java API for XML Web Services
LBS	Serviços Baseados em Localização
MLP	Mobile Location Protocol
NB	Soluções Baseadas em Rede
OGC	Open Geospatial Consortium

OMA	Open Mobile Alliance
OpenLS	OpenGIS Location Services
POI	Pontos de Interesse
PPS	Precision Positioning Service
RPC	Chamadas Remotas de Procedimentos
SA	Avaliação Seletiva
SDE	Serviço de Diretório Específico
SDP	Serviço de Diretório por Proximidade
SGBD	Sistemas Gerenciadores de Banco de Dados
SGBDG	Sistemas Gerenciadores de Banco de Dados Geográficos
SGML	Linguagem Padronizada de Marcação Genérica
SMTP	Protocolo de Transferência de Correio Simples
SOAP	Protocolo de Acesso Simples a Objetos
Spring-WS	Spring Web Services
SPS	Stantard Positioning Service
SRID	Identificador de Referência Espacial
SRS	Sistema de Referência de Coordenadas
UNESC	Universidade do Extremo Sul Catarinense
URI	Identificador Uniforme de Recursos
URL	Localizador Padrão de Recursos
W3C	World Wide Web Consortium
WS	Web Services
WSDL	Linguagem de Descrição de Web Services (WSDL)
XLS	XML para Serviços de Localização

XML	Linguagem de Marcação Extensível
XSD	Definição de Esquemas XML

SUMÁRIO

1 INTRODUÇÃO

As inovações tecnológicas das redes de comunicação de dados sem fio integradas à web vêm promovendo um crescimento exponencial das comunicações móveis recentemente. Dispositivos Móveis (DM) possibilitam a utilização em deslocamento, viabilizando a mobilidade no uso. A mobilidade possibilita ampliação da abrangência dos domínios de uso de sistemas e dispositivos, garantindo acesso às informações independente da localização. A flexibilidade de acesso às informações pode ser alcançada com a utilização de DM (Celulares, Smartphones, PDA, NetBooks, Notebooks, etc.) com suporte a tecnologias/protocolos tais como WAP, GPRS/EDGE, IrDA, Bluetooth, IEEE 802.11, entre outras. Sua utilização vem abrindo novas aplicações para o acesso à informação em um vasto conjunto de atividades. Dentre as possibilidades de Sistemas Móveis e DM destacam-se os Serviços Baseados em Localização, do inglês Location Based Services (LBS).

LBS são serviços baseados em informações geográficas acessíveis por meio de DM através de redes de comunicações, combinadas ou não com a posição geográfica dos respectivos DM. LBS podem ser aplicados em áreas tais como Transporte, Logística, Rastreamento e Localização de Cargas, Serviços de Gerenciamento e Rastreamento de Frotas e Emergências. Pode-se acompanhar, por exemplo, a localização de um veículo de cargas por meio de monitoramento e rastreamento baseado em LBS.

Um LBS apresenta tipicamente a seguinte arquitetura: DM com Serviço de Localização (GPS ou Serviço de Localização de Operadoras de Telefonia móvel) integrado; Sistema LBS; Sistema de Comunicação; Servidor da Aplicação LBS e; Banco de Dados de Informações Geográficas (LIZAKOSKI, 2008).

Dentre os serviços em um LBS, contempla-se tipicamente Serviço de Diretório, Serviço de Gateway, Serviço de Geocodificação/Geocodificação Reversa, Serviço de Apresentação de Mapas e Serviço de Determinação de Rotas.

Serviços de Diretório são serviços de armazenamento hierárquico de informações, com objetivo principal de facilitar a pesquisa e a recuperação dessas informações (TRIGO, 2007). Nos LBS, os Serviços de Diretório provêm acesso a um diretório online para localização de um determinado lugar, produto ou serviço, por meio da especificação de identificadores, tais como nome, tipo, categoria, dentre outros, ou por meio da posição geográfica, podendo ser utilizada para consultar determinados locais, produtos ou serviços próximos a uma determinada localização, em uma localização específica, ou dentro de uma determinada área (SILVA, 2005).

LBS requerem cada vez mais interoperabilidade entre diferentes plataformas, sistemas operacionais e infraestruturas de rede (SILVA, 2005). O uso da tecnologia Web Services em soluções LBS possibilita contemplar estes requisitos (ARSANJANI et al, 2003). Web Services (WS) é uma arquitetura de software projetada para suportar interoperabilidade nas interações entre computadores em uma rede a partir de conjuntos de protocolos e padrões abertos definidos pela World Wide Web Consortium (W3C). A tecnologia WS tipicamente possui uma interface descritiva em um formato de linguagem denominada Web Services Description Language (WSDL). Outros sistemas podem interagir com a tecnologia WS por meio de mensagens Simple Object Access Protocol (SOAP), tipicamente transmitidas utilizando HTTP com XML em conjunto com outros padrões Web para prover interoperabilidade (W3C, 2010, tradução nossa).

Padrões abertos proporcionam acessibilidade e integração na forma de comunicação entre aplicações desenvolvidas sob estes padrões, explorando as características das redes. Atualmente há padrões abertos com o objetivo de garantir interoperabilidade entre aplicações

LBS. Destacam-se as especificações OpenGIS Location Services (OpenLS), do Open Geospatial Consortium (OGC) (OGC, 2008), e o Mobile Location Protocol (MLP), do Open Mobile Alliance (OMA) (OMA, 2004).

Encontra-se em desenvolvimento na Pré-Incubadora de Base Tecnológica Softsul, do curso de Ciência de Computação da Universidade do Extremo Sul Catarinense (UNESC), o projeto Tecnologia para Gerenciamento Logístico e de Risco em Transporte, voltado ao desenvolvimento de um Sistema de Serviços que envolvem Monitoramento de Localização e Roteirização de Veículos de Carga, dentre outros. Um módulo do sistema englobará os Serviços de Diretório para Localização de Pontos de Interesse relacionados às Posições de Veículos de Transporte Rodoviário de Cargas.

Partindo deste diagnóstico, o presente trabalho busca o desenvolvimento de um Protótipo de LBS para Serviços de Diretório, com ênfase na localização de Posições de Veículos de Transporte Rodoviário de Cargas, baseado em padrões abertos combinados com WS, aplicado ao gerenciamento e rastreamento de frota de veículos de Transporte Rodoviário de Cargas. Para tanto, o desenvolvimento deve seguir uma padronização que garanta interoperabilidade.

1.1 OBJETIVO GERAL

Desenvolver um Protótipo de LBS para Serviços de Diretório para Localização de Posições, baseado em padrões abertos que garantam interoperabilidade.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos que serão apresentados neste trabalho são:

a) compreender os conceitos envolvidos em LBS;
b) compreender Web Services na provisão de interoperabilidade entre aplicações;
c) conhecer os padrões abertos para LBS que definem as formas de acesso aos núcleos (Serviços de Apresentação; Serviços de Diretório; Serviços de Gateway; Serviços de Geocodificação e Geocodificação Reversa e; Serviços de Rotas) de um LBS;
d) comparar os padrões abertos para LBS;
e) selecionar o padrão aberto que garanta a melhor adequação às características do sistema proposto;
f) modelar e desenvolver um protótipo de LBS para Serviços de Diretório baseado em padrões abertos;
g) fornecer suporte tecnológico para o Transporte Rodoviário de Cargas.

1.3 JUSTIFICATIVA

O Brasil tem sua malha viária baseada essencialmente no Sistema de Transporte Rodoviário. Sua matriz rodoviária contém 1.634.071 quilômetros de rodovias, estradas, ruas e outras vias, pavimentadas ou não, com a intenção de deslocar materiais, pessoas ou animais de um determinado ponto a outro (CNT, 2009).

O Transporte Rodoviário é atualmente o sistema de transportes mais utilizado no país, com mais de 90% do deslocamento de passageiros (CNT; COPPEAD/UFRJ, 2002) e 61,1% do movimento de cargas (CNT, 2009). É realizado basicamente por veículos automotores, tipicamente automóveis, ônibus e caminhões. Há aproximadamente 130 mil empresas de Transporte Rodoviário de Cargas instaladas no Brasil com 2.279.141 caminhões e cavalos mecânicos, provendo trabalho, diretamente, a pelo menos 5 milhões de pessoas (CNT, 2009). O transporte resulta em um faturamento anual superior a R$ 40 bilhões e corresponde a participação de 5,3% do PIB nacional (ANTCL, 2001).

Destaca-se a tendência de agregação de valor a produtos e serviços no transporte rodoviário com base em ciência, tecnologia e inovação. Há atualmente 1,45 milhões de veículos rastreados no país, que representam 5% da frota, sendo destes 1,2 milhões de carros de passeio e o restante de veículos de carga (CAMPASSI, 2009). O setor prevê forte crescimento a partir de 2010, quando todos os veículos novos, de passeio e de carga, sairão de fábrica com rastreadores instalados por determinação da resolução nº 245 do Departamento Nacional de Trânsito (DENATRAN, 2009).

Os LBS já são realidade atualmente no mercado rodoviário e vêm conquistando cada vez mais espaço nos dias atuais graças à convergência de tecnologias tais como a de DM e a de redes de comunicação de dados sem fio integradas à web. A implementação de Serviços de

Diretório incorporados a um sistema LBS pode prover melhoria a áreas tais como a de Transporte, envolvendo serviços de localização de cargas e serviços de gerenciamento e rastreamento de frotas, além de promover uma ligação entre a área estudada e a computação. Um sistema LBS envolve aspectos científicos, tecnológicos e inovadores.

Motivado pela carência de soluções disponíveis aplicadas ao Gerenciamento Logístico e de Risco no Transporte Rodoviário de Cargas, encontra-se em desenvolvimento na Pré-Incubadora de Base Tecnológica Softsul, do curso de Ciência de Computação da UNESC, o projeto Tecnologia para Gerenciamento Logístico e de Risco em Transporte para o desenvolvimento de um LBS.

Para o desenvolvimento dos Serviços de Diretório será realizado um estudo acerca dos padrões OpenLS, do OGC (OGC, 2008), e MLP, da OMA (OMA, 2004), voltados à utilização e ao desenvolvimento da aplicação visando garantir interoperabilidade. Este estudo será importante para a concepção e o desenvolvimento de aplicações que futuramente venham a ser construídas e incorporadas ao sistema, facilitando a integração, a comunicação interna, e promovendo reuso.

Diante deste contexto, têm-se a necessidade de estudo de padrões e tecnologias envolvidas no desenvolvimento destes serviços, especificamente Serviços de Diretório, analisando-se a possibilidade de implementação e buscando garantir a interoperabilidade entre as aplicações do LBS.

1.4 ESTRUTURA DO TRABALHO

Esta pesquisa é composta de 7 capítulos. O primeiro capítulo contém uma introdução à pesquisa desenvolvida, e apresentando também os objetivos e a justificativa para sua realização.

No Capítulo 2 são abordados conceitos envolvidos em DM, apresentando as principais soluções para obtenção de suas posições geográficas.

No Capítulo 3 são apresentados os principais conceitos envolvidos em LBS, identificados os diversos núcleos existentes e descritas suas características mais importantes. O capítulo também oferece aprofundamento no núcleo de Serviços de Diretório e apresenta conceitos de Bancos de Dados Geográficos utilizados em LBS

O Capítulo 4 contempla um estudo das tecnologias envolvidas na provisão de interoperabilidade a LBS, abordando conceitos de Web Services e padrões abertos para LBS existentes, onde é selecionando o padrão adequado às características do sistema proposto. Também é feita uma apresentação dos principais conceitos da XML, linguagem utilizada frequentemente tanto em Web Services quanto nos padrões abertos abordados.

No Capítulo 5 são mostrados e discutidos trabalhos correlatos a esta pesquisa.

O trabalho desenvolvido, recursos utilizados, desenvolvimento e provisão de interoperabilidade do protótipo de Serviço de Diretório, e os resultados obtidos estão presentes no Capítulo 6.

Por fim, tem-se a Conclusão, onde são abordadas as considerações finais deste trabalho e propostas de trabalhos futuros.

2 DISPOSITIVOS MÓVEIS

Um Dispositivo Móvel (DM) é um equipamento, com dimensões reduzidas, que possa prover mobilidade no seu uso e esteja acessível independente de tempo e localização.

Devido à suas dimensões reduzidas e consequente miniaturização de seus componentes de hardware, DM tipicamente possuem memória e capacidade de processamento limitados. Deste modo, o desenvolvimento de softwares para tais dispositivos exige atenção especial dos desenvolvedores. Da mesma forma, as necessidades de interface de usuário em DM são diferentes das de computadores de mesa, em consequência do tamanho de seus periféricos de entrada (teclado, botões, etc.) e saída (tela) (ADOBE SYSTEMS INCORPORATED, 2010, tradução nossa). Uma vez que oferecem mobilidade e dependem do tempo de duração da bateria, DM também precisam dispor de mecanismos de monitoramento de nível de energia, bem como prevenção de perda de dados em sua interrupção (PANDEY, 2009, tradução nossa).

Atualmente observa-se o uso de DM em diversas áreas, tais como negócios, estudos, entretenimento, localização, dentre outras.

Com base no conceito de dispositivos móveis, incluem-se diversos componentes da Tecnologia de Informação, tais como telefones celulares, palmtops, laptops, computadores de bordo, componentes baseados em satélites, dentre outros (OGC, 2008, tradução nossa).

Podem-se incluir também os rastreadores veiculares utilizados frequentemente no Transporte Rodoviário de Cargas.

2.1 TECNOLOGIAS DE POSICIONAMENTO

Existem diversas soluções para posicionamento geográfico de DM, categorizadas de acordo com a forma de obtenção de suas coordenadas geográficas. Tais soluções podem ser divididas em três categorias: Baseadas em Terminais, Baseadas em Rede e Soluções Híbridas (GUEDES, 2003).

2.1.1 Soluções Baseadas em Terminais

Soluções Baseadas em Terminais, do inglês Handset Based (HB), consistem na disponibilização de equipamentos capazes de detectar parâmetros necessários para que o DM obtenha suas próprias coordenadas geográficas (GUEDES, 2003), tipicamente um receptor GPS embutido e/ou acoplado (NOKIA, 2010).

2.1.1.1 Sistema de Posicionamento Global

O Sistema de Posicionamento Global[1], do inglês Global Positioning System (GPS), é HB, o qual faz uso de uma rede composta por 24 satélites (GUEDES, 2003) controlados e operados pelo Departamento de Defesa dos Estados Unidos (SILVA, 2005). Cada satélite possui em sua composição quatro relógios atômicos, que podem prover uma referência de horário com erro de 1 nanosegundo (CARARO, 2006), ou seja, com uma precisão de 1 segundo em aproximadamente 30 anos.

[1] Desenvolvido em 1973 para fim militar e disponibilizado em 1980 para uso civil (SILVA, 2005).

O Departamento de Defesa dos Estados Unidos disponibiliza dois tipos de serviços para GPS (GUEDES, 2003):

a) Precision Positioning Service (PPS) – restrito a uso governamental e militar autorizado pelos Estados Unidos. Requer o uso de equipamentos de criptografia e receptores especiais;

b) Stantard Positioning Service (SPS) – utilizado por civis, sem custos e restrições de uso. Possui precisão inferior ao PPS. É o serviço utilizado pelos receptores GPS comerciais.

O GPS provê suas informações com cobertura mundial, 24 horas por dia, 7 dias por semana (GUEDES, 2003). Inicialmente, a precisão das informações para obtenção do posicionamento providas aos usuários do SPS era degradada por uma perturbação intencional denominada Avaliação Seletiva, do inglês Selective Availability, (SA) (CARARO, 2006), resultando em precisão de aproximadamente 100 metros (GUEDES, 2003). Em 02 de maio de 2000 a SA foi desativada (CARARO, 2006), aumentando a precisão para aproximadamente 10 metros, sendo que na prática a precisão tem variado entre 1 e 10 metros (GUEDES, 2003).

Com base nas informações provindas dos satélites GPS, o terminal pode determinar informações como latitude, longitude, altitude e velocidade. Para cálculos de posicionamento por GPS, utilizam-se mecanismos de triangularização de satélites (GUEDES, 2003), conforme demonstra a Figura 1.

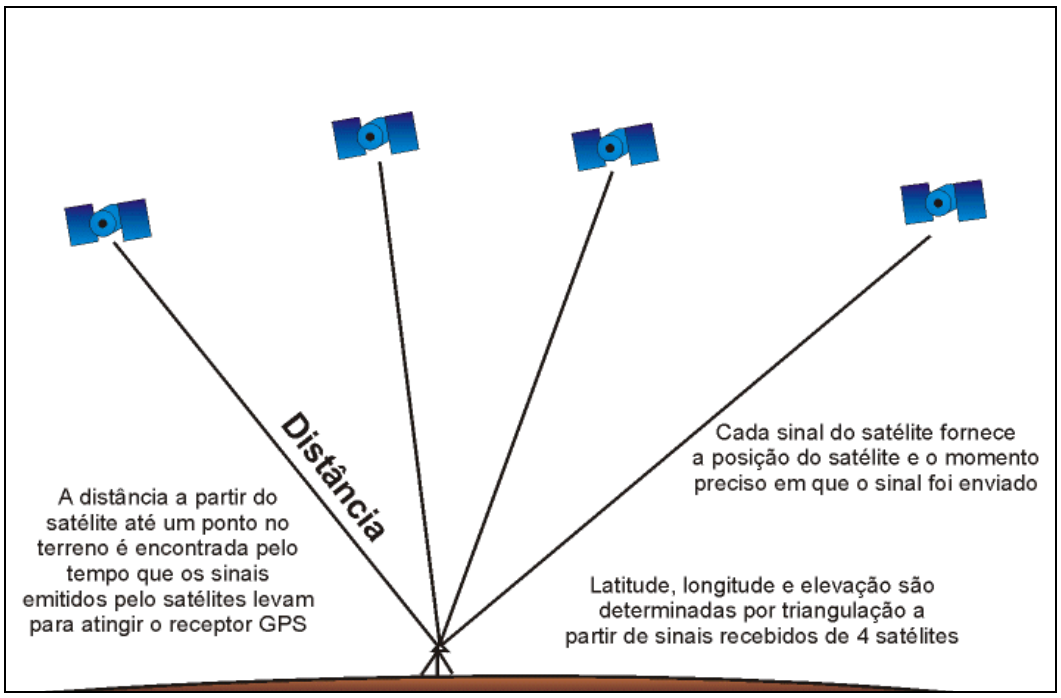


Figura 1. Determinação do posicionamento por GPS
Fonte: CÉZARO, R. (2003)

Para obtenção das coordenadas geográficas são necessários no mínimo 3 satélites. Para a altitude (ou elevação) são necessários no mínimo 4 dos 24 satélites disponíveis (GUEDES, 2003).

Dentre as desvantagens do GPS está a dificuldade da obtenção do posicionamento em ambientes cobertos (LIZAKOSKI, 2008).

2.1.2 Soluções Baseadas em Rede

Soluções Baseadas em Rede, do inglês Network Based (NB), consistem na obtenção do posicionamento geográfico dos DM por meio da localização das antenas da rede de telefonia móvel celular, de forma que todo o processo de obtenção das coordenadas geográficas seja efetuado nas Estações Rádio-Base (ERB) das redes (GUEDES, 2003).

2.1.2.1 Identificação da Célula

A identificação da célula, do inglês Cell Identification (Cell-ID), é uma solução NB onde o DM obtém sua posição geográfica baseado na célula na qual se encontra em uma rede celular (SILVA, 2005). Uma célula é delimita pela área de cobertura de uma ERB, conforme ilustra a Figura 2.

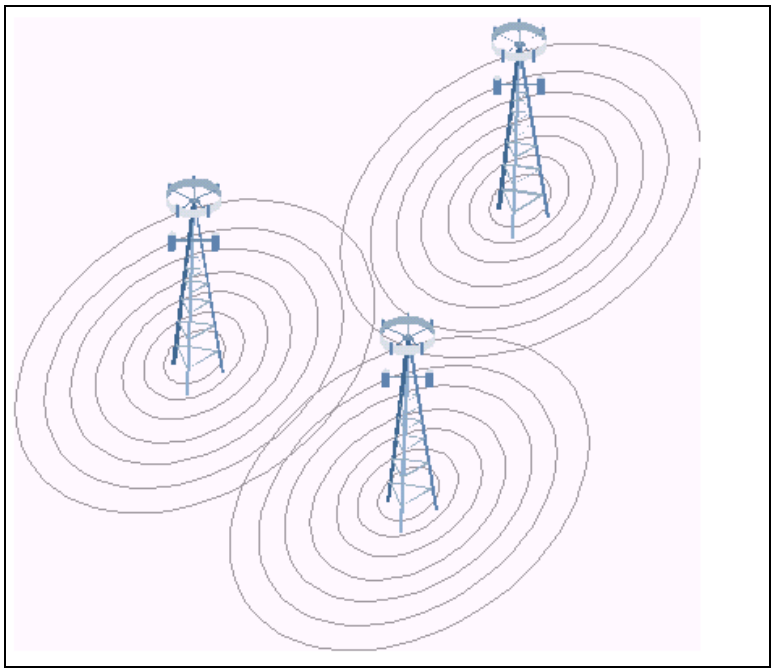


Figura 2. Delimitação de células das ERB
Fonte: FREEDMAN, A. (2000)

Cabe à rede identificar a ERB na qual o DM está conectado, bem como a localização do centroide de cada célula. Esta localização corresponde efetivamente às coordenadas geográficas do DM fornecidas pelo serviço de localização Cell-ID (SILVA, 2005).

A precisão do Cell-ID depende do tamanho da célula na qual o DM encontra-se presente, podendo variar de alguns metros até quilômetros (GUEDES, 2003). O tamanho das

células, por sua vez, é inversamente proporcional à quantidade de ERB disponíveis em determinada área. Áreas urbanas tipicamente possuem maior densidade de ERB, dispondo de células menores e consequentemente maiores precisão e exatidão no posicionamento por Cell-ID (SILVA, 2005). Em redes celulares 3G, onde as células são menores, há maior precisão no posicionamento (LIZAKOSKI, 2008). Áreas rurais tipicamente possuem menor densidade de ERB, resultando em precisão e exatidão inferiores (SILVA, 2005).

A precisão da localização por Cell-ID pode ser melhorada através do cálculo do tempo decorrido para recebimento do sinal enviado pelo DM à ERB. Desta forma, é possível determinar a que distância o DM está da ERB, reduzindo o erro de posicionamento (SILVA, 2005).

Apesar das desvantagens desta tecnologia, sua alta disponibilidade e baixo tempo de resposta a tornam útil em aplicações onde não sejam requeridas altas precisão e exatidão no posicionamento. Um ponto positivo para o Cell-ID é o bom funcionamento em ambientes cobertos (LIZAKOSKI, 2008).

2.1.3 Soluções Híbridas

Soluções Híbridas consistem na união de ambas as soluções (HB e NB) para obtenção da localização do DM (GUEDES, 2003).

2.1.3.1 GPS Assistido

O GPS Assistido, do inglês Assisted GPS (A-GPS), é uma solução híbrida, pois obtém o posicionamento do DM por meio da combinação de informações provindas dos satélites

GPS (HB) com informações provindas de rede celular (NB), estas fornecidas por Servidores de Localização GPS pertencentes às ERB nas proximidades (GUEDES, 2003), conforme se observa na Figura 3.

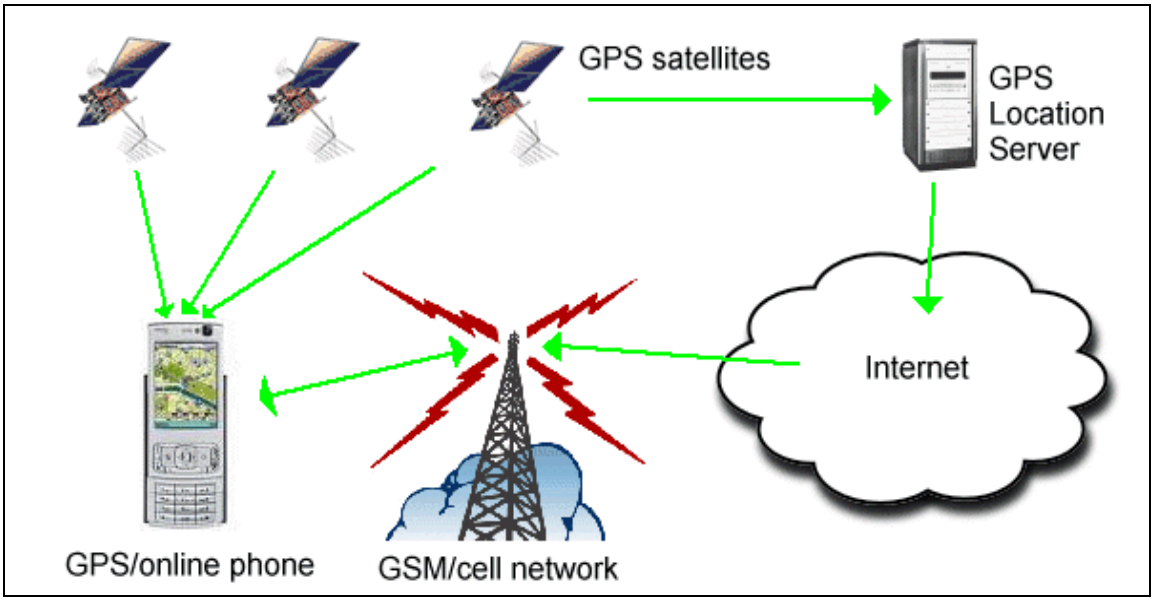


Figura 3. A-GPS
Fonte: LITCHFIELD, S. (2007)

A tarefa de localização do DM por A-GPS é dividida entre o terminal e o Servidor de Localização da ERB. Este servidor comunica-se com os satélites GPS e mapeia sua região de controle, informando ao DM quais satélites deve utilizar para triangularização (GUEDES, 2003). Com isso, algumas funções até então desempenhadas exclusivamente pelos DM para obtenção do posicionamento por GPS, passam a ser executadas pelos Servidores de Localização das redes celulares, reduzindo o tempo de localização e economizando energia do DM (LIZAKOSKI, 2008).

3 SERVIÇOS BASEADOS EM LOCALIZAÇÃO

Em 1996 foi regulamentado nos Estados Unidos o mandato E-911 pelo Federal Communications Comission, determinando que as operadoras de rede celular do país deveriam possuir mecanismos de localização das chamadas de emergência. Este fato impulsionou o crescimento da indústria de DM dotados de tecnologias de posicionamento, fazendo com que surgissem as primeiras aplicações de Serviços Baseados em Localização (LBS) (SILVA, 2005).

O termo localização adotado na definição de LBS refere-se à localização de dispositivos, utilizado para a troca de informações entre usuários e dispositivos, ou entre dois ou mais dispositivos. Exemplos destas informações são o local onde o DM se encontra, a melhor rota entre a Posição de um DM e um Ponto de Interesse, a velocidade que o DM se movimenta, o tempo estimado de chegada do DM ao destino, entre outras (OGC, 2008, tradução nossa).

Tendo como origem a convergência de múltiplas tecnologias, incluindo Sistemas de Informação Geográfica, Internet, comunicação sem fio e DM (ESRI, 2010, tradução nossa), pode-se conceituar LBS qualquer serviço de aplicação que explore a posição geográfica de DM, fornecendo informações úteis aos seus usuários (OGC, 2008, tradução nossa).

Os LBS podem ser divididos, de acordo com sua aplicação, nas seguintes categorias (FAGGION; TROCHERIS, 2003, tradução nossa):

a) informação – dedicada ao mercado consumidor. Incluem-se serviços tais como navegação, informações turísticas, propagandas baseadas em localização, informações de tráfego e localização de Pontos de Interesse próximos (comércio, hospitais, locais de lazer);
b) rastreamento e gerenciamento de frotas – no mercado consumidor, estes serviços podem ser utilizados, por exemplo, para rastrear a localização de pessoas com mal

de Alzheimer, ou para os pais localizarem seus filhos. No mercado de negócios, podem ser aplicados ao rastreamento de frotas de caminhões, provendo, por exemplo, a otimização de rotas, resultando em diminuição do tempo de viagem e aumento de segurança no transporte;

c) assistência e emergência – esta categoria surgiu devido ao longo tempo gasto pelas ambulâncias e demais veículos de resgate para encontrar o local do socorro, em consequência das informações imprecisas sobre a localização das vítimas. Estes serviços estão sendo adotados em diversos países, e requerem precisão apurada para assegurar o envio da assistência ao local correto;
d) diversão – incluem-se serviços tais como jogos baseados em localização, captura e compartilhamento de fotos com informações geográficas, dentre outros.

3.1 NÚCLEOS DE LBS

Os núcleos de um LBS consistem na categorização dos serviços essenciais possíveis nos LBS, de acordo com o tipo de funcionalidade provida. Seus principais núcleos são (OGC, 2008, tradução nossa):

a) Serviço de Diretório – provê acesso a um diretório online, onde é possível encontrar a localização de um Ponto de Interesse específico, ou mais próximo de determinada localização;
b) Serviço de Gateway – busca e fornece a posição de DM conhecidos na rede;
c) Serviço de Geocodificação – transforma informações de endereços (cep, rua, cidade, entre outras) nas posições geográficas correspondentes;

d) Serviço de Geocodificação Reversa – transforma posições geográficas em uma descrição completa dos endereços correspondentes;
e) Serviço de Apresentação – provê a retratação de mapas com base em dados geoespaciais;
f) Serviço de Determinação de Rotas – determina rotas para viagens e informações de navegação entre dois ou mais pontos.

3.2 SERVIÇOS DE DIRETÓRIO

Serviços de Diretório são serviços que fornecem um diretório online para obtenção da localização geográfica de locais, produtos ou serviços de interesse do usuário, conhecidos como Pontos de Interesse (POI). Para consultar um Serviço de Diretório, deve-se essencialmente fornecer ao serviço algum tipo de identificador de fácil interpretação, por exemplo, nome, telefone, ou categoria do POI desejado (OGC, 2008, tradução nossa).

Além de possibilitar a localização de POI específicos, os Serviços de Diretório possibilitam o uso de posições geográficas (tipicamente a de DM) para a localização de POI mais próximos, em uma localização específica, em determinada distância ou dentro de uma determinada área (OGC, 2008, tradução nossa).

Observa-se a possibilidade de aplicação de Serviços de Diretório no Transporte Rodoviário de Cargas onde, por meio de rastreadores instalados nos veículos, obtêm-se as coordenadas geográficas da localização exata dos mesmos. Podem-se utilizar estas coordenadas para consultar um Serviço de Diretório, a fim obter a localização geográfica exata de clientes, oficinas, postos de combustíveis e demais pontos de apoio nas proximidades dos veículos. Pode-

se também localizar clientes, locais de embarque ou descarga em uma área específica, delimitada por um conjunto de coordenadas geográficas.

Como resposta a cada requisição, os Serviços de Diretório devem retornar a localização geográfica e a descrição completa dos POI encontrados (OGC, 2008, tradução nossa).

3.3 BANCOS DE DADOS GEOGRÁFICOS

Para serem compreendidos pelos usuários de LBS, dados geográficos devem estar associados a informações mais amigáveis, tais como nome, endereço, descrição, dentre outras (SILVA, 2005).

Atualmente há Sistemas Gerenciadores de Banco de Dados (SGBD) com suporte a dados geográficos, os quais podem ser relacionados com dados convencionais (SILVA, 2005). Esta característica os denomina Sistemas Gerenciadores de Banco de Dados Geográficos (SGBDG) (CÂMARA et al, 2005). Exemplos de SGBDG são o Oracle Spatial, Oracle Locator, MySQL e PostgreSQL com a extensão espacial PostGIS habilitada (SILVA, 2005). Cada um destes SGBDG define seus próprios tipos de dados geográficos, funções para processamento, formas de armazenamento e recuperação dos dados. (CÂMARA et al, 2005).

SGDBG representam os limites espaciais de cada entidade geográfica através de pontos, linhas e áreas (ou polígonos) (CÂMARA et al, 2005). Estas representações podem ser visualizadas na Figura 4.

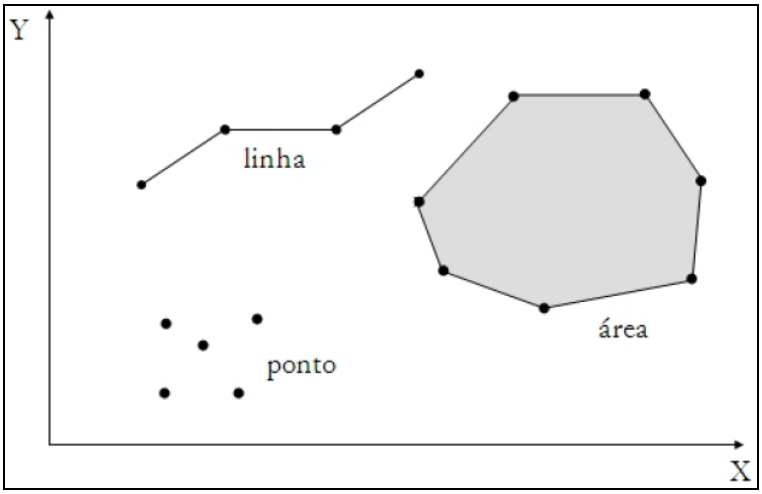


Figura 4. Representações vetoriais em duas dimensões
Fonte: CÂMARA, G. et al (2005)

Os pontos consistem em pares ordenados (x, y), onde x é a latitude e y a longitude que definem suas localizações no espaço. As linhas são formadas por um conjunto de pontos interconectados. Os polígonos são regiões do plano delimitadas por uma série de linhas conectadas em suas extremidades, de modo que o último ponto de cada linha seja o primeiro da próxima (CÂMARA et al, 2005).

Entidades do mundo real, como POI, estradas e limites administrativos podem ser representadas em bancos de dados geográficos respectivamente por pontos, linhas e polígonos (SILVA, 2005).

4 INTEROPERABILIDADE EM LBS

Interoperabilidade é a capacidade de dois ou mais sistemas[2] ou componentes trocarem informações e as utilizarem (IEEE, 1990, tradução nossa). Uma das formas de prover

[2] Tais sistemas podem estar rodando ou não em diferentes dispositivos e/ou sistemas operacionais, e serem desenvolvidos ou não em diferentes linguagens de programação (SILVA, 2005).

interoperabilidade é possibilitando que a troca de informações ocorra por meio de um conjunto padrão de interfaces e formatos abertos, conhecidos como padrões abertos (CHEDE, 2008).

Padrões abertos são padrões disponibilizados publicamente, criados com intuito de prover interoperabilidade, de forma que não beneficiem um produto específico, mas possam ser utilizados livremente, sem cobranças ou restrições de uso, por quaisquer indivíduos ou empresas que desejem trocar informações entre sistemas de forma transparente e padronizada (CHEDE, 2008).

Atualmente, encontra-se bastante difundido o desenvolvimento de padrões abertos para diversas áreas de aplicação. Para tal, há organizações dedicadas ao desenvolvimento de padrões abertos, as quais tipicamente contam com equipes internas de desenvolvedores, juntamente com a participação do público e de organizações filiadas. Um exemplo é o W3C, consórcio internacional dedicado ao desenvolvimento de padrões abertos para a Web (W3C, 2010, tradução nossa).

O uso de padrões, além de garantir interoperabilidade entre sistemas, proporciona diminuição no custo de desenvolvimento de software, beneficiando-se dos esforços de análise e projeto já embutidos nos padrões, reduz a necessidade de aquisição de ferramentas proprietárias, aumenta a oferta de mão-de-obra qualificada, diminui a curva de aprendizado de novos funcionários e reduz o tempo de desenvolvimento (DELPHI GROUP, 2010, tradução nossa).

Aplicações que acessam LBS podem estar disponíveis em diferentes dispositivos, possuir interface com sistemas de banco de dados legados e englobar diversas tecnologias de infraestrutura de rede. A adoção de padrões abertos é uma forma de garantir interoperabilidade no acesso à LBS (SILVA, 2005).

4.1 LINGUAGEM DE MARCAÇÃO EXTENSÍVEL

Desenvolvido e mantido pela W3C desde 1996 (W3C, 2008, tradução nossa), a Linguagem de Marcação Extensível, do inglês Extensible Markup Language (XML), consiste em uma metalinguagem de marcação derivada da Linguagem Padronizada de Marcação Genérica, do inglês Standard Generalized Markup Language[3] (SGML), porém mais simples, que vem sendo utilizada como base para a construção de diversos padrões de troca de informações voltados a domínios e aplicações específicas (MANSANO, 2002).

Com o avanço da World Wide Web, observou-se a necessidade de uma abordagem simplificada da SGML, fato que motivou o desenvolvimento da XML. Características como disponibilidade e expansibilidade da SGML, juntamente com sua funcionalidade, encorajaram a utilização da sua especificação como ponto de partida para o desenvolvimento da XML[4] (ANDERSON, 2001).

Uma das principais características da XML, e que a rotula, é sua extensibilidade, visto que sua especificação possibilita a construção arbitrária de conjuntos de tags[5] para contextos específicos de maneira rápida e fácil. Deste modo, torna-se possível a criação de vocabulários para domínios de atividades específicos, o que além possibilitar reutilização de soluções já testadas, proporciona interoperabilidade na troca de informações entre sistemas relacionados (ANDERSON, 2001).

---

[3] A SGML é uma complexa e poderosa linguagem de marcação, cuja finalidade é a troca e o armazenamento de dados (MANSANO, 2002).

[4] Inclusive, a XML possui compatibilidade retroativa com sistemas que utilizam o padrão SGML (ANDERSON, 2001).

[5] Uma tag XML constitui-se de uma palavra-chave representando uma informação, entreposta em sinais de maior (>) e menor (<) (ANDERSON, 2001).

A possibilidade de ser utilizada em uma ampla variedade de plataformas e ser interpretada por inúmeras ferramentas torna o uso da XML eficaz na provisão de interoperabilidade (MANSANO, 2002).

Segundo Anderson (2001) a estrutura de um documento XML possui tipicamente três partes básicas:

a) prólogo (opcional) – composto por declaração XML, instruções de processamento, codificações de caractere, declarações de tipo de documento e comentários;
b) corpo – composto por dados de caracteres e elementos aninhados hierarquicamente em forma de árvore;
c) epílogo (opcional) – composto por instruções de processamento, comentários e/ou espaços em branco.

4.1.1 Elementos

Forma mais comum de marcação em um documento XML, um elemento é uma espécie de contêiner, cujo conteúdo é delimitado por uma tag inicial e uma tag final. Tags finais diferenciam-se de tags iniciais por conterem uma barra (/) antes do nome do elemento.

Elementos podem conter dados de caractere, elementos filhos, e/ou demais marcações XML (ANDERSON, 2001). A Figura 5 contém um exemplo de elemento com conteúdo:

```
<nome_elemento> Conteúdo </nome_elemento>
```

Figura 5. Exemplo de elemento com conteúdo

Elementos também podem não possuir conteúdo, sendo conhecidos como elementos vazios. Neste caso, utiliza-se uma barra após seu nome (/), conforme ilustra a Figura 6 (ANDERSON, 2001).

```
<nome_elemento_vazio/>
```

Figura 6. Exemplo de elemento vazio

O elemento situado no topo da hierarquia que define o corpo de um documento XML denomina-se elemento de documento, ou elemento raiz. Um elemento raiz é obrigatório e deve ser único em um documento XML. Todos os demais elementos devem ser descendentes do elemento raiz, e denominam-se elementos filhos (PITTS-MOULTIS, 2000).

Elementos também podem possuir atributos, que os anexam informações relativas às suas propriedades. Atributos são compostos por pares de nome e valor, e devem ser inseridos dentro das tags iniciais, após o nome do elemento. Para melhor entendimento, pode-se pensar nos elementos como “substantivos” do XML, e nos atributos como seus “adjetivos” (ANDERSON, 2001). Um exemplo de elemento com atributo pode ser visualizado na Figura 7.

```
<nome_elemento  nome_atributo="valor_atributo">
```

Figura 7. Exemplo de elemento com atributo

4.1.2 Dados de caracteres

Consideram-se dados de caracteres todo o texto de um documento XML que não possui marcação (W3C, 2008, tradução nossa). Valores de atributos e conteúdo de elementos em forma de texto são exemplos de dados de caracteres (ANDERSON, 2001).

A linguagem XML reserva uma série de caracteres para fins de delimitação e marcação. Entretanto, podem ser utilizados na forma de dados de caracteres por meio de sequências de escape denominadas referências de entidade. Referências de entidade devem ser precedidas por um "E" comercial (*&*) e seguidas de ponto-e-vírgula (*;*). Exemplo disto são os caracteres reservados > e <, onde se utilizam respectivamente as referências de entidade *>* e *&lt* (ANDERSON, 2001).

4.1.3 Comentários

Comentários são partes de documentos XML que não devem ser exibidos ou processador pela aplicação (ANDERSON, 2001).

Comentários podem ser inseridos em qualquer parte do documento XML, devendo iniciar com a sequência de caracteres <*!*-- e terminar com -->. Textos comentados podem conter quaisquer tipos de dados, exceto o literal --, que pode ser interpretado erroneamente como um delimitador de fechamento de comentário (ANDERSON, 2001). A Figura 8 exemplifica um comentário de texto em um documento XML.

```
<!-- texto comentado -->
```

Figura 8. Exemplo de comentário XML

4.1.4 Instruções de Processamento

Instruções de processamento fornecem informações para a aplicação, as quais devem ser processadas juntamente com o documento. Assim como nos comentários, as instruções de processamento não são exibidas pela aplicação, mas diferentemente, passam pelo processador XML (ANDERSON, 2001).

Instruções de processamento são opcionais e suas interpretações dependem do processador XML. As instruções de processamento devem iniciar com a sequência de caracteres *<?* e terminar com *?>*[6] (HOLZNER, 2001).

A Figura 9 demonstra uma instrução de processamento compatível com o processador XML do navegador Microsoft Internet Explorer *(*HOLZNER, 2001):

```
<?xml-stylesheet type="text/css" href="arquivo.css"?>
```

Figura 9. Exemplo de instrução de processamento
Fonte: Adaptado de HOLZNER, S. (2001)

[6] A tag <?xml?>, apesar de possuir a sintaxe de uma instrução de processamento, não está inclusa nesta categoria, pois é reservada para o uso em declarações XML (ANDERSON, 2001).

4.1.5 Declaração XML

Documentos XML devem iniciar com uma declaração (W3C, 2008, tradução nossa), indicando que o documento está escrito na linguagem XML. A sintaxe desta declaração deve iniciar com a tag *<?xml?>* seguida de três possíveis atributos, os quais, dependendo de sua obrigatoriedade, devem ser dispostos na seguinte ordem (ANDERSON, 2001):

a) *version* – especifica a versão da XML utilizada;

b) *encoding* (opcional) – especifica a codificação de caracteres utilizada. O valor padrão é *UTF-8*;

c) *standalone* (opcional) – pode conter os valores y*es* (sim) ou *no* (não), indicando se o documento faz ou não referência à entidades externas.

A Figura 10 contém um exemplo de declaração XML.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

Figura 10. Exemplo de declaração XML
Fonte: Adaptado de ANDERSON, R. (2001)

4.1.6 Espaço de Nomes

A linguagem XML permite a criação de *tags* personalizadas. Esta característica, no entanto, propicia a ocorrência de problemas de ambiguidade e colisão de nomes. Para solucioná-los, há uma especificação chamada *Namespaces in XML*, do W3C, que permite a definição de um contexto para os elementos dentro de um espaço identificador (ANDERSON, 2001).

Espaço de nomes consiste em uma coleção de nomes, definidas com um Identificador Uniforme de Recursos, do inglês Uniform Resource Identifier[7] (URI), utilizada em documentos XML para identificar o vocabulário ao qual pertencem elementos e nomes de atributos específicos (W3C, 2009, tradução nossa).

Para declaração de um espaço de nomes, deve-se informar a tag *xmlns*, seguida de um prefixo vinculado a URI do vocabulário que define o uso do elemento. Um dos espaços de nomes de um documento XML pode ser declarado sem prefixo, que será o espaço de nomes padrão do documento (ANDERSON, 2001).

A Figura 11 contém exemplos de declaração de espaço de nomes com e sem prefixo, respectivamente.

Figura 11. Exemplos de declaração de espaço de nomes com e sem prefixo
Fonte: Adaptado de OGC (2010)

Para utilização de um elemento de um espaço de nomes, deve-se informar o prefixo em que se encontra o elemento, seguido do nome do elemento, conforme demonstra a Figura 12 (ANDERSON, 2001):

Figura 12. Exemplo de utilização de um elemento em um espaço de nomes

[7] URI é uma sequência de caracteres que identifica um recurso local ou disponível na Internet (IETF, 2005, tradução nossa).

Deste modo, identifica-se à qual espaço de nomes o elemento pertence, evitando-se a ocorrência de ambiguidades e colisão de nomes.

4.1.7 Esquemas XML

Esquemas XML, do inglês XML Schemas, é uma recomendação da W3C que possibilita o uso de vocabulários compartilhados, fornecendo uma forma de definir e validar estrutura, conteúdo e semântica de documentos XML. Sua construção é feita por meio de Definição de Esquemas XML, do inglês XML Schema Definition (XSD). Seus arquivos devem possuir a extensão *.xsd* (W3C, 2004b, tradução nossa).

Esquemas XML são escritos na linguagem XML, e devem seguir suas recomendações, herdando características tais como extensibilidade e facilidade de uso (ANDERSON, 2001).

Dentre as informações que podem ser definidas em um esquema XML estão: elementos; atributos; quais e quantos elementos são filhos e sua respectiva ordem; se um elemento é vazio ou possui conteúdo; tipos de dados aceitáveis e; valor padrão para elementos e atributos (W3SCHOOLS, 2010, tradução nossa).

Um exemplo de esquema XML é mostrado na Figura 13.

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema">
  <element name="nota">
    <complexType>
      <sequence>
        <element name="de" type="xs:string" />
        <element name="para" type="xs:string" />
        <element name="título" type="xs:string" />
        <element name="mensagem" type="xs:string" />
      </sequence>
    </complexType>
  </element>
</schema>
```

Figura 13. Exemplo de esquema XML
Fonte: Adaptado de W3SCHOOLS (2010)

4.2 WEB SERVICES

Web Services (WS) fornecem uma arquitetura padrão para interoperabilidade entre diferentes aplicações (W3C, 2004a, tradução nossa), de forma que estas possam se comunicar utilizando padrões web (ARSANJANI et al, 2003, tradução nossa).

Um WS é identificado por um Localizador Padrão de Recursos, do inglês Uniform Resource Locator (URL), e difere-se das páginas web comuns devido ao conteúdo que é enviado nas mensagens de requisição e resposta entre cliente e servidor. Este conteúdo pode ser um documento XML formatado de acordo com as regras do Protocolo de Acesso Simples a Objetos, do inglês Simple Object Access Protocol (SOAP) (POTTS; KOPACK, 2003).

A descrição dos WS SOAP é feita por meio de Linguagem de Descrição de Web Services, do inglês Web Services Description Language (WSDL), que possibilita a especificação de detalhes relativos aos serviços, tais como onde e como as funcionalidades são oferecidas (W3C, 2004a, tradução nossa).

Embora suas especificações ofereçam uma base sólida para seu desenvolvimento, WS requerem atenção dos desenvolvedores em relação à segurança, de forma que garanta autenticidade, privacidade e integridade das informações (POTTS; KOPACK, 2003).

4.2.1 Protocolo Simples de Acesso a Objetos

O Protocolo Simples de Acesso a Objetos (SOAP) é uma especificação que define um protocolo para troca de mensagens com WS, visando independência de arquiteturas específicas de hardware e software (POTTS; KOPACK, 2003).

O SOAP define duas partes básicas para as mensagens: cabeçalho (elemento *Header*) e corpo (elemento *Body*). Ambos são contidos em um elemento raiz, chamado *Envelope*. O cabeçalho é opcional, e pode conter desde instruções de processamento até credenciais para autenticação. O corpo contém o *payload*, que representa as próprias informações a serem enviadas. O *payload* pode ser desde um documento baseado em XML, até Chamadas Remotas de Procedimentos, do inglês Remote Procedure Calls (RPC) (POTTS; KOPACK, 2003).

A troca de mensagens SOAP com WS tipicamente é feita por meio do Protocolo de Transferência de Hipertexto, do inglês Hypertext Transport Protocol (HTTP) (SILVA, 2005), mas sua especificação viabiliza a utilização de outros protocolos de comunicação, por exemplo, Protocolo de Transferência de Correio Simples, do inglês Simple Mail Transfer Protocol (SMTP) e Protocolo de Transferência de Arquivos, do inglês File Transfer Protocol (FTP) (W3C, 2004a, tradução nossa).

A Figura 14 apresenta a estrutura de uma mensagem SOAP.

```
<?xml version="1.0"?>
<env:Envelope xlmns:env="ttp://www.w3.org/2003/05/soap-envelope">
  <env:Header>
    <!--Cabeçalho-->
  </env:Header>
  <env:Body>
    <!--Corpo da mensagem-->
  </env:Body>
</env:Envelope>
```

Figura 14. Exemplo contendo a estrutura de mensagens SOAP
Fonte: Adaptado de POTTS, S; KOPACK, M. (2003)

4.2.2 Linguagem de Descrição de Web Services

A Linguagem de Descrição de Web Services (WSDL) baseia-se em XML, utilizada para a criação de documentos de descrição de WS. Estes documentos são conhecidos como documentos WSDL, e fornecem todas as informações necessárias para acesso ao WS (POTTS; KOPACK, 2003).

Os possíveis elementos de um documento WSDL são: (W3C, 2007, tradução nossa):

a) *types* (tipos) – define os tipos de dados que serão utilizados na troca de mensagens. Para garantir interoperabilidade, recomenda-se o uso de XSD;

b) *message* (mensagem) – representa a definição dos dados a serem transmitidos;

c) *portType* (tipo de porta) – contém as operações disponíveis no WS, representadas pelo elemento *operation (operação)*. Cada operação faz referência a uma mensagem de entrada e uma de saída;

d) *binding* (ligação) – especifica protocolo e formato para as mensagens das operações definidas em um tipo de porta particular;

e) *port* (porta) – especifica um endereço, tipicamente uma URL, para cada ligação;

f) *service* (serviço) – agrega um conjunto de portas relacionados entre si.

Um exemplo de documento WSDL pode ser visto na Figura 15.

```
<?xml version="1.0"?>
<definitions name="StockQuote"
  targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.com/stockquote.wsdl"
  xmlns:xsd1="http://example.com/stockquote.xsd"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">
  <types>
    <schema targetNamespace="http://example.com/stockquote.xsd"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <element name="TradePriceRequest">
        <complexType>
          <all>
            <element name="tickerSymbol" type="string" />
          </all>
        </complexType>
      </element>
      <element name="TradePrice">
        <complexType>
          <all>
            <element name="price" type="float" />
          </all>
        </complexType>
      </element>
    </schema>
  </types>
  <message name="GetLastTradePriceInput">
    <part name="body" element="xsd1:TradePriceRequest" />
  </message>
  <message name="GetLastTradePriceOutput">
    <part name="body" element="xsd1:TradePrice" />
  </message>
  <portType name="StockQuotePortType">
    <operation name="GetLastTradePrice">
      <input message="tns:GetLastTradePriceInput" />
      <output message="tns:GetLastTradePriceOutput" />
    </operation>
  </portType>
  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
    <soap:binding style="document"
      transport="http://schemas.xmlsoap.org/soap/http" />
    <operation name="GetLastTradePrice">
      <soap:operation soapAction="http://example.com/GetLastTradePrice" />
      <input>
        <soap:body use="literal" />
      </input>
      <output>
        <soap:body use="literal" />
      </output>
    </operation>
  </binding>
  <service name="StockQuoteService">
    <documentation>My first service</documentation>
    <port name="StockQuotePort" binding="tns:StockQuoteBinding">
      <soap:address location="http://example.com/stockquote" />
    </port>
  </service>
</definitions>
```

Figura 15. Exemplo de documento WSDL
Fonte: Adaptado de POTTS, S; KOPACK, M. (2003)

4.3 PADRÕES ABERTOS PARA LBS

Em 1996, com o surgimento dos primeiros LBS, não havia padronização das interfaces de acesso às informações de localização dos DM. Da mesma forma, não havia interfaces padrões para acesso aos serviços de Sistemas de Informações Geográficas, do inglês Geographic Information Systems (GIS), os quais tipicamente utilizavam padrões proprietários[8] (SILVA, 2005).

Em meados de 2000, houve o surgimento de uma nova fase, onde foram propostos padrões abertos visando solucionar os problemas de interoperabilidade até então enfrentados no acesso à LBS. Pode-se citar o Mobile Location Protocol (MLP), do Open Mobile Alliance (OMA); e o Open Location Services (OpenLS), do Open Geospatial Consortium (OGC) (SILVA, 2005). Tais padrões possuem aplicações distintas, mas complementares. Dentre eles, o padrão aberto que contempla as soluções para acesso aos principais núcleos de LBS é o OpenLS (OMA, 2004, tradução nossa; OGC, 2008, tradução nossa).

4.3.1 Mobile Location Protocol

O Mobile Location Protocol (MLP) é um padrão aberto para LBS que fornece uma interface padronizada entre o Servidor de Localização e a aplicação cliente, visando prover interoperabilidade no acesso a Serviços de Gateway. Sua especificação provê uma infraestrutura para a obtenção da posição de DM dividida em camadas bem definidas (OMA, 2004, tradução nossa).

[8] Padrões proprietários não garantem interoperabilidade, mas intraoperabilidade; o proprietário controla suas funcionalidades e evolução, de forma que favoreça seu próprio negócio (CHEDE, 2008).

O padrão MLP baseia-se na especificação XML, e suas estruturas são definidas por meio de Definição de Tipo de Documento, do inglês Document Type Definition (DTD) (OMA, 2004, tradução nossa), um padrão para definição e validação de documentos XML similar ao XSD, porém mais antigo e com menos funcionalidades (W3SCHOOLS, 2010, tradução nossa).

Apesar das limitações conhecidas no uso de DTD, a especificação do MLP fornece suporte a Serviços de Gateway de forma ampla e bem definida. Contudo, não são fornecidas interfaces de acesso aos demais núcleos de LBS (OMA, 2004, tradução nossa), como Serviço de Diretório, que se encontra no escopo deste trabalho. Com base nesta constatação, conclui-se que a adoção do padrão MLP torna-se imprópria para o desenvolvimento do protótipo de LBS a ser desenvolvido.

4.3.2 Open Location Services

A especificação OpenLS disponibiliza um conjunto de interfaces padronizadas baseadas em XML, que garantem o desenvolvimento de LBS altamente interoperáveis. Estas interfaces definem e provém acesso aos principais núcleos de um LBS, incluindo Serviços de Diretório, Serviços de Gateway, Serviços de Determinação de Rotas, Serviços de Apresentação, Serviços de Geocodificação e Serviços de Geocodificação Reversa (SILVA, 2005). A definição de sua estrutura é feita por meio de XSD.

Por contemplar características relativas ao trabalho, tais como acesso a Serviços de Diretório, requisições e repostas baseadas em XML (que possibilita o uso com mensagens SOAP), o padrão aberto OpenLS será adotado para a provisão de interoperabilidade na comunicação com os serviços do protótipo.

4.3.2.1 Arquitetura dos Serviços

Visando prover mobilidade, os serviços do OpenLS são disponibilizados tipicamente em um servidor Web. Tanto suas requisições quanto suas respostas baseiam-se em XML, e a comunicação com o servidor é realizada por meio do protocolo HTTP (SILVA, 2005), que por sua vez baseia-se em um modelo de requisição e resposta (IETF, 1999, tradução nossa).

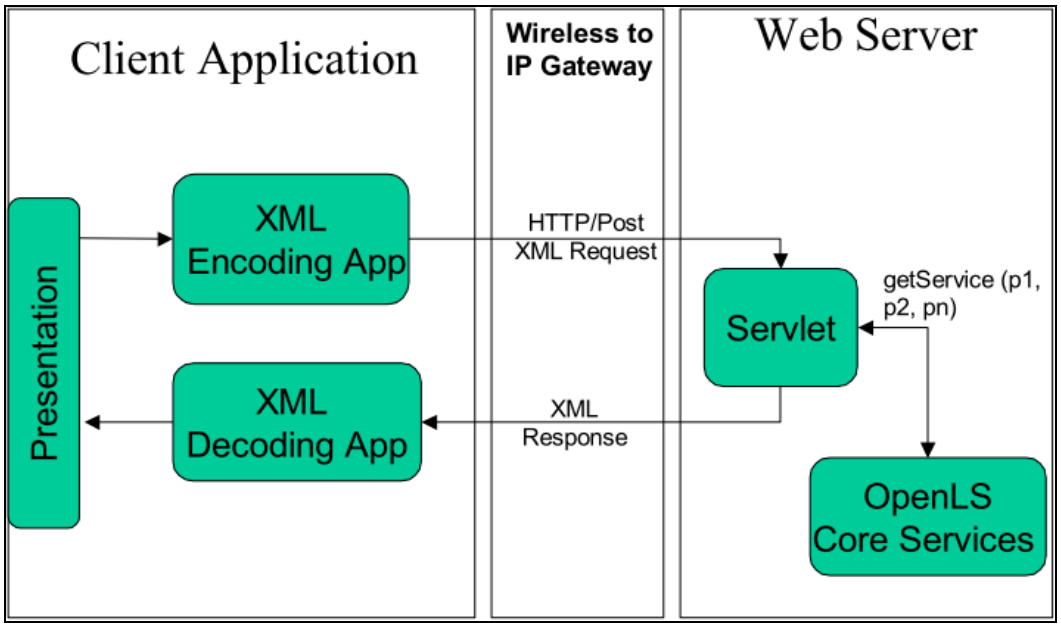


Figura 16. Padrão de Requisição / Resposta do OpenLS
Fonte: OGC (2008)

A Figura 16 exibe a arquitetura padrão do OpenLS, utilizando requisições e respostas baseados em XML. Inicialmente, ocorre o processamento de uma requisição do usuário a um serviço específico na Aplicação Cliente. Esta, por sua vez, processa a requisição e a codifica em XML, encaminhando-a a uma Servlet situada no Servidor Web por meio do método HTTP Post. A Servlet analisa a requisição XML e, de acordo com suas tags, gera a função adequada para chamar o devido núcleo do LBS. O serviço de núcleo processa a requisição, gera a resposta e a

devolve à Servlet, que a codifica em XML e a encaminha para a Aplicação Cliente. Esta, por sua vez, decodifica e processa a resposta XML e, finalmente, formata o conteúdo para exibição no dispositivo[9] (OGC, 2008, tradução nossa).

4.3.2.2 Estrutura das Requisições e Respostas

As interfaces do OpenLS utilizam estruturas e tipos de dados definidos em XSD, denominados Tipos Abstratos de Dados, do inglês Abstract Data Type (ADT) (OGC, 2008, tradução nossa). Tanto os ADT quanto as mensagens de requisição e resposta utilizados nos serviços do OpenLS são codificados em XML para Serviços de Localização[10], do inglês XML for Location Services (XLS), devendo declarar o seguinte espaço de nomes (OGC, 2008, tradução nossa): *xmlns:xls="http://www.opengis.net/xls"*.

No corpo do documento XML que define as requisições e respostas ao LBS através das interfaces do OpenLS, há um elemento raiz chamado *XLS*, o qual possui dois elementos filhos: *RequestHeader* e *Request* para as requisições e; *ResponseHeader* e *Response* para as respostas (OGC, 2008, tradução nossa).

O elemento *RequestHeader* define o cabeçalho da mensagem de requisição, fornecendo informações úteis para autenticação e identificação. Os atributos de um elemento *RequestHeader* são (OGC, 2008, tradução nossa):

a) *clientName* (opcional) – nome para autenticação do cliente;

b) *clientPassword* (opcional) – senha para autenticação do cliente;

[9] Caso o dispositivo possua recursos limitados, é possível que o processamento da resposta seja efetuado diretamente no Servidor Web, que enviará o conteúdo à Aplicação Cliente pronto para exibição no dispositivo (OGC, 2008, tradução nossa).

[10] XLS é um esquema XML que define um vocabulário específico para codificação de mensagens de requisição e resposta do OpenLS, e seus respectivos ADT (OGC, 2008, tradução nossa).

c) *sessionID* (opcional) – identificador da sessão. Deve ser retornado no cabeçalho da resposta;

d) *srsName* (opcional) – referencia um Sistema de Referência de Coordenadas, do inglês Coordinate Reference Systems (CRS). O valor padrão é *WGS84*, que se refere ao Identificador de Referência Espacial, do inglês Spatial Reference Identifier (SRID) 4326;

e) *MSID* – identificador único que pode ser utilizado para diferentes propósitos, por exemplo, para faturamento do serviço.

O elemento *Request* define o corpo da requisição ao serviço e contém atributos com informações em comum a todas as requisições, juntamente com um elemento identificado na especificação OpenLS como *_RequestParameters*[11], o qual representa o conjunto de parâmetros que formam uma requisição a um núcleo específico de um LBS. Os atributos de um elemento *Request* são (OGC, 2008, tradução nosa):

a) *methodName* – nome do método que deverá ser chamado pelo serviço (ex.: *DirectoryRequest*, para Serviços de Diretório);

b) *version* – versão dos parâmetros de requisição suportada pelo cliente;

c) *requestID* – identificador da requisição, único dentro do escopo da sessão;

d) *maximumResponses* (opcional) – número máximo de respostas que deverão ser geradas com o processamento da requisição.

O elemento *ResponseHeader* define o cabeçalho da mensagem de resposta, podendo conter, além do identificador da sessão especificado na requisição (*sessionID*), uma lista com os erros encontrados no processamento no seu processamento (OGC, 2008, tradução nossa).

[11] O *_RequestParameters* do Serviço de Diretório será detalhado na seção 4.3.2.3.1.

O elemento *Response* define o corpo da resposta do serviço, e contém informações em comum a todas as respostas, juntamente com um elemento identificado na especificação como *_ResponseParameters*[12], o qual representa o conjunto de parâmetros que formam a resposta de um núcleo específico do LBS. Os atributos de um elemento *Response* são (OGC, 2008, tradução nossa):

a) *version* – versão dos parâmetros da resposta suportada pelo cliente;
b) *requestID* – identificador da resposta, único dentro do escopo da sessão;
c) *numberOfResponses* (opcional) – número de respostas geradas com o processamento da requisição.

Cada uma das interfaces de acesso do OpenLS (por exemplo, a de Serviços de Diretório) possui seus próprios parâmetros de requisição (*_RequestParameters*) e resposta (*_ResponseParameters*) (SILVA, 2005).

4.3.2.3 Serviço de Diretório do OpenLS

A especificação do OpenLS sugere dois tipos de busca que podem ser realizadas utilizando Serviços de Diretório: Serviço de Diretório Específico (SDE) e Serviço de Diretório por Proximidade (SDP) (WILBRINK, 2010, tradução nossa).

O SDE provê acesso a um diretório online para encontrar a localização de POI específicos, ou seja, deve-se especificar exatamente qual o POI procurado através de propriedades que o identifiquem. Este serviço independe da localização do DM, por exemplo, pode-se querer localizar um local específico em um país distante (OGC, 2008, tradução nossa).

[12] O *_ResponseParameters* do Serviço de Diretório será detalhado na seção 4.3.2.3.2.

Para consultar o SDE, é necessário informar no mínimo um dos parâmetros que identificam o POI ao qual se deseja localizar (WILBRINK, 2010, tradução nossa).

O SDP provê acesso a um diretório online para localizar POI mais próximos de um ponto referencial (OGC, 2008, tradução nossa), por meio do uso de um filtro espacial que pode ser configurado para localizar POI mais próximos, em determinada distância, dentro de determinada área ou em um endereço específico. Para a definição da proximidade pode-se utilizar a posição de DM, ou a de um POI qualquer. Caso se opte pela segunda opção, pode-se obter a posição do POI através de uma consulta ao SDE (WILBRINK, 2010, tradução nossa). Por exemplo, pode-se querer localizar as transportadoras mais próximas de um posto de gasolina específico, num limite máximo de 500 metros de distância. Neste caso, a localização do posto de gasolina pode ser obtida através de consulta ao SDE. Esta deverá ser fornecida ao SDP para a localização das transportadoras, juntamente com o uso de um filtro espacial propriamente configurado.

4.3.2.3.1 Parâmetros de Requisição do Serviço de Diretório

Os parâmetros de requisição (*_RequestParameters*) de um Serviço de Diretório definidos pelo OpenLS devem iniciar com o elemento *DirectoryRequest*. Os possíveis atributos para o elemento *DirectoryRequest são:* (OGC, 2008, tradução nossa):

a) *sortCriteria* (opcional) – define a propriedade do POI que será usada para ordenar o resultado. Pode assumir os valores *Name*, *Distance*, ou demais propriedades dos POI;

b) *sortDirection* (opcional) – define a ordenação das respostas. Pode assumir os valores *Ascending* (valor padrão) para ordem crescente, ou *Descending* para ordem decrescente;

c) *distanceUnit* (opcional) – especifica a unidade de medida para a distância. Pode assumir os valores *M* (valor padrão) para metros, *KM* para quilômetros, *DM* para decímetros, *MI* para milhas, *YD* para jardas e *FT* para pés.

Além dos atributos opcionais citados anteriormente, o elemento *DirectoryRequest* contém dois elemento filhos: *POILocation* e *POIProperties* (OGC, 2008, tradução nossa).

O elemento *POILocation* é opcional e define o filtro para busca dos POI, podendo conter um dos seguintes elementos filhos (OGC, 2008, tradução nossa):

a) *Address* – define um endereço para busca do POI

b) *Nearest* – define um filtro espacial que seleciona o POI mais próximo de uma determinada localização;

c) *WithinDistance* – define um filtro espacial que seleciona POI localizados em determinada distância de uma localização;

d) *WhitinBoundary* – define um filtro espacial que seleciona POI localizados dentro de uma determinada área delimitada por um polígono.

O elemento *POIProperties* é obrigatório e define critérios de seleção dos POI em uma lista de propriedades, podendo possuir o atributo *directoryType* (opcional), que define o tipo de diretório para a busca, além de um ou mais elementos filhos *POIProperty* (OGC, 2008, tradução nossa).

Cada elemento *POIProperty* especifica uma propriedade para busca do POI, e deve possuir um par de nome representados pelos seguintes atributos (OGC, 2008, tradução nossa):

a) *name* – nome da propriedade do POI. Pode assumir os valores *ID*, *POIName*, *PhoneNumber*, *Keyword* (palavra-chave qualquer, como a categoria do POI), *NAICS_type*, *NAICS_subType*, *NAICS_category*, *SIC_type*, *SIC_subType*, *SIC_category*, *SIC_code* ou *other*;

b) *value* – valor da propriedade do POI.

A Figura 17 demonstra uma requisição utilizando a interface de acesso ao Serviço de Diretório do OpenLS.

```
<?xml version="1.0" encoding="UTF-8" ?>
<XLS xmlns=http://www.opengis.net/xls xmlns:gml=http://www.opengis.net/gml
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
  xsi:schemaLocation="http://www.opengis.net/xls…" version="1.0">
  <RequestHeader clientName="someName" clientPassword="password" />
  <Request requestID="123" maximumResponses="1" version="1.1"
    methodName="DirectoryRequest">
    <DirectoryRequest>
      <POILocation>
        <Address countryCode="US">
        </Address>
      </POILocation>
      <POIProperties>
        <POIProperty name="SIC_code" value="1234567890" />
      </POIProperties>
    </DirectoryRequest>
  </Request>
</XLS>
```

Figura 17. Exemplo de requisição a um Serviço de Diretório utilizando o OpenLS
Fonte: ORACLE (2010)

4.3.2.3.2 Parâmetros de Resposta do Serviço de Diretório

Os parâmetros de resposta (*_ResponseParameters*) de um Serviço de Diretório definidos pelo OpenLS devem iniciar com o elemento *DirectoryResponse*, que contém um elemento filho *POIContext* para cada POI que satisfaz os critérios de busca da requisição (OGC, 2008, tradução nossa).

O elemento *POIContext* possui um par de elementos filhos *POI* e *Distance* que representam respectivamente o POI encontrado e sua distância[13] da localização determinada (OGC, 2008, tradução nossa).

O elemento *POI* pode possuir os seguintes atributos (OGC, 2008, tradução nossa):

a) *ID* – número de identificação do POI;

b) *POIName* (opcional) – nome do POI;

c) *phoneNumber* (opcional) – número de telefone do POI;

d) description (opcional) – descrição do POI;

Além dos atributos citados anteriormente, o elemento *POI* também pode conter os seguintes elementos (OGC, 2008, tradução nossa):

a) *POIAttributeList* – lista com um ou mais atributos relativos às propriedades do POI, possuindo um ou mais pares de nome e valor.

b) *Point* – coordenadas geográficas do POI;

c) *Address* – endereço do POI.

A Figura 18 contém a resposta de um Serviço de Diretório para uma requisição OpenLS.

[13] A distância será zero para requisições com a restrição WhitinBoundary (OGC, 2008, tradução nossa).

```
<?xml version="1.0" encoding="UTF-8" ?>
<xls:XLS xmlns:xls=http://www.opengis.net/xls xmlns:gml=http://www.opengis.net/gml
  xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance version="1.0">
  <xls:ResponseHeader />
  <xls:Response requestID="123" version="1.1">
    <DirectoryResponse xmlns="http://www.opengis.net/xls">
      <xls:POIContext xmlns:xls="http://www.opengis.net/xls">
        <xls:POI ID="1" POIName="Borders Books &amp;amp; More"
          phoneNumber="415-731-0665" description="Books &amp;amp; more">
          <POIAttributeList xmlns="http://www.opengis.net/xls">
            <xls:SIC xmlns:xls=http://www.opengis.net/xls
              category="Book stores" code="1234567890" subType="" type="" />
          </POIAttributeList>
          <gml:Point xmlns:gml="http://www.opengis.net/gml">
            <gml:pos dimension="2" srsName="4326">
              -122.4753965
              37.7269066
           </gml:pos>
          </gml:Point>
          <xls:Address countryCode="US">
            <xls:StreetAddress>
              <xls:Building number="233" />
              <xls:Street>Winston Drive</xls:Street>
            </xls:StreetAddress>
            <xls:Place type="CountrySubdivision">CA</xls:Place>
            <xls:Place type="CountrySecondarySubdivision" />
            <xls:Place type="Municipality">
              San Francisco
           </xls:Place>
            <xls:Place type="MunicipalitySubdivision" />
            <xls:PostalCode>94132</xls:PostalCode>
          </xls:Address>
        </xls:POI>
      </xls:POIContext>
    </DirectoryResponse>
  </xls:Response>
</xls:XLS>
```

Figura 18. Exemplo de resposta de um Serviço de Diretório utilizando o OpenLS
Fonte: ORACLE (2010)

5 TRABALHOS CORRELATOS

5.1 IMPLEMENTAÇÃO DE SERVIÇOS BASEADOS EM LOCALIZAÇÃO UTILIZANDO ARQUITETURAS E PADRÕES ABERTOS

Dissertação de mestrado desenvolvida por Grace Kelly de Castro Silva, apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Estadual de Campinas (UNICAMP) no ano de 2005.

A dissertação apresentou um estudo sobre as tecnologias envolvidas no desenvolvimento de LBS e propôs uma arquitetura baseada em padrões abertos para disponibilização destes serviços, utilizando WS. Esta arquitetura foi validada através do desenvolvimento de um protótipo do Serviço de Apresentação de Mapas, utilizando a interface de acesso definida na especificação OpenLS.

Os resultados estabeleceram que, devido à versão da especificação OpenLS publicada no período de desenvolvimento do trabalho (OpenLS 1.0) ainda não estar totalmente preparada para o uso com a tecnologia Web Services, foram encontradas dificuldades na implementação do protótipo. No entanto, tais problemas puderam ser contornados, viabilizando o uso de WS com o OpenLS no desenvolvimento de LBS (SILVA, 2005).

5.2 OPENROUTESERVICE

Projeto desenvolvido por Pascal Neis e Alexander Zipf, do Grupo de Pesquisa em Cartografia do Departamento de Geografia da Universidade de Bonn, Alemanha.

O OpenRouteService é um Serviço de Rota baseado nas especificações OpenLS, que utiliza dados geográficos de rodovias providos pelo OpenStreetMap[14] (NEIS, 2008, tradução nossa). O serviço e destina-se a projetos acadêmicos e sem fins lucrativos (ORS, 2010, tradução nossa), e pode ser acessado livre e gratuitamente através do website *http://www.openrouteservice.org/*.

Há também uma versão do OpenRouteService chamada *Haiti Special*, também desenvolvida com base nas especificações OpenLS, que provê um Serviço de Rotas para a região afetada pelo terremoto no Haiti em 2010, onde é possível criar rotas com desvio dos locais atingidos (ORS, 2010, tradução nossa), disponibilizado em *http://openls.geog.uni-heidelberg.de/osm-haiti/*.

[14] O projeto OpenStreetMap visa criação e disponibilização de dados geográficos com licença de uso livre (OSM, 2010, tradução nossa).

6 TRABALHO DESENVOLVIDO

O presente trabalho refere-se ao desenvolvimento de um protótipo de LBS, especificamente um Serviço de Diretório, utilizando o padrão OpenLS, da OGC, juntamente com Web Services (WS), com intuito de prover interoperabilidade.

Este capítulo descreve e detalha as etapas realizadas para a construção do referido protótipo.

A metodologia deste trabalho baseou-se nas seguintes etapas: descrição dos recursos utilizados, modelagem e desenvolvimento do Serviço de Diretório, provisão de interoperabilidade no acesso ao protótipo desenvolvido, execução dos testes e resultados obtidos.

6.1 RECURSOS UTILIZADOS

O protótipo proposto contempla uma série de características cujo desenvolvimento requer planejamento e contextualização entre si. Incluem-se características tais como implementação de padrão aberto baseado em XML, distribuição de serviços, disponibilização de acesso aos serviços por meio de WS, comunicação com banco de dados, representação e processamento de informações georeferenciadas, entre outras. Com base nesta constatação, entende-se que a seleção adequada dos recursos de software implica diretamente na redução do tempo e dos esforços necessários para o desenvolvimento. Partindo deste princípio, foram levantados os recursos disponíveis com licença de uso livres para o desenvolvimento deste tipo de aplicação. Após o levantamento foram selecionados os que melhor atendem às necessidades do sistema proposto.

6.1.1 Ambiente de Desenvolvimento

A linguagem Java foi considerada apropriada devido a características tais como orientação a objetos e suporte a múltiplas plataformas.

Primeiramente, verificou-se a necessidade de representação das informações contidas nos esquemas XML do OpenLS em linguagem de programação, de forma que pudessem ser interpretadas e manipuladas pelo serviço para processamento das requisições e geração das respostas. A orientação a objetos, além de proporcionar extensibilidade e permitir a reutilização de código, possibilitou a representação dos elementos e atributos XML na memória através de objetos, os quais puderam ser manipulados facilmente utilizando recursos da linguagem escolhida.

O suporte a múltiplas plataformas garante que o serviço possa ser implantado em servidores rodando em diferentes sistemas operacionais, bastando que estes sejam providos de Máquina Virtual Java.

O Ambiente de Desenvolvimento Integrado, do inglês Integrated Development Environment (IDE), utilizado foi o Eclipse, que possui funcionalidades que facilitaram o desenvolvimento do protótipo, dentre as quais se podem citar integração com Apache Maven e Apache Tomcat.

O Apache Maven foi utilizado para gerenciamento de automação do projeto, provendo facilidades como automatização da compilação, gerenciamento de dependências e distribuição da aplicação. Sua configuração centralizou-se em um único arquivo chamado “pom.xml”, e sua adoção facilitou a construção do ambiente de desenvolvimento, permitindo foco na aplicação em si. O Apache Tomcat, por sua vez, escolhido como servidor de aplicação Java, devido a características com acessibilidade e facilidade no uso.

Entre as dependências obtidas por meio do Maven, estão o Hibernate, o Hibernate Spatial, e o Apache CXF. O Hibernate é um framework de mapeamento objeto-relacional para Java que facilita o mapeamento e a persistência dos objetos da aplicação em uma base de dados relacional. Possui diversos recursos para facilitar o acesso ao banco de dados, acelerando o processo de desenvolvimento e aumentando a produtividade. O Hibernate Spatial é uma extensão do Hibernate focada no mapeamento e persistência de objetos com tipo de dados geográficos.

Após definidos os recursos do ambiente de desenvolvimento para desenvolvimento do serviço, foram selecionados os recursos para a provisão da interoperabilidade. Neste ponto, foram identificadas, testadas e avaliadas diversas Interfaces de Programação de Aplicações, do inglês Application Programming Interface (API), voltadas ao desenvolvimento de WS e implementação de padrões baseados em XML na linguagem Java.

Para a implementação do OpenLS, foram testadas as API Java Architecture for XML Binding (JAXB) e XMLBeans. Ambas as API se propõe a automatizar a criação de classes Java com base em esquemas XML por meio de uma ferramenta chamada Binding Compiler, e a converter objetos Java para documentos XML (Marshalling) e documentos XML para objetos Java (Unmarshalling). A Figura 19 ilustra o funcionamento do JAXB, que se assemelha ao do XMLBeans.

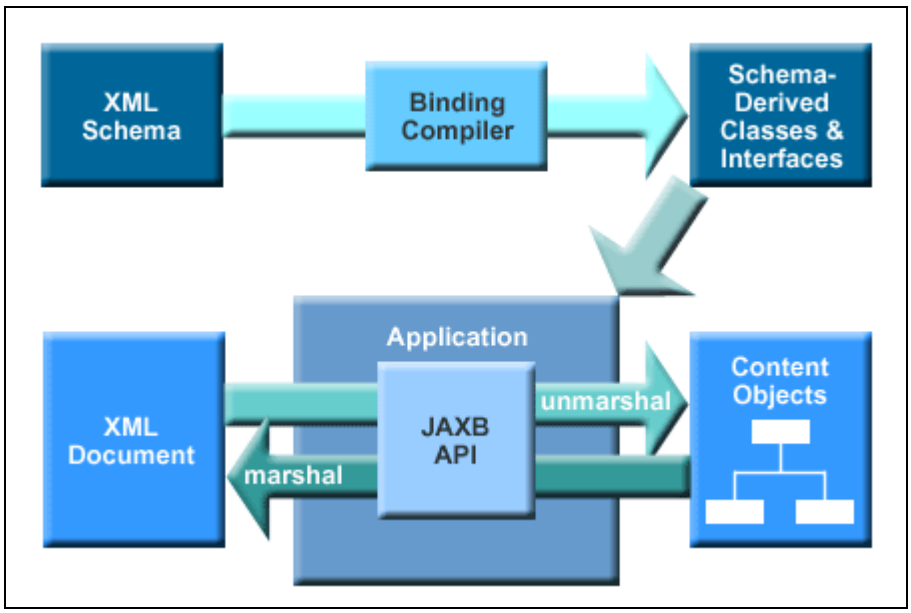


Figura 19. Funcionamento da API JAXB
Fonte: ORT, E.; MEHTA, B. (2003)

O XMLBeans utiliza como Binding Compiler a aplicação executável em linha de comando *scomp*, que gera as classes Java dentro de arquivos com extensão Java Archive (JAR). Nos testes executados, a aplicação não conseguiu localizar o compilador Java declarado nas variáveis de ambiente do sistema operacional, cujo caminho precisou ser adicionado explicitamente por meio do parâmetro *–compiler*. Também houve dificuldade na geração das classes a partir de esquemas XML com dependências (importações) de outros esquemas. O problema foi contornado com a adição do parâmetro *–cp* seguido do arquivo JAR correspondente à dependência.

O JAXB, por sua vez, utiliza como Binding Compiler a aplicação também executável em linha de comando *xjc*, que gera as classes diretamente em uma árvore de diretórios definida de acordo com o pacote fornecido na compilação. Nos testes iniciais, a aplicação apresentou conflito entre alguns elementos do esquema XML. Entretanto, este problema pôde ser contornado facilmente com a criação de um arquivo de customização de binding, o qual foi fornecido ao

Binding Compiler por meio do parâmetro *–b*. O JAXB foi considerado a melhor opção, e selecionado para utilização no protótipo.

Para o desenvolvimento do WS, foram testadas as API Spring Web Services (Spring-WS) e Java API for XML Web Services (JAX-WS).

O Spring-WS é um subprojeto do Spring que oferece suporte ao desenvolvimento de WS. Após diversos testes, o Spring-WS mostrou-se mais complexo e limitado que o JAX-WS. A principal dificuldade, que descartou seu uso, foi o fato de a API ter se mostrado incapaz de gerenciar múltiplos esquemas XML, necessidade notória da especificação OpenLS, que faz uso constante de herança na definição de sua estrutura.

O JAX-WS tem como principal característica o uso de anotações para a construção de WS. Todos os testes executados com esta API foram satisfatórios, motivo pelo qual foi adotada para uso no trabalho. Seu uso tornou simples o desenvolvimento e a configuração do WS do protótipo desenvolvido.

Outro fato que pesou na escolha do JAXB e do JAX-WS é a existência do Apache CXF, um framework de código fonte aberto que provê uma infraestrutura robusta para a construção e desenvolvimento de WS. O framework faz uso do JAX-WS como API de WS, e integra-se com o Spring, que é utilizado no protótipo. O CXF também se integra com diversas API para Binding, sendo sua opção padrão o JAXB. A integração com o JAXB promove abstração do processo de Marshalling e Unmarshalling das requisições e respostas do serviço, além de facilitar a validação de documento XML com base nos esquemas.

6.1.2 Banco de Dados

Após comparação entre os SGBD existentes, optou-se pelo PostgreSQL. Diversas características contribuíram para a sua escolha, dentre as quais se pode citar suporte aos principais sistemas operacionais, licença de uso livre, estabilidade, popularidade e, principalmente, a existência do PostGIS. O PostGIS é uma extensão espacial disponibilizada livremente, que adiciona suporte aos principais tipos de dados geográficos existentes, além de diversas funções para análise e processamento dos dados. A instalação do PostgreSQL inclui a ferramenta front-end PGAdmin, que foi utilizada para a administração do banco de dados.

Apesar de o PostgreSQL com o PostGIS ser considerado a melhor opção gratuita, o uso do Hibernate e do Hibernate Spatial tornam a aplicação independente de banco de dados, ou seja, caso o desenvolvedor opte por passar a utilizar outro SGBD, como o software proprietário Oracle Spatial, o Hibernate converterá internamente a sintaxe de acesso para a linguagem apropriada ao novo banco, eliminando a necessidade de alterações no código-fonte da aplicação.

6.1.3 Ambiente de Testes

Para a realização dos testes, utilizou-se a aplicação de código fonte aberto SoapUI. A ferramenta permite consumir e efetuar testes em WS SOAP, possibilitando a importação e geração automática de modelos das requisições descritas no WSDL. Com o seu uso, foi possível testar o acesso ao protótipo do serviço desenvolvido por meio da troca de mensagens SOAP com o WS, utilizando no payload das mensagens requisições XML baseadas no padrão OpenLS.

A Figura 20 ilustra a integração de todos os recursos utilizados no desenvolvimento do trabalho, nos lado do cliente e servidor.

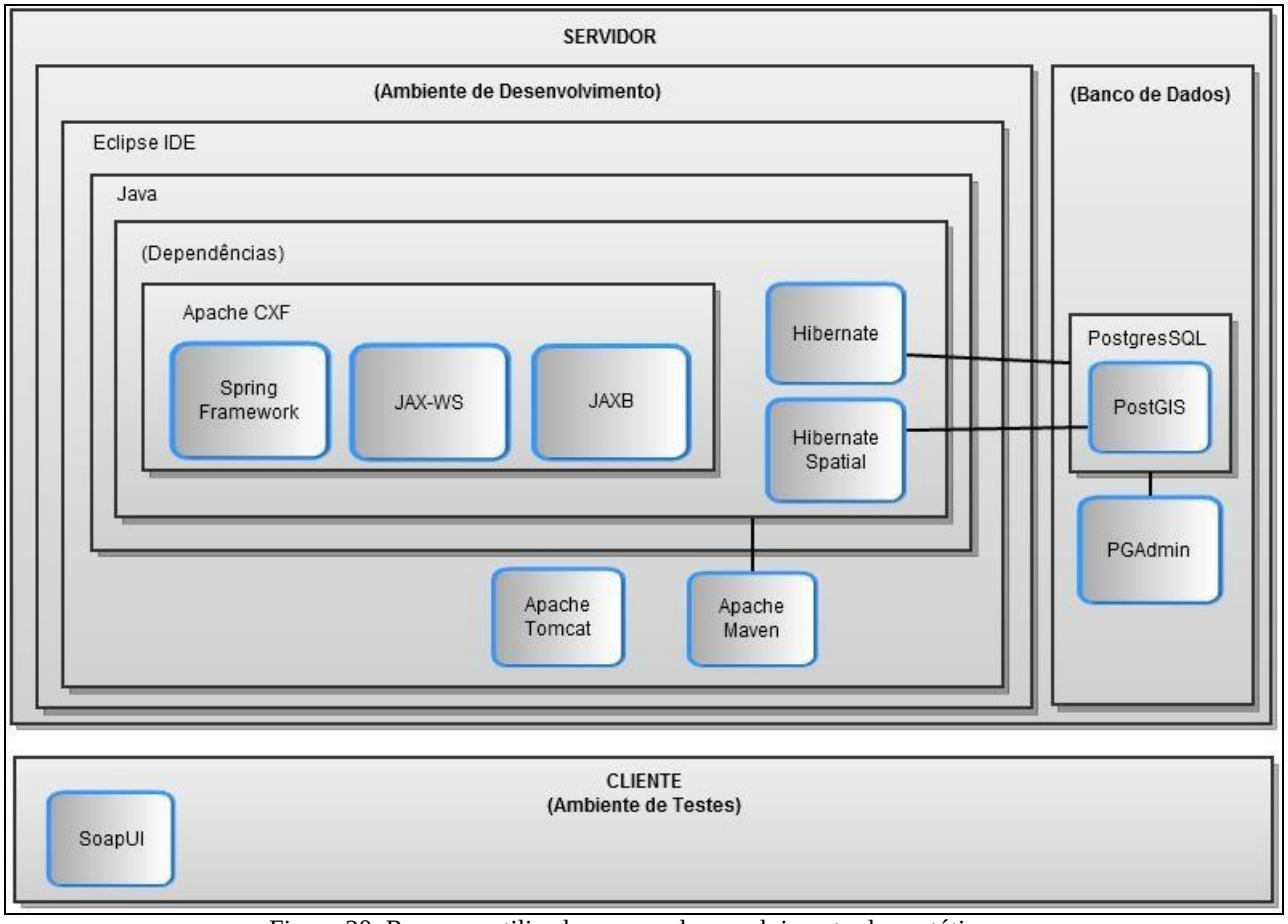


Figura 20. Recursos utilizados para o desenvolvimento do protótipo

6.2 MODELAGEM E DESENVOLVIMENTO DO PROTÓTIPO DE SERVIÇO DE DIRETÓRIO

Serviços de Diretório fornecem o posicionamento de POI aos usuários. Para que isto seja possível, os serviços necessitam de uma base de dados contendo, além de atributos de identificação dos POI, a localização geográfica exata dos mesmos. A base de dados para o protótipo foi criada por meio do PGAdmin. Para prover suporte aos tipos de dados geográficos, instalou-se o PostGIS à base de dados, por meio da adição da linguagem *plpgsql*, e dos scripts *postgis.sql*, *spatial_ref_sys.sql* e *postgis_comments.sql*.

Criada e configurada a base de dados, criou-se a entidade *POI*, modelada para conter informações que representem Pontos de Interesse. A entidade possui atributos para identificação de POI, além de um atributo do tipo *com.vividsolutions.jts.geom.Point* para conter as coordenadas geográfica do POI, e um atributo do tipo *Endereco* para conter informações relativas ao endereço do POI[15]. Sua modelagem pode ser vista na Figura 21.

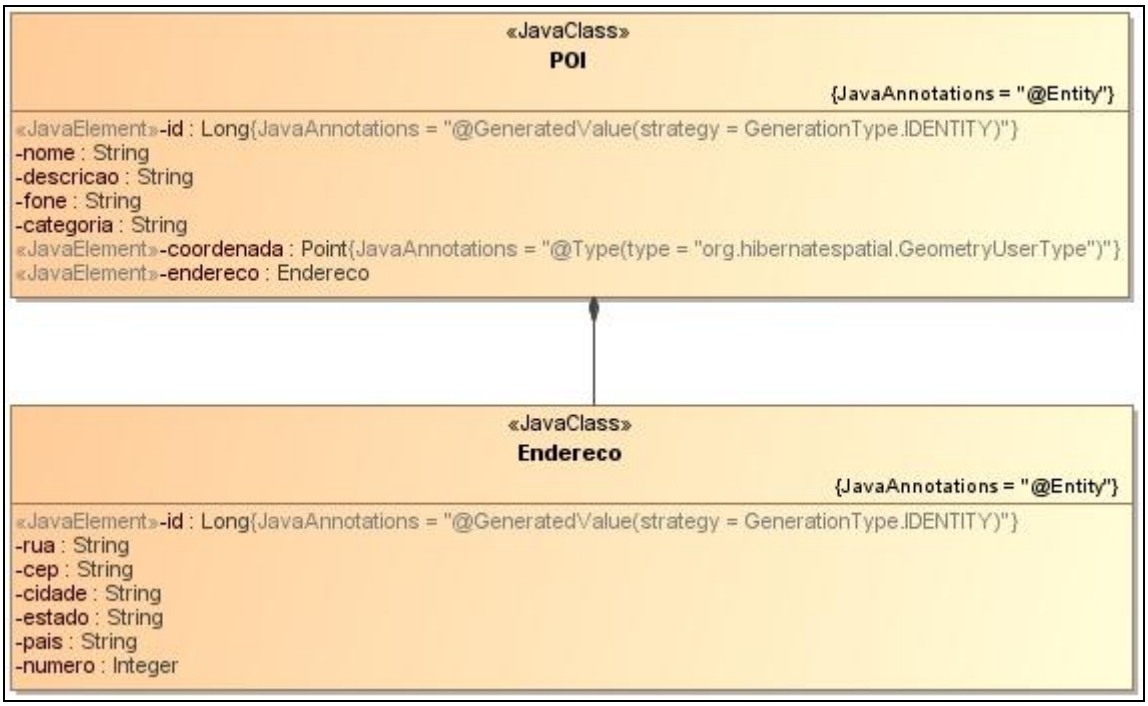


Figura 21. Entidade POI

Para a criação das tabelas *poi* e *endereço* na base de dados, utilizou-se um recurso do Hibernate que as gera automaticamente com base nas entidades por ele gerenciadas. Para isto, foi necessário o mapeamento das entidades da aplicação com anotações do Hibernate, sendo que o Hibernate Spatial ficou responsável pelo mapeamento dos atributos com tipos de dados geográficos. O uso das anotações também pode ser visto na Figura 21.

[15] O uso da classe *Endereco* poderia ser substituído pelo uso de Geocodificação Reversa, que não está incluso no escopo deste trabalho. No entanto, a forma utilizada foi suficiente para o desenvolvimento do protótipo.

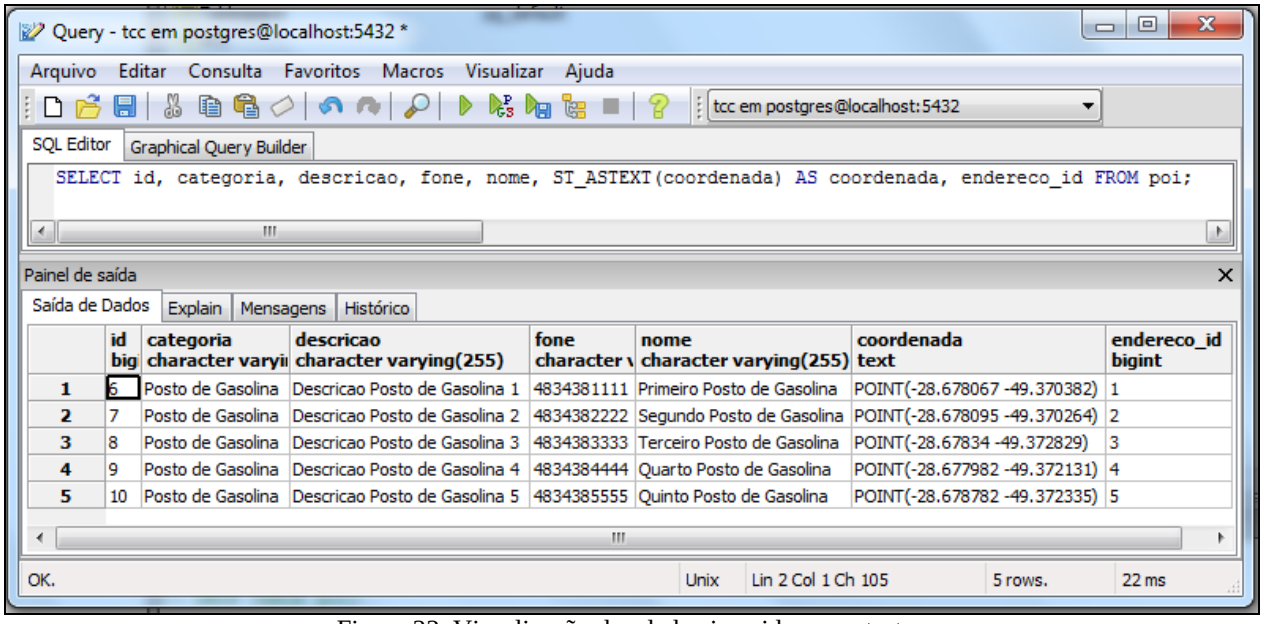
Apesar de o Hibernate permitir a criação automática das tabelas, houve dificuldade na adição da coluna geográfica *coordenada*, problema que foi contornado por meio de chamada à função *AddGeometryColumn* do PostGIS no PGAdmin. A sintaxe utilizada foi a seguinte:

a) *SELECT AddGeometryColumn('poi', 'coordenada', 4326, 'POINT', 2)*;

b) *SELECT AddGeometryColumn('posicao', 'coordenada', 4326, 'POINT', 2)*.

Passaram-se como parâmetro, respectivamente, nome da tabela, nome da coluna, SRID, tipo de dados da coluna e dimensão.

Criadas as entidades na aplicação e as tabelas na base de dados, inseriram-se POI fictícios para testes, os quais tiveram suas localizações geográficas definidas aleatoriamente, dentro dos limites da cidade de Criciúma, onde se desenvolveu o protótipo. A Figura 22 mostra uma consulta efetuada no PGAdmin na tabela *poi*, selecionando os POI inseridos para testes.



	id big	categoria character varyi	descricao character varying(255)	fone character v	nome character varying(255)	coordenada text	endereco_id bigint
1	6	Posto de Gasolina	Descricao Posto de Gasolina 1	4834381111	Primeiro Posto de Gasolina	POINT(-28.678067 -49.370382)	1
2	7	Posto de Gasolina	Descricao Posto de Gasolina 2	4834382222	Segundo Posto de Gasolina	POINT(-28.678095 -49.370264)	2
3	8	Posto de Gasolina	Descricao Posto de Gasolina 3	4834383333	Terceiro Posto de Gasolina	POINT(-28.67834 -49.372829)	3
4	9	Posto de Gasolina	Descricao Posto de Gasolina 4	4834384444	Quarto Posto de Gasolina	POINT(-28.677982 -49.372131)	4
5	10	Posto de Gasolina	Descricao Posto de Gasolina 5	4834385555	Quinto Posto de Gasolina	POINT(-28.678782 -49.372335)	5

Figura 22. Visualização dos dados inseridos para testes

Inseridos os dados a serem utilizados, criou-se o serviço propriamente dito. Sua implementação definiu-se diretamente na classe *DirectoryService*, responsável por atender requisições à Serviços de Diretório. Esta classe referencia à interface *DirectoryDAO*, que define um contrato para o Objeto de Acesso a Dados, do inglês Data Access Object (DAO), determinando os métodos de consulta ao banco de dados que devem ser implementados para atender as necessidades do serviço. Para melhor entendimento do serviço, pode-se visualizar a Figura 23.

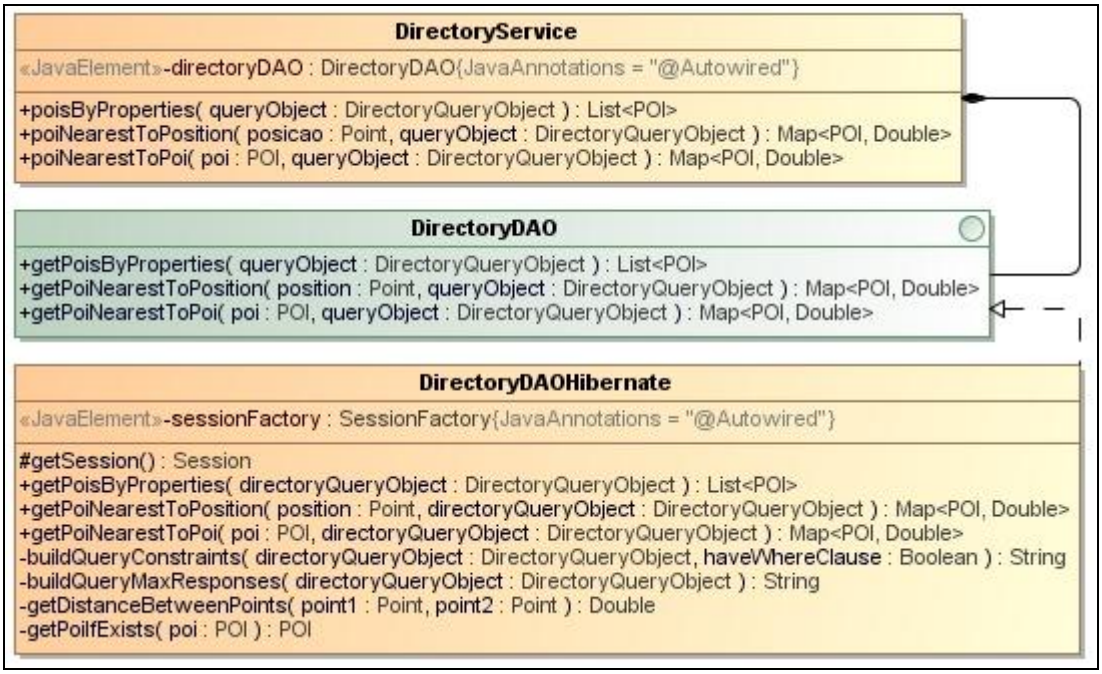


Figura 23. Classes do Serviço de Diretório

A implementação do DAO do protótipo se deu na classe *DirectoryDAOHibernate*, que utilizou funções do Hibernate e Hibernate Spatial para consultar à base de dados. Além dos métodos para implementação da interface *DirectoryDAO*, a classe *DirectoryDAOHibernate* possui métodos auxiliares, utilizados internamente para construir suas consultas.

A Figura 23 também mostra que o Serviço de Diretório desenvolvido provê acesso a três tipos de serviço; um Serviço de Diretório Específico (SDE) e dois Serviços de Diretório por Proximidade (SDP). O SDE *poisByProperties* fornece POI de acordo as propriedades especificadas na consulta. Os SDP *poiNearestToPoi* e *poiNearestToPosition* fornecem, respectivamente, uma consulta ao POI mais próximo do POI especificado e, ao POI mais próximo de determinada posição.

As funções de SDE recebem por parâmetro um objeto do tipo *DirectoryQueryObject*, criado especificamente para conter as propriedades do POI que serão utilizadas nas consultas. As funções de SDP recebem por parâmetro, além de um objeto do tipo *DirectoryQueryObject*, um objeto relativo à referência da proximidade, que pode ser um POI ou um ponto com as coordenadas geográficas de uma posição.

Ao receber uma requisição, o serviço consulta a base de dados por meio da implementação do DAO, que é injetada pelo Spring com o uso da anotação *@Autowired*. Obtidos os dados, é gerada a devida resposta para a requisição.

6.3 PROVISÃO DE INTEROPERABILIDADE NO ACESSO AO SERVIÇO DE DIRETÓRIO

Finalizadas as etapas da seção anterior, já se tem um Serviço de Diretório funcional. Entretanto, o serviço só pode ser acessado internamente, ou seja, pela própria aplicação. Uma das formas de disponibilizar acesso ao serviço em meio externo, provendo interoperabilidade, é por meio da implantação de WS. Todavia, apesar de sua especificação prover interoperabilidade no acesso aos serviços, seu uso não garante que as informações transmitidas no payload das mensagens SOAP sejam interpretadas por ambos os lados da comunicação. A adoção do padrão aberto OpenLS permite definir um protocolo na comunicação com o LBS, e sua utilização

simultânea com WS mostra-se uma solução eficiente para o problema de interoperabilidade apresentado.

6.3.1 Aplicação do OpenLS

O padrão OpenLS é definido e disponibilizado pela OGC por meio de XSD. Os esquemas XML do OpenLS, além de fornecerem estruturas para as requisições e respostas dos LBS, definem os tipos de dados que poderão ser utilizados. Exemplos são o elemento *PointOfInterestType*, que representa um POI e seus possíveis atributos; o elemento *PositionType,* que representa uma posição geográfica e; o elemento *AddressType*, que representa um endereço.

A aplicação do OpenLS no protótipo envolve a criação de classes Java representando os elementos contidos nos esquemas XML do padrão. O JAXB permite a geração automática dessas classes. Contudo, a complexidade do padrão implica na necessidade de uma compreensão prévia de sua estrutura.

Características relevantes a serem consideradas na definição do OpenLS são o uso de herança e polimorfismo em XML, por meio do uso de *substitutionGroup*. Outro ponto fundamental é a divisão do esquema XML em fragmentos (arquivos XSD) contendo partes específicas de sua estrutura.

O Serviço de Diretório do OpenLS é definido no esquema XML *DirectoryService.xsd*, que interliga-se por meio da cláusula *include*, respectivamente, com os esquemas *XLS.xsd*; *ADT.xsd*; *geometry.xsd* e; *UOM.xsd*. O conjunto destes, num todo, forma a

estrutura completa do Serviço de Diretório, fazendo com que estes arquivos devam estar presentes na execução do JAXB[16].

Os esquemas XML *DirectoryService.xsd*, *ADT.xsd* e *geometry.xsd* também utilizam a cláusula *import* para importar o *gml4xls.xsd*, que é uma definição da Linguagem de Marcação Geográfica, do inglês Geography Markup Language[17] (GML), adaptado ao XLS. Este arquivo é disponibilizado juntamente com os outros esquemas XML do OpenLS, necessitando também sua inclusão na compilação.

Por fim, é importante ressaltar que os esquemas XML *geometry.xsd* e *UOM.xsd* importam o *xlinks.xsd* por meio do link *http://schemas.opengis.net/xlink/1.0.0/xlinks.xsd*, fazendo com que seja necessário estar conectado à internet no momento da execução do Binding Compiler.

Após conhecida a estrutura do padrão, executou-se o Binding Compiler *xjc*, do JAXB, com a passagem da localização do *DirectoryService.xsd* por parâmetro. Entretanto, ocorreram conflitos de colisão de nomes entre alguns elementos dos esquemas XML. Os elementos de base *_POI*, *_Position*, *_RouteSummary*, *_NextSegment* e *_POIProperty* conflitaram respectivamente com os elementos *POI*, *Position*, *RouteSummary*, *NextSegment* e *POIPropert*. Isto se deve à forma na qual o JAXB nomeia as classes geradas. Por padrão, o compilador utilizou o nome dos elementos XML e ignorou o caracter _ (underline), fazendo com que fossem geradas classes com o mesmo nome, causando os conflitos.

O problema foi contornado com a criação de um arquivo de customização de binding, nomeado *custombinding.xjb*. O conteúdo do arquivo pode ser visualizado na Figura 24.

[16] Caso necessite-se implementar as interfaces OpenLS para os demais núcleos de LBS, utiliza-se o mesmo conceito, substituindo-se o *DirectoryService.xsd* pelo XSD correspondente ao serviço desejado.
[17] A GML também é mantida pela OGC, e define uma gramática para tipo de dados geográficos.

```
<jxb:bindings version="1.0" xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <jxb:bindings schemaLocation="ADT.xsd" node="//xs:element[@name='_POI']">
    <jxb:class name="POIBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="ADT.xsd"
    node="//xs:element[@name='_Position']">
    <jxb:class name="PositionBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="ADT.xsd"
    node="//xs:element[@name='_RouteSummary']">
    <jxb:class name="RouteSummaryBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="ADT.xsd"
    node="//xs:element[@name='_NextSegment']">
    <jxb:class name="NextSegmentBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="DirectoryService.xsd"
    node="//xs:element[@name='_POIProperty']">
    <jxb:class name="POIPropertyBaseElement" />
  </jxb:bindings>
</jxb:bindings>
```

Figura 24. Customização de binding para solucionar conflitos do JAXB

Nota-se que foram definidos explicitamente os nomes das classes a serem geradas para os elementos que apresentaram conflito. O comando final para geração das classes foi o seguinte: *xjc -extension -nv -b custombinding.xjb DirectoryService.xsd*.

Após executado o comando, o JAXB gerou um pacote chamado *gml*, contendo as classes referentes aos elementos do esquema XML *gml4xls.xsd*, frequentemente utilizados no OpenLS. O JAXB também gerou uma classe chamada *ObjectFactory*, cuja finalidade é a criação de objetos para as classes geradas no pacote. A Figura 25 mostra o conteúdo do pacote *gml*.

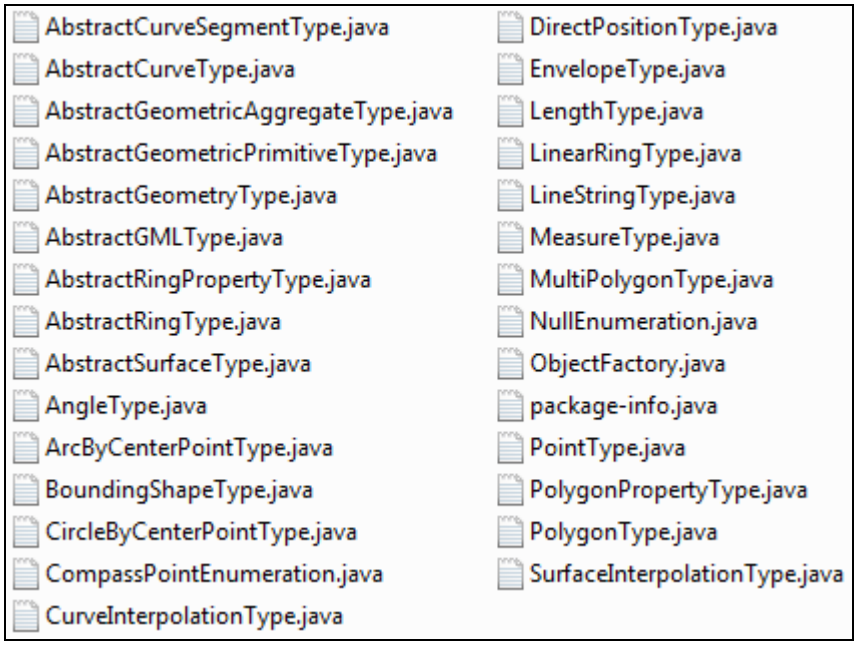


Figura 25. Arquivos gerados pelo JAXB no pacote *gml*

O JAXB também gerou um pacote chamado *xls*, contendo as classes referentes aos elementos do OpenLS utilizados nos Serviços de Diretório. Assim como no pacote *gml*, foi gerada uma fábrica para a criação de objetos, de nome *ObjectFactory*. O conteúdo do pacote *xls* pode ser visualizado na Figura 26.

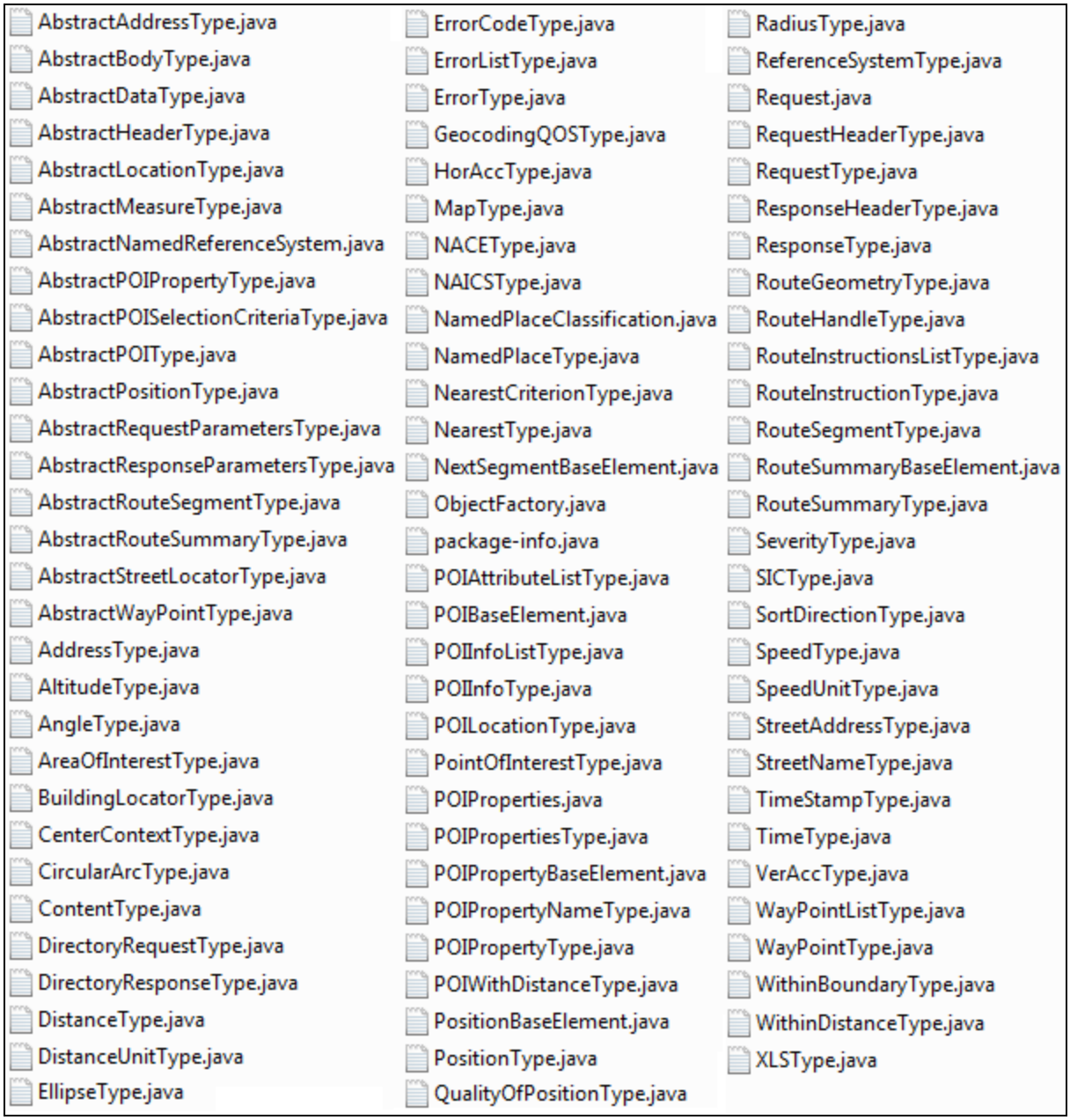


Figura 26. Arquivos gerados pelo JAXB no pacote *xls*

Criadas as classes Java que implementam o padrão OpenLS, tem-se a possibilidade de gerar objetos de acordo com sua especificação. No entanto, observou-se que a aplicação direta do padrão no serviço existente, impactaria em necessidade de modificações no seu código. Isto geraria, além de custos referentes ao tempo de planejamento das modificações, uma série de incompatibilidades com as implementações atuais de acesso ao serviço. Chegou-se a este impasse

propositalmente, visando simular uma situação real, onde uma suposta empresa fornece um Serviço de Diretório operável, e necessita adotar um padrão para interoperabilidade. Deve-se pensar em uma solução que mantenha a modelagem dos dados e serviços já existentes, de modo que não interfira nas funcionalidades operantes.

A solução criada foi o desenvolvimento de uma camada de código superior à da aplicação, voltada unicamente a fins de distribuição e interoperabilidade. Esta camada consiste em duas etapas: criação de uma interface de serviço específica para o OpenLS, que faz chamada ao serviço existente e; criação de uma classe para a conversão de objetos OpenLS em objetos da aplicação e vice-versa. A solução encontra-se ilustrada na Figura 27.

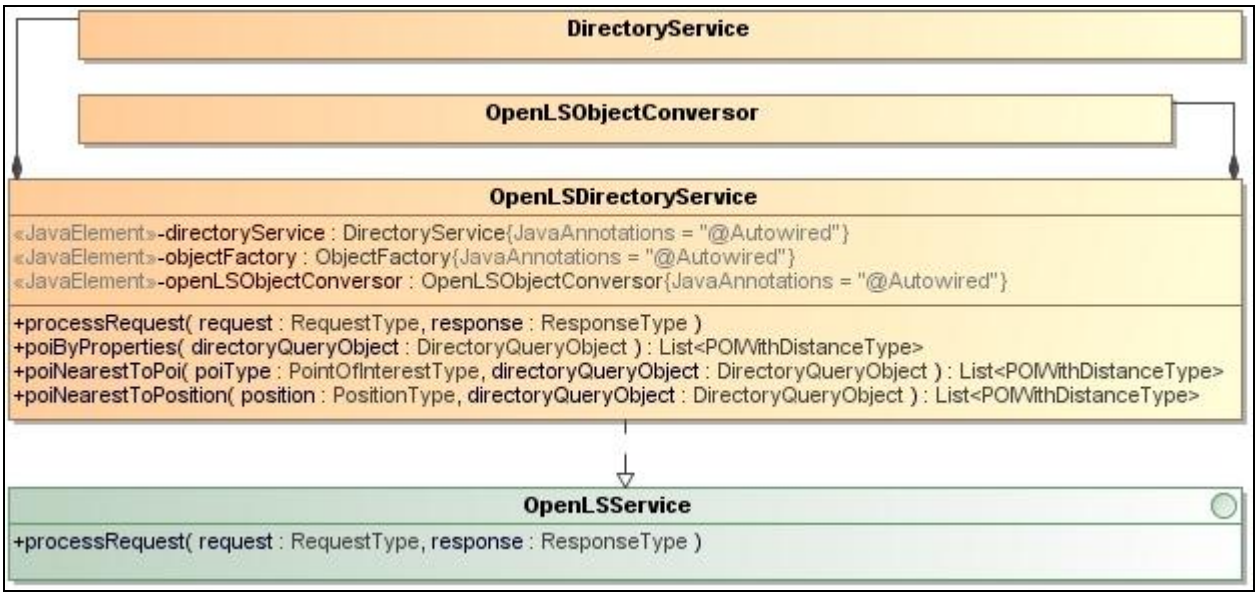


Figura 27. Camada de serviço OpenLS para acesso ao serviço da aplicação

A interface *OpenLSService* é quem define a forma como os serviços do OpenLS atenderão as requisições. Seu método *processRequest*, de uso comum a todos os núcleos do LBS, define que o serviço deve receber referência de um objeto *RequestType*, referente à requisição, e de um objeto *ResponseType*, referente à resposta. O serviço do OpenLS processa o objeto de

requisição e atualiza o objeto de resposta diretamente na fonte de sua chamada (quem chamará este método é o endpoint do WS, que será explicado posteriormente).

Para a implementação do Serviço de Diretório do OpenLS, criou-se a classe *OpenLSDirectoryService*, responsável pela implementação do método *processRequest* com lógicas de processamento específicas deste tipo de serviço.

6.3.2 Implantação do Web Services

A implantação do OpenLS no protótipo permite que as aplicações clientes possam se comunicar com o LBS de forma interoperável. Não menos importante, a implantação do WS garante a consistência e acessibilidade desta comunicação.

Conforme mencionado na seção sobre os recursos utilizados, o framework utilizado para a construção do WS, Apache CXF, utiliza essencialmente a API JAX-WS. Dentre as funcionalidades conhecidas da API, está a criação automática do contrato WSDL por meio do código Java. Apesar de este recurso possibilitar a criação do arquivo de definição do WS sem a necessidade de domínio da linguagem WSDL, optou-se por cria-lo manualmente, antes da implementação do código Java. Esta abordagem diminuiu o acoplamento entre o contrato e a implementação, aumentando a interoperabilidade e facilitando a manutenção do WS.

Outro ponto fundamental, que pesou na opção pela implementação manual da definição do WS, foi a necessidade de aprofundamento na linguagem, o que resultou em maior compreensão de WS; um dos objetivos deste trabalho. Na seção seguinte será apresentado e detalhado o contrato WSDL criado para o protótipo de LBS.

6.3.2.1 Definição do Web Services

O elemento raiz do documento WSDL criado é o *wsdl:definitions*, que pode ser visto na Figura 28. Este elemento contém atributos que definem os espaços de nomes utilizados pelos elementos do documento. Todos os demais elementos requeridos na definição de WS devem ser filhos deste.

```
<wsdl:definitions targetNamespace=http://ws.tcc.unesc.net/wsdl
  xmlns="http://www.opengis.net/xls"
  xmlns:wsdl=http://schemas.xmlsoap.org/wsdl /
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns=http://ws.tcc.unesc.net/wsdl
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:gml="http://www.opengis.net/gml">
```

Figura 28. Fragmento WSDL referente à definição dos espaços de nomes

O elemento *wsdl:types*, mostrado na Figura 29, define os tipos de dados utilizados na interação com o WS. Utilizou-se o esquema XML do Serviço de Diretório do OpenLS. Sua importação por meio do elemento *import* do XSD eliminou a necessidade de inserção do conteúdo do arquivo.

```
<wsdl:types>
  <xsd:schema elementFormDefault="qualified">
    <xsd:import namespace="http://www.opengis.net/xls"
      schemaLocation="../../opengis/schema/DirectoryService.xsd" />
  </xsd:schema>
</wsdl:types>
```

Figura 29. Fragmento WSDL referente à definição dos tipos de dados

Os elementos *wsdl:message* (Figura 30) definem as mensagens de interação com o WS. Definiu-se uma mensagem para requisição e outra para resposta, cada uma contendo um

elemento filho *wsdl:part*, que define o tipo dos dados utilizado. Para satisfazer as especificações do padrão OpenLS, definiu-se que ambas as mensagens devem possuir como elemento raiz o *XLS*.

```
<wsdl:message name="Request">
  <wsdl:part name="XLS" element="XLS" />
</wsdl:message>
<wsdl:message name="Response">
  <wsdl:part name="XLS" element="XLS" />
</wsdl:message>
```

Figura 30. Fragmento WSDL referente à definição das mensagens

Um elemento que requer atenção especial é o *wsdl:portType* (Figura 31). Ele define as operações que serão disponibilizadas pelo WS, bem como as mensagens envolvidas. O WS desenvolvido possui somente a operação *getService*, que recebe uma mensagem *Request*, e devolve uma mensagem *Response*.

```
<wsdl:portType name="ServicePort">
  <wsdl:operation name="getService">
    <wsdl:input message="tns:Request" />
    <wsdl:output message="tns:Response" />
  </wsdl:operation>
</wsdl:portType>
```

Figura 31. Fragmento WSDL referente à definição das operações

Destaca-se que a operação *getService*, definida no *portType*, corresponde ao método que deverá ser implementado no endpoint[18]. Observa-se que o método deverá receber por parâmetro um objeto do tipo *XLS* correspondente à requisição, e retornar um objeto do mesmo tipo correspondente à resposta do WS.

[18] Endpoints são classes ou entidades responsáveis por receber e processar as requisições do Web Services.

O elemento *wsdl:binding*, apresentado na Figura 32, especifica o protocolo e formato das mensagens das operações definidas no *portType*. Seu elemento filho *soap:binding* possui dois atributos: *style* e *transport*.

```
<wsdl:binding name="ServiceBinding" type="tns:ServicePort">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <wsdl:operation name="getService">
    <soap:operation soapAction="Service\#getService" />
    <wsdl:input>
      <soap:body namespace="http://ws.tcc.unesc.net/wsdl" use="literal" />
    </wsdl:input>
    <wsdl:output>
      <soap:body namespace="http://ws.tcc.unesc.net/wsdl" use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
```

Figura 32. Fragmento WSDL referente à definição de protocolo e formato de mensagens

O atributo *transport* define o protocolo SOAP utilizado. Neste caso optou-se pelo HTTP.

O atributo *style* define o tipo do payload que deverá ser utilizado nas mensagens SOAP, podendo assumir os valores *rpc* ou *document*. A opção *rpc* define que o payload conterá uma Chamada de Remota de Procedimentos. Neste caso, o conteúdo do elemento *body* da mensagem SOAP deverá ser envolvido em um elemento externo correspondente ao nome da operação chamada, e deverá seguir as regras definidas na especificação SOAP. A opção *document* foi adotada. Seu uso permite que o payload seja definido de acordo com o esquema XML contido no elemento *types*, neste caso, o esquema do Serviço de Diretório do OpenLS.

O elemento *wsdl:operation* configura as operações definidas nos elementos *portType*. Seu elemento filho *soap:body* define em seu atributo *use* a codificação do payload das mensagens de entrada e saída, podendo conter os valores *encoded* ou *literal*. Na opção *encoded*, os dados são codificados de acordo com a especificação SOAP, necessitando o acréscimo de um atributo

encodingStyle contendo uma URL indicando o local das regras de codificação. A opção adotada foi a *literal*. Nesta opção, os dados devem seguir literalmente a definição do esquema XML. Segundo Silva (2005), a opção pela combinação de *Document/Literal* permite um menor acoplamento entre as aplicações, e um maior grau de interoperabilidade.

Por fim, tem-se o elemento *wsdl:service* (Figura 33), que define a localização do serviço na rede.

```
<wsdl:service name="Service">
  <wsdl:port name="ServicePort" binding="tns:ServiceBinding">
    <soap:address location=" http://localhost:8080/TccLeandroNatal/LbsEndpoint " />
  </wsdl:port>
</wsdl:service>
```

Figura 33. Fragmento WSDL referente à definição da localização do serviço

6.3.2.2 Implementação do Endpoint

Requisições enviadas à WS são encaminhadas a entidades denominadas endpoints, onde são processadas e devolvidas aos clientes. Para atender às requisições definidas no contrato WSDL, os endpoints de um WS devem possuir métodos que implementem as operações nele especificadas.

Para o protótipo de LBS desenvolvido, criou-se a interface *LbsEndpoint*, que define um método *getService* correspondente à operação definida no portType *ServicePort* do WSDL. Também respeitando o WSDL, definiu-se na assinatura do método *getService* o recebimento de um parâmetro do tipo XLSType referente ao elemento XLS do OpenLS, e o retorno de um objeto também do tipo XLSType. Estas configurações devem-se às mensagens de entrada e saída definidas no WSDL.

A interface *LbsEndpoint* foi implementada pela classe *LbsEndpointImpl*, composta por um objeto do tipo *ObjectFactory* injetado pelo Spring. A classe implementa, além do método *getService*, métodos auxiliares utilizados para geração das listas de erros de acordo com a especificação OpenLS. Pode-se visualizar a modelagem do endpoint na Figura 34.

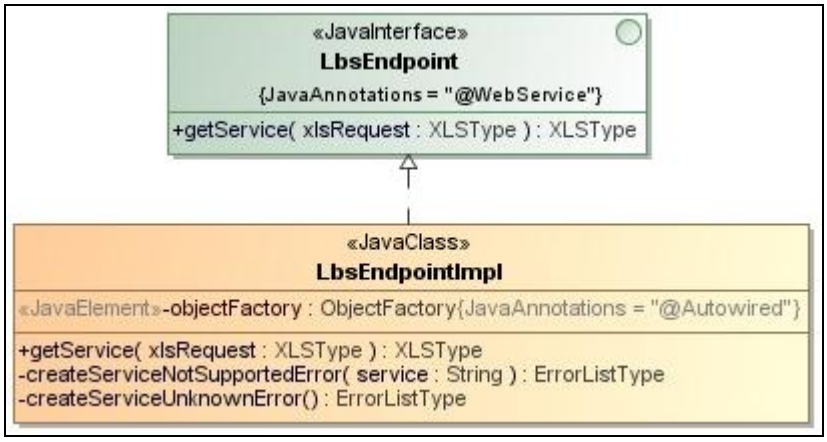


Figura 34. Endpoint do Web Services

O trecho de código apresentado na Figura 35 é adicionado ao arquivo de configuração do Apache CXF, *beans.xml*. O código informa ao framework qual a classe utilizada para implementar o endpoint. Também habilita a validação das mensagens de requisição e resposta de acordo com os esquemas XML que definem o uso do OpenLS.

```
<jaxws:endpoint id="lbsEndpoint" implementor="#lbsEndpointImpl"
  address="/LbsEndpoint">
  <jaxws:properties>
    <entry key="schema-validation-enabled" value="true" />
  </jaxws:properties>
</jaxws:endpoint>
```

Figura 35. Validação das mensagens e especificação do endpoint do Web Services

Antes de chegar ao endpoint, as requisições são validadas e, caso haja alguma inconsistência, o próprio WS retorna uma mensagem de erro ao cliente informando quais regras dos esquemas XML foram violadas. Caso as requisições estejam de acordo, o Apache CXF invoca automaticamente os procedimentos de Unmarshalling do JAXB, transformando a requisição XML em objeto Java do tipo *XLSType*, passando-o por parâmetro para o método *getService*.

Ao ser chamado, o método *getService* cria uma variável de referência do tipo *OpenLSService*. Esta referência inicialmente é nula, e somente referenciará um objeto após analisado à que núcleo destina-se a requisição. Cria-se então um objeto *XLSType* para a resposta e configura-se atributos como versão do XLS, ID da sessão, ID da requisição, dentre outros.

Depois de criada e pré-configurada a resposta, verifica-se à que núcleo destina-se a requisição. Isto se faz por meio de análise do atributo *MethodName* do objeto *RequestType*, que compõe o objeto *XLSType* da requisição. Caso contenha o valor *DirectoryRequest*, a variável *OpenLSService* criada anteriormente solicita ao Spring e referencia por polimorfismo um objeto do tipo *OpenLSDirectoryService*. Isto indica que a implementação do Serviço de Diretório do OpenLS é quem deverá processar a requisição. Caso a requisição se destine aos demais núcleos de LBS existentes, configura-se no cabeçalho da resposta uma lista de erros do OpenLS, informando que o serviço não é suportado. Neste caso utiliza-se o código de erro *NOT_SUPPORTED*, definido no padrão por meio da enumeração *ErrorCodeType*. Caso o nome do método chamado seja inválido, ou seja, não corresponda a nenhum dos núcleos conhecidos de LBS, utiliza-se o código de erro *UNKNOWN*, informando que o serviço é desconhecido.

Após analisado o tipo de requisição, caso a requisição pertença a um dos núcleos suportados pela aplicação, é chamado o método *processRequest* do serviço OpenLS. O método se

encarregará de processar a requisição, gerar a resposta e incluí-la no corpo da resposta do OpenLS, que é devolvida ao cliente por meio de mensagem SOAP de resposta.

Apesar de este trabalho envolver somente o núcleo de Serviços de Diretório, a estrutura desenvolvida provê total extensibilidade para que os demais núcleos sejam implementados. Para melhor compreensão do funcionamento do protótipo de LBS desenvolvido, foi criado um diagrama de atividades contendo as etapas realizadas desde a requisição ao LBS pela Aplicação Cliente, até o envio da resposta processada. O diagrama pode ser observado na Figura 36.

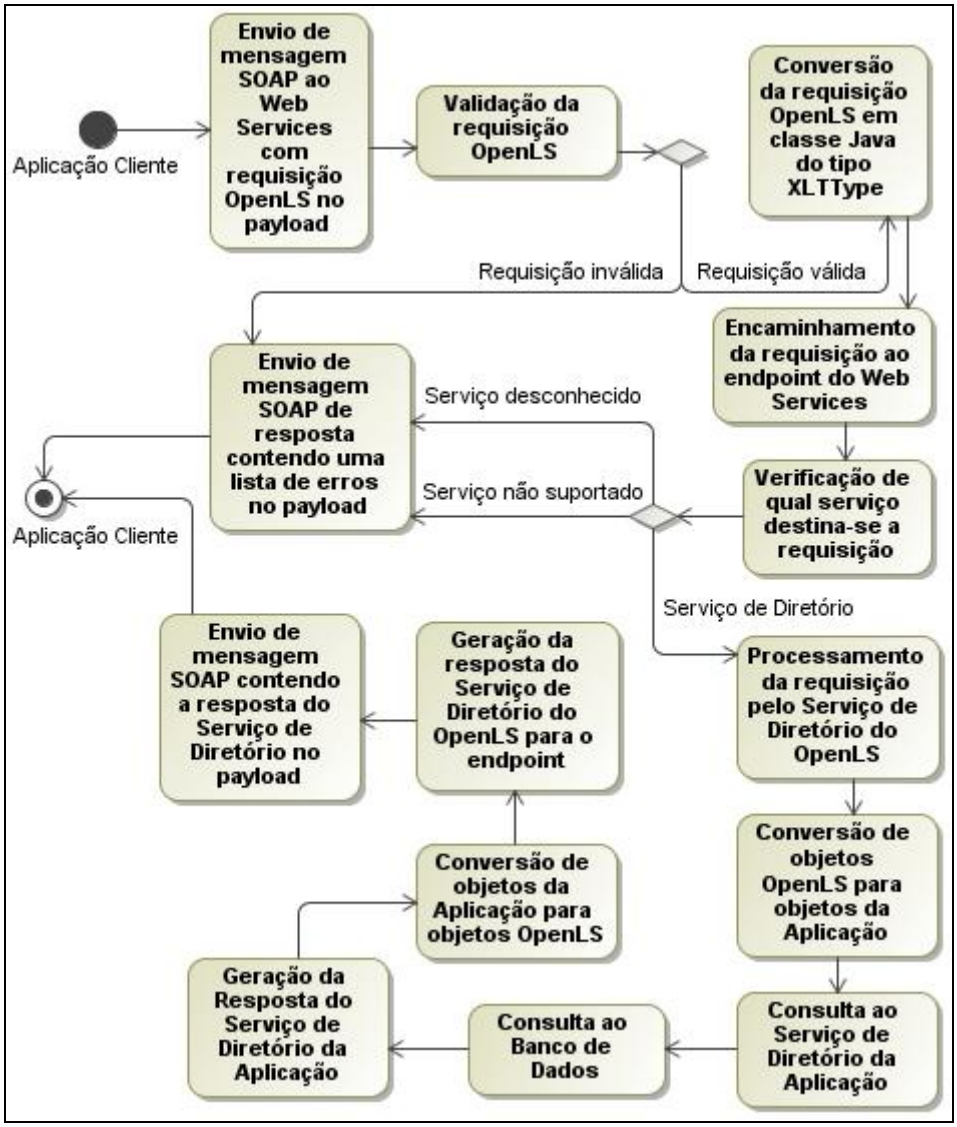


Figura 36. Diagrama de atividades com as etapas realizadas na consulta ao protótipo de LBS

6.4 EXECUÇÃO DOS TESTES DE ACESSO AO LBS E RESULTADOS

A execução de testes em aplicações acessadas por meio de WS difere-se das aplicações convencionais, uma vez que seus serviços são requisitados por meio de mensagens formatadas de acordo com suas especificações. Para isto, devem-se utilizar aplicações clientes aptas a consumir este tipo de serviço.

Uma prática comum utilizada para testar WS no decorrer de seu desenvolvimento, é o uso de ferramentas disponibilizadas gratuitamente, desenvolvidas com intuito de funcionar como aplicações clientes para WS. Esta prática foi adotada neste trabalho por meio do uso da ferramenta SoapUI. A ferramenta permitiu que o serviço desenvolvido fosse acessado e testado facilmente. Seu uso como aplicação cliente do LBS desenvolvido se mostrou satisfatório para realização dos testes.

Inicialmente criou-se um projeto no SoapUI e indicou-se a URL de publicação do documento WSDL. A ferramenta importou automaticamente as configurações do WS, adicionando, além da operação disponível *getService*, um modelo para sua requisição. O estado inicial da ferramenta após a criação do projeto pode ser visto na Figura 37.

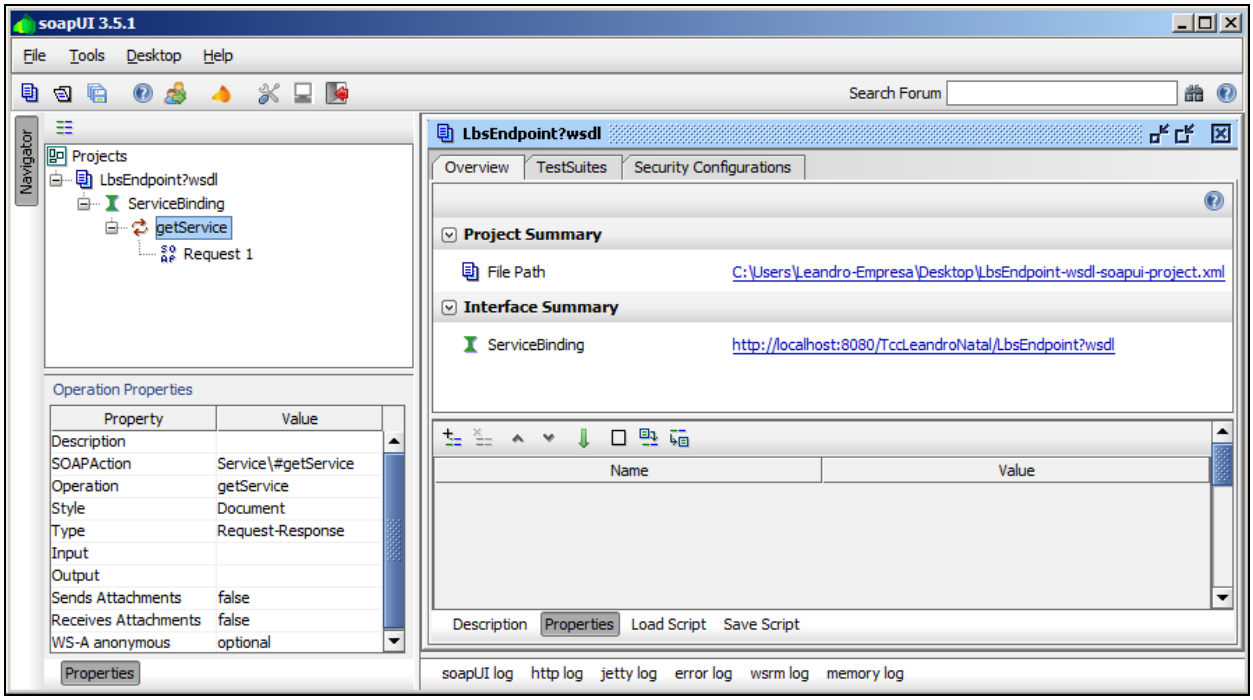


Figura 37. Importação do documento WSDL no SoapUI, gerando um modelo para a requisição

A Figura 38 mostra a ferramenta já com um projeto de testes configurado, contendo algumas requisições para o LBS desenvolvido. Todas as requisições foram formatadas de acordo com a especificação SOAP, e foram utilizadas no payload documentos contendo requisições de acordo com o padrão OpenLS.

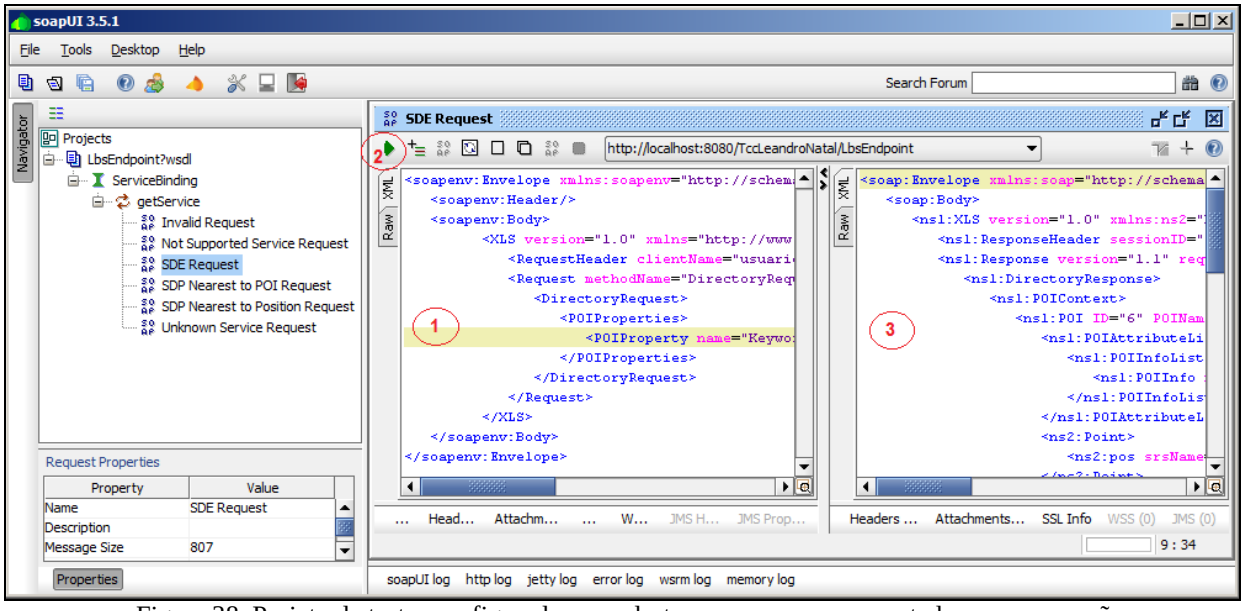


Figura 38. Projeto de testes configurado, com destaque nos passos executados para execução

A Figura 38 também destaca os passos utilizados para acessar o WS por meio da ferramenta SoapUI. Primeiramente, adiciona-se a requisição no local indicado pelo número *1*. Após definida a mensagem de requisição, pressiona-se o botão indicado pelo número *2*. A mensagem é enviada ao endpoint do WS, onde é processada e retornada. A resposta pode ser visualizada no local indicado pelo número *3*.

6.4.1 Análise das Requisições e Respostas utilizadas nos testes

Para certificar o bom funcionamento do protótipo, foram criados os diversos tipos requisições possíveis ao LBS: requisições para os SDE e SDP disponíveis; requisições para serviços não suportados e serviços inválidos e; requisições inválidas (com problemas na estruturação). Esta seção apresenta e detalha os resultados obtidos para as requisições citadas, enviadas ao LBS por meio da ferramenta SoapUI.

A primeira mensagem SOAP enviada ao LBS para teste, mostrada na Figura 39, contém uma requisição a SDE. Seu elemento *POIProperties* restringe a busca à POI cujas categorias sejam *Posto de Gasolina*, e define o número máximo de respostas como 2.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope
  xmlns:wsdl="http://ws.tcc.unesc.net/wsdl">
  <soapenv:Header />
  <soapenv:Body>
    <XLS version="1.0" xmlns=http://www.opengis.net/xls
      xmlns:gml="http://www.opengis.net/gml">
      <RequestHeader sessionID="1234" />
      <Request methodName="DirectoryRequest" version="1.1"
        requestID="123" maximumResponses="2">
        <DirectoryRequest>
          <POIProperties>
            <POIProperty name="Keyword" value="Posto de Gasolina" />
          </POIProperties>
        </DirectoryRequest>
      </Request>
    </XLS>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 39. Mensagem SOAP com requisição a SDE

A Figura 40 mostra a resposta gerada pelo serviço para a requisição da Figura 39. Como a requisição limitou o número de respostas em 2, o serviço retornou dois POI, definidos dentro de elementos *POIContext*.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:XLS version="1.0" xmlns:ns2=http://www.opengis.net/gml
      xmlns:ns1="http://www.opengis.net/xls">
      <ns1:ResponseHeader sessionID="1234" />
      <ns1:Response version="1.1" requestID="123"
        numberOfResponses="2">
        <ns1:DirectoryResponse>
          <ns1:POIContext>
            <ns1:POI ID="6" POIName="Primeiro Posto de Gasolina"
              phoneNumber="4834381111" description="Descrição Posto de Gasolina 1">
              <ns1:POIAttributeList>
                <ns1:POIInfoList>
                  <ns1:POIInfo name="categoria" value="Posto de Gasolina" />
                </ns1:POIInfoList>
              </ns1:POIAttributeList>
              <ns2:Point>
                <ns2:pos srsName="4326">-28.678067 -49.370382</ns2:pos>
              </ns2:Point>
              <ns1:Address countryCode="BR">
                <ns1:StreetAddress locator="500">
                  <ns1:Street>Av. dos Imigrantes</ns1:Street>
                </ns1:StreetAddress>
                <ns1:Place type="CountrySubdivision">Santa Catarina
                        </ns1:Place>
                <ns1:Place type="Municipality">Criciúma</ns1:Place>
              </ns1:Address>
            </ns1:POI>
          </ns1:POIContext>
          <ns1:POIContext>
            <ns1:POI ID="9" POIName="Quarto Posto de Gasolina"
              phoneNumber="4834384444" description="Descricao Posto de Gasolina 4">
              <ns1:POIAttributeList>
                <ns1:POIInfoList>
                  <ns1:POIInfo name="categoria" value="Posto de Gasolina" />
                </ns1:POIInfoList>
              </ns1:POIAttributeList>
              <ns2:Point>
                <ns2:pos srsName="4326">-28.677982 -49.372131</ns2:pos>
              </ns2:Point>
              <ns1:Address countryCode="BR">
                <ns1:StreetAddress locator="800">
                  <ns1:Street>Av. Centenário</ns1:Street>
                </ns1:StreetAddress>
                <ns1:Place type="CountrySubdivision">Santa Catarina
                        </ns1:Place>
                <ns1:Place type="Municipality">Criciúma</ns1:Place>
              </ns1:Address>
            </ns1:POI>
          </ns1:POIContext>
        </ns1:DirectoryResponse>
      </ns1:Response>
    </ns1:XLS>
  </soap:Body>
</soap:Envelope>
```

Figura 40. Resposta gerada à requisição a SDE

A segunda mensagem SOAP enviada ao LBS, mostrada na Figura 41, contém uma requisição a SDP. O que a qualifica como consulta a SDP é presença do elemento *POILocation*, que define um filtro espacial.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope
  xmlns:wsdl="http://ws.tcc.unesc.net/wsdl">
  <soapenv:Header />
  <soapenv:Body>
    <XLS version="1.0" xmlns="http://www.opengis.net/xls"
      xmlns:gml="http://www.opengis.net/gml">
      <RequestHeader sessionID="1313" />
      <Request methodName="DirectoryRequest" version="1.1"
        requestID="123">
        <DirectoryRequest>
          <POILocation>
            <Nearest>
              <Position>
                <gml:Point>
                  <gml:pos srsName="4326">-28.67934 -49.372899
                  </gml:pos>
                </gml:Point>
              </Position>
            </Nearest>
          </POILocation>
          <POIProperties>
            <POIProperty name="Keyword" value="Posto de Gasolina" />
          </POIProperties>
        </DirectoryRequest>
      </Request>
    </XLS>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 41. Mensagem SOAP com requisição a SDP (POI mais próximo de uma localização)

O elemento *POILocation* contém um elemento filho *Nearest*, que por sua vez contém um elemento filho *Position*, onde é definido o SRID e as coordenadas geográficas de uma posição. Este filtro representa uma busca aos POI mais próximos de determinada posição, no caso, a posição definida no elemento *Position*. Esta é a requisição utilizada para localizar os POI mais próximos de veículos de transporte. Para simulação dos rastreadores veiculares, foi utilizado DM dotado de tecnologias de posicionamento (GPS e Cell-id) e comunicação sem fio (GPRS, Wi-fi). A localização geográfica adicionada na requisição foi a do DM.

A resposta para a requisição da Figura 41 é mostrada na Figura 42.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:XLS version="1.0" xmlns:ns2="http://www.opengis.net/gml"
      xmlns:ns1="http://www.opengis.net/xls">
      <ns1:ResponseHeader sessionID="1313" />
      <ns1:Response version="1.1" requestID="123"
        numberOfResponses="1">
        <ns1:DirectoryResponse>
          <ns1:POIContext>
            <ns1:POI ID="10" POIName="Quinto Posto de Gasolina"
              phoneNumber="4834385555" description="Posto de Gasolina 5">
              <ns1:POIAttributeList>
                <ns1:POIInfoList>
                  <ns1:POIInfo name="categoria" value="Posto de Gasolina" />
                </ns1:POIInfoList>
              </ns1:POIAttributeList>
              <ns2:Point>
                <ns2:pos srsName="4326">-28.678782 -49.372335</ns2:pos>
              </ns2:Point>
              <ns1:Address countryCode="BR">
                <ns1:StreetAddress locator="900">
                  <ns1:Street>Rua Alvaro Catão</ns1:Street>
                </ns1:StreetAddress>
                <ns1:Place type="CountrySubdivision">Santa Catarina
                        </ns1:Place>
                <ns1:Place type="Municipality">Criciúma</ns1:Place>
              </ns1:Address>
            </ns1:POI>
            <ns1:Distance uom="M" value="74.6" />
          </ns1:POIContext>
        </ns1:DirectoryResponse>
      </ns1:Response>
    </ns1:XLS>
  </soap:Body>
</soap:Envelope>
```

Figura 42. Resposta gerada à requisição a SDP, mostrando o POI mais próximo da localização

A resposta contém o posto de gasolina mais próximo da posição informada no filtro da requisição. O elemento *Distance* informa que a distância da posição até o POI é de 74,6 metros. Nota-se que em um filtro de requisição “mais próximo” deve haver somente uma resposta.

A terceira mensagem SOAP enviada ao LBS, mostrada na Figura 43, contém uma requisição a SDP, desta vez utilizando como referencia de proximidade um POI, que foi obtido na resposta da segunda requisição. Conforme se pode observar, utilizou-se diretamente o atributo *ID* do *POI*, que é o único conteúdo obrigatório segundo a especificação OpenLS, e é suficiente para este tipo de consulta.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope
  xmlns:wsdl="http://ws.tcc.unesc.net/wsdl">
  <soapenv:Header />
  <soapenv:Body>
    <XLS version="1.0" xmlns=http://www.opengis.net/xls
      xmlns:gml="http://www.opengis.net/gml">
      <RequestHeader sessionID="1313" />
      <Request methodName="DirectoryRequest" version="1.1"
        requestID="123">
        <DirectoryRequest>
          <POILocation>
            <Nearest>
              <POI ID="10">
              </POI>
            </Nearest>
          </POILocation>
          <POIProperties>
            <POIProperty name="Keyword" value="Posto de Gasolina" />
          </POIProperties>
        </DirectoryRequest>
      </Request>
    </XLS>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 43. Mensagem SOAP com requisição a SDP (POI mais próximo de um POI)

A Figura 44 mostra a resposta da requisição da Figura 43.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:XLS version="1.0" xmlns:ns2="http://www.opengis.net/gml"
      xmlns:ns1="http://www.opengis.net/xls">
      <ns1:ResponseHeader sessionID="1313" />
      <ns1:Response version="1.1" requestID="123"
        numberOfResponses="1">
        <ns1:DirectoryResponse>
          <ns1:POIContext>
            <ns1:POI ID="8" POIName="Terceiro Posto de Gasolina"
              phoneNumber="4834383333" description="Descricao Posto de Gasolina 3">
              <ns1:POIAttributeList>
                <ns1:POIInfoList>
                  <ns1:POIInfo name="categoria" value="Posto de Gasolina" />
                </ns1:POIInfoList>
              </ns1:POIAttributeList>
              <ns2:Point>
                <ns2:pos srsName="4326">-28.67834 -49.372829</ns2:pos>
              </ns2:Point>
              <ns1:Address countryCode="BR">
                <ns1:StreetAddress locator="700">
                  <ns1:Street>Rua João Ronchi</ns1:Street>
                </ns1:StreetAddress>
                <ns1:Place type="CountrySubdivision">Santa Catarina
                    </ns1:Place>
                <ns1:Place type="Municipality">Criciúma</ns1:Place>
              </ns1:Address>
            </ns1:POI>
            <ns1:Distance uom="M" value="63.57" />
          </ns1:POIContext>
        </ns1:DirectoryResponse>
      </ns1:Response>
    </ns1:XLS>
  </soap:Body>
</soap:Envelope>
```

Figura 44. Resposta gerada à requisição a SDP, mostrando o POI mais próximo do POI

A resposta informou que o posto de gasolina mais próximo do POI com *ID* 10 (Quinto Posto de gasolina, conforme resposta da segunda requisição) é o Terceiro Posto de Gasolina, que fica a uma distância de 63,57 metros.

A quarta mensagem SOAP enviada ao LBS, mostrada na Figura 45, contém uma requisição ao Serviço de Geocodificação, que não é suportado pelo protótipo desenvolvido. Sua resposta, mostrada na Figura 46, informa por meio de uma lista de erros definida no OpenLS que o serviço não é suportado.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope
  xmlns:wsdl="http://ws.tcc.unesc.net/wsdl">
  <soapenv:Header />
  <soapenv:Body>
    <XLS version="1.0" xmlns=http://www.opengis.net/xls
      xmlns:gml="http://www.opengis.net/gml">
      <RequestHeader sessionID="1313" />
      <Request methodName="GeocodeRequest" version="1.1"
        requestID="123">
        <!-- _RequestParameters do servido de geocodificação-->
      </Request>
    </XLS>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 45. Mensagem SOAP com requisição a um serviço não suportado pelo protótipo

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:XLS version="1.0" xmlns:ns2=http://www.opengis.net/gml
      xmlns:ns1="http://www.opengis.net/xls">
      <ns1:ResponseHeader sessionID="1313">
        <ns1:ErrorList>
          <ns1:Error errorCode="NotSupported"
            message="Serviço GeocodeRequest não suportado." />
        </ns1:ErrorList>
      </ns1:ResponseHeader>
      <ns1:Response version="1.1" requestID="123" />
    </ns1:XLS>
  </soap:Body>
</soap:Envelope>
```

Figura 46. Resposta gerada à requisição ao serviço não suportado

A Figura 47 mostra uma requisição a um serviço desconhecido. Sua resposta é mostrada na Figura 48.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope
  xmlns:wsdl="http://ws.tcc.unesc.net/wsdl">
  <soapenv:Header />
  <soapenv:Body>
    <XLS version="1.0" xmlns=http://www.opengis.net/xls
      xmlns:gml="http://www.opengis.net/gml">
      <RequestHeader sessionID="1313" />
      <Request methodName="AnyRequest" version="1.1" requestID="123">
        <!-- _RequestParameters do servido inválido-->
      </Request>
    </XLS>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 47. Mensagem SOAP com requisição a um serviço desconhecido em LBS

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <ns1:XLS version="1.0" xmlns:ns2=http://www.opengis.net/gml
      xmlns:ns1="http://www.opengis.net/xls">
      <ns1:ResponseHeader sessionID="1313">
        <ns1:ErrorList>
          <ns1:Error errorCode="Unknown" message="Serviço desconhecido." />
        </ns1:ErrorList>
      </ns1:ResponseHeader>
      <ns1:Response version="1.1" requestID="123" />
    </ns1:XLS>
  </soap:Body>
</soap:Envelope>
```

Figura 48. Resposta gerada à requisição ao serviço desconhecido

A quinta e última mensagem SOAP enviada ao LBS, mostrada na Figura 49, contém uma requisição inválida, ou seja, que viola as regras dos esquemas XML do OpenLS. Nesta requisição violou-se propositalmente o elemento *Request*, removendo-se o atributo obrigatório *methodName*.

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope
  xmlns:wsdl="http://ws.tcc.unesc.net/wsdl">
  <soapenv:Header />
  <soapenv:Body>
    <XLS version="1.0" xmlns=http://www.opengis.net/xls
      xmlns:gml="http://www.opengis.net/gml">
      <RequestHeader sessionID="1313" />
      <Request version="1.1" requestID="123">
        <!-- _RequestParameters-->
      </Request>
    </XLS>
  </soapenv:Body>
</soapenv:Envelope>
```

Figura 49. Mensagem SOAP com requisição inválida segundo a especificação OpenLS

A resposta da Figura 49 é mostrada na Figura 50. O próprio validador do Apache CXF detectou o erro, informando-o ao cliente no corpo da mensagem SOAP de resposta.

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>Unmarshalling Error: cvc-complex-type.4: Attribute
        'methodName' must appear on element 'Request'.</faultstring>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Figura 50. Resposta gerada à requisição inválida

Conforme se pode observar, para todas as requisições possíveis ao LBS, foram geradas as devidas respostas. A formatação das mensagens SOAP do LBS com o tipo Document/Literal garantiu que o documento XML no payload das mensagens de requisição e resposta respeitassem integramente a especificação OpenLS, contendo como raiz o elemento *XLS*, juntamente com os demais elementos filhos definidos no padrão. Mesmo requisições inválidas, cujas limitações na estrutura não podem ser interpretadas pelo LBS, puderam ser detectadas e respondidas a Aplicação Cliente, informando claramente os erros cometidos. Isto mostra que os resultados obtidos com a arquitetura utilizada são satisfatórios, e seu uso viável em LBS.

7 CONCLUSÃO

Este trabalho apresentou um estudo sobre um importante tipo de aplicação que vêm crescendo constantemente com a evolução e acessibilidade de tecnologias tais como redes de comunicação sem fio, Dispositivos Móveis e tecnologias de posicionamento, os Serviços Baseados em Localização (LBS). LBS podem ser aplicados e gerar informações úteis a diversas áreas, das quais se pode citar o Transporte Rodoviário de Cargas, onde é possível obter informações como localização de veículos, seu posicionamento em relação a Pontos de Interesse (POI), melhores rotas entre dois ou mais pontos, dentre outras. LBS são categorizados em diferentes núcleos, de acordo com o tipo de funcionalidade provida. Dentre os núcleos possíveis de LBS está o Serviço de Diretório, que permite consultar POI específicos ou mais próximos de determinada posição.

LBS requerem cada vez mais interoperabilidade, uma vez que podem estar disponíveis em diferentes dispositivos, possuir interface com sistemas de banco de dados legados e englobar diversas tecnologias de infraestrutura de rede.

O uso de padrões abertos é uma forma de garantir interoperabilidade na comunicação entre os serviços e as aplicações que o acessam. Os padrões abertos voltados à interoperabilidade em LBS são o Mobile Location Protocol (MLP), do Open Mobile Alliance (OMA), e o Open Location Services (OpenLS), do Open Geospatial Consortium (OGC). Estes padrões possuem aplicações distintas, mas complementares. Dentre eles, o que contempla as soluções para acesso aos principais núcleos de LBS, fornecendo interfaces padronizadas, é o OpenLS.

A tecnologia Web Services (WS) fornece uma arquitetura padronizada para interoperabilidade no acesso a serviços, possibilitando o uso de padrões baseados em XML na

comunicação. O padrão aberto OpenLS é baseado em XML, e sua utilização em WS fornece uma arquitetura totalmente padronizada no acesso a LBS.

Foi modelado e desenvolvido um protótipo de LBS, com abordagem para um Serviço de Diretório, com ênfase na obtenção da posição de Veículos de Transporte Rodoviários de Cargas, provido de interoperabilidade por meio da combinação de WS com as interfaces do OpenLS. Com o trabalho constatou-se que o desenvolvimento deste tipo de aplicação requer um estudo de tecnologias complementares tais como: XML, utilizada frequentemente em WS e no OpenLS e; Bancos de Dados Geográficos, utilizados para armazenar e recuperar informações georeferenciadas. A seleção dos recursos de software também foi importante, pois implicou diretamente na redução do tempo e esforços necessários para o desenvolvimento. Mesmo com a seleção adequada dos recursos, foram encontradas problemas no decorrer do desenvolvimento, os quais puderam ser contornados com a elaboração de algumas soluções.

Testes executados no protótipo mostraram que o sistema desenvolvido é totalmente viável para incorporação em um LBS. A formatação das mensagens SOAP do LBS com o tipo Document/Literal garantiu que as requisições respostas respeitassem integramente a especificação OpenLS. Foram executados testes para todos os tipos de requisições possíveis ao protótipo, e os resultados obtidos foram satisfatórios.

Em relação a trabalhos futuros, sugere-se a implementação de um framework que contemple todos os núcleos da especificação OpenLS, que seja flexível e possa ser usado para a provisão de interoperabilidade em LBS existentes.

REFERÊNCIAS

ADOBE SYSTEMS INCORPORATED. **Execução de exemplos do ActionScript 3.0 em dispositivos móveis.** Disponível em: <http://help.adobe.com/pt_BR/as3/dev/WS701a38e0c62007d33298b7371246d7d49ec-8000.html>. Acesso em: 05 abr. 2010.

ANDERSON, Richard. **Professional XML.** Rio de Janeiro: Ciência Moderna, 2001. 1266 p.

ANTCL - Associação Nacional do Transporte de Cargas e Logística. **Perfil do Transporte Rodoviário de Cargas: Peso do Transporte na Economia Brasileira.** 2001. Disponível em <http://www.ntc.org.br/perfil.htm>. Acesso em: 29 set. 2009.

ARSANJANI, A.; HAILPERN, B.; MARTIN, J; TARR, P. L. Web services: promises and compromises. **Acm Queue**, New York, v. 1, n. 1, p.48-58, mar. 2003.

CÂMARA, G.; CASANOVA, M. A.; JUNIOR, C. D.; VINHAS, L; QUEIROZ, G. **Bancos de Dados Geográficos.** Curitiba: Mundogeo, 2005. 490 p.

CAMPASSI, R. **ZATIX Compra Controle da Controlsat e Busca outros Ativos no Setor.** Jornal Valor Econômico. São Paulo. 24 ago. 2009.`

CARARO, Angela Cristina. **CORREÇÕES RELATIVÍSTICAS SOBRE AS MEDIDAS DE TEMPO GPS.** 2006. 100 f. Dissertação (Mestrado) - Universidade Federal do Paraná, Curitiba, 2006. Disponível em: <http://dspace.c3sl.ufpr.br:8080/dspace/handle/1884/5785>. Acesso em: 20 fev. 2010.

CÉZARO, Rangel Leandro. Geoprocessamento aplicado em Geografia. **Revista Geonotas**, Maringá, v. 7, n. 1, p.1-2, jan. 2003. Disponível em: <http://www.dge.uem.br/geonotas/vol7-1/rangel.shtml>. Acesso em: 21 jun. 2010.

CHEDE, Cezar Taurion. Padrões abertos, interoperabilidade e interesse público. **PoliTics**, Rio de Janeiro, n. , p.29-35, nov. 2008. Disponível em: <http://www.politics.org.br/edicao_02/downloads/poliTICS_n2_CezarTaurion.pdf>. Acesso em: 11 mar. 2010.

CNT. **Boletim Estatístico.** Confederação Nacional do Transporte. Brasília. Março. 2009. Disponível em: <http://www.cnt.org.br/portal/webcnt/>. Acesso em: 25 set. 2009.

CNT; COPPEAD/UFRJ. **Transporte de Passageiros**. Confederação Nacional do Transporte e Centro de Estudos em Logística/COPPEAD/UFRJ Junho. 2002. Disponível em: <http://www.cnt.org.br/portal/webcnt/>. Acesso em: 25 set. 2009.

DELPHI GROUP. **The Value of Standards.** Disponível em: <http://www.delphigroup.com/whitepapers/pdf/20030728-standards.pdf>. Acesso em: 11 mar. 2010.

DENATRAN. **Deliberação no. 83.** Departamento Nacional de Trânsito. Julho, 2009. Disponível em: <http://www.denatran.gov.br/download/Deliberacoes/DELIBERACAO_CONTRAN_83_09.pdf> . 2009. Acesso em: 02 out. 2009.

ESRI. **What are Location Services? – From a GIS Perspective.** Disponível em: <http://lbs360.directionsmag.com/LBSArticles/ESRI.What are LS Whitepaper.pdf>. Acesso em: 03 mar. 2010.

FAGGION, Nadège; TROCHERIS, André. Location-Based Services Strengthen the Strategic Position of Mobile Operators. **Alcatel Telecommunications Review**, Paris, p. 1-9. jun. 2003.

FREEDMAN, Alan. **Computer Desktop Encyclopedia.** 2000. Disponível em: <http://www.computerlanguage.com/>. Acesso em: 26 mai. 2010.

GUEDES, Edna Maria Pereira. **ESTUDO DE TÉCNICA HÍBRIDA DE LOCALIZAÇÃO DE ESTAÇÕES MÓVEIS BASEADA EM TDoA E AoA.** 2003. 119 f. Dissertação (Mestrado) - Instituto Militar de Engenharia, Rio de Janeiro, 2003. Disponível em: <http://www.pgee.ime.eb.br/pdf/edna_guedes.pdf>. Acesso em: 10 fev. 2010.

HOLZNER, Steven. **Desvendando XML.** Rio de Janeiro: New Riders, 2001. 858 p.

IEEE. **IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.** New York: IEEE Computer Society, 1990. 1360 p.

IETF. **Hypertext Transfer Protocol:** HTTP/1.1. 1999. Disponível em: <http://www.ietf.org/rfc/rfc2616.txt>. Acesso em: 10 fev. 2010.

IETF. **Uniform Resource Identifier (URI):** Generic Syntax. 2005. Disponível em: <http://www.ietf.org/rfc/rfc3986.txt>. Acesso em: 10 fev. 2010.

LITCHFIELD, Steve. **Assisted GPS and the future of smartphones.** 2007. Disponível em: <http://www.allaboutsymbian.com/features/item/The_future_of_GPS-equipped_smartphones.php>. Acesso em: 11 nov. 2010.

LIZAKOSKI, Ismael. **Desenvolvimento de protótipo LBS Webmobile com uso da API J2ME.** 2008. 52 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Centro Universitário Feevale, Novo Hamburgo. Disponível em: <http://tconline.feevale.br/tc/files/0001_1526.pdf>. Acesso em: 04 nov. 2009.

MANSANO, Willian. **Aplicação de XML para intercâmbio de documentos complexos.** 2002. 169 f. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas, 2002. Disponível em: <http://libdigi.unicamp.br/document/?code=vtls000290978>. Acesso em: 23 out. 2009.

NEIS, Pascal; ZIPF, Alexander. OpenRouteService.org is three times “Open”: Combining OpenSource, OpenLS and OpenStreetMaps. In: UNIGIS, 16., 2008, Manchester. **Proceedings of GISRUK 2008.** Manchester: Gisruk, 2008. p. 248 - 251. Disponível em: <http://www.unigis.org/gisruk_2008/proceedings.htm>. Acesso em: 01 maio 2010.

NOKIA. **MIDP: Location API Developer's Guide.** Versão 2.0. Disponível em: <http://sw.nokia.com/id/175bf8e6-a1f5-4d3d-a591-6fc936506a6b/MIDP_Location_API_Developers_Guide_v2_0_en.pdf>. Acesso em: 06 mar. 2010.

OGC. **OpenGIS Location Services (OpenLS): Core Services.** Versão 1.2. 2008. Disponível em: <http://www.opengeospatial.org/standards/ols>. Acesso em: 05 fev. 2010.

OMA. **OMA Mobile Location Protocol.** Versão 3.1. 2004. Disponível em: <http://www.openmobilealliance.org/Technical/release_program/mlp_v31.aspx>. Acesso em: 05 fev. 2010.

ORACLE. **Oracle Spatial Developer’s Guide.** Disponível em: <http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28400.pdf>. Acesso em: 22 jun. 2010.

ORS. **OpenRouteService.** Disponível em: <http://www.openrouteservice.org>. Acesso em: 01 mai. 2010.

ORT, Ed; MEHTA, Bhakti. **Java Architecture for XML Binding.** 2003. Disponível em: <http://www.oracle.com/technetwork/articles/javase/index-140168.html>. Acesso em: 12 nov. 2010.

OSM. **OpenStreetMap.** Disponível em: <http://www.openstreetmap.org/>. Acesso em: 01 mai. 2010.

PANDEY, Aradhana. **Image Compression for mobile based location search engine.** In: GEOMATRIX, 1., 2009, Indore. Proceedings of Conference on Geospatial Technologies. Bombay: Rea Of Csre, 2009. p. 1 - 13.

PITTS-MOULTIS, Natanya. **XML black book.** São Paulo: Makron Books, 2000. 627 p.

POTTS, Stephen; KOPACK, Mike. **Aprenda Web services em 24 horas.** Rio de Janeiro: Campus, 2003. 367 p.

SILVA, Grace Kelly de Castro. **Implementação de Serviços Baseados em Localização utilizando arquiteturas e padrões abertos.** 2005. 121 f. Dissertação (Mestrado) - Universidade Estadual de Campinas, Campinas. Disponível em: <http://libdigi.unicamp.br/document/?code=vtls000359246>. Acesso em: 23 out. 2009.

TRIGO, Clodonil Honório. **OpenLDAP**: uma abordagem integrada. São Paulo: Novatec, 2007.

W3C. **Extensible Markup Language (XML) 1.0 (Fifth Edition):** W3C Recommendation. 2008. Disponível em: <http://www.w3.org/TR/REC-xml/>. Acesso em: 05 fev. 2010.
W3C. **Namespaces in XML 1.0 (Third Edition):** W3C Recommendation. 2009. Disponível em: <http://www.w3.org/TR/xml-names/>. Acesso em: 05 fev. 2010.

W3C. **Web Services Architecture.** 2004a. Disponível em: <http://www.w3.org/TR/ws-arch/>. Acesso em: 05 fev. 2010.

W3C. **Web Services Description Language (WSDL) Version 2.0.** 2007. Disponível em: <http://www.w3.org/TR/wsdl20/>. Acesso em: 14 fev. 2010.

W3C. **World Wide Web Consoritum.** Disponível em: <http://www.w3.org>. Acesso em: 23 mai. 2010.

W3C. **XML Schema.** 2004b. Disponível em: <http://www.w3.org/TR/xmlschema-0/>. Acesso em: 05 fev. 2010.

W3SCHOOLS**. XML Schema.** Disponível em: <http://www.w3schools.com/schema/default.asp>. Acesso em: 10 fev. 2010.

WILBRINK, Will. **OpenLS Assisting Government.** Disponível em: <http://portal.opengeospatial.org/files/index.php?artifact_id=6185&version=1&format=pdf>. Acesso em: 19 mar. 2010.

APÊNDICE A – ARTIGO

Web Services e OpenLS para Interoperabilidade no Acesso a LBS

Leandro Natal Coral[1], Evânio Ramos Nicoleit[1]

[1]Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
Caixa Postal 3.167 – 88.806-000 – Criciúma – SC – Brasil
{leandronatalcoral,evnic}@gmail.com

Abstract. *This paper describes the development of a LBS Directory Service prototype using Web Services (WS) and the OpenLS open standard to provide interoperability in its access. Location Based Services (LBS) are services that explore the geographical position of Mobile Devices, providing useful information to their users. LBS require interoperability, because they are accessed by different platforms, operating systems and network infrastructures. Among the existents LBS types is the Directory Service, which provides Points of Interest specific or nearest a location. The results indicate that the use of WS with OpenLS is viable for the interoperability provision in LBS.*

Resumo. *Este artigo descreve o desenvolvimento de um protótipo de Serviço de Diretório de LBS, utilizando Web Services (WS) e o padrão aberto OpenLS para prover interoperabilidade em seu acesso. Serviços Baseados em Localização (LBS) são serviços que exploram a posição geográfica de Dispositivos Móveis, fornecendo informações úteis aos seus usuários. LBS requerem interoperabilidade, pois são acessados por diferentes plataformas, sistemas operacionais e infraestruturas de rede. Dentre os tipos de LBS existentes está o Serviço de Diretório, que fornece Pontos de Interesse específicos, ou mais próximo de determinada localização. Os resultados indicam que o uso de WS com o OpenLS é viável para a provisão de interoperabilidade em LBS.*

1. Introdução

As inovações tecnológicas das redes de comunicação de dados sem fio integradas à web vêm promovendo um crescimento exponencial no que se refere a Dispositivos Móveis (DM). DM podem ser utilizados em deslocamento, estando disponíveis em diferentes localizações geográficas. Atualmente existem soluções que possibilitam a obtenção da posição geográfica de DM. Exemplos são as Tecnologias de Posicionamento (TP) Sistema de Posicionamento Global (GPS), a Identificação da Célula (Cell-ID) e o GPS Assistido (A-GPS). Rastreadores veiculares, utilizados frequentemente no TR, são exemplos de DM providos de TP.

As possibilidades de Dispositivos Móveis (DM) juntamente com as necessidades cotidianas da sociedade atual deram origem a novos serviços, conhecidos como Serviços

Baseados em Localização (LBS). Pode-se conceituar LBS qualquer serviço de aplicação que explore a posição geográfica de DM, fornecendo informações úteis aos seus usuários [OGC 2008]. LBS podem ser aplicados a diversas áreas, como a de Rastreamento e Gerenciamento de Frotas, onde se podem fornecer informações relacionadas à localização de veículos de transporte por meio de monitoramento e rastreamento baseado em LBS.

Os dados utilizados pelos LBS tipicamente são obtidos por meio de Sistemas Gerenciadores de Banco de Dados Geográficos (SGBDG), que possibilitam representar dados geográficos e associá-los com dados convencionais [Câmara et al 2005]. Entidades do mundo real, como POI, estradas e limites administrativos podem ser representadas em SGBDG respectivamente por pontos, linhas e polígonos [Silva 2005].

LBS são categorizados em núcleos, de acordo com o tipo de serviço provido. Um dos possíveis núcleos é o de Serviços de Diretório (SD), onde se pode obter a localização de Pontos de Interesse (POI [1]) específicos ou mais próximo de determinada localização, que pode ser a posição de DM.

LBS requerem cada vez mais interoperabilidade, pois são acessados por diferentes plataformas, sistemas operacionais e infraestruturas de rede [Silva 2005]. O uso da tecnologia Web Services (WS) e de padrões abertos visam contemplar estes requisitos.

Este artigo discute os resultados obtidos em um Trabalho de Conclusão de Curso desenvolvido pelo primeiro autor na Universidade do Extremo Sul Catarinense (UNESC), especificamente um SD, utilizando WS e o padrão aberto para OpenLS para a provisão de interoperabilidade. Por contemplar as características do sistema proposto, o padrão abertos para LBS utilizado foi o OpenLS, da OGC. O trabalho dá ênfase nos seus exemplos à área de Transporte Rodoviário (TR).

2. Serviços de Diretório de LBS

Para consultar um SD em um LBS, deve-se essencialmente fornecer ao serviço algum tipo de identificador de fácil interpretação, como nome, telefone ou categoria do POI desejado [OGC 2008].

Há dois de SD: Serviço de Diretório Específico (SDE), que fornece a localização de POI específicos, onde é especificado qual o POI procurado e; Serviço de Diretório por Proximidade (SDP), que usa um filtro espacial para localizar POI mais próximos, em determinada distância, ou dentro de uma determinada área.

Observa-se o uso de SD no TR onde, por meio de rastreadores, obtêm-se as coordenadas geográficas dois veículos, podendo-se utilizá-las para consultar um SD, a fim de localizar clientes, oficinas, postos de combustíveis os quaisquer pontos de apoio nas proximidades. Pode-se também localizar clientes, locais de embarque ou descarga em uma área específica, delimitada por um conjunto de coordenadas geográficas.

---

[1] POI podem ser locais, produtos ou serviços existentes em um espaço geográfico.

3. A Tecnologia Web Services

WS fornecem uma arquitetura padronizada para interoperabilidade entre diferentes aplicações [W3C 2004], de forma que estas possam se comunicar utilizando padrões web [Arsanjani et al 2003].

Uma das formas de acesso a WS é por meio do Protocolo de Acesso Simples a Objetos (SOAP) [Potts e Kopack 2003], que define um protocolo baseado em XML para troca de mensagens com WS [Potts e Kopack 2003]. O SOAP define duas partes básicas para as mensagens: cabeçalho (elemento Header) e corpo (elemento Body). Ambos são contidos em um elemento raiz, chamado Envelope. O cabeçalho é opcional. O corpo contém o payload, que pode ser um documento baseado em XML ou Chamadas Remotas de Procedimentos (RPC).

A descrição de WS SOAP é feita por meio de um contrato definido em Linguagem de Descrição de Web Services (WSDL), onde são especificados detalhes de acesso ao WS [W3C 2004].

Requisições enviadas a WS são processadas em classes conhecidas como *endpoints*. Endpoints devem implementar os métodos definidos no contrato do WS.

4. Padrão Aberto OpenLS

A especificação OpenLS disponibiliza interfaces padronizadas baseadas em XML, que garantem o desenvolvimento de LBS altamente interoperáveis. Estas interfaces definem e provém acesso aos principais núcleos de LBS, incluindo além do SD (implementado no presente trabalho), os demais núcleos de LBS (Serviços de Gateway, Serviços de Determinação de Rotas, Serviços de Apresentação, Serviços de Geocodificação e Serviços de Geocodificação Reversa) [Silva 2005].

As interfaces do OpenLS são definidas por meio de Definição de Esquemas XML (XSD). Os esquemas XML do OpenLS definem estrutura das requisições e respostas de LBS e tipos de dados que poderão ser utilizados.

A estrutura básica das requisições e respostas do OpenLS está no uso de um elemento raiz *XLS*, que deve possuir dois elementos filhos: *RequestHeader* e *Request* para as requisições e; *ResponseHeader* e *Response* para as respostas [OGC 2008]. Destaca-se o atributo *methodName*, do elemento *Request*, que informa a qual núcleo destina-se a requisição e; os elementos abstratos *_RequestParameters* e *_ResponseParameters*, que são substituídos respectivamente pelos parâmetros de requisição e resposta específicos do núcleo utilizado.

Uma característica visível na definição do OpenLS é o uso constante de polimorfismo em XML, por meio dá cláusula *substitutionGroup* da especificação XSD. Outro ponto fundamental é a divisão do esquema XML em fragmentos (arquivos XSD menores) contendo partes específicas de sua estrutura.

O SD do OpenLS é definido no esquema DirectoryService.xsd, e interliga-se por meio da cláusula include, respectivamente, com os esquemas *XLS.xsd*; *ADT.xsd*; *geometry.xsd* e; *UOM.xsd*. Os esquemas DirectoryService.xsd, ADT.xsd e geometry.xsd também utilizam a cláusula import para a importação do *gml4xls.xsd*, que é uma definição da Linguagem de Marcação Geográfica (GML) adaptada ao OpenLS. Os

esquemas *geometry.xsd* e *UOM.xsd* importam o *xlinks.xsd* por meio da URL *http://schemas.opengis.net/xlink/1.0.0/xlinks.xsd.*

Todos os tipos de dados necessários para LBS são definidos nos esquemas XML do OpenLS. Exemplos são o elemento *PointOfInterestType*, que representa um POI; o elemento *PositionType,* que representa uma posição geográfica e; o elemento *AddressType*, que representa um endereço.

5. Metodologia

A metodologia para o desenvolvimento do protótipo baseou-se nas seguintes etapas: seleção dos recursos de software; modelagem e desenvolvimento do LBS com implementação do núcleo de SD; aplicação do OpenLS e implantação do WS para provisão de interoperabilidade no acesso serviço e; execução de testes para análise do resultados obtidos.

5.1. Recursos Utilizados

Inicialmente foram selecionados os recursos disponíveis com licença de uso livres voltados ao desenvolvimento deste tipo de aplicação.

No ambiente de desenvolvimento, utilizou-se a linguagem Java com a IDE Eclipse. O Apache Maven foi utilizado para o gerenciamento das dependências. O Hibernate e o Hibernate Spatial auxiliaram no mapeamento objeto-relacional (o segundo para tipos de dados geográficos). O Apache Tomcat foi escolhido como servidor de aplicação Java. O Apache CXF foi utilizado como framework para Web Services por integrar-se com o framework Spring e com as API JAXB e JAX-WS. O JAX-WS foi adotado como API para desenvolvimento do WS. O JAXB foi utilizado como API de Binding entre Java e XML, possibilitando além da criação de classes Java com base em esquemas XML, a conversão de objetos XML em objetos Java e vice-versa.

Como banco de dados, foi utilizado o PostgreSQL com sua extensão espacial PostGIS, o que provê funcionalidades de SGBDG. A ferramenta front-end PGAdmin foi utilizada para administração do banco.

Para os testes, utilizou-se a aplicação de código fonte aberto SoapUI, que permite a troca de mensagens SOAP com o WS, possibilitando testá-lo sem necessidade de implementação de uma aplicação cliente.

5.2. Modelagem e Desenvolvimento do SD para o LBS

SD necessitam acessar bases de dados contendo informações de POI. Primeiramente criou-se uma base de dados no PostgreSQL, na qual foi adicionado o PostGIS por meio da execução dos scripts postgis.sql, spatial_ref_sys.sql e postgis_comments.sql.

Após preparação da base de dados, foram criadas entidades para representar POI. Para isto, foram implementadas em Java as classes POI e Endereco, mapeadas com o Hibernate e Hibernate Spatial. As entidades podem ser visualizadas na Figura 1.

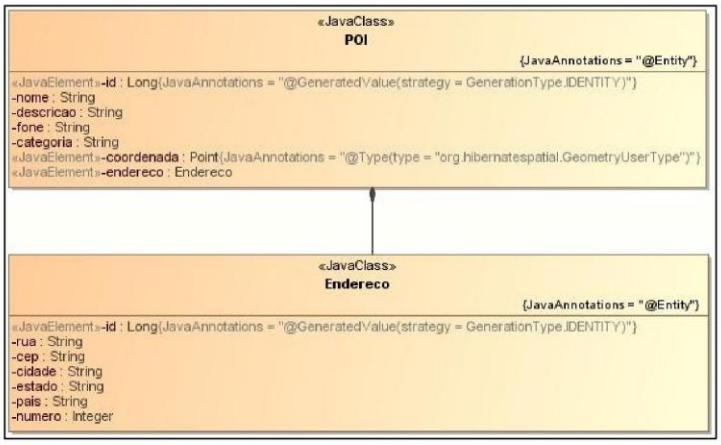


Figura 1. Entidades para conter POI

O Hibernate ficou responsável pela criação automática das tabelas na base de dados. Entretanto, houve dificuldade na adição da coluna geográfica *coordenada* pelo Hibernate Spatial, problema contornado como chamada à função *AddGeometryColumn* do PostGIS diretamente no PGAdmin.

Criadas as entidades inseriram-se POI fictícios para testes, os quais tiveram suas localizações geográficas definidas aleatoriamente, dentro dos limites da cidade de Criciúma, onde se desenvolveu o protótipo. A Figura 22 mostra uma consulta efetuada no PGAdmin na tabela *poi*, selecionando os POI inseridos para testes.

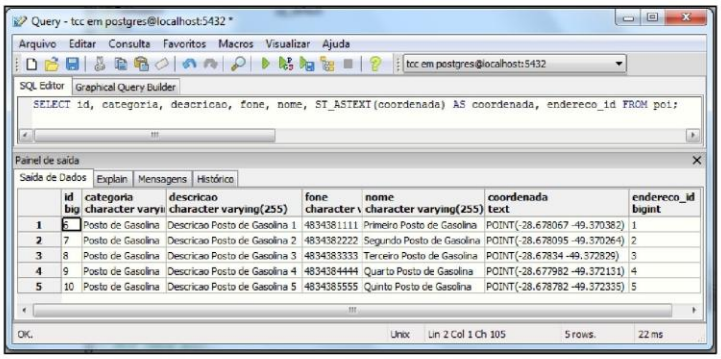


Figura 2. Visualização dos dados inseridos para testes

Disponibilizados os dados, criou-se o SD. Foram criados três tipos de SD: um SDE, denominado *poiByProperties*, que fornecem POI por meio de propriedades que o identifiquem; um SDP denominado *poiNearestToPosition*, que fornece o POI mais próximo de determinada posição geográfica, podendo esta ser obtida de um DM por meio de TP (possivelmente um rastreador veicular) e; um SDP denominado *poiNearestToPoi*, que fornece o POI mais próximo de determinado POI. Os três SD descritos são implementados em Java na classe *DirectoryService*, que é mostrada na Figura 3.

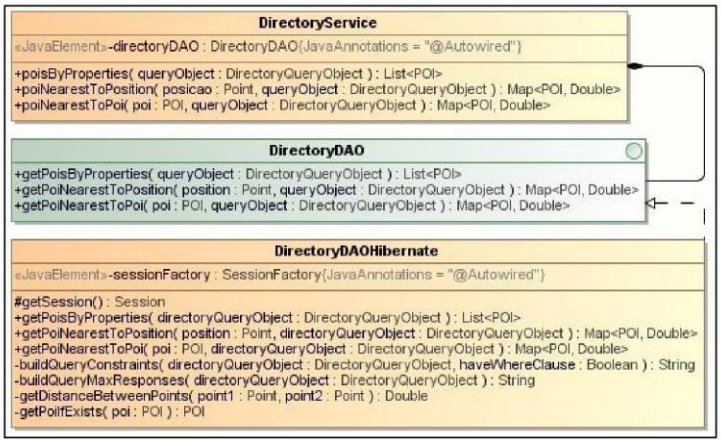


Figura 3. Classes criadas para o SD

Criou-se uma classe *DirectoryQueryObject*, que representa um objeto de consulta ao SD contendo as propriedades de POI desejadas. Também foi criada uma interface que define um contrato para o Objeto de Acesso a Dados (DAO), determinando os métodos de consulta ao banco de dados requeridos pelo serviço. A classe *DirectoryDAOHibernate* implementou a interface com funções do Hibernate.

Neste ponto já se tem um SD funcional. Entretanto, o serviço só pode ser acessado pela própria aplicação. Uma das formas de disponibilizar o serviço em meio externo, provendo interoperabilidade, é por meio de WS. Todavia, apesar de sua especificação prover interoperabilidade no acesso, seu uso não garante que as informações transmitidas no payload das mensagens SOAP sejam interpretadas por ambos os lados da comunicação. A adoção do padrão aberto OpenLS permite definir um protocolo na comunicação com o LBS. Por serem baseadas em XML, requisições e respostas do OpenLS podem ser utilizadas como payload das mensagens SOAP.

5.3. Aplicação do OpenLS no LBS

A estrutura do OpenLS é disponibilizada em esquemas XML. A geração automática das classes Java a partir de esquemas XML se dá meio de uma ferramenta conhecida como Binding Compiler (BC). O JAXB fornece o BC *xjc*, que foi utilizado para a geração das classes Java a partir do esquema XML do OpenLS, facilitando a aplicação deste padrão aberto no protótipo de LBS.

Na execução inicial do xjc para o esquema *DirectoryService.xsd*, ocorreram conflitos de colisão de nomes entre alguns elementos envolvidos. Os elementos de base _POI, _Position, _RouteSummary, _NextSegment e _POIProperty conflitaram respectivamente com os elementos POI, Position, RouteSummary, NextSegment e POIPropert. Isto se deve à forma na qual o BC do JAXB nomeia as classes geradas. A ferramenta ignorou o caracter _ (underline) precedente ao nome dos elementos, tentando gerar classes com nomes idênticos para ambos, o que resultou em conflitos. O problema foi contornado com a criação de um arquivo de customização de Binding nomeado *custombinding.xjb*, onde foram definidos explicitamente os nomes das classes a serem geradas para os elementos conflitantes, cujo conteúdo pode ser visualizado na Figura 4.

```
<jxb:bindings version="1.0" xmlns:jxb="http://java.sun.com/xml/ns/jaxb"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <jxb:bindings schemaLocation="ADT.xsd" node="//xs:element[@name='_POI']">
    <jxb:class name="POIBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="ADT.xsd"
    node="//xs:element[@name='_Position']">
    <jxb:class name="PositionBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="ADT.xsd"
    node="//xs:element[@name='_RouteSummary']">
    <jxb:class name="RouteSummaryBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="ADT.xsd"
    node="//xs:element[@name='_NextSegment']">
    <jxb:class name="NextSegmentBaseElement" />
  </jxb:bindings>
  <jxb:bindings schemaLocation="DirectoryService.xsd"
    node="//xs:element[@name='_POIProperty']">
    <jxb:class name="POIPropertyBaseElement" />
  </jxb:bindings>
</jxb:bindings>
```

Figura 4. Customização de Binding solucionando conflitos do JAXB

O comando final para geração das classes foi o seguinte: *xjc -extension -nv -b custombinding.xjb DirectoryService.xsd*. O x*jc* gerou um pacote *gml* e um pacote *xls*, contendo as classes utilizadas no OpenLS para SD. O BC também gerou uma classe *ObjectFactory*, que serve como uma fábrica para a criação dos demais objetos.

Criadas as classes Java para os elementos que compõe o OpenLS, tem-se a possibilidade de instanciar objetos em conformidade com sua especificação. No entanto, observou-se que a aplicação direta do padrão no serviço existente, impactaria em necessidade de modificações no seu código. Isto geraria, além de custos referentes ao tempo de planejamento das modificações, uma série de incompatibilidades com as aplicações que acessam o serviço. Chegou-se a este impasse propositalmente, visando simular uma situação real, onde uma suposta empresa fornece um Serviço de Diretório operável, e necessita adotar um padrão para interoperabilidade. Deve-se pensar em uma solução que mantenha a modelagem dos dados e serviços existentes, de modo que não interfira nas funcionalidades operantes.

A solução criada consiste no desenvolvimento de uma camada de código superior à da aplicação, voltada unicamente a fins de distribuição e interoperabilidade. Esta camada envolve duas etapas: criação de uma interface de serviço específica para o OpenLS, que faz chamada ao serviço existente na aplicação e; criação de uma classe para a conversão de objetos OpenLS em objetos da aplicação e vice-versa. A solução encontra-se ilustrada na Figura 5.

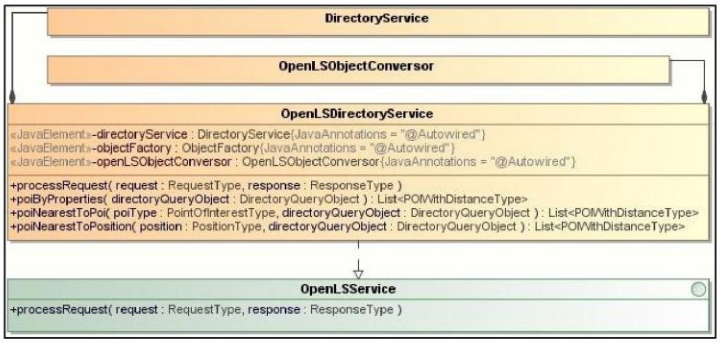


Figura 5. Camada de serviço OpenLS com acesso ao serviço da aplicação

A interface *OpenLSService* é quem define a forma como os serviços do OpenLS atenderão as requisições. Seu método *processRequest*, de uso comum a todos os núcleos do LBS, define que o serviço deve receber referência de um objeto *RequestType*, referente à requisição, e de um objeto *ResponseType*, referente à resposta. O serviço do OpenLS processa o objeto de requisição e atualiza o objeto de resposta diretamente na fonte de sua chamada (neste protótipo, quem deverá chamar o método *processRequest* é o endpoint do WS).

Para implementação do SD do OpenLS, criou-se a classe *OpenLSDirectoryService*, responsável pela implementação do método da interface com lógicas de processamento específicas deste tipo de serviço.

5.4. Implantação do WS

Após a aplicação do OpenLS, que permite que as aplicações clientes possam se comunicar com o LBS de forma interoperável, foi implantado o WS, que garante a consistência e acessibilidade desta comunicação.

Inicialmente, criou-se o contrato WSDL para o WS. A API JAX-WS permite a criação automática do contrato com base no código Java. Neste caso, optou-se pela criação manual, sabendo que esta abordagem diminui o acoplamento entre o contrato e a implementação, aumenta a interoperabilidade e facilita a manutenção do WS.

No elemento *wsdl:types, d*efiniu-se a utilização do esquema XML *DirectoryService.xsd*. No elemento *wsdl:message*, definiu-se mensagens de requisição e resposta, ambas contendo o elemento *XLS* como raiz. No elemento *esdl:portType*, definiu-se uma operação *getService*, que recebe e retorna as mensagens de requisição e resposta definidas no elemento wsdl:*message*. Esta operação corresponde ao método que deve ser implementado pelo endpoint. No elemento *wsdl:binding*, definiu-se o protocolo e o formato das mensagens. Utilizou-se o protocolo HTTP para SOAP, e o formato das mensagens *Document/Literal*, *que* permite um menor acoplamento entre as aplicações e um maior grau de interoperabilidade, além de possibilitar que o próprio esquema XML defina as regras das mensagens. O uso de RPC resultaria na violação de algumas regras da especificação OpenLS. A Figura 6 mostra parte do contrato WSDL criado para o WS do LBS.

```
...
   <wsdl:types>
      <xsd:schema elementFormDefault="qualified">
         <xsd:import namespace="http://www.opengis.net/xls"
                     schemaLocation="../../opengis/schema/DirectoryService.xsd" />
      </xsd:schema>
   </wsdl:types>
   <wsdl:message name="Request">
      <wsdl:part name="XLS" element="XLS"/>
   </wsdl:message>
   <wsdl:message name="Response">
      <wsdl:part name="XLS" element="XLS" />
   </wsdl:message>
   <wsdl:portType name="ServicePort">
      <wsdl:operation name="getService">
         <wsdl:input message="tns:Request" />
         <wsdl:output message="tns:Response" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="ServiceBinding" type="tns:ServicePort">
      <soap:binding style="document"
                    transport="http://schemas.xmlsoap.org/soap/http" />
```

```
<wsdl:operation name="getService">
   <soap:operation soapAction="Service\#getService" />
   <wsdl:input>
      <soap:body namespace="http://ws.tcc.unesc.net/wsdl" use="literal" />
   </wsdl:input>
   <wsdl:output>
      <soap:body namespace="http://ws.tcc.unesc.net/wsdl" use="literal" />
   </wsdl:output>
...
```

Figura 6. Contrato WSDL

Criado o contrato, desenvolveu-se o endpoint para processamento das requisições ao WS. Para isto, criou-se em Java uma interface *LbsEndpoint*, contendo o método *getService* correspondente à operação definida no elemento *wsdl:portType* do contrato. A interface foi implementada pela classe LbsEndpointImpl, que também contém métodos auxiliares para geração de listas de erros de acordo com a especificação OpenLS. Pode-se visualizar a modelagem do endpoint na Figura 7.

O trecho de código apresentado na Figura 8 é um recurso do Apache CXF que informa ao a classe utilizada para implementação do endpoint e habilita a validação das mensagens de acordo com os esquemas XML utilizado, neste caso, o OpenLS.

A Figura 9 contém um diagrama de atividades mostrando todas as etapas realizadas desde a requisição ao LBS pela aplicação cliente, até o envio da resposta processada pelo LBS.

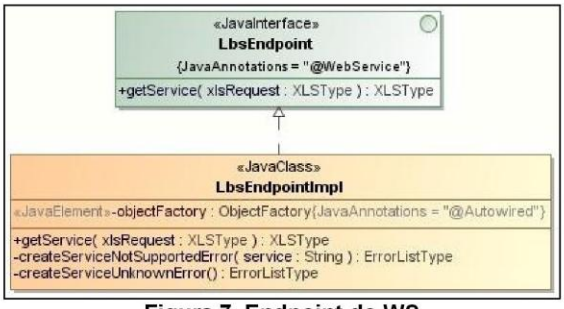


Figura 7. Endpoint do WS

```
<jaxws:endpoint id="lbsEndpoint" implementor="#lbsEndpointImpl" address="/LbsEndpoint">
<jaxws:properties>
  <entry key="schema-validation-enabled" value="true" />
  </jaxws:properties>
</jaxws:endpoint>
```

Figura 8. Recurso do CXF para validação das mensagens com o esquema XML

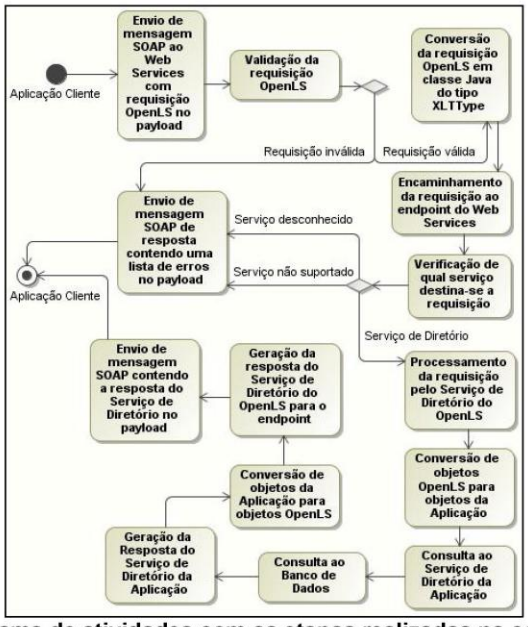


Figura 9. Diagrama de atividades com as etapas realizadas na consulta ao LBS

5.5. Execução dos Testes e Resultados

Inicialmente criou-se um projeto de testes no SoapUI e indicou-se a URL de publicação do documento WSDL. A ferramenta importou automaticamente as configurações do WS, adicionando, além da operação disponível *getService*, um modelo para sua requisição.

Para certificar o bom funcionamento do protótipo, foram criados os diversos tipos requisições possíveis ao LBS: requisições para os SDE e SDP disponíveis; requisições para serviços não suportados e serviços inválidos e; requisições inválidas (com problemas na estruturação).

Para todas as requisições possíveis ao LBS, foram geradas as devidas respostas. A formatação das mensagens SOAP do LBS com o tipo Document/Literal garantiu que o documento XML no payload das mensagens de requisição e resposta respeitassem integramente a especificação OpenLS, contendo como raiz o elemento *XLS*, juntamente com os demais elementos filhos definidos no padrão. Mesmo requisições inválidas, cujas limitações na estrutura não podem ser interpretadas pelo LBS, puderam ser detectadas e respondidas à aplicação cliente, informando claramente os erros cometidos.

6. Conclusão

Este trabalho discutiu um importante tipo de aplicação que vêm crescendo constantemente com a evolução e acessibilidade de tecnologias tais como redes de comunicação sem fio, Dispositivos Móveis e tecnologias de posicionamento, os LBS.

LBS podem prover informações úteis a diversas áreas, dentre elas o Transporte Rodoviário de Cargas, onde é possível obter informações tais como localização de veículos, seu posicionamento em relação a Pontos de Interesse (POI), melhores rotas

entre dois ou mais pontos, etc.. LBS são categorizados em diferentes núcleos, de acordo com o tipo de funcionalidade provida. Dentre os núcleos possíveis de LBS está o Serviço de Diretório, que permite consultar POI específicos ou mais próximos de determinada posição. LBS requerem cada vez mais interoperabilidade, uma vez que podem estar disponíveis em diferentes dispositivos, possuir interface com sistemas de banco de dados legados e englobar diversas tecnologias de infraestrutura de rede.

Os resultados mostraram que a arquitetura Web Services combinada com o padrão aberto OpenLS é viável para a provisão de interoperabilidade em LBS, e seu uso em aplicações LBS fornece uma interface de acesso totalmente padronizada, facilitando a integração e utilização dos serviços.

Referências

Arsanjani, A., Hailpern, B., Martin, J. e Tarr, P. L. (2003) "Web services: promises and compromises", Acm Queue, New York, p.48-58.

Câmara, G., Casanova, M. A., Junior, C. D., Vinhas, L. e Queiroz, G. (2005) "Bancos de Dados Geográficos", Mundogeo, Curitiba.

OGC (2008) "OpenGIS Location Services (OpenLS): Core Services", Versão 1.2, http://www.opengeospatial.org/standards/ols, Fevereiro.

Potts, S. e Kopack, M. (2003) "Aprenda Web services em 24 horas", Campus, Rio de Janeiro.

Silva, G. K. C. (2005) "Implementação de Serviços Baseados em Localização utilizando arquiteturas e padrões abertos", Universidade Estadual de Campinas, Campinas, http://libdigi.unicamp.br/document/?code=vtls000359246, Outubro.

W3C (2004) "Web Services Architecture", http://www.w3.org/TR/ws-arch, Fevereiro.