Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

DANIEL PAULO DE ASSIS

Uma Arquitetura de Software Baseada em Web Services

2012

Assis, SP

Fundação Educacional do Município de Assis
Instituto Municípal de Ensino Superior de Assis - IMESA

DANIEL PAULO DE ASSIS

Uma Arquitetura de Software Baseada em Web Services

Trabalho apresentado ao programa de iniciação Científica (PIC) do Instituto Municipal de Ensino Superior de Assis - IMESA e a Fundação Educacional do Munícipio de Assis – FEMA

Orientando: Daniel Paulo de Assis
Orientadora: Profa. Dra. Marisa Atsuko Nitto

Linha de pesquisa: Ciências Exatas e da Terra

2012

Assis, SP

FICHA CATALOGRÁFICA

PAULO DE ASSIS, Daniel
Uma Arquitetura de Software Baseada em Web Services/ Daniel Paulo de Assis.
Fundação Educacional do Município de Assis – Fema : Assis, 2012
77p.

Orientadora: Profa. Dra. Marisa Atsuko Nitto
Projeto de Iniciação Científica (PIC) – Ciência da Computação - Instituto Municipal de Ensino Superior de Assis

1. Web Services 2. Software. 3. Arquitetura

CDD: 001.6
Biblioteca da FEMA

RESUMO

Na atualidade a informatização de conteúdos, e a integração entre sistemas através da internet é uma realidade, o consumo de serviços fornecidos através dessa grande rede provê inúmeras facilidades a pequenas, médias, grandes empresas e também aos usuários que podem usufruir de todas essas facilidades.

O consumo de serviços disponibilizados online através da internet vem crescendo à passos largos, e é uma tendência que deve prosseguir com força total, o grande reaproveitamento de código, econômia de processamento e a possibilidade de interação independente de plataforma e linguagem de programação tornam os web services uma ferramenta extraordinária, facilitando em muito a vida de desenvolvedores de sistemas.

O projeto do desenvolvimento de um aplicativo utilizando a tecnologia de Arquitetura Orientada a Serviço baseado em *Web Services*. A linguagem de programação utilizada para a implementação do aplicativo é Java, com o servidor de aplicações *web* Apache Tomcat e banco de dados PostgreSQL. O estudo de caso utilizado é um aplicativo *web* integrado com o Google Maps API.

Lista de Figuras

Figura 1 - Web Services independente da plataforma (WIEHLER, 2004) 10

Figura 2 - Funcionamento simplificado de um Web Service (AYALA et al., 2002) 11

Figura 3 - Arquitetura SOA baseada em Web Services (WIEHLER, 2004) 14

Figura 4 - Modelo de Comunicação SOAP sobre protocolo HTTP (ENDEL O, 2003) 18

Figura 5 - Arquitetura de um *Web Service* (ENDEL D, 2012) 20

Figura 6 - Três componentes básicos utilizados pelos serviços web (CESCONETI e GLAZAR, 2006) 21

Figura 7 - Diagrama SOA (FRONDANA et al., 2009) 23

Figura 8 - Arquitetura SOA (SILVA e SIROTHEAU, 2006) 24

Figura 9 - Visão da arquitetura com SOA (SILVA e SIROTHEAU, 2006) 25

Figura 10 - XML *Document Example* (ENDEL F, 2012) 30

Figura 11 - Arquitetura WSDL (ENDEL M, 2012) 32

Figura 12 - Estrutura de um documento SOAP (ENDEL G, 2012) 35

Figura 13 - Esqueleto de mensagem Soap (ENDEL G, 2012) 35

Figura 14 - Arquitetura UDDI (ENDEL H, 2012) 39

Figura 15 - Visão geral do sistema (DE ASSIS, 2011) 42

Figura 16 -Visualização de marcador no mapa (DE ASSIS, 2011) 42

Figura 17 - Visão geral do sistema para consumo dos *web services* 44

Figura 18 - Diagrama de Caso de Uso (Aplicativo Cadastro) 45

Figura 19 - Diagrama de Caso de Uso (Aplicativo Servidor) 46

Figura 20 - Diagrama de Caso de Uso (Aplicativo Cliente) 47

Figura 21 - Pacote weblocalcadastro.uteis 49

Figura 22 - Pacote weblocalcadastro.dao 50

Figura 23 - Pacote weblocalcadastro.bean 51

Figura 24 - Pacote weblocais.uteis 52

Figura 25 - Pacote weblocais.dao 53

Figura 26 - Pacote weblocais.model 53
Figura 27 - Pacote weblocais.services 54
Figura 28 - Pacote weblocais.business 55
Figura 29 - Pacote weblocaiscliente.beans 56
Figura 30 - Pacote weblocaiscliente.control 57
Figura 31 - Diagrama de Atividades 58
Figura 32 - DER (Diagrama Entidade-Relacionamento) 59
Figura 33 - Mapa do Site (Cadastro) 59
Figura 34 - Mapa do site (cliente) 60
Figura 35 - WebServices Disponibilizados (IDE Netbeans) 62
Figura 36 - Arquivo WSDL gerado 63
Figura 37 - Funcionamento do aplicativo de coleta de dados 64
Figura 38 - Página web (aplicativo servidor online) 65
Figura 39 - Home page (aplicativo cliente) 66
Figura 40 - Menu (aplicativo cliente) 66
Figura 41 - Busca visualização por lista (filtro por palavra chave) 68
Figura 42 - Busca visualização por lista (filtro por categoria) 68
Figura 43 -Busca visualização por lista (filtro por rua) 69
Figura 44 - Busca visualização por lista (filtro por cep) 69
Figura 45 - Visualização de Locais no mapa 70
Figura 46 - Visualização da infoWindow no mapa 71
Figura 47 - Visualização dos "markers" no modo "map" 71
Figura 48 - Utilização do recurso *Street View* na busca por locais 72

Sumário

1 INTRODUÇÃO 9

1.1 PROBLEMATIZAÇÃO 12

1.2 OBJETIVOS 15

1.2.1 OBJETIVO GERAL 15

1.2.2 OBJETIVOS ESPECÍFICOS 15

1.3 RELEVÂNCIA OU JUSTIFICATIVA 16

2 FUNDAMENTAÇÃO TEÓRICA BÁSICA 16

2.1 Web Services 17

2.1.1 Arquitetura dos *Web Services* 21

2.2 Arquitetura Orientada a Serviços 22

2.2.1 Benefícios Web Services 25

2.2.2 Arquitetura Web Services 26

2.2.3 Web Service Roles (Papéis) 26

2.2.4 *Web Service Stack Protocols* (Pilha de Protocolos) 27

2.2.5 Meios de Comunicação Web Services 28

2.3 XML (eXtensible Markup Language) 28

2.4 WSDL (*Web Services Description Language*) 30

2.5 SOAP (*Simple Object Access Protocol*) 33

2.5.1 Sintax SOAP 34

2.6 UDDI (*Universal Description, Discovery and Integration*) 36

2.6.1 UDDI Arquitetura Técnica 38

2.7 Tecnologias Java 39

2.8 Apache Tomcat 40

2.9 PostgreSQL 41

2.10 Integração de Um Aplicativo Web em Java Com o Google Maps API 41

3 DESENVOLVIMENTO DO PROJETO 43

3.1 Descrição do Problema 43
3.2 Diagrama de Casos de Uso (Use Case) 45
3.2.1 Diagrama de Casos de Uso (Aplicativo Cadastro) 45
3.2.2 Diagrama de Casos de Uso (Aplicativo Servidor) 46
3.2.3 Diagrama de Casos de Uso (Aplicativo Cliente) 47
3.3 Diagrama de Classe 48
3.3.1 Diagrama de Classe (Aplicativo Cadastro) 49
3.3.2 Diagrama de Classe (Aplicativo Servidor) 51
3.3.3 Diagrama de Classe (Aplicativo Cliente) 55
3.4 Diagrama de Atividades 57
3.5 Diagrama Entidade-Relacionamento (DER) 58
3.6 Mapas dos Sites (aplicativo cadastro e cliente) 59
3.7 Serviços disponibilizados e consumidos 61
3.7.1 Web Services 61
3.7.2 WSDL 62
4 FUNCIONAMENTO DOS APLICATIVOS 63
4.1 Aplicativo de Cadastro 63
4.2 Aplicativo Servidor 64
4.3 Aplicativo Cliente 65
5 CONCLUSÃO 72
REFERÊNCIAS 74

1 INTRODUÇÃO

A *World Wide Web* teve como princípio permitir a troca de documentos entre os computadores distribuídos por essa rede, e que com o seu crescimento e popularização passou a ser utilizada como base para comunicação entre aplicações distribuídas que necessitam de um método eficiente para intercâmbio de dados (informações). Com isso, uma parcela significativa dos sistemas computacionais passou a utilizar este modelo de comunicação, permitindo vislumbrar a sua integração, uma vez que uma das maiores dificuldades existentes atualmente no mundo computacional é a integração de sistemas.

As tecnologias para integração buscam prover, em diferentes níveis, meios de realizar trocas de informação entre aplicações distribuídas em um ambiente heterogêneo. Dentre as tecnologias existentes destacam se:

- **uPnP** (*Universal Plug and Play Device Architecture*), utilizada para descoberta e auto-configuração de dispositivos;
- **CORBA** (*Common Object Request Broker Architecture*), utilizada para integração de sistemas computacionais orientadas a objetos;
- **DCOM** (*Distributed Component Object Model*), utilizada para criar componentes de *software* distribuídos em computadores interligados em rede;
- **Web Services**, utilizada para integração das mais diversas aplicações distribuídas sem se preocupar com a heterogeneidade intrínseca dos ambientes distribuídos.

Algumas limitações de cada uma destas tecnologias impedem que sejam usadas em grande escala. Essas limitações resultam de causas como: falta de interoperabilidade, complexidade de implementação, centrada numa dada plataforma, altamente dependente do código e comunicação tipicamente síncrona (ENDEL P, 2000; ENDEL N, 2005; ENDEL Q et al., 2004).

Os *Web Services* apresentam se como uma evolução das tecnologias de comunicação baseadas em objetos distribuídos (VOGELS, 2003). Os *Web*

Services são o mais recente e promissor modelo de computação distribuída baseada em serviços, representando a última proposta de solução para o desenvolvimento e integração de aplicações. Surgem em resposta aos problemas levantados durante décadas pela falta de interoperabilidade e integração de aplicações. Representa também uma mudança na forma de atuação das empresas da área, que estão empenhadas em trabalhar juntas, no desenvolvimento de protocolos abertos para este novo paradigma, deixando a competição para outros planos (MOREIRA, 2005).

Um dos pontos importante dessa evolução é que ao invés de fazer referência a uma interface de um objeto, um *Web Services* busca uma mudança deste paradigma para uma Arquitetura Orientada a Serviços (SOA - *Service-Oriented Architecture*) (CHAPPELL; JEWELL, 2002). Devido á adoção desta arquitetura, os *softwares* ou sistemas deixam de ser orientados a objetos para serem compostos por vários serviços descritos por um ou mais *Web Services*. A figura 1 mostra o contraste da tecnologia CORBA/DCOM com SOA baseada em *Web Services*.

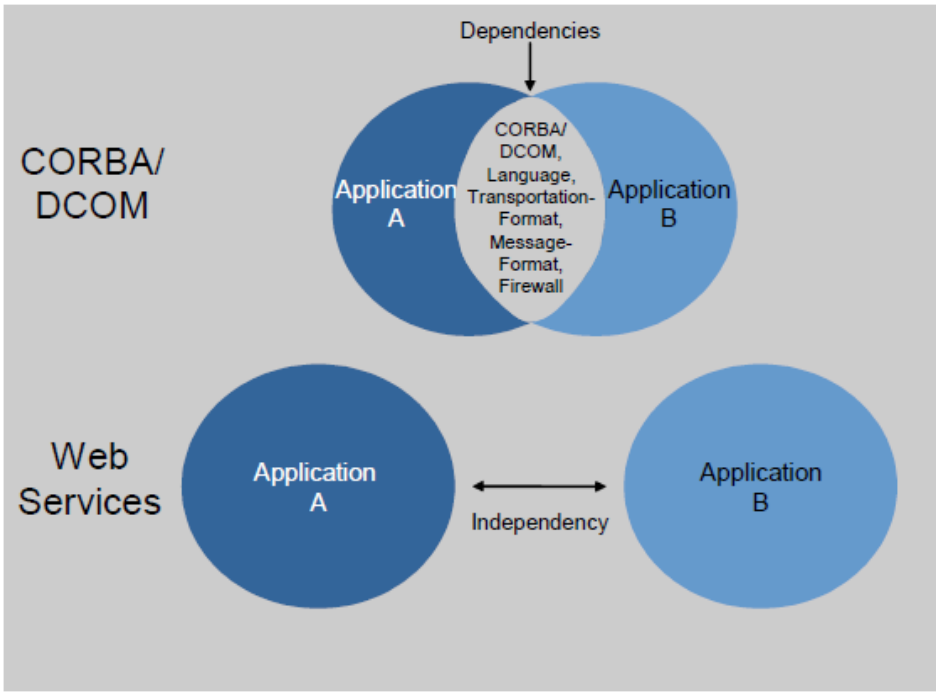


Figura 1 - Web Services independente da plataforma (WIEHLER, 2004)

A vantagem da tecnologia SOA baseada em *Web Service* é que ela independe de plataforma utilizada, bem como a linguagem, o transporte e a independência no formato de mensagem.

A ideia dos *Web Services* é criar componentes de *software* que podem ser ativados a distância via *internet*, trocando informações pela rede na forma de arquivos no padrão XML (*Extensible Markup Language*). Cada componente se autodescreve e eles podem ser listados em diretórios para que sejam encontrados mais facilmente na *internet* (GREGO, 2002).

A figura 2 mostra o funcionamento simplificado de um *Web Service*.

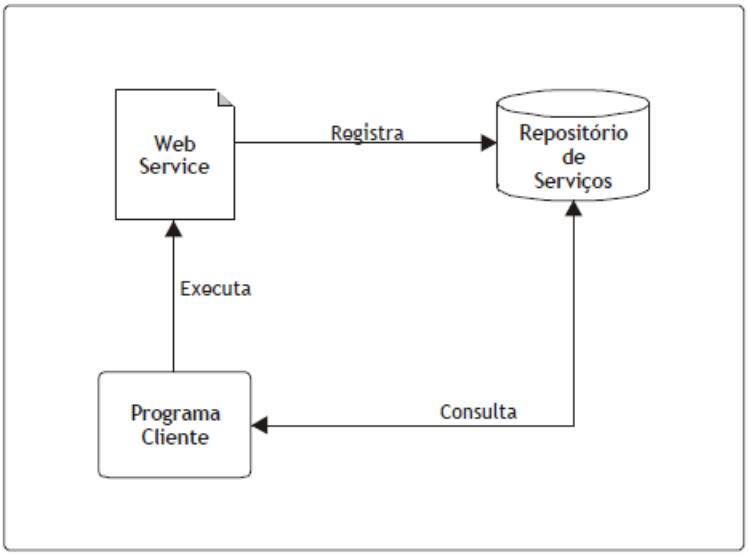


Figura 2 - Funcionamento simplificado de um Web Service (AYALA et al., 2002)

O *Web Service* pode se registrar em um repositório central, onde *softwares* clientes que necessitam dos serviços oferecidos pelo Web Service poderão pesquisá-lo. Uma vez localizado, o *software* cliente receberá uma referência para o serviço encontrado, podendo usufruir dos serviços apenas executando os vários métodos implementados com a ajuda das interfaces públicas. Assim, cada serviço precisa publicar sua interface para que os *softwares* clientes a utilizem.

Também é importante ressaltar que essa troca de informações será efetuada de forma segura, o que significa que esse modelo poderá ser utilizado na esfera

comercial. Para assegurar a troca dos dados, deverão ser utilizadas técnicas como o HTTPS (*Secure Hyper Text Transfer Protocol*), que apresenta o uso de HTTP (*Hyper Text Transfer Protocol*) encapsulado sobre SSL (*Secure Sockets Layer*) permitindo o envio e recebimento de mensagens cifradas. Também será utilizada uma autenticação prévia, permitindo o acesso ao *Web Service* somente a sistemas credenciados.

Nesta pesquisa foi desenvolvida uma aplicação utilizando *Web Services* e as arquiteturas de *software* orientadas a serviço, que são responsáveis pela publicação e localização dos serviços, tendo em vista que esta tecnologia tem sido muito utilizada devido ás vantagens quanto à adequação a ambientes heterogêneos (multiplataformas) como a *internet* e permite que os códigos possam ser reutilizados (BARALE, 2007; ALONSO et al., 2004; RODRIGUES, 2008 e SILVA, 2007). Para o desenvolvimento do aplicativo será utilizada a tecnologia Java por prover algumas facilidades no desenvolvimento de *softwares* ou sistemas e também por ser *open-source*.

A linguagem de programação Java apresenta uma série de características que a torna muito atrativa para o desenvolvimento de aplicações. Essas características abrangem a orientação a objetos, independência da plataforma, robustez, segurança, confiabilidade e conectividade com a *web* (LEMAY; PERKINS, 1996 e DEITEL, 2003). Isto proporciona o desenvolvimento de projetos com maior qualidade, portabilidade e com menor tempo de desenvolvimento.

1.1 PROBLEMATIZAÇÃO

A partir do fenômeno da *interne*t, no início dos anos noventa, que a computação distribuída passou a ter relevância definitiva, a ponto de a *internet* tornar-se a plataforma almejada pela maioria dos desenvolvedores de *software*. Em outras palavras, os projetos de Software passaram a incluir requisitos tais como acesso via WWW a sistemas corporativos legados, desenvolvimento de aplicações tipo cliente/servidor sobre o TCP/IP (*Transmission Control Protocol / Internet Protocol*) que é o protocolo padrão da *internet.* Adicionalmente, a proliferação da computação distribuída (ou dos sistemas distribuídos) sobre a *internet* trouxe consigo a necessidade crescente de comunicação entre ambientes

computacionais heterogêneos. Todavia, a comunicação entre aplicativos desenvolvidos com tecnologias heterogêneos apresentava diversas dificuldades resultantes do fato de que estes não foram originalmente projetados para trabalhar em conjunto, a exemplo da heterogeneidade de sistemas operacionais, protocolos, equipamentos e banco de dados.

Várias plataformas comerciais e experimentais desenvolvidas, principalmente nos anos noventa, resolvem boa parte desses problemas: CORBA (*Common Object Request Broker Architeture*), DCOM (*Distributed Component Object Model*), JAVA RMI (*Remote Method Invocation*), etc.

No entanto, nenhuma dessas infraestruturas de desenvolvimento e gerenciamento de *software* distribuído visava primordialmente o ambiente *internet*, que é reconhecidamente o grande desafio dos dias atuais. Para atender a essa demanda, nos anos dois mil surgiram os chamados *Web Services* (Serviços Web) e as arquiteturas de software orientadas a serviço (SOA – *Service Oriented Architecture*). Por utilizarem a infraestrutura já disponível para a WWW (como o protocolo HTTP - *Hyper Text Transfer Protocol*), amplamente usada para armazenamento e recuperação de páginas *web*, e também por incorporarem os conceitos de *middlewares* distribuídos. Assim como o CORBA, os serviços *web* nasceram com o intuito de possibilitar uma integração e disponibilização global das computações distribuídas.

Neste projeto de pesquisa será abordado o problema de integração de *softwares* utilizando a tecnologia de *Web Services* e a Arquitetura Orientada a Serviços (SOA). Será utilizada para a modelagem do problema uma arquitetura SOA baseada em *Web Services*, como mostra a figura 3.

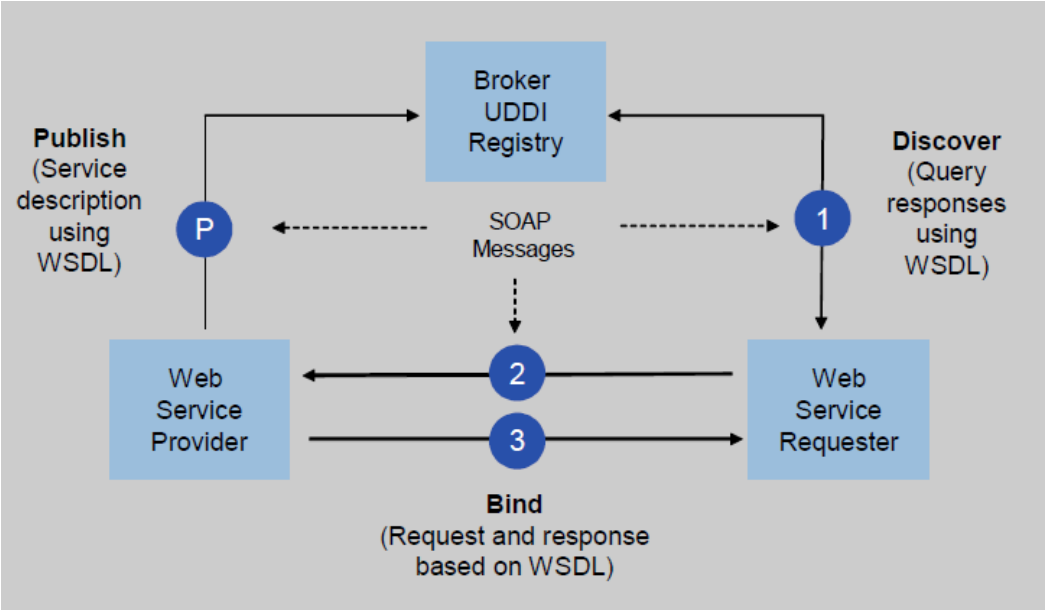


Figura 3 - Arquitetura SOA baseada em Web Services (WIEHLER, 2004)

A arquitetura de *Web Services* é composta pelos seguintes elementos:

- **WSDL** (*Web Services Description Language*) para descrever os serviços;
- **SOAP** (*Simple Object Access Protocol*) utilizado para publicar, localizar e invocar um Web Services em um registro;
- **UDDI** (*Universal Description, Discovery and Integration*), um registro que é acessado por clientes para localizar os serviços de que necessitem.

Na comunicação de dados e na interligação em rede, o protocolo utilizado é um padrão que especifica o formato de dados e as regras a serem seguidas. Sem protocolos, uma rede não funciona. Um protocolo especifica como um programa deve preparar os dados para serem enviados para o estágio seguinte do processo de comunicação.

1.2 OBJETIVOS

Este trabalho tem como objetivo apresentar um conjunto de tecnologias, chamadas *Web Services* (serviços *web*), para suporte a aplicações orientadas a serviços. Essa tecnologia está sendo muito utilizada devido às suas vantagens, por exemplo, a sua adequação a ambientes heterogêneos (multiplataformas) como a internet. Outra vantagem é que permite que os códigos possam ser reutilizados.

Para desenvolver alguma aplicação utilizando serviços *web*, é necessário conhecer outras tecnologias que compõem a arquitetura em quatro camadas: XML (*Extensible Markup Language*), SOAP (*Simple Object Application Protocol*), WSDL (*Web Service Definition Language*) e UDDI (*Universal Discovery Description Integration*), que são responsáveis pela publicação e localização dos serviços (MCGOVERN et al., 2003; GRAHAM et al., 2001; HANSEN, 2007).

1.2.1 OBJETIVO GERAL

Desenvolver um aplicativo utilizando a tecnologia de Arquitetura Orientada a Serviço baseado em *Web Services*. A linguagem de programação utilizada para a implementação do aplicativo é Java, com o servidor de aplicações *web* Apache Tomcat e banco de dados PostgreSQL. Será utilizado neste projeto o aplicativo desenvolvido no programa de iniciação cientifica 2011 como um estudo de caso. O aplicativo desenvolvido foi integração de um aplicativo *web* em Java com o Google Maps API (DE ASSIS, 2011; DE ASSIS e NITTO, 2011).

1.2.2 OBJETIVOS ESPECÍFICOS

- Adquirir conhecimentos sobre Web-Services;
- Adquirir conhecimento sobre Arquitetura Orientada a Serviços (SOA);
- Adquirir conhecimento em programação *web*;
- Modelagem e desenvolvimento do Web Services;
- Implementação dos algoritmos do Web Services;
- Implementação do aplicativo;
- Criação da base de dados;

- Integração entre tecnologias;
- Validar e testar o modelo de comunicação entre as tecnologias;
- Divulgação de resultados parciais e finais.

1.3 RELEVÂNCIA OU JUSTIFICATIVA

O desenvolvimento deste projeto tem grande relevância, tendo em vista que os serviços *web* têm facilitado em muito a vida dos desenvolvedores de aplicações *web* devido à integração que elas proporcionam.

Além disso, o crescimento da *internet* como o maior canal de comunicação e tráfego de informações aponta para um mercado promissor, e o desenvolvimento de *softwares* e aplicativos que utilizam essa tecnologia torna se uma experiência única e também preparar melhor para o mercado de trabalho. O desenvolvimento de Web Services orientado a serviços é uma realidade nos dias atuais, e a tendência é que eles continuem no mercado por um bom tempo devido a sua grande capacidade de integrar diferentes tecnologias.

Esta tecnologia é recente e para preencher esta lacuna de mercado é necessário pesquisar e desenvolver novas arquiteturas de *softwares* baseadas em *Web Services* para coletar dados sobre a sua usabilidade.

2 FUNDAMENTAÇÃO TEÓRICA BÁSICA

Nesta sessão serão apresentados os conceitos que norteiam esse projeto de pesquisa. A princípio um breve resumo sobre Web Services, em seguida será abordado o conceito da tecnologia XML que é a base para a padronização dos Web Services dentro desse aspecto também se enquadram os conceito sobre WSDL (*Web Service Description Language*), SOAP (*Simple Object Access Protocol*) e UDDI (*Universal Description, Discovery, and Integration*). Também será necessário abordar os modelos de comunicação para Web Services que são: RPC(*Remote Procedure Call*) e SOA(*Service Oriented Architecture).*

Dentro dessa sessão ainda serão abordados os conceitos das tecnologias de desenvolvimento que serão utilizadas como a linguagem Java o Servidor de Aplicações Web Apache Tomcat e SGBD PostgreSQL.

2.1 Web Services

Serviços da Web fornecem um meio padrão de interoperabilidade entre diferentes aplicações de software, rodando em uma variedade de plataformas e / ou frameworks (ENDEL Q, 2004).

Um serviço Web é um sistema de software projetado para suportar interação máquina-máquina interoperável sobre uma rede. Ele tem uma interface descrita em um formato processável por computador (especificamente WSDL). Outros sistemas interagem com o serviço Web de uma maneira prescrita por sua descrição usando mensagens SOAP, tipicamente transmitidas utilizando HTTP com uma serialização XML em conjunto com outros relacionados a padrões Web (ENDEL K, 2012).

Serviços *web* representam parte da lógica de negócio, executando em sistemas remotos que os hospedam e os mantém distribuídos na rede. Eles podem ser acessados através de protocolos padronizados da internet como HTTP (*ENDEL B,* 2012).

A figura 4 abaixo mostra um modelo de comunicação usando SOAP sobre o protocolo HTTP.

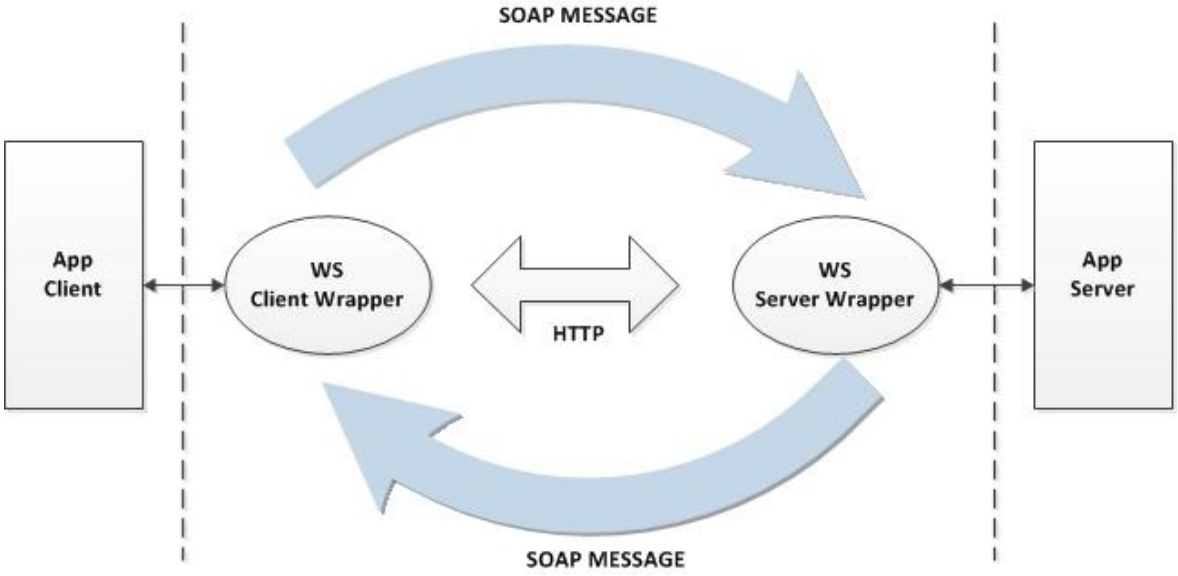


Figura 4 - Modelo de Comunicação SOAP sobre protocolo HTTP (ENDEL O, 2003)

O processo de comunicação baseada em padrões permite que aplicações diferentes se comuniquem através dos protocolos utilizados pelos Web Services sem que haja necessidade de se conhecer detalhes de implementação.

De acordo com (ENDEL C, 2012) as principais características dos serviços *Web* são:

- **Baseado em XML:** usado para representar os dados. Como transporte de dados, XML (*eXtensible Markup Language*) elimina qualquer dependência com rede e sistema operacional;
- **Fracamente acoplado:** a interface de um serviço *Web* pode mudar durante o tempo sem comprometer a habilidade do cliente de interagir com o serviço;

- **Granularidade grossa:** provê uma maneira natural de definir serviços de granularidade grossa que acessam a quantidade correta de lógica de negócio;
- **Chamadas síncronas e assíncronas:** um cliente pode invocá-lo de forma síncrona e assíncrona. Possibilitar chamadas assíncronas é a chave para permitir sistemas fracamente acoplados;
- **Suporta Remote Procedure Calls (RPCs):** serviços web permitem que os clientes invoquem procedimentos, funções e métodos em objetos remotos usando um protocolo baseado em XML. Procedimentos remotos expõem entrada e saída de parâmetros que um serviço web deve suportar. Desenvolvimento de componentes através de Enterprise JavaBeans (EJBs) e Componentes. Tem se tornado uma parte de arquiteturas e implementações empresariais nos últimos anos. Ambas as tecnologias são distribuídas e acessíveis através de uma variedade de mecanismos de RPC. Um serviço de web suporta RPC pela prestação de serviços próprios, equivalentes às de um componente tradicional, ou, traduzindo invocações de entrada em uma invocação de um EJB ou um componente. NET;
- **Suporta a troca de documentos:** uma das principais vantagens do XML é a sua maneira genérica de representar não apenas dados, mas também documentos complexos. Estes documentos podem ser simples, como quando o que representa um endereço atual, ou eles podem ser complexas, o que representa um livro inteiro ou PDO. Serviços *Web* apoiam o intercâmbio transparente de documentos para facilitar a integração de negócios.

Algumas das características de um *Web Service* são descritas por (ENDEL C, 2012):

- Está disponível através da Internet ou privada (*intranet*) redes;
- Usa um sistema de mensagens XML padronizadas;
- Não está vinculado a qualquer outro sistema operacional ou uma linguagem de programação;
- É auto descritivo através de uma gramática XML comum;

- É detectável através de um mecanismo simples encontra.

Uma plataforma básica de *Web Service* trabalha com o protocolo HTTP em conjunto com XML, utilizando os seguintes componentes:

- **SOAP** (*Simple Object Access Protocol*)
- **UDDI** (*Universal Description, Discovery and Integration*)
- **WSDL** (*Web Services Description Language*)

A figura 5 abaixo ilustra a arquitetura de um *Web Service* composta por suas três entidades essenciais: Provedor de Serviços, Consumidor de Serviços e Catálogo de Serviços (UDDI). As entidades interagem entre si:

1 - Publicar

2 e **3 -** Localizar

4 e **5 –** Ligar

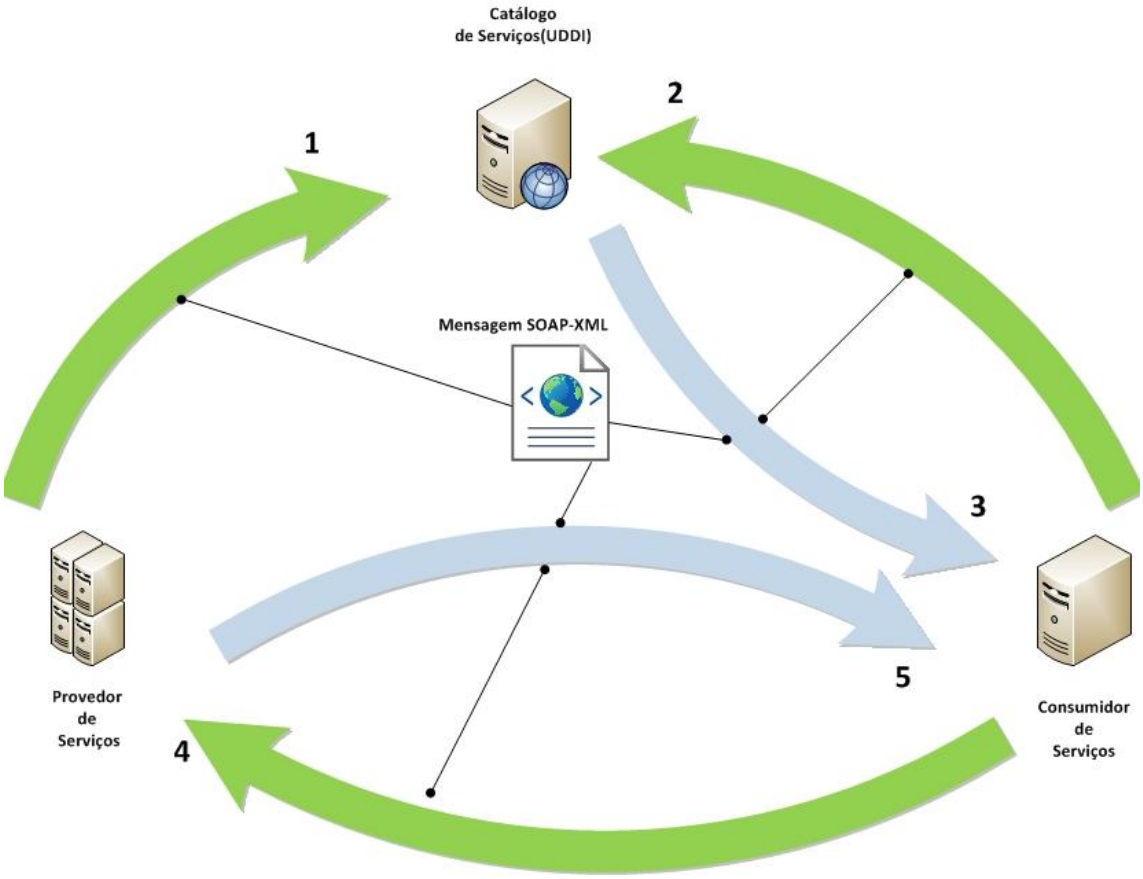


Figura 5 - Arquitetura de um *Web Service* (ENDEL D, 2012)

2.1.1 Arquitetura dos *Web Services*

A arquitetura orientada a serviços (SOA) fornece o modelo teórico para todos os serviços *web* (ENDEL R, 2004). A figura 6 mostra um modelo que contém três entidades e três operações.

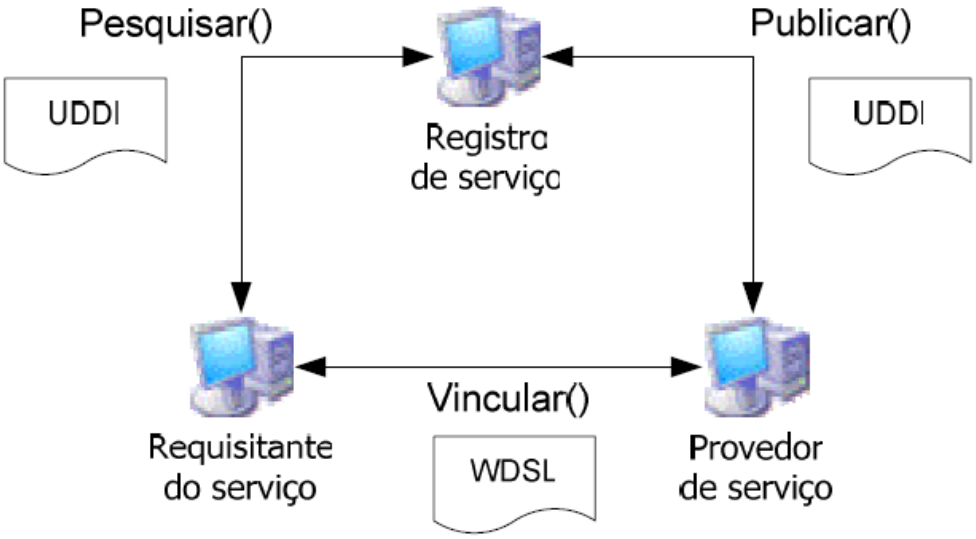


Figura 6 - Três componentes básicos utilizados pelos serviços web (CESCONETI e GLAZAR, 2006)

O Provedor de serviços é onde estão os serviços *web*. O primeiro passo é publicar as informações desses serviços, detalhes de classificação, detalhes de conexão e outras informações, no registro dos serviços. Essas informações são especificadas pela UDDI. Nesse passo, o provedor de serviços também gera o documento WSDL com os detalhes dos serviços, formas de conexão, métodos, parâmetros e tipo de retorno. O documento WSDL será usado pelo requisitante de serviços para se vincular ao provedor de serviços.

O registro de serviços é uma aplicação que retorna informações sobre os serviços *web* em resposta a uma requisição que tenha sido submetida por um requisitante de serviços na operação Pesquisar(). Essas informações (UDDI) retornam os detalhes de contato para os serviços *web*, com base em suas classificações que correspondem ao critério de pesquisa. Esse registro também inclui informações sobre como descobrir os detalhes de conexão.

O requisitante de serviços é uma aplicação que deseja utilizar os serviços publicados pelo provedor. Ele não sabe onde estão os serviços, nem como encontrá-los; então, recorre ao registro de serviços e requisita a operação pesquisar. De posse da UDDI, consegue descobrir onde estão os serviços, então se conecta ao provedor e obtém o documento WSDL, com os detalhes dos serviços. Com a aplicação pronta pode se vincular aos serviços no provedor.

Segundo (ENDEL R et al., 2004), um dos objetivos do uso da SOA é prover o desenvolvimento de aplicações distribuídas pela composição rápida e de baixo custo de componentes de *software* autônomos e independentes de plataforma.

Os serviços *web*, para fazerem a comunicação entre as aplicações, utilizam quatro camadas que empacotam a requisição e a resposta entre cliente e servidor. As camadas utilizadas são:

- *Extensible Markup Language* (XML);
- *Simple Object Application Protocol* (SOAP);
- *Web Service Definition Language* (WSDL);
- *Universal Discovery Description Integration* (UDDI).

2.2 Arquitetura Orientada a Serviços

A definição de Arquitetura Orientada a Serviços, ou simplesmente SOA, é um pouco complexa, uma vez que envolvem os conceitos de arquitetura e de serviço, conceitos esses que também não são claros, tanto na literatura acadêmica quanto na literatura não acadêmica. A Arquitetura Orientada a Serviços pode ser entendida como um estilo arquitetural para a construção de sistemas baseados em componentes modularizados, autônomos e fracamente acoplados,

denominados serviços. Cada serviço expõe processos e comportamentos, através de contratos, que são compostos de mensagens em endereços detectáveis chamados terminais. O comportamento dos serviços obedece a uma política, externa ao próprio serviço. A figura 7 mostra um diagrama SOA.

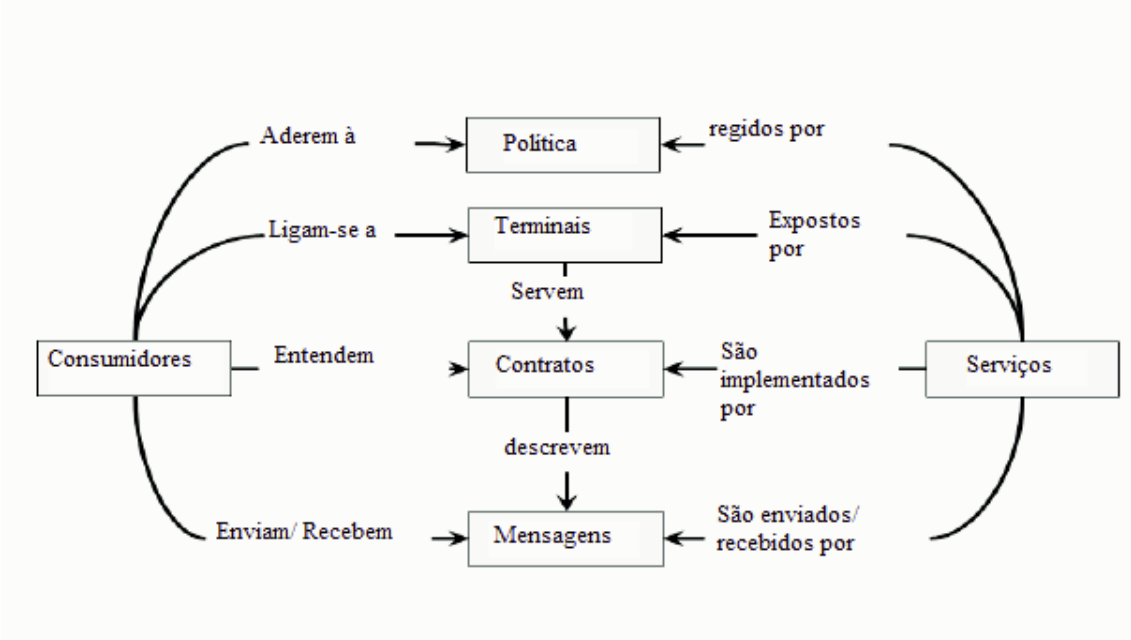


Figura 7 - Diagrama SOA (FRONDANA et al., 2009)

As descrições do diagrama segundo (FRONDANA et al., 2009) são:

Serviço: é o pilar central de SOA. O serviço é uma facilidade que atende a uma necessidade, e deve promover alta coesão e distinta função de negócio. Serviço deve ser lógica modularizada, e deve implementar no mínimo toda a funcionalidade prometida pelos contratos expostos. Uma de suas características é a autonomia. Autonomia implica em auto-suficiência, em certa medida, e possuir manutenibilidade própria.

Contrato: o conjunto de todas as mensagens suportadas pelo serviço é conhecido como o contrato do serviço. O contrato pode ser unilateral, onde ele é um conjunto fechado de mensagens que o serviço fornece. Ele também pode ser multilateral ou bilateral, ou seja, com um grupo predefinido de participantes. O contrato pode ser considerado a interface do Serviço.

Terminal: é um endereço, um local específico onde o serviço pode ser encontrado. Um contrato específico pode ser exposto em um terminal específico.

Mensagem: é a unidade de comunicação de SOA. Mensagens podem vir em diferentes modos e formatos, por instâncias, mensagens do tipo http GET, mensagens tipo *Simple Object Access Protocol*, mensagens *Java Message Service* ou até mesmo mensagens de e-mail são formatos válidos de mensagem.

Politica: uma importante diferença entre a Orientação a Objetos e SOA é a existência de uma política. A política separa a especificação dinâmica da especificação semântica, e representa as condições da disponibilidade da especificação semântica para os consumidores do serviço. Características únicas da política são a de poder ser atualizada em tempo de execução, e ser externa à lógica do negócio. Ela especifica propriedades dinâmicas como segurança (criptografia e autenticação, por exemplo), auditoria, dentre demais outros itens.

Consumidores de Serviço: Um serviço não tem muito sentido se não houver algo ou alguém que se beneficie dele. Então é necessária a existência do Consumidor do Serviço, ou simplesmente, Consumidor. O Consumidor é qualquer programa que interaja com o serviço, trocando mensagens com ele. Consumidores podem ser tanto aplicações do cliente como serviços “vizinhos”, cujo único requisito seja aderir ao contrato do serviço.

A figura 8 mostra a arquitetura SOA.

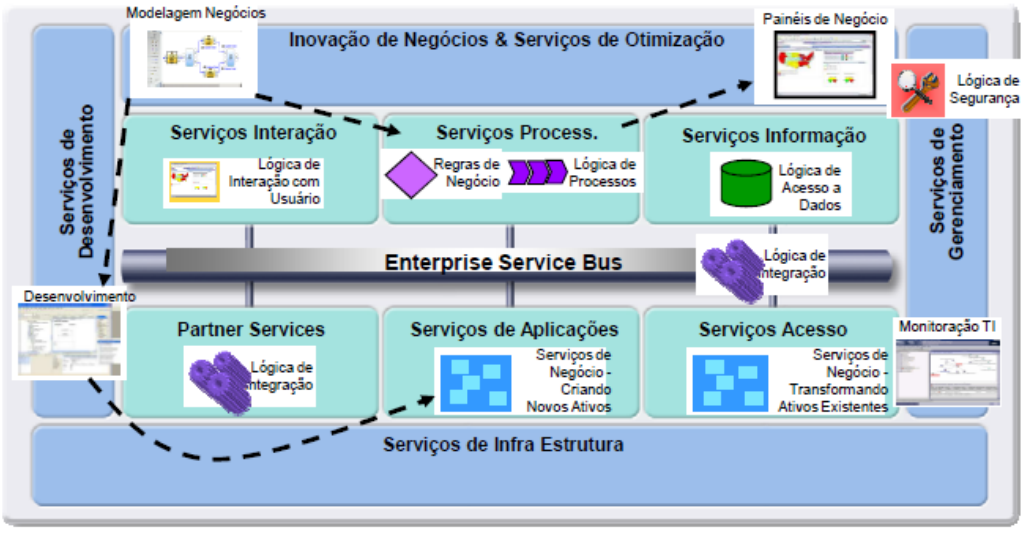


Figura 8 - Arquitetura SOA (SILVA e SIROTHEAU, 2006)

A figura 9 mostra a visão da arquitetura com SOA.

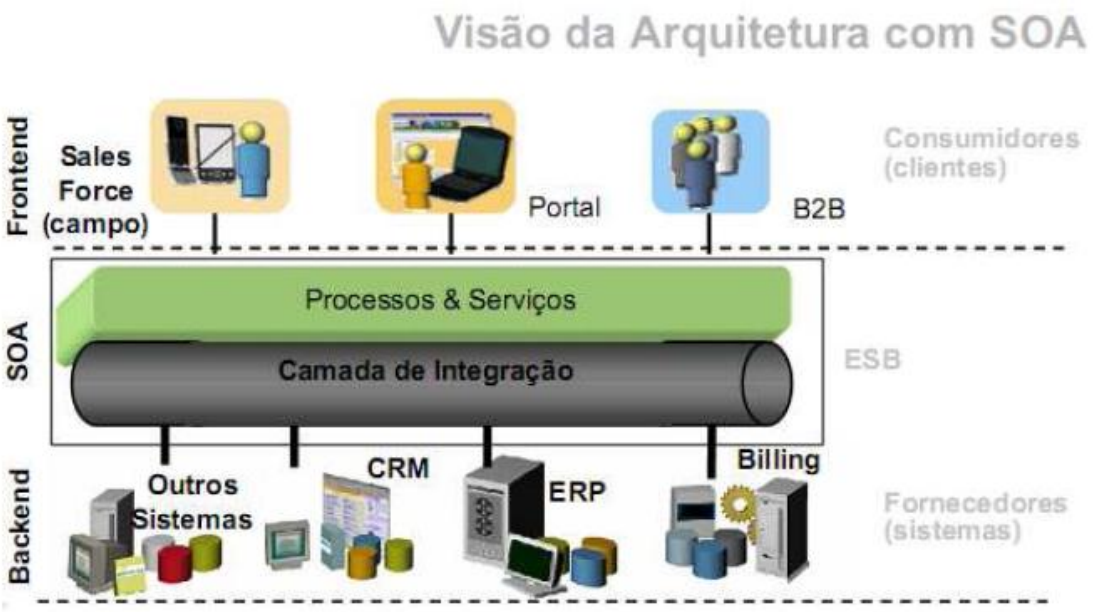


Figura 9 - Visão da arquitetura com SOA (SILVA e SIROTHEAU, 2006)

Esta arquitetura mostra a camada *Backend*, SOA e *Frontend*.

2.2.1 Benefícios Web Services

- **Expondo a função existente sobre a rede:**

Web Service permite expor a funcionalidade do código na rede. Uma vez que é exposto na rede, outro aplicativo pode usar a funcionalidade do seu programa.

- **Interoperabilidade**

Permite que aplicações diferentes possam interagir entre si. Por exemplo, aplicações VB ou. NET podem “falar” com os serviços Java *web* e vice versa.

- **Protocolo padronizado**

Web Services utiliza o protocolo padrão da indústria a comunicação. Esta padronização da pilha de protocolos dá ao negócio muitas vantagens como

ampla gama de opções, a redução no custo devido à concorrência e aumento da qualidade.

- **Baixo custo de comunicação**

Web Services usam SOAP sobre o protocolo HTTP para a comunicação, assim pode-se usar a internet (custo relativamente baixo) para implentação dos *Web Services*.

2.2.2 Arquitetura Web Services

Existem duas maneiras de se observar a arquitetura de um *Web Service*, examinando o papel de cada serviço dentro da arquitetura ou examinado a pilha de protocolos emergentes do *Web Service*.

2.2.3 Web Service Roles (Papéis)

Dentro da arquitetura de um *Web Service* existem três papéis que são muito importantes:

- **Prestador de serviços:**
  Este é o prestador do serviço web. O prestador de serviços implementa o serviço e o torna disponível na Internet.

- **Serviço solicitante**
  Este é qualquer consumidor do serviço web. O solicitante utiliza um serviço de web já existente, abrindo uma conexão de rede e enviando um pedido XML.
- **Serviço de registro**
  Este é um diretório lógico centralizador de serviços. O registro fornece um local central onde os desenvolvedores podem publicar novos serviços ou encontrar os já existentes. Portanto, serve como uma agência centralizadora para as empresas e seus serviços. (ENDEL C, 2012).

2.2.4 *Web Service Stack Protocols* (Pilha de Protocolos)

Examinando a pilha de protocolos também podemos observar a arquitetura do *Web Service*. A pilha ainda está evoluindo, mas atualmente tem quatro camadas principais:

- **Serviço de transporte**

  Esta camada é responsável pelo transporte de mensagens entre aplicações. Atualmente, esta camada inclui *Hypertext Transfer Protocol* (HTTP), *Simple Mail Transfer Protocol* (SMTP), *File Transfer Protocol* (FTP), e os protocolos mais recentes, como *Blocks Extensible Exchange Protocol* (BEEP).

- **Mensagens XML**

  Esta camada é responsável pela codificação de mensagens em um formato XML comum para que as mensagens possam ser compreendidas em cada extremidade. Atualmente, esta camada inclui XML-RPC e SOAP.

- **Descrição do serviço**

  Esta camada é responsável por descrever a interface pública de um serviço de web específicos. Atualmente, a descrição do serviço é feita através do *Web Service Description Language* (WSDL).

- **Descoberta de serviços**

  Esta camada é responsável pela centralização dos serviços em um registro comum, e oferece uma funcionalidade fácil para publicar ou encontrar um WS . Atualmente, o serviço de descoberta é feita via *Universal Description, Discovery, and Integration* (UDDI).

Como serviços web tendem a evoluir, camadas adicionais podem surgir, e tecnologias também podem ser adicionadas a cada camada. (ENDEL C, 2012).

2.2.5 Meios de Comunicação Web Services

Existem dois modelos de comunicação para serviços *Web*. São eles:

- **Remote Procedure Call (RPC):** representa uma chamada de método distribuída, neste caso a unidade básica é a operação definida no WSDL. Este método é criticado por não ser fracamente acoplado, pois é freqüentemente implementado mapeando serviços diretamente para chamadas de método especificas. Esta forma geralmente é a mais usada.
- **Arquitetura Orientada a Serviço (*Service Oriented Architecture* - SOA):** a unidade básica de comunicação é a mensagem. Este estilo também é conhecido como “serviços orientados a mensagem”. Este estilo é mais fracamente acoplado, pois o foco está no “contrato” que o WSDL fornece (ENDEL D, 2012)

2.3 XML (eXtensible Markup Language)

Extensible Markup Language (XML) é um simples formato de texto muito flexível derivado do SGML (ISO 8879). Originalmente concebido para enfrentar os desafios de grande escala a publicação eletrônica, XML também está jogando um papel cada vez mais importante na troca de uma ampla variedade de dados na Web e em outros lugares (ENDEL E, 2012).

Algumas das principais características da linguagem XML são:

- XML é uma linguagem de marcação muito parecida com HTML;
- XML foi projetado para transportar dados, não para exibir dados;
- Tags XML não são predefinidas. Você deve definir suas próprias tags;
- XML foi projetado para ser auto-descritivo.

XML é usado em muitos aspectos do desenvolvimento web, muitas vezes, para simplificar o armazenamento de dados e compartilhamento. (ENDEL F, 2012)

O XML é baseado em padrões de tecnologia comprovadamente otimizados para a Web. Os padrões que compõem o XML são definidos pelo W3C (*World Wide Web Consortium*) e são os seguintes:

- ***Extensible Markup Language* (XML)**: é uma Recomendação, que é vista como o último estágio de aprovação do W3C. Isso significa que o padrão é estável e pode ser aplicado à Web e utilizado pelos desenvolvedores de ferramentas.
- **XML *Namespaces*:** é também uma Recomendação, a qual descreve a sintaxe de namespace, ou espaço de nomes, e que serve para criar prefixos para os nomes de tags, evitando confusões que possam surgir com nomes iguais para tags que definem dados diferentes.
- ***Document Object Model* (DOM) Level 1 -** é uma Recomendação que provê formas de acesso aos dados estruturados utilizando scripts, permitindo aos desenvolvedores interagir e computar tais dados consistentemente.
- ***Extensible Stylesheet Language* (XSL)-** é atualmente um rascunho. O XSL apresenta duas seções: a linguagem de transformação e a formatação de objetos. A linguagem de transformação pode ser usada para transformar documentos XML em algo agradável para ser visto, assim como transformar para documentos HTML, e pode ser usada independentemente da segunda seção (formatação de objetos). O Cascade Style Sheet (CSS) pode ser usado para XML simplesmente estruturado, mas não pode apresentar informações em uma ordem diferente de como ela foi recebida.
- **XML *Linking Language* (XLL)** e **XML *Pointer Language* (XPointer)** - são também rascunhos. O XLL é uma linguagem de construção de links que é similar aos links HTML, sendo que é mais poderosa, porque os links podem ser multidirecionais, e podem existir á nível de objetos, e não somente á nível de página. (ENDEL L, 2012).

A figura 10 mostra um exemplo de uma estrutura de arquivo XML:

XML Document Example

```
<?xml version="1.0"?>
<note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

Figura 10 - XML *Document Example* (ENDEL F, 2012)

No mundo real, sistemas de computação e bancos de dados contêm dados em formatos incompatíveis, a vantagem da utilização de XML é que através dessa padronização ele independe de padrões de *hardware* ou *software* para armazenar os dados, tornando muito mais fácil o compartilhamento desses dados entre aplicações diferentes inclusive no que se diz respeito a linguagem de programação utilizada para o desenvolvimento (ENDEL F, 2012).

2.4 WSDL (*Web Services Description Language*)

WSDL é um formato XML para descrever serviços de rede como um conjunto de terminais que operam em mensagens contendo um ou outro documento orientado ou procedimento orientado informações. As operações e mensagens são descritas abstratamente, e em seguida ligado a um protocolo de rede concreto e formato de mensagem para definir um endpoint. (ENDEL A, 2012).

Um documento WSDL define serviços como coleções de terminais de rede, ou portas. De acordo com (ENDEL M, 2012), alguns dos elementos encontrados na definição dos serviços de rede de um documento WSDL são:

- **Type** - um recipiente para definições de tipo de dados usando algum tipo de sistema (como XSD).

- **Message** - uma definição, resumo digitado dos dados sendo comunicados.

- **Operation** - uma descrição abstrata de uma ação suportada pelo serviço.

- **Port-Type** um conjunto abstrato de operações suportadas por um ou mais endpoints.

- **Binding** - um protocolo concreto e especificação de formato de dados para um tipo de porta particular.

- **Port** - um único terminal definido como uma combinação de uma ligação e um endereço de rede.

- **Service** - uma coleção de endpoints relacionados.

A figura 11 a seguir mostra os principais que elementos que compõem o WSDL.

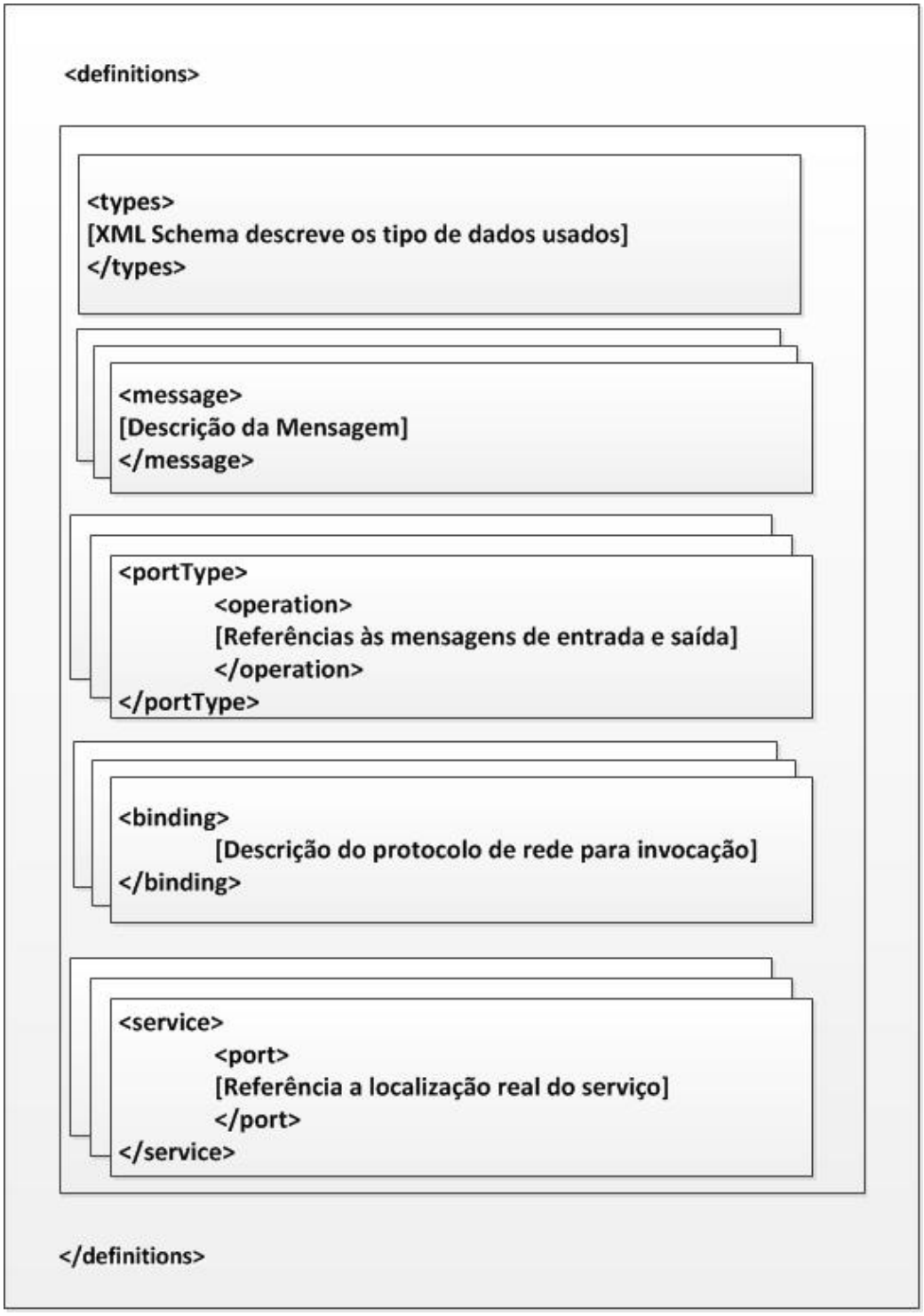


Figura 11 - Arquitetura WSDL (ENDEL M, 2012)

WSDL tem quatro primitivas de transmissão que um terminal pode suportar:

- **One-way:** O endpoint recebe uma mensagem.
- **Request-response:** O endpoint recebe uma mensagem e envia uma mensagem de resposta.
- **Solicit-response:** O endpoint envia uma mensagem, e aguarda uma mensagem de resposta
- **Notification:** O endpoint envia uma mensagem, porém sem aguardar a resposta.

Resumidamente podemos dizer que WSDL é um formato XML que permite que os serviços sejam descritos, usado para gerar código para o cliente, para o servidor e para configuração.

2.5 SOAP (*Simple Object Access Protocol*)

SOAP é um protocolo baseado em XML simples que permiti aos aplicativos trocar informações sobre o protocolo HTTP, ou simplesmente SOAP é um protocolo utilizado para acessar Web Service (ENDEL G, 2012 e ENDEL S, 2012).

SOAP é um protocolo para comunicação que encapsula os dados transferidos no formato XML. O protocolo SOAP estende o XML para que aplicações clientes possam enviar facilmente parâmetros para aplicações servidoras e então receber e entender o documento XML retornado. Graças à simplicidade de um arquivo XML (texto puro), dados são facilmente trafegados entre sistemas de computadores com arquiteturas e formatos de dados heterogêneos. O transporte é realizado pelos protocolos: HTTP e HTTPS, também é possível usar outros protocolos como SMTP e XMPP. (ENDEL D, 2012; OASIS, 2006 e ENDEL N, 2005).

Algumas das principais características do protocolo SOAP:

- SOAP é um protocolo de comunicação;
- SOAP é para a comunicação entre aplicações;
- SOAP é um formato para envio de mensagens;
- SOAP comunica via Internet;
- SOAP é independente de plataforma;
- SOAP é independente de linguagem;
- SOAP é baseado em XML;
- SOAP é simples e extensível;
- SOAP permite contornar firewalls.

O protocolo SOAP é extremamente importante para o desenvolvimento de aplicações que necessitam se comunicar pela Internet, foi desenvolvido especificamente para se comunicar através do protocolo HTTP (Hypertext Transfer Protocol), protocolo que é suportado por todos os navegadores.

Portanto, SOAP fornece um meio para a comunicação entre aplicações rodando em diferentes sistemas operacionais, com diferentes tecnologias e linguagens de programação (ENDEL G, 2012).

2.5.1 Sintax SOAP

De acordo com (ENDEL G, 2012), uma mensagem SOAP é um documento XML contendo os seguintes elementos:

- **Um elemento *Envelope***, que identifica o documento XML como uma mensagem SOAP (elemento raiz do documento);
- **Um elemento de *Header***, que contém informações de cabeçalho;
- **Um elemento de *Body***, que contém informações de resposta e de chamada;

- **Um elemento *Fault***, contém erros e informações de status

A figura 12 abaixo mostra a estrutura de um documento SOAP.

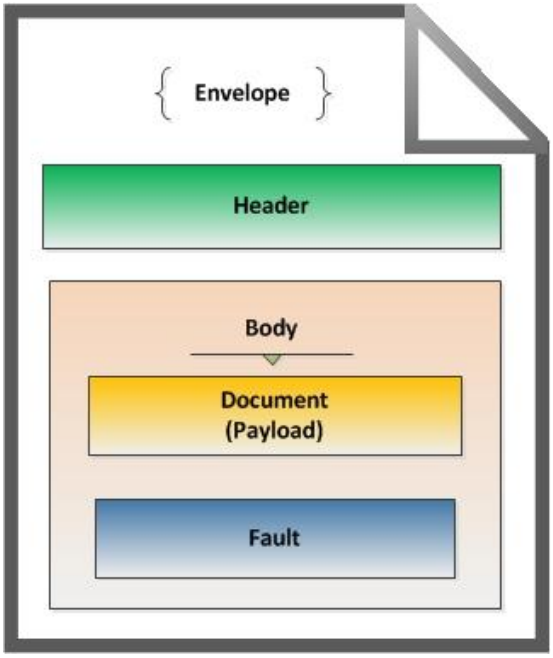


Figura 12 - Estrutura de um documento SOAP (ENDEL G, 2012)

A figura 13 traz o esqueleto de uma mensagem SOAP.

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
...
</soap:Header>

<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Figura 13 - Esqueleto de mensagem Soap (ENDEL G, 2012)

2.6 UDDI (*Universal Description, Discovery and Integration*)

UDDI é um catálogo de serviços para publicar e descobrir metadados sobre serviços *Web*, permitindo que aplicações descubram nos tanto em tempo de projeto quanto de execução. Uma vez que um serviço *Web* é definido usando WSDL, é necessária alguma forma de torná-lo conhecido para utilização. Isso é feito através do registro UDDI (*Universal Description Discovery and Integration*) que fornece métodos para publicar e encontrar descrições de serviços. Dessa forma, provedores de serviços podem publicar a existência de seus serviços para que potenciais usuários os encontrem. O registro UDDI armazena uma descrição do serviço (conforme o tipo de negócio, dividindo por categorias e organizado hierarquicamente) e a localização do mesmo (*binding*) (ENDEL D, 2012).

UDDI é uma estrutura independente de plataforma para descrever serviços, descobrindo negócios e integração dos serviços de negócios usando a Internet.

De acordo com (GRAHAM et al., 2001) algumas das características do UDDI são:

- UDDI é um diretório para armazenar informações sobre os serviços web;
- UDDI é um diretório de interfaces web serviço descrito pelo WSDL;
- UDDI comunica via SOAP;
- UDDI é construído no Microsoft. NET.

Antes da criação do UDDI não havia um padrão na Internet para as empresas divulgarem seus produtos e serviços, tampouco um método para se integrar a outros sistemas e processos.

UDDI é um esforço inter profissional conduzido por todas as principais plataformas e fornecedores de software como a Dell, Fujitsu, HP, Hitachi, IBM, Intel, Microsoft, Oracle, SAP e Sun, assim como uma grande comunidade de operadores de mercado, e ex-líderes empresariais (ENDEL A, 2012).

UDDI é composto por duas partes:

- Um registro de todos os metadados de serviço uma web, incluindo um ponteiro para a descrição WSDL de um serviço;
- Um conjunto de definições de tipo de porta WSDL para manipular e procurar que o registro.

Um negócio ou empresa pode registrar três tipos de informações em um registro UDDI. Esta informação está contida em três elementos de UDDI. Esses três elementos são:

- **Páginas Brancas:**

Esta categoria contém:

- Informações básicas sobre a Companhia e seus negócios;
- Informações básicas de contato, incluindo nome, endereço, número de telefone de contato etc;
- Identificadores únicos para os IDs fiscais da empresa. Esta informação permite que os outros possam descobrir o seu serviço web baseado em sua identificação de negócio.

- **Páginas Amarelas:**

Esta categoria contém:

- Este tem mais detalhes sobre a empresa, e inclui descrições do tipo de recursos eletrônicos, a empresa pode oferecer para quem quer fazer negócios com ele;
- Comumente aceita esquemas de categorização industrial, códigos da indústria, códigos de produtos, códigos de identificação de negócios e gostaria de torná-lo mais fácil para as empresas a busca através das listas e encontrar exatamente o que eles querem.

- **Páginas Verdes:**

Esta categoria contém informações técnicas sobre um serviço web. Isto é o que permite que alguém possa ligar-se a um serviço de Web depois que foi encontrado. Isto inclui:

  - As várias interfaces;
  - Os locais de URL;
  - Descoberta de informações e dados semelhantes necessários para encontrar e executar o serviço Web.

2.6.1 UDDI Arquitetura Técnica

A arquitetura técnica do UDDI é dívida em três partes:

- **UDDI modelo de dados:**
  Um esquema XML para descrever negócios e serviços web. O modelo de dados é descrito em detalhes em "Dados UDDI Model" seção.

- **UDDI API Specification:**
  A Especificação da API para pesquisa e publicação de dados UDDI.

- **UDDI cloud serviços:**
  Este são sites de operadores que fornecem implementações da especificação UDDI e sincronizam todos os dados em uma base programada.

A figura 14 traz a arquitetura técnica do UDDI.

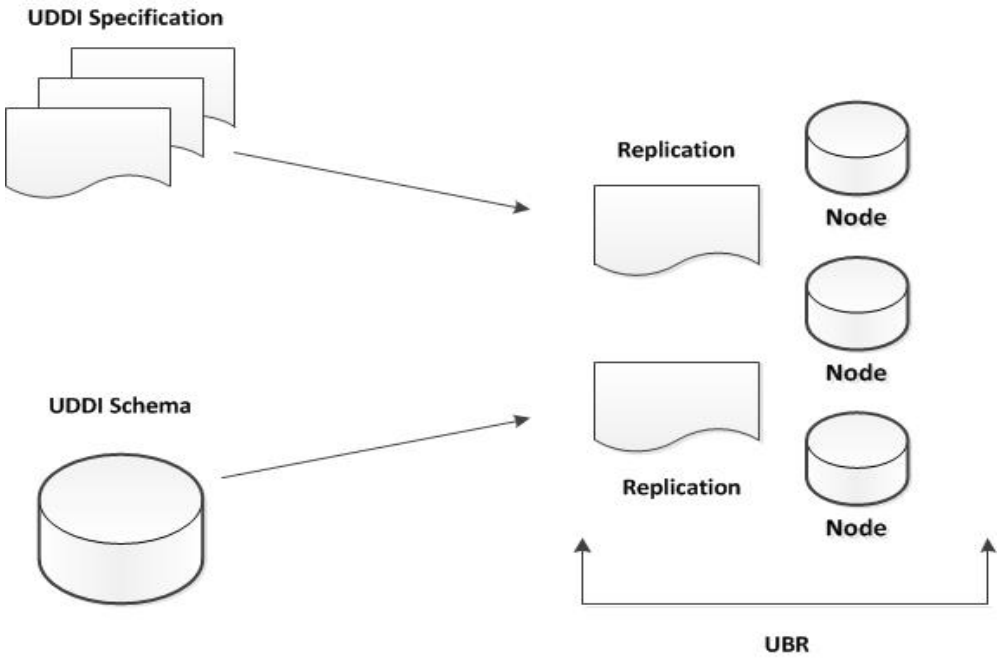


Figura 14 - Arquitetura UDDI (ENDEL H, 2012)

O UDDI Registry Negócios (UBR), também conhecida como a nuvem pública, é um sistema conceitualmente único construído a partir de vários nós que tem seus dados sincronizados através da replicação. (ENDEL H, 2012).

2.7 Tecnologias Java

Aplicações *Web* estão cada vez mais presentes e seu desenvolvimento representa boa parte da produção de organizações desenvolvedoras de *software* bem como de mídia em geral. O rápido crescimento das aplicações *Web*, tanto em seu escopo quanto na extensão de seu uso, tem afetado todos os aspectos de nossas vidas (GINIGE e MURUGESAN, 2000). Por representar uma evolução do software convencional, algumas preocupações adicionais motivaram as pesquisas relacionadas à engenharia de aplicações *Web* (*Web Engineering*), mantendo o objetivo de aplicar princípios de engenharia para desenvolver aplicações *Web* de qualidade (PRESSMAN, 2002). Com isso, a tecnologia Java ganhou espaço, pois foi projetada para se mover através de redes de dispositivos heterogêneos, como

a *Internet*. Agora as aplicações podem ser executadas dentro dos *Browsers* nos *Applets* Java e tudo seria disponibilizado pela *Internet* instantaneamente. Foi o estático HTML dos *Browsers* que promoveu a rápida disseminação da dinâmica tecnologia Java.

O berço da linguagem Java é um projeto da *Sun Microsystems* de 1991, chamado de Projeto Green, que tinha como mentores Patrick Naughton, Mike Sheridan, e James Gosling. Era um projeto sobre *software* para produtos eletrônicos de consumo. O objetivo era que os eletrônicos tivessem um *software* embarcado que possibilitasse o seu controle via rede de computadores e uma maior interatividade com o usuário (DEITEL; DEITEL, 2005, SILVEIRA; COSENTINO, 2009 e ENDEL T, 2010).

2.8 Apache Tomcat

O software Tomcat, desenvolvido pela Fundação Apache, permite a execução de aplicações para web. Sua principal característica técnica é estar centrada na linguagem de programação Java, mais especificamente nas tecnologias de *Servlets* e de *Java Server Pages* (JSP). Esta abordagem rivaliza, por exemplo, com a usada pela Microsoft com o ASP (baseada na linguagem Visual Basic).

A Fundação Apache, mais conhecida pelo seu servidor web de mesmo nome, permite, como no caso do servidor Apache, que o Tomcat seja usado livremente, seja para fins comerciais ou não.

O Tomcat está escrito em Java e, por isso, necessita que a versão Java 2 Standard Edition (J2SE) esteja instalada no mesmo computador onde ele será executado. No entanto, não basta ter a versão runtime de Java instalada, pois o Tomcat necessita compilar (e não apenas executar) programas escritos em Java. O projeto Jakarta da Fundação Apache, do qual o subprojeto Tomcat é o representante mais ilustre, tem como objetivo o desenvolvimento de soluções código aberto baseadas na plataforma Java. (ENDEL I, 2012).

2.9 PostgreSQL

O PostgreSQL é um banco de dados de nível corporativo com a vantagem de ser gratuito. O PostgreSQL dá total suporte a programação Java, pois trabalha com o *driver* JDBC (*Java Database Connectivity*) que permite que programas Java possam se conectar a um banco de dados PostgreSQL utilizando todos os recursos disponíveis pelo SGBD.

O PostgreSQL é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. Tem mais de 15 anos de desenvolvimento ativo e uma arquitetura que comprovadamente ganhou forte reputação de confiabilidade, integridade de dados e conformidade a padrões. Roda em todos os grandes sistemas operacionais, incluindo GNU/Linux, Unix (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), e MS Windows. (ENDEL J, 2012).

2.10 Integração de Um Aplicativo Web em Java Com o Google Maps API

Este projeto foi realizado no programa de iniciação cientifica em 2011, onde foi desenvolvido um aplicativo *web* que se integra com a ferramenta Google Maps. A maior dificuldade para esse tipo de problema é determinar uma solução de fácil implementação e rapidez na resposta. O aplicativo *web* visa disponibilizar um serviço diferenciado para a área educacional e neste projeto os *markers* são dinâmicos, onde são inseridas informações e imagens da escola. *Markers* são figuras que marcam um determinado ponto no mapa. Foi incluída também no aplicativo a parte de gerenciamento de uma das escolas, para que no futuro todas as outras escolas sejam integradas no sistema. O aplicativo é um sistema *web* com um servidor de serviços web, que será desenvolvido em JSF (*Java Server Faces*) com comunicação XML com o Servidor da Google através da Google Maps API para manipulação de dados espaciais e com o banco de dados PostgreSQL. A figura 15 mostra a visão geral do sistema.

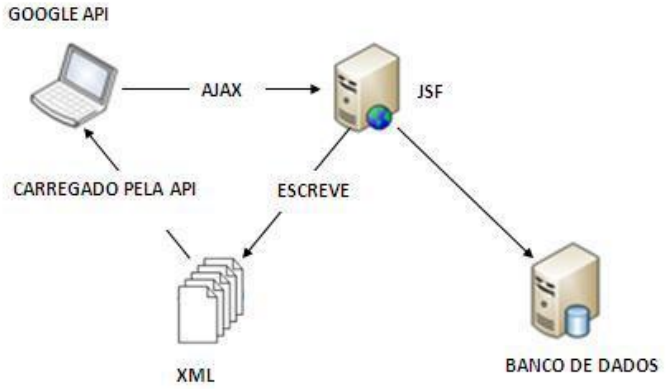


Figura 15 - Visão geral do sistema (DE ASSIS, 2011)

A figura 16 mostra a localização de uma escola de musica cadastrada no sistema no mapa.

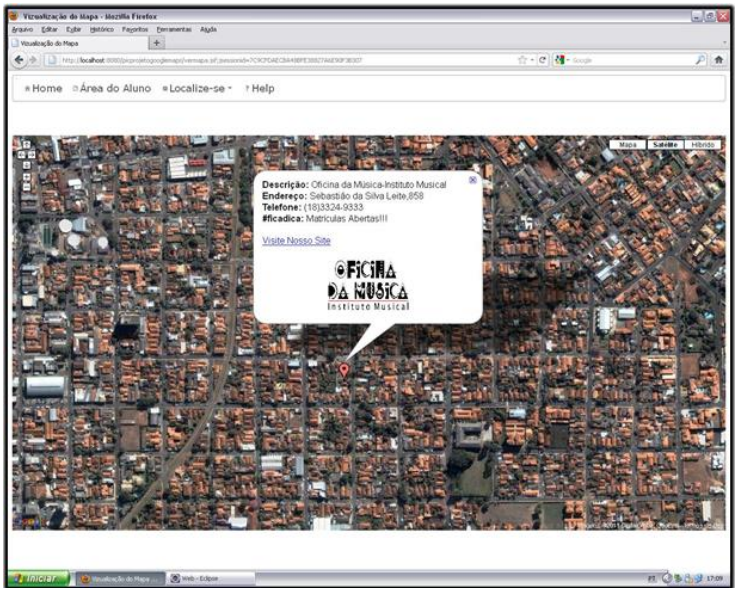


Figura 16 -Visualização de marcador no mapa (DE ASSIS, 2011)

Mais detalhes podem ser vistos em (DE ASSIS, 2011; DE ASSIS e NITTO, 2011).

3 DESENVOLVIMENTO DO PROJETO

Neste capítulo serão descritas todas as ações pertinentes e necessárias ao desenvolvimento dos aplicativos para a exemplificação de todo o processo de criação e consumo de um *web service*.

3.1 Descrição do Problema

O aplicativo utilizado para cadastro dos dados que serão posteriormente "consumidos" pelos serviços *web* foi baseado no desenvolvimento resultante do Projeto de Iniciação Científica do ano de 2011: Integração de um Aplicativo Web em Java com a API Google Maps (DE ASSIS, 2011; DE ASSIS e NITTO, 2011). O aplicativo de cadastro tem como objetivo obter dados de locais como: hospitais, restaurantes, empresas, juntamente com dados pertinentes onde a latitude e longitude também é recuperada através do Google Maps.

Para a demonstração do funcionamento de uma arquitetura de software com base em *web services* foram necessários o desenvolvimento de dois aplicativos *web*, um aplicativo servidor (*server*) que disponibiliza os serviços e um aplicativo cliente (*client*) responsável pelos consumo dos serviços.

A figura 17 mostra uma visão geral do sistema para consumo dos *web services*.

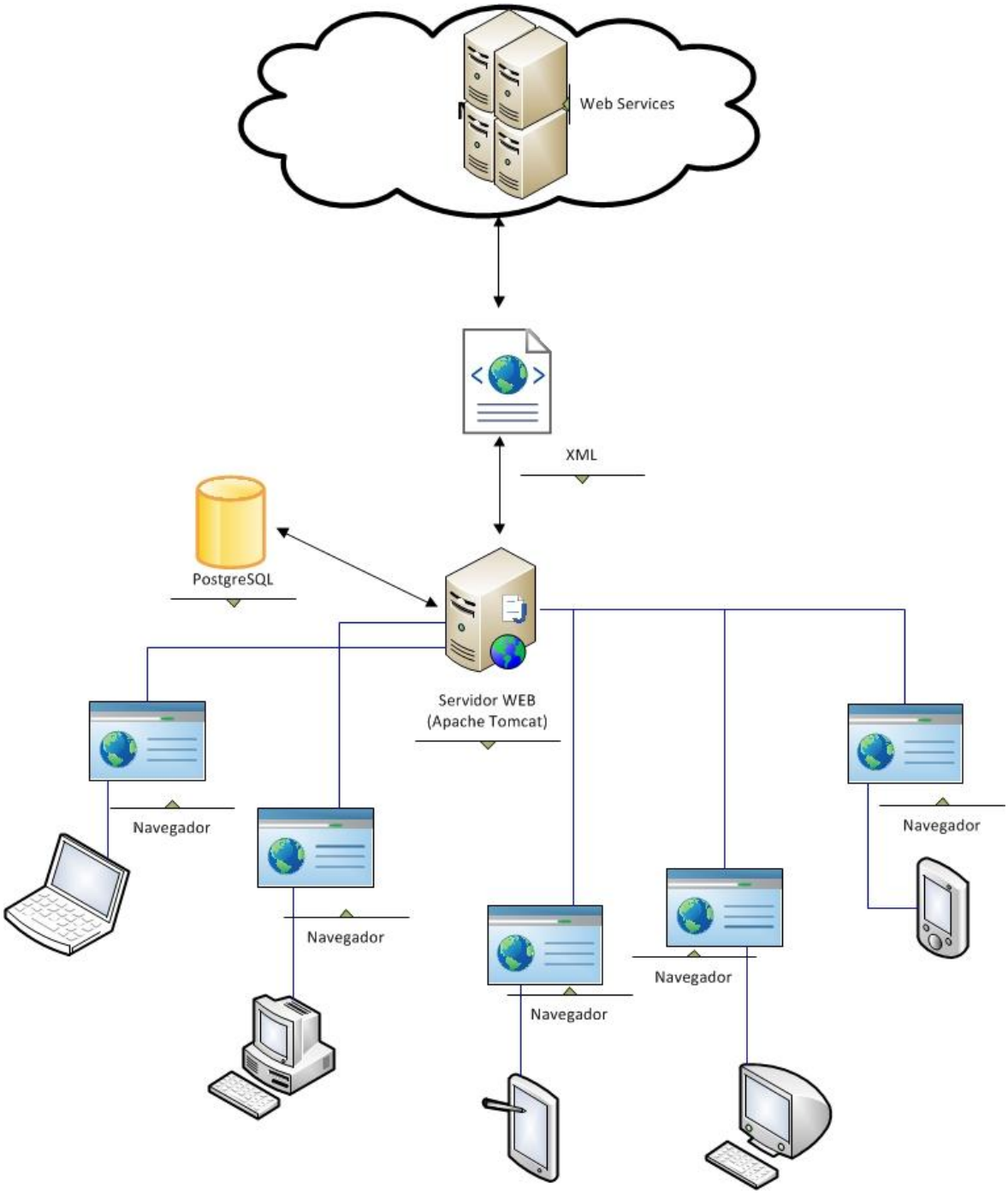


Figura 17 - Visão geral do sistema para consumo dos *web services*

Os serviços podem ser consumidos através de qualquer aparelho (celular, smartphone, computador) que possuam acesso à internet de maneira transparente. Os dados e serviços ficam armazenados em servidores remotos. As especificações serão apresentadas em diagramas utilizados para a modelagem do sistema.

3.2 Diagrama de Casos de Uso (Use Case)

Os casos de uso bem como outros diagramas que serão apresentados serão divididos sempre em três partes: a que se refere ao aplicativo de cadastro, ao aplicativo servidor e ao aplicativo cliente.

3.2.1 Diagrama de Casos de Uso (Aplicativo Cadastro)

O aplicativo de cadastro possui somente um caso de uso, onde o administrador do sistema tem como objetivo injetar dados na base. A figura 18 mostra o diagrama de caso de uso para a aplicação cadastro.

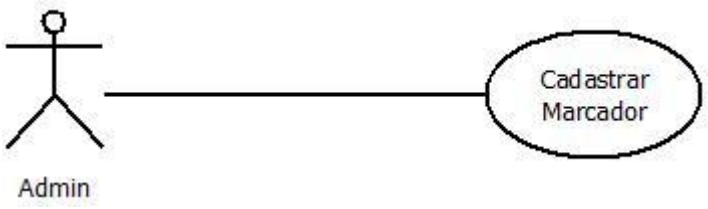


Figura 18 - Diagrama de Caso de Uso (Aplicativo Cadastro)

As descrições do caso de uso são:

- **Cadastrar Marcador**: caso de uso responsável pela inserção de dados na base, através de formulário exibido através de um click no mapa.

3.2.2 Diagrama de Casos de Uso (Aplicativo Servidor)

O aplicativo servidor possui quatro casos de uso, que são justamente os serviços disponibilizados pela aplicação. A figura 19 mostra o diagrama de caso de uso para a aplicação servidor.

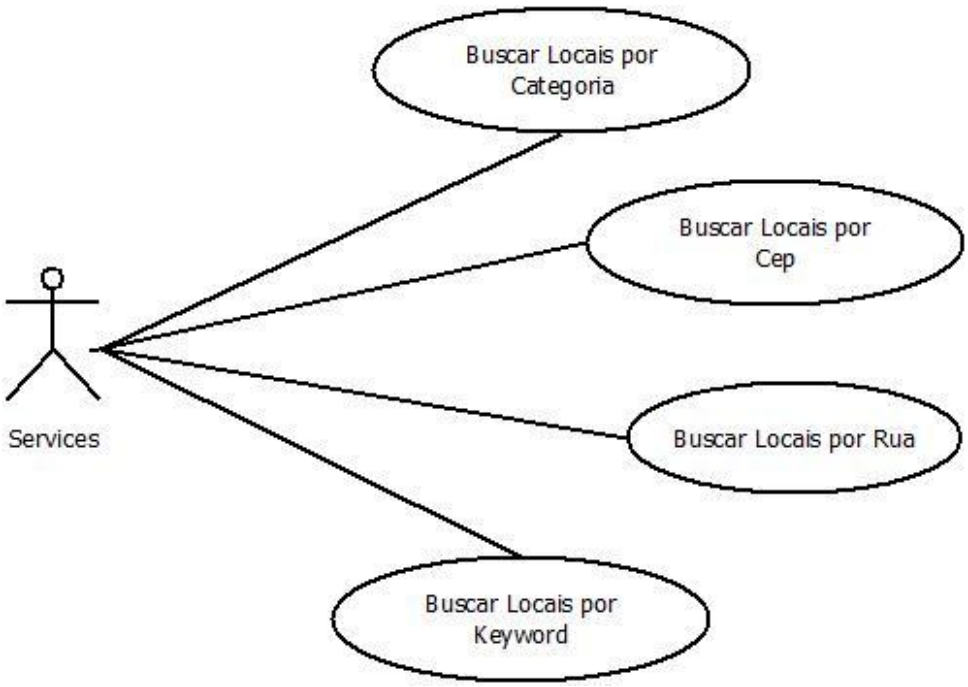


Figura 19 - Diagrama de Caso de Uso (Aplicativo Servidor)

As descrições dos casos de uso são:

- **Buscar Locais por Categoria:** caso de uso que recebe uma categoria como parâmetro para filtro e retorna uma lista de locais, caso haja itens correspondentes;
- **Buscar Locais por Cep:** caso de uso que recebe um cep como parâmetro para filtro e retorna uma lista de locais, caso haja itens correspondentes;

- **Buscar Locais por Rua:** caso de uso que recebe uma rua como parâmetro para filtro e retorna uma lista de locais, caso haja itens correspondentes;
- **Buscar Locais por Keyword:** caso de uso que recebe uma palavra-chave como filtro e retorna uma lista de locais, caso haja itens correspondentes.

3.2.3 Diagrama de Casos de Uso (Aplicativo Cliente)

O aplicativo cliente possui nove casos de uso, sendo que para cada caso de uso que se utilizam do recurso de visualização no mapa, há um caso de uso dependente e comum à todos. Esses casos de uso representam o consumo dos serviços disponibilizado pelo aplicativo servidor. A figura 20 mostra o diagrama de caso de uso para a aplicação cliente.

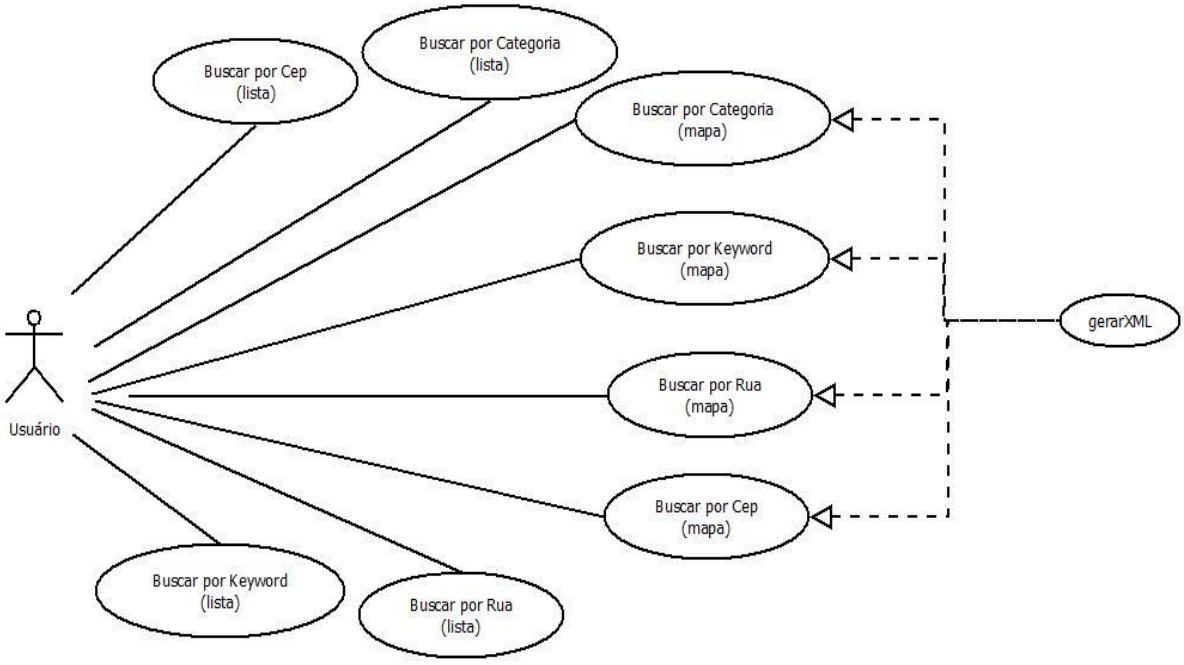


Figura 20 - Diagrama de Caso de Uso (Aplicativo Cliente)

As descrições dos casos de uso são:

- **Buscar por cep (lista):** caso de uso que recebe um cep como parâmetro para filtro e retorna uma lista de locais, caso haja itens correspondentes;
- **Buscar por categoria (lista):** caso de uso que recebe uma categoria como parâmetro para filtro e retorna uma lista de locais, caso haja itens correspondentes;
- **Buscar por Keyword (lista):** caso de uso que recebe uma palavra-chave como parâmetro para filtro e retorna uma lista de locais, caso haja itens correspondentes;
- **Buscar por Rua (lista):** caso de uso que recebe uma rua como parâmetro para filtro e retorna uma lista de locais, caso haja itens correspondentes;
- **Buscar por cep (mapa):** caso de uso que recebe um cep como parâmetro para filtro e retorna uma lista de marcadores no mapa;
- **Buscar por categoria (mapa):** caso de uso que recebe um cep como parâmetro para filtro e retorna uma lista de marcadores no mapa;
- **Buscar por Keyword (mapa):** caso de uso que recebe um cep como parâmetro para filtro e retorna uma lista de marcadores no mapa;
- **Buscar por Rua (mapa):** caso de uso que recebe um cep como parâmetro para filtro e retorna uma lista de marcadores no mapa;
- **gerarXML:** caso de uso que gera um arquivo xml, utilizado pela API Google Maps para inserir os marcadores no mapa.

3.3 Diagrama de Classe

Os diagramas de classe também serão divididos entre aplicativo cadastro, servidor e cliente, dentro dessas divisões as classes serão apresentadas dentro de pacotes tal qual organizado na aplicação.

3.3.1 Diagrama de Classe (Aplicativo Cadastro)

Serão feitas as descrições dos diagramas de classes da aplicação cadastro. O aplicativo de cadastro possui três pacotes com suas respectivas classes.

O pacote weblocalcadastro.uteis contem a classe responsável por prover a conexão com a base de dados. A figura 21 mostra o pacote weblocalcadastro.uteis.

- **weblocalcadastro.uteis**
    - **Conexão:** classe que provê a conexão com o SGBD.

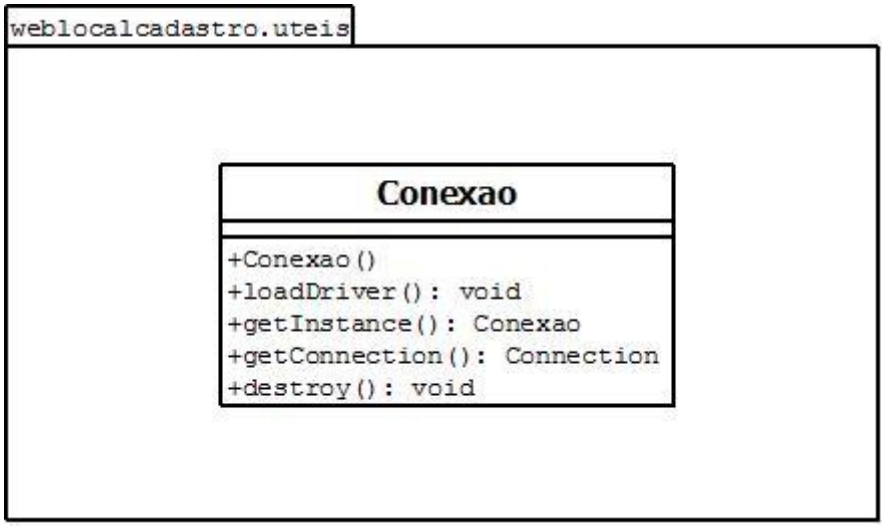


Figura 21 - Pacote weblocalcadastro.uteis

O pacote weblocalcadastro.dao contem a classe MapDao que disponibiliza o método de inserção de dados na base. A figura 22 mostra o pacote weblocalcadastro.dao.

- **weblocalcadastro.dao**
    - **MapDao:** classe que possui o método para inserção dos dados.

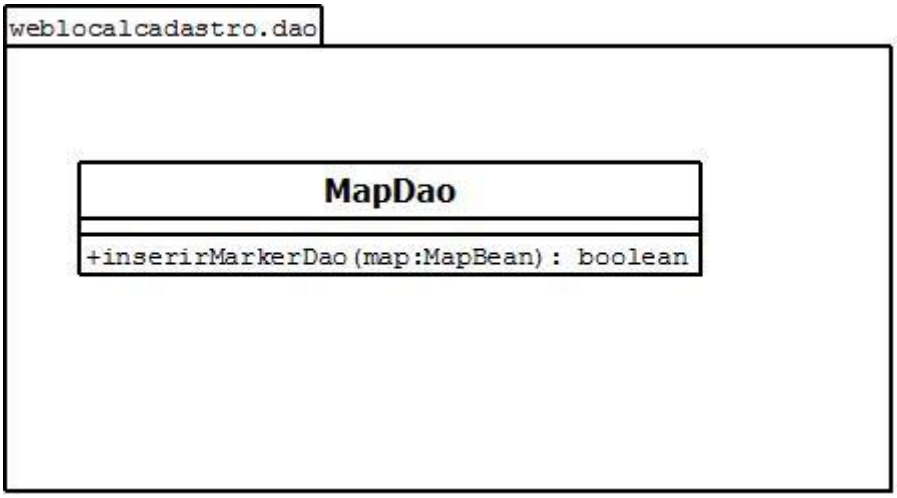


Figura 22 - Pacote weblocalcadastro.dao

O pacote weblocaldastro.bean contem a classe MapBean, responsável pela comunicação da página JSF com os objetos do tipo Marcadores.

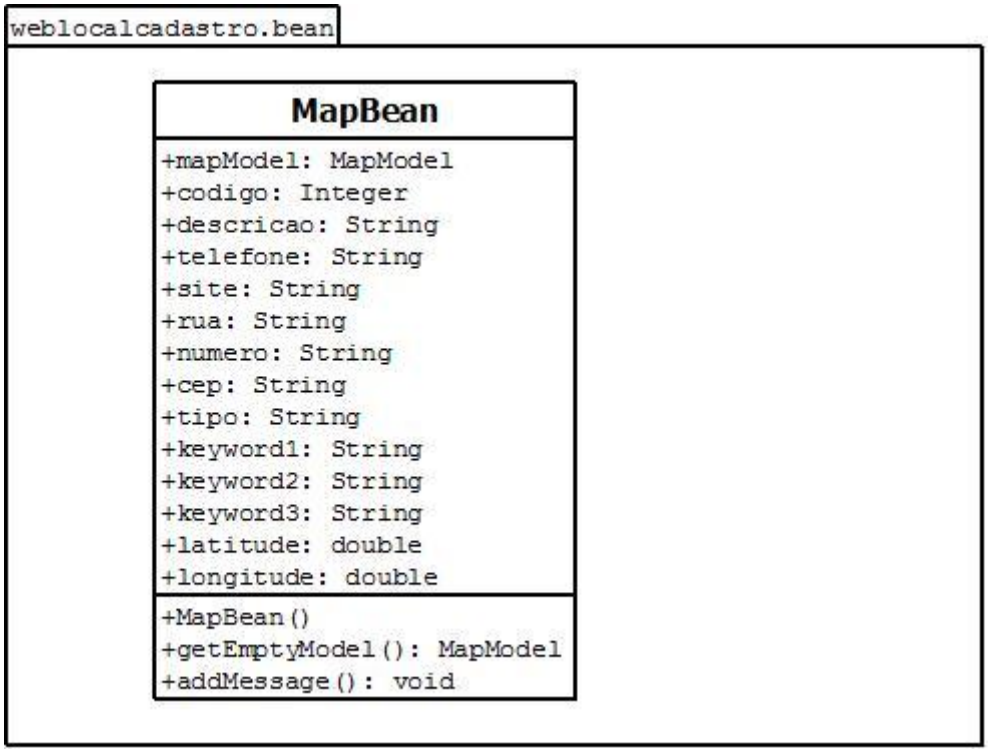


Figura 23 - Pacote weblocalcadastro.bean

- **weblocalcadastro.bean**
    - **MapBean:** classe responsável pela comunicação com a página JSF, captura dos dados que serão inseridos base de dados.

3.3.2 Diagrama de Classe (Aplicativo Servidor)

Serão feitas as descrições dos diagramas de classes da aplicação servidor. O aplicativo servidor possui cinco pacotes com suas respectivas classes.

O pacote weblocais.uteis contem a classe conexão, responsável pela conexão com a base dados. A figura 24 mostra o pacote weblocais.uteis.

- **weblocais.uteis**
    - **Conexão:** classe que contém os métodos responsáveis pela conexão com a base de dados.

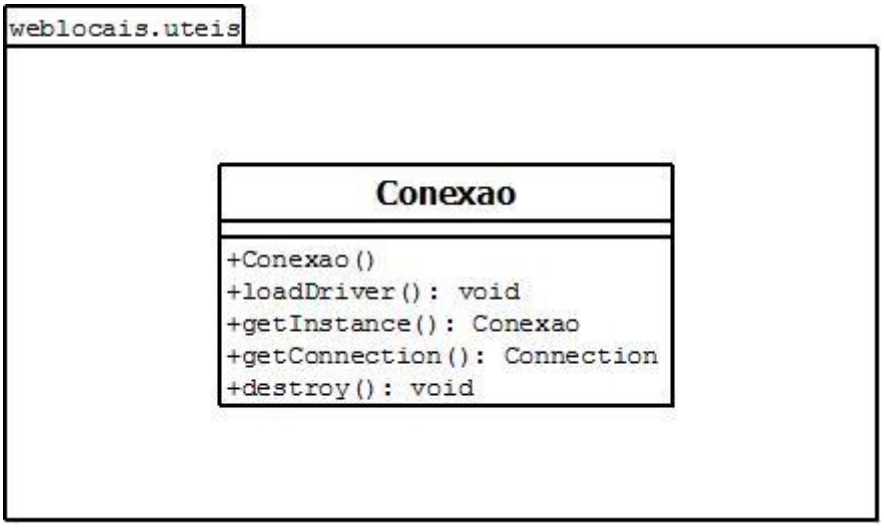


Figura 24 - Pacote weblocais.uteis

O pacote weblocais.dao contem a classe WebLocaisDao que apresenta os métodos que recuperam da base de dados os valore solicitados. A figura 25 mostra o pacote weblocais.dao.

- **weblocais.dao**
    - **WeblocaisDao:** classe que contém os métodos responsáveis pela recuperação de dados.

Figura 25 - Pacote weblocais.dao

O pacote weblocal.model contem a classe WebLocal que apresenta os atributos pertinentes aos dados, bem como seus métodos getters e setters. A figura 26 mostra o pacote weblocais.model.

- **weblocal.model**
    - **WebLocal:** classe que contém os atributos do objeto tipo WebLocal juntamente com seus métodos getters e setters.

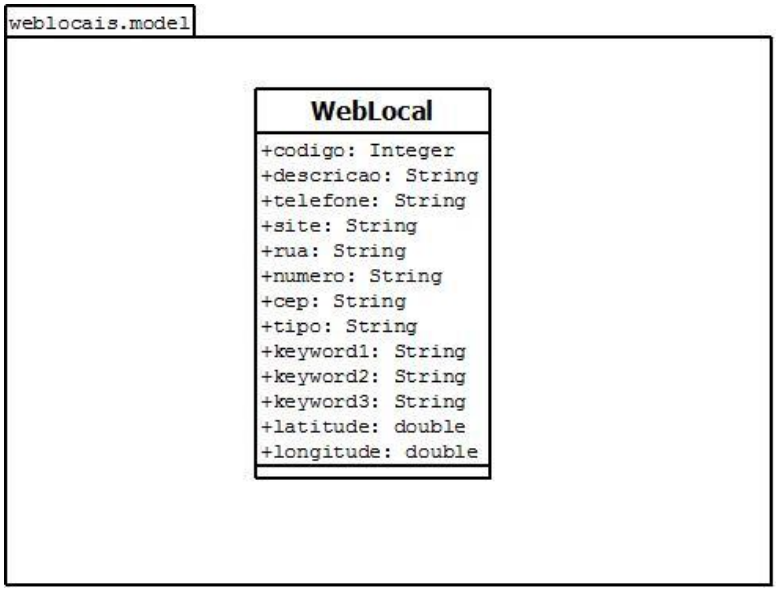


Figura 26 - Pacote weblocais.model

O pacote weblocais.services contém a classe WebServices que apresenta os métodos que são exatamente os serviços *web* disponiblizados pelo aplicativo servidor. A figura 27 mostra o pacote weblocais.services.

- **weblocal.model**
    - **WebServices:** classe que contém os métodos que através de anotações específicas da linguagem Java se tornam serviços web.

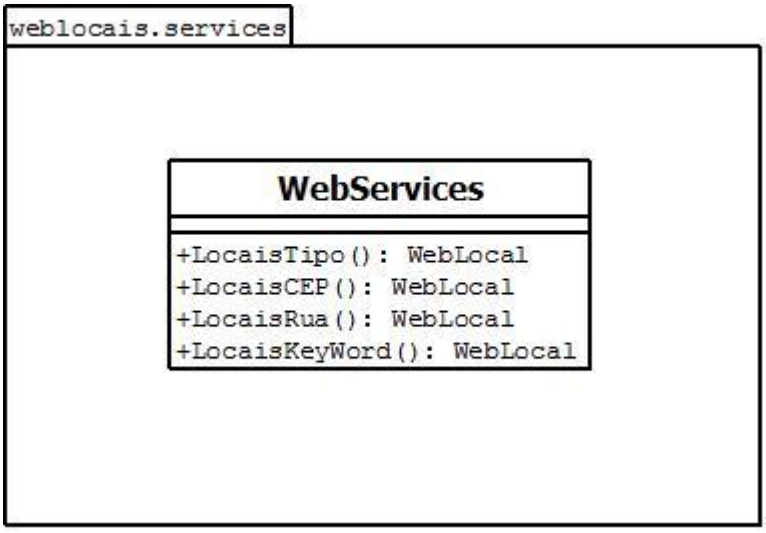


Figura 27 - Pacote weblocais.services

O pacote welocais.business contem a classe WebLocaisBusiness que abstrai os métodos do pacote weblocais.dao, facilitando uma reutilização do código e eliminado tratamentos de exceptions em excesso. A figura 28 mostra o pacote weblocal.business.

- **weblocais.business**
    - **WebLocaisBusiness:** classe que abstrai a chamada aos métodos da classe WebLocaisDao, para futura reutilização de código.

Figura 28 - Pacote weblocais.business

3.3.3 Diagrama de Classe (Aplicativo Cliente)

Serão feitas as descrições dos diagramas de classes da aplicação cliente. O aplicativo cliente possui dois pacotes acompanhados de suas respectivas classes.

O pacote weblocaiscliente.beans contem as classes: LocaisCepBean, LocaisKeywordBean, LocaisTipoBean e LocaisRuaBean. A figura 29 mostra o pacote weblocaiscliente.beans.

- **weblocaiscliente.beans**
  - **LocaisCepBean:** classe que contém os métodos responsáveis pelo consumo do serviço web que provém uma busca filtrando por cep e retorando uma lista de locais.
  - **LocaisKeywordBean:** classe que contém os métodos responsáveis pelo consumo do serviço web que provém uma busca filtrando por palavra-chave e retorando uma lista de locais.
  - **LocaisTipoBean:** classe que contém os métodos responsáveis pelo consumo do serviço web que provém uma busca filtrando por categoria/tipo e retorando uma lista de locais.

- **LocaisRuaBean:** classe que contém os métodos responsáveis pelo consumo do serviço web que provém uma busca filtrando por rua e retorando uma lista de locais.

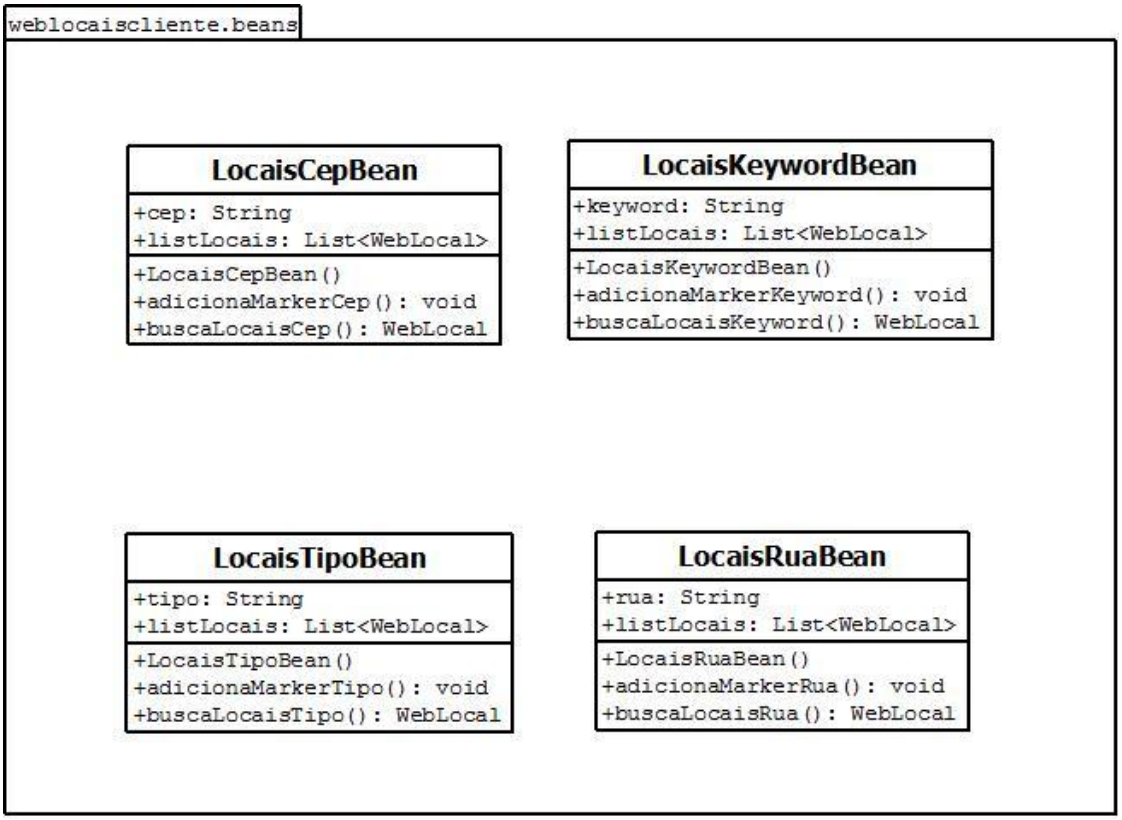


Figura 29 - Pacote weblocaiscliente.beans

O pacote weblocaiscliente.control contem a classe MarkerXML que apresenta o método responsável pela geração do arquivo XML através das listas retornas pelo consumo dos *web services*, este arquivo XML é utilizado pelo API Google Maps para a exibição dos marcadores no mapa. A figura 30 mostra o pacote weblocais.control.

- **weblocais.control**
  - **MarkerXML:** classe que contém o método responsável pela geração do arquivo XML, utilizado pela API Google Maps para exibição dos marcadores no mapa.

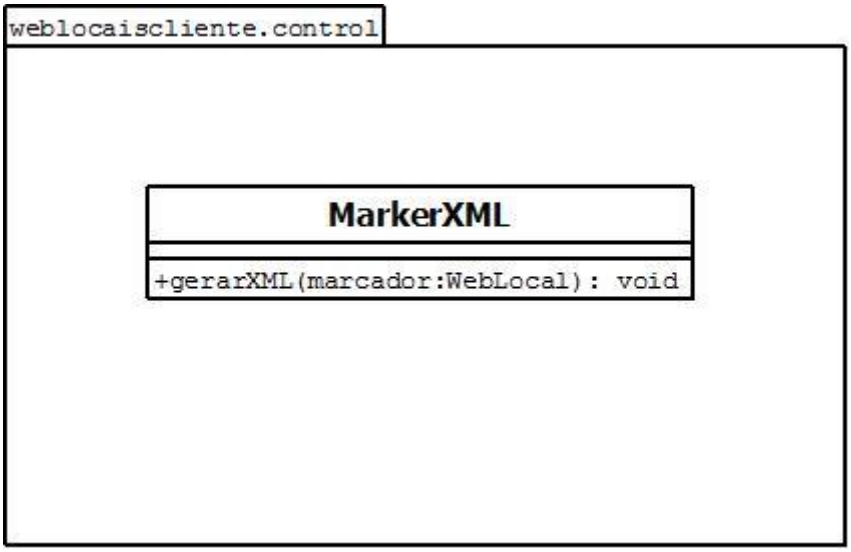


Figura 30 - Pacote weblocaiscliente.control

3.4 Diagrama de Atividades

Esse tipo de diagrama representa o controle de fluxo de dados de aplicativo, tem como característica as possíveis tomadas de decisão que possam vir a ocorrer. A figura 31 mostra o diagrama de atividades do aplicativo, representando o consumo dos serviços *webs* e os resultados que podem ser obtidos, esse diagrama representa o aplicativo cliente em funcionamento.

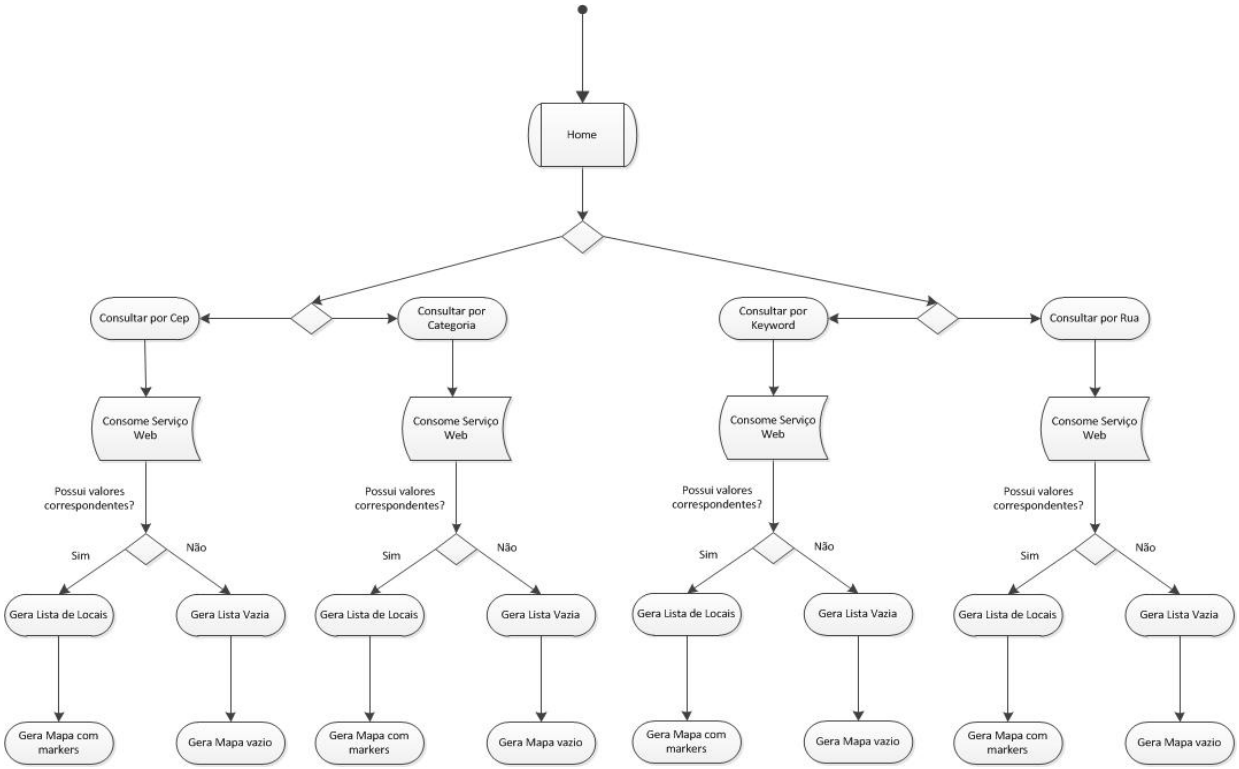


Figura 31 - Diagrama de Atividades

3.5 Diagrama Entidade-Relacionamento (DER)

O Diagrama Entidade Relacionamento é de extrema importância para o bom funcionamento do aplicativo, e é utilizado para fazer toda a modelagem da base de dados, através dele se sabe exatamente quais tabelas, campos seus respectivos tipos e tamanhos serão necessários além dos relacionamentos existentes entre as tabelas.

A figura 32 mostra a única tabela utilizada para esse projeto, nesse caso não houve a necessidade de relacionamentos.

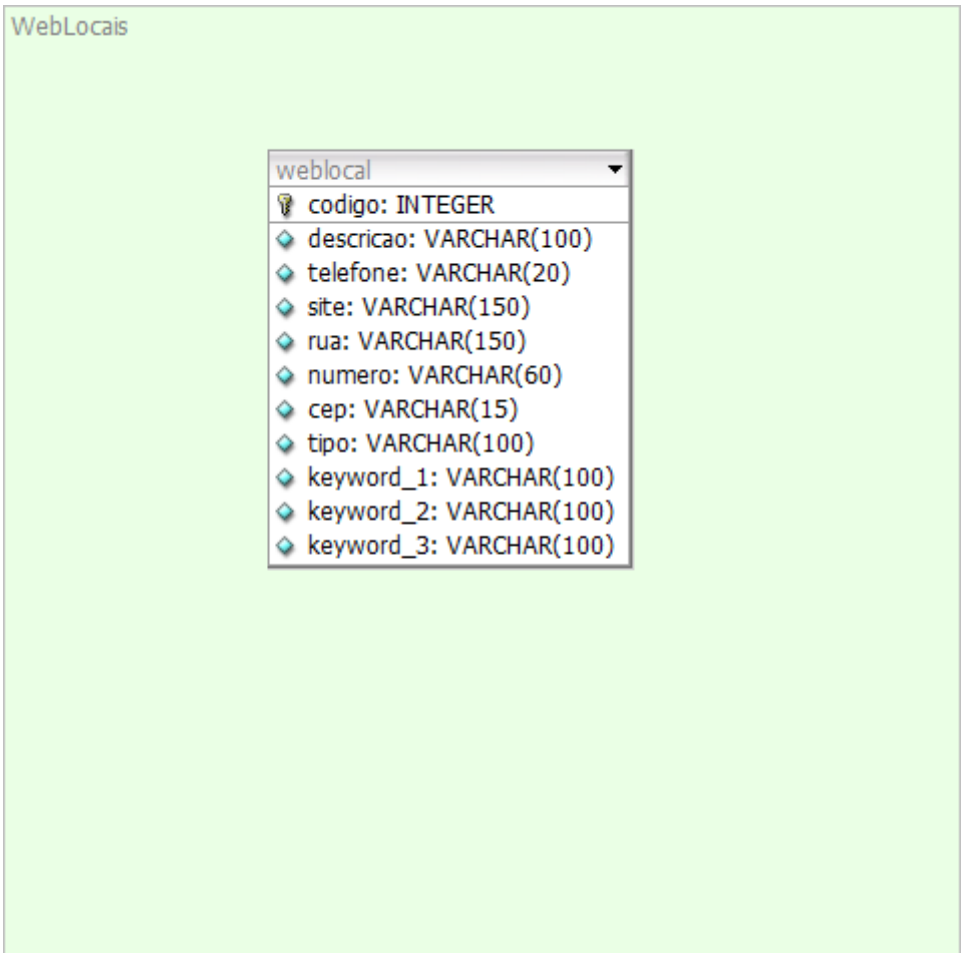


Figura 32 - DER (Diagrama Entidade-Relacionamento)

3.6 Mapas dos Sites (aplicativo cadastro e cliente)

Nesta seção serão exibidos diagramas que facilitam a navegação pelo site de cadastro de dados e também o site cliente responsável pelo consumo dos serviços web. A figura 33 mostra o mapa do site utilizado para cadastrar os dados.

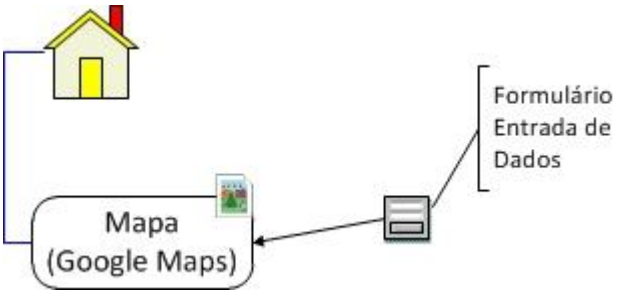


Figura 33 - Mapa do Site (Cadastro)

A figura 34 mostra uma visão bem detalhada do *website* criado para consumir os serviços disponibilizados, lembrando que para consumir um *web service*. O aplicativo em questão não precisa ser um *website*, a única obrigatoriedade é que o aplicativo *web*, *desktop* ou *mobile* possua acesso à internet.

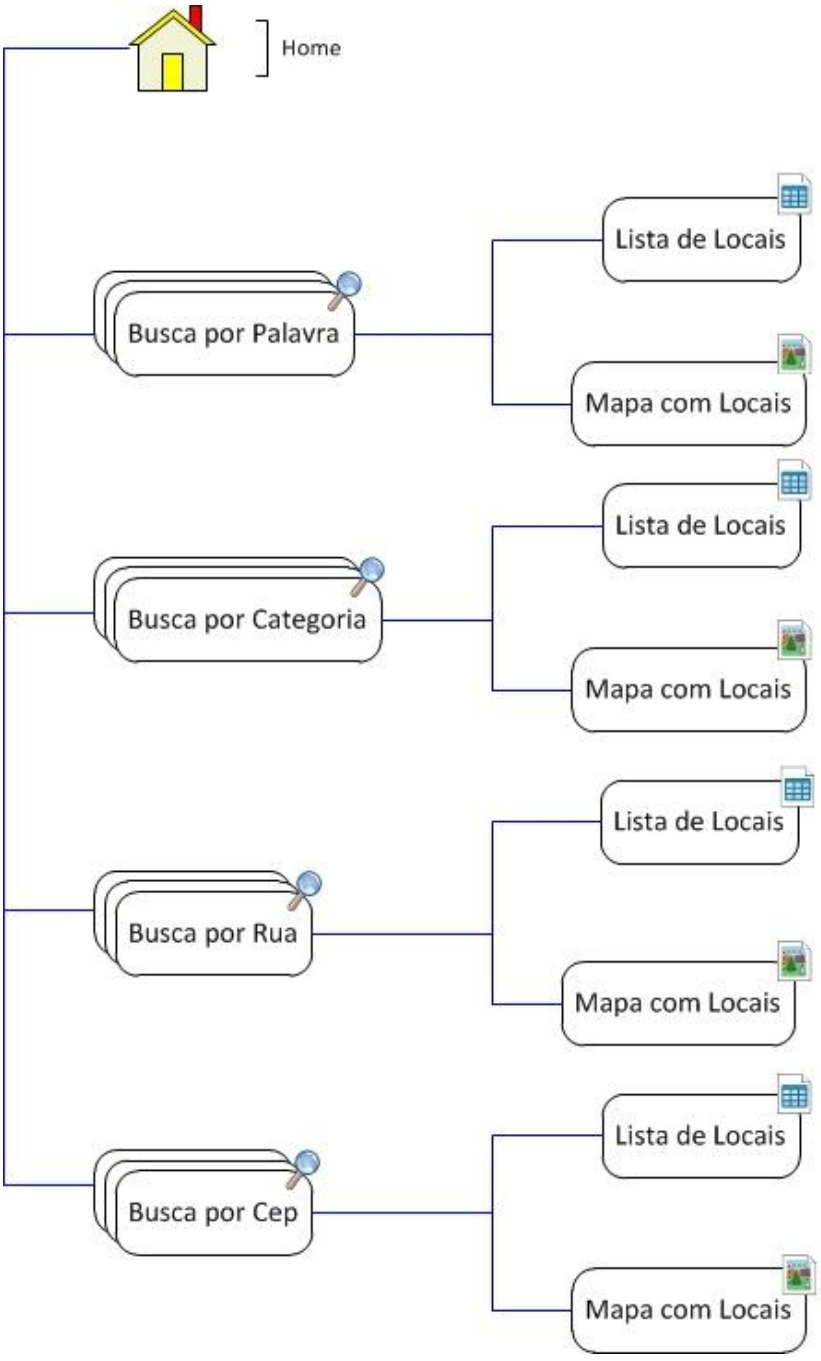


Figura 34 - Mapa do site (cliente)

3.7 Serviços disponibilizados e consumidos

Neste projeto foram disponibilizados quatro serviços *web*, sendo que todos os aplicativos foram desenvolvidos na IDE Netbeans.

3.7.1 Web Services

A figura 35 mostra os serviços *web* visualizada pelo ambiente visual de geração de serviços da IDE Netbeans. As operações (serviços *web*) podem ser criadas pelo ambiente visual ou através de código com o uso de “annotations”.

- **LocaisTipo**
  - **Parâmetro** (String tipo)
  - **retorno** (List< WebLocal>)
- **LocaisCep**
  - **Parâmetro** (String cep)
  - **retorno** (List< WebLocal>)
- **LocaisRua**
  - **Parâmetro** (String rua)
  - **retorno** (List< WebLocal>)
- **LocaisKeyword**
  - **Parâmetro** (String keyword)
  - **retorno** (List< WebLocal>)

Figura 35 - WebServices Disponibilizados (IDE Netbeans)

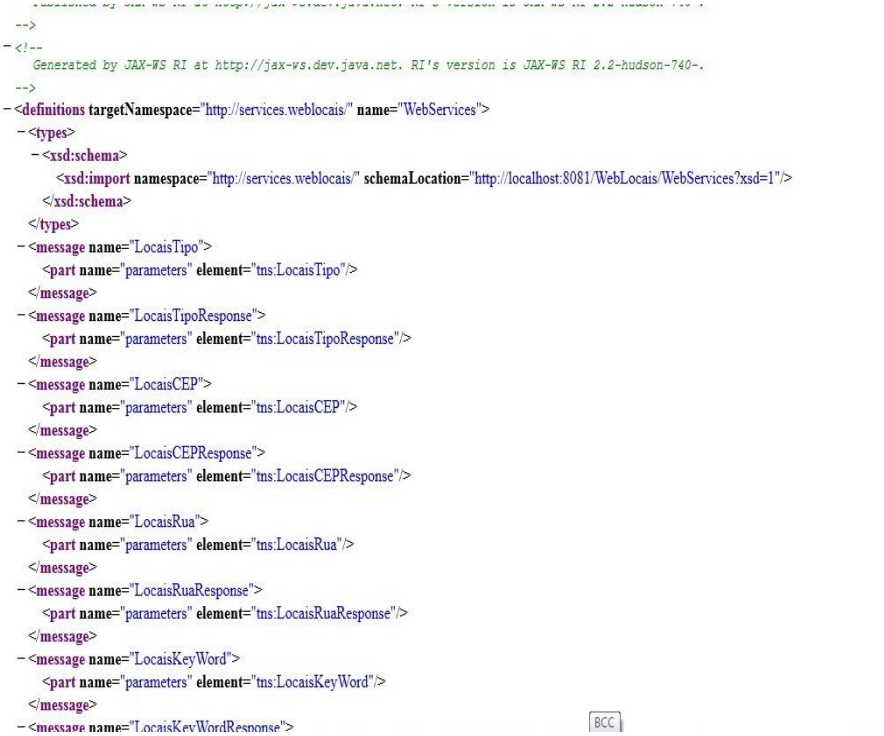
3.7.2 WSDL

O WSDL é o arquivo em formato XML usado para que o aplicativo cliente possa gerar os códigos necessários, todas as informações referentes aos serviços como os métodos disponíveis, parâmetros de entrada, tipos de dados e retorno dos métodos entre outros.

O WSDL pode ser usado de duas maneiras:

- **Arquivo Local:** o arquivo WSDL pode ser salvo localmente e referenciado;
- **WSDL URL**: caso você tenha o endereço (URL) do serviço basta fazer a referência ao serviço através da url. A URL do WSDL é composta na seguinte ordem:
  - http://<servidor>:<porta>/<nome_aplicacao>/<nome_web_service>?wsdl

A figura 36 abaixo mostra um “snapshot” de parte do arquivo WSDL gerado à partir dos serviços.



```
-->
<!--
    Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2-hudson-740-.
-->
<definitions targetNamespace="http://services.weblocais/" name="WebServices">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://services.weblocais/" schemaLocation="http://localhost:8081/WebLocais/WebServices?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="LocaisTipo">
    <part name="parameters" element="tns:LocaisTipo"/>
  </message>
  <message name="LocaisTipoResponse">
    <part name="parameters" element="tns:LocaisTipoResponse"/>
  </message>
  <message name="LocaisCEP">
    <part name="parameters" element="tns:LocaisCEP"/>
  </message>
  <message name="LocaisCEPResponse">
    <part name="parameters" element="tns:LocaisCEPResponse"/>
  </message>
  <message name="LocaisRua">
    <part name="parameters" element="tns:LocaisRua"/>
  </message>
  <message name="LocaisRuaResponse">
    <part name="parameters" element="tns:LocaisRuaResponse"/>
  </message>
  <message name="LocaisKeyWord">
    <part name="parameters" element="tns:LocaisKeyWord"/>
  </message>
  <message name="LocaisKeyWordResponse">
```

Figura 36 - Arquivo WSDL gerado

4 FUNCIONAMENTO DOS APLICATIVOS

Este capítulo é dedicado à demonstração do funcionamento dos aplicativos que foram necessários para coleta de dados, disponibilização e consumo dos serviços.

4.1 Aplicativo de Cadastro

O aplicativo utilizado para coleta de dados tem como base o Projeto de Iniciação Científica desenvolvido no ano de 2011: Integração de um Aplicativo Web em Java com a API Google Maps (DE ASSIS, 2011; DE ASSIS e NITTO, 2011). Esse é um aplicativo online que realiza a coleta de dados através de um formulário e

também recuperada latitude e longitude através do click do mouse, os dados obtidos são armazenados em uma base de dados. A figura 37 mostra um print do aplicativo de cadastro em funcionamento.

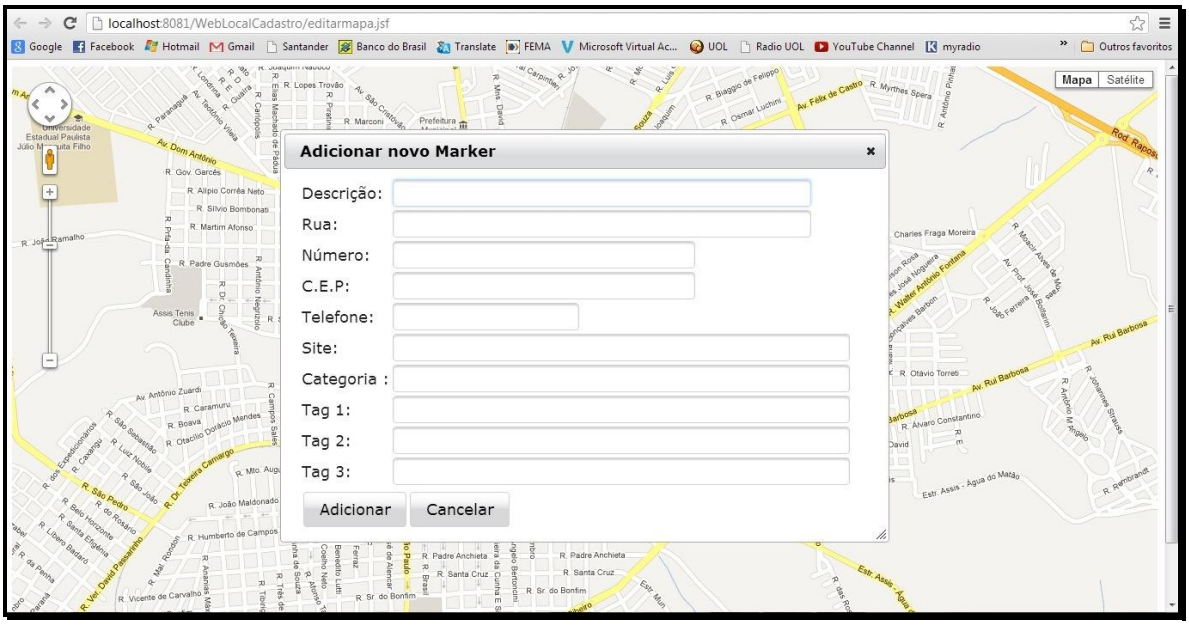


Figura 37 - Funcionamento do aplicativo de coleta de dados

4.2 Aplicativo Servidor

O aplicativo servidor é quem disponibiliza os serviços a serem consumidos, é nele que estão os códigos que fazem conexão com a base de dados, códigos que fazem a recuperação dos dados e onde se encontram todas as bibliotecas necessárias. Nesse caso o processamento de dados é feito totalmente do lado servidor, ficando a encargo do aplicativo cliente somente a exibição e ou utilização dos dados recebidos.

Para que os serviços possam ser consumidos é necessário que o aplicativo *web* esteja no ar (online). A figura 38 mostra uma página *web* criada para informar que o aplicativo servidor está online e quais os serviços disponibilizados.

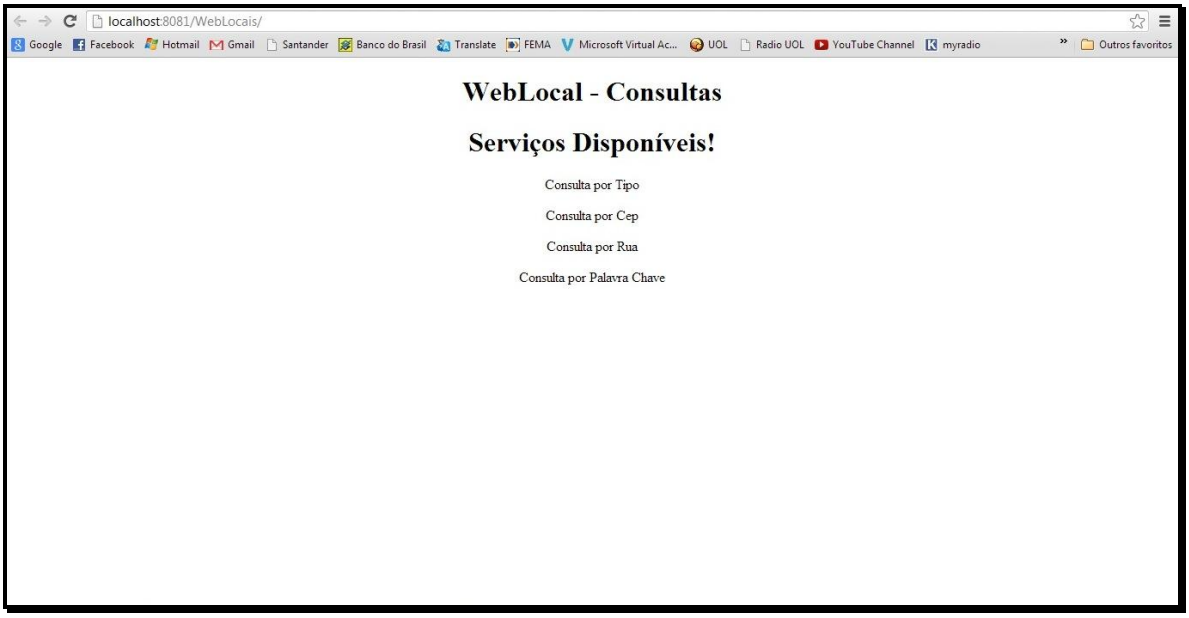


Figura 38 - Página web (aplicativo servidor online)

4.3 Aplicativo Cliente

O aplicativo cliente é um website desenvolvido explicitamente para consumir os serviços disponibilizados pelo servidor, lembrando que o consumo desses serviços poderia ser feito em qualquer linguagem de programação que dê suporte à construção de *web services*, bem como o aplicativo cliente poderia ser *mobile* ou mesmo *desktop*. A figura 39 exibe a home page do aplicativo cliente.

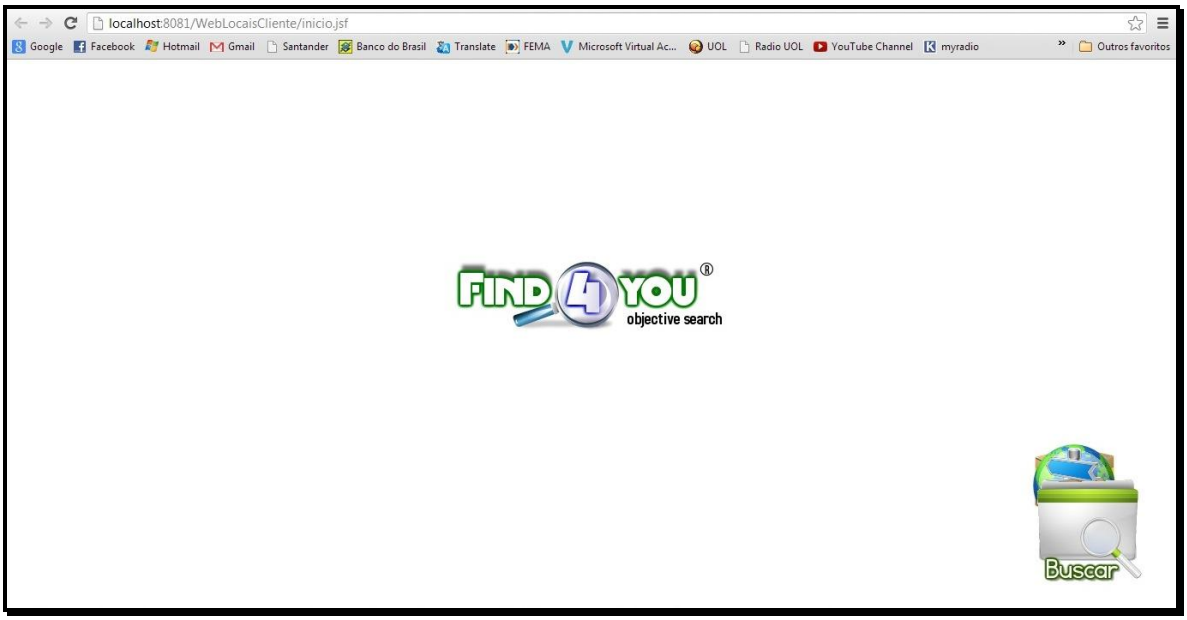


Figura 39 - Home page (aplicativo cliente)

A figura 40 mostra o menu do aplicativo, onde exibe as opções possíveis que consequente levam ao consumo dos serviços *web*.

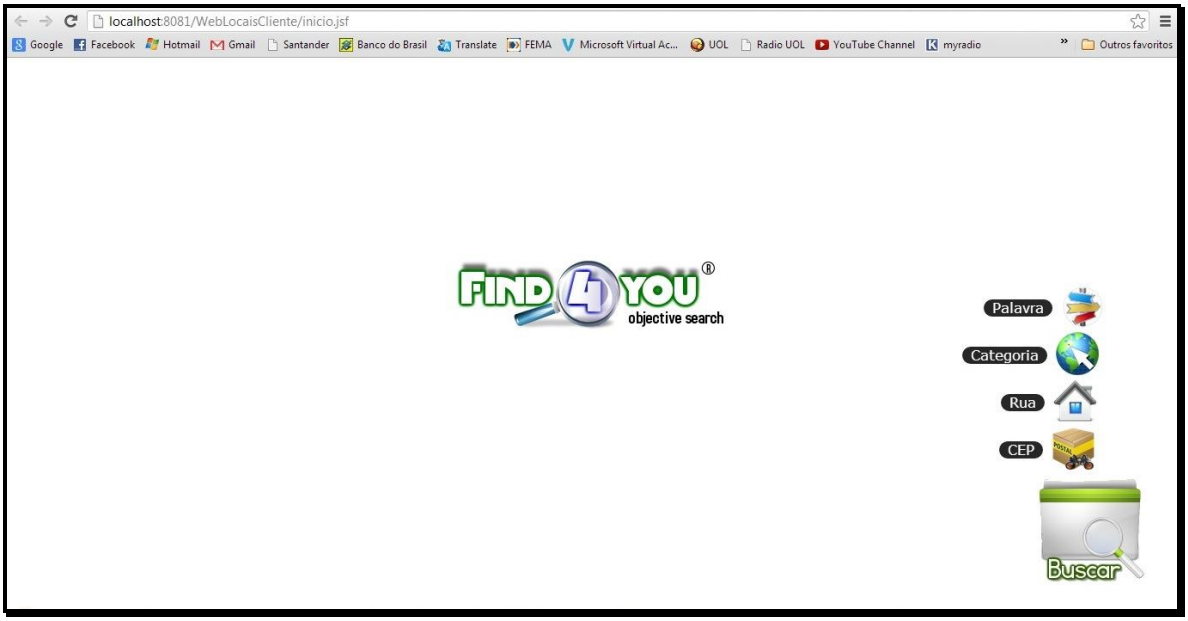


Figura 40 - Menu (aplicativo cliente)

As opções disponibilizadas no menu são:

- **Buscar por palavra**
    - visualiza lista de locais
    - visualiza marcadores no mapa
- **Buscar por categoria**
    - visualiza lista de locais
    - visualiza marcadores no mapa
- **Busca por rua**
    - visualiza lista de locais
    - visualiza marcadores no mapa
- **Buscar por cep**
    - visualiza lista de locais
    - visualiza marcadores no mapa

A figura 41 mostra a exibição de uma consulta onde a busca foi “filtrada” por palavra-chave, no caso a palavra médico.

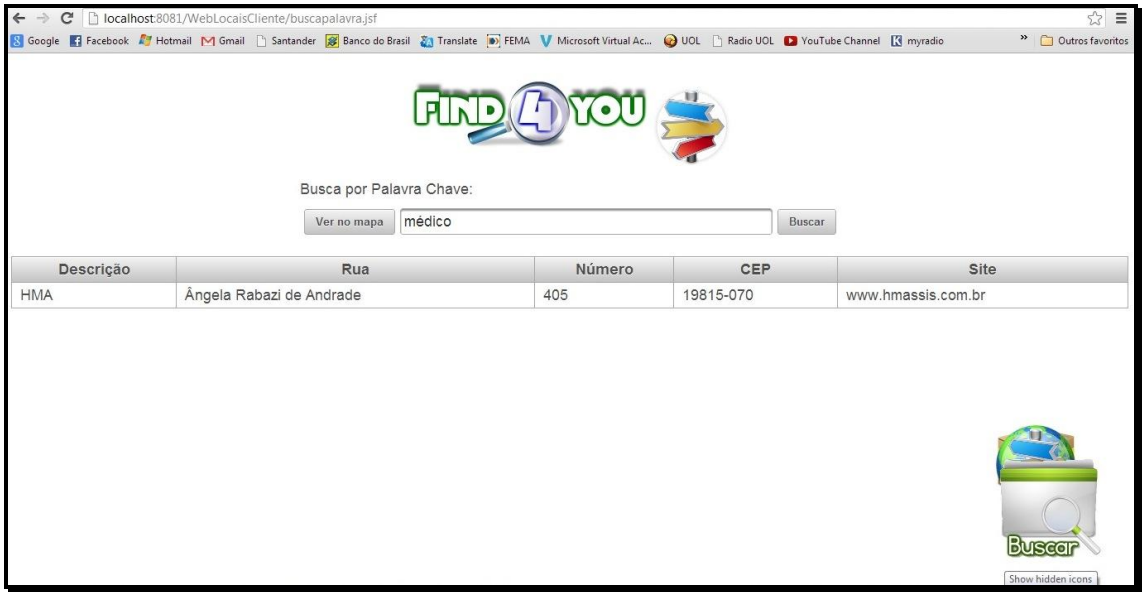


Figura 41 - Busca visualização por lista (filtro por palavra chave)

A figura 42 mostra uma busca onde a opção de filtro selecionada é a categoria, no caso a categoria seleciona foi hotel.

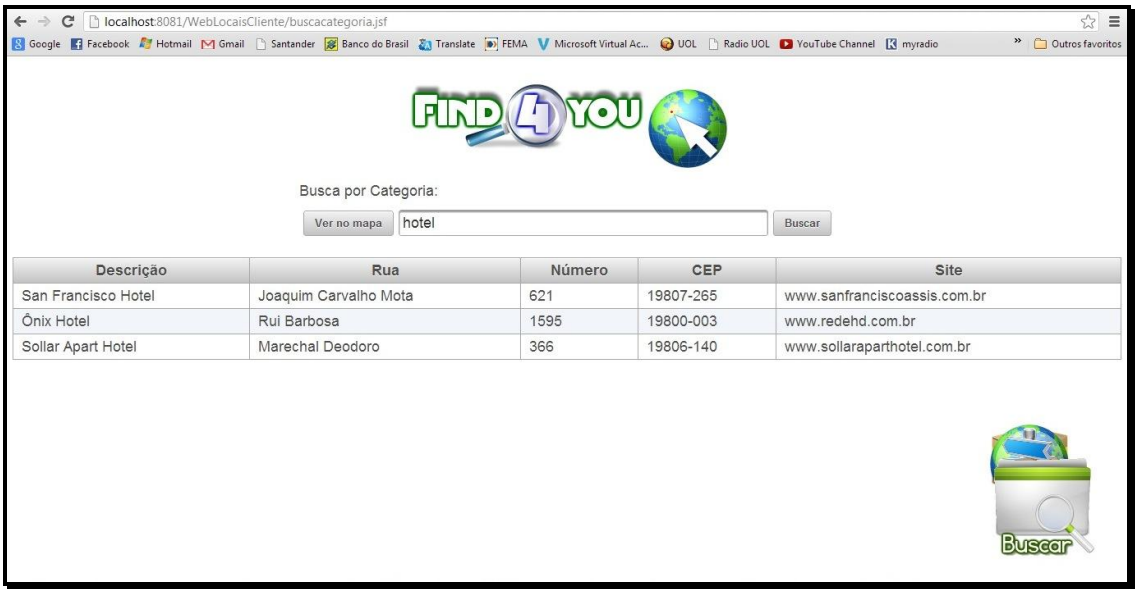


Figura 42 - Busca visualização por lista (filtro por categoria)

A figura 43 mostra uma busca, onde a opção de busca é feita utilizando-se o nome da rua como parâmetro para a consulta. Por exemplo: rua Quinze de Novembro.

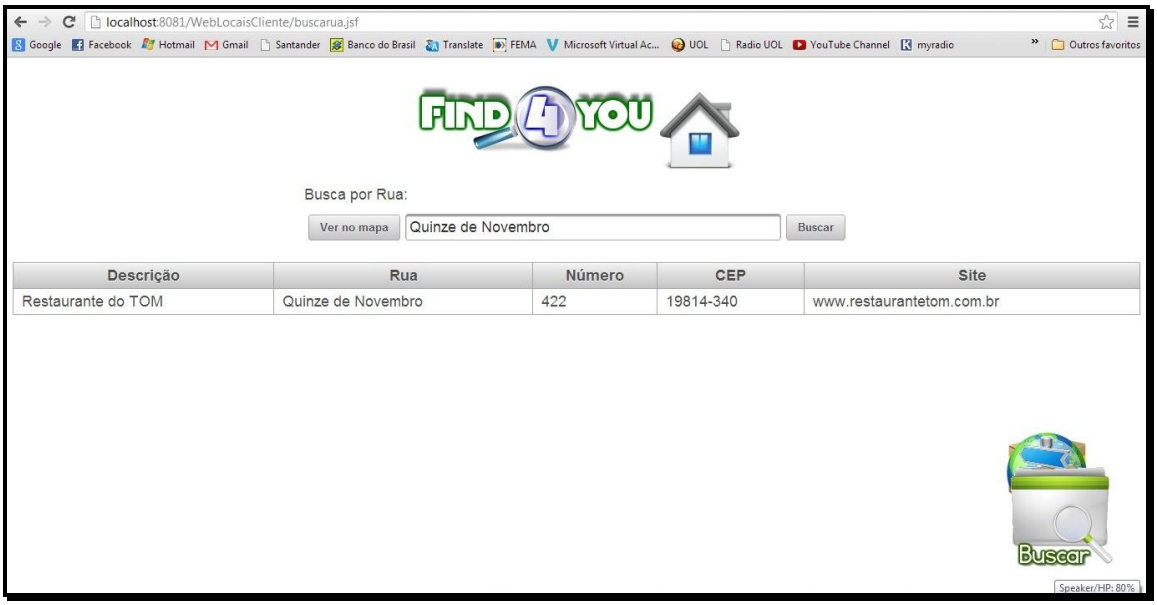


Figura 43 -Busca visualização por lista (filtro por rua)

A figura 44 mostra uma busca utilizando o cep como opção de filtro. Por exemplo: cep número 19814-50.

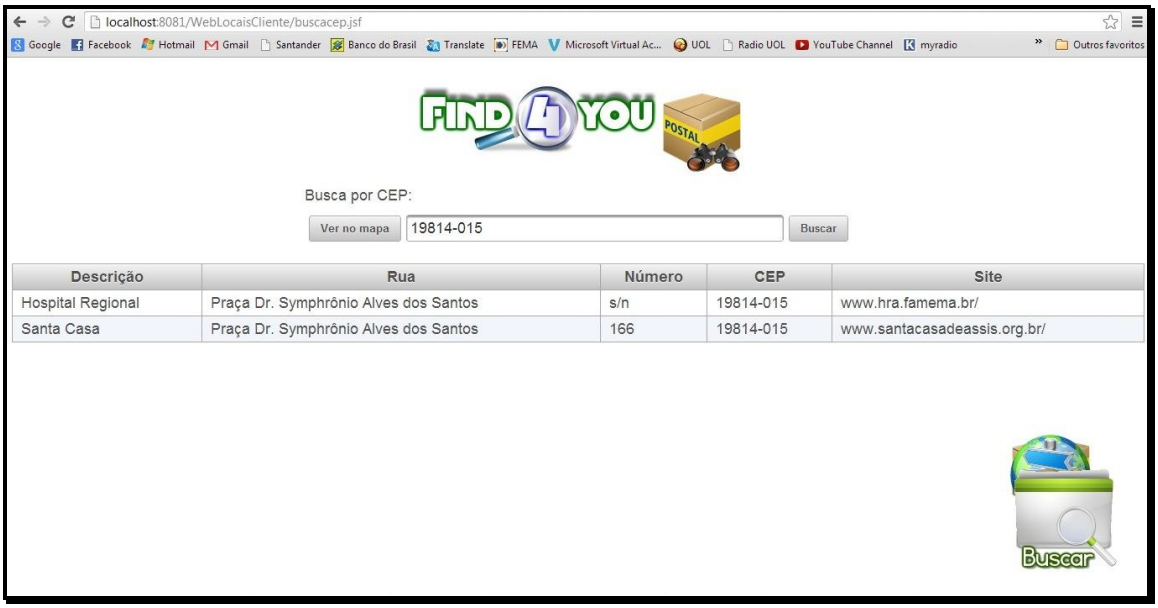


Figura 44 - Busca visualização por lista (filtro por cep)

A figura 45 mostra uma visualização dos locais no mapa. Na escolha pela visualização dos locais no mapa, é exibida uma lista de links à direita que ao receberem o click levam até o marcador correspondente no mapa (API Google Maps) integrado ao aplicativo, também é possível ver alguns marcadores que correspondem aos locais no mapa, essa busca teve como opção de filtro universidade.

Figura 45 - Visualização de Locais no mapa

A figura 46 mostra como base a mesma consulta da figura 45 só que nessa imagem podem ser visualizada a “infoWindow” exibida tanto no click no “marker” quanto no link correspondente à direita. No caso, a “infoWindow” exibida contém as informações do “marker” relacionado à FEMA.

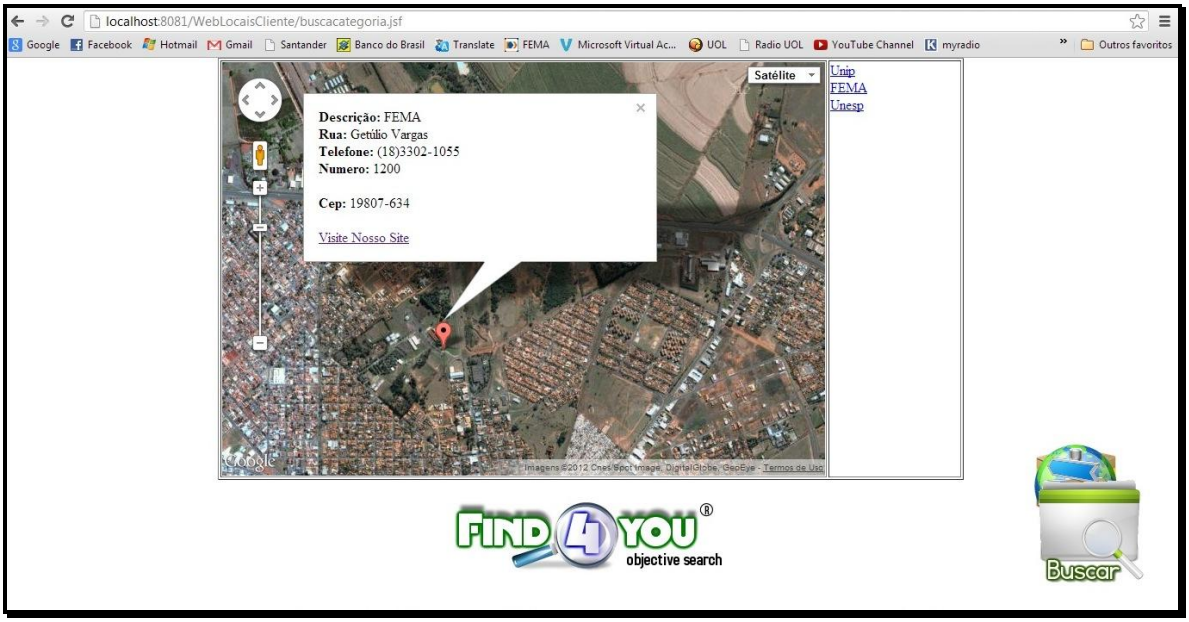


Figura 46 - Visualização da infoWindow no mapa

A figura 47 mostra a mesma busca utilizando o recurso da API Google Maps, onde o mapa pode ser visualizado nas opções map ou satélite, neste caso foi escolhido a opção map. Também se podem visualizar os marcadores da lista espalhados pelo mapa.

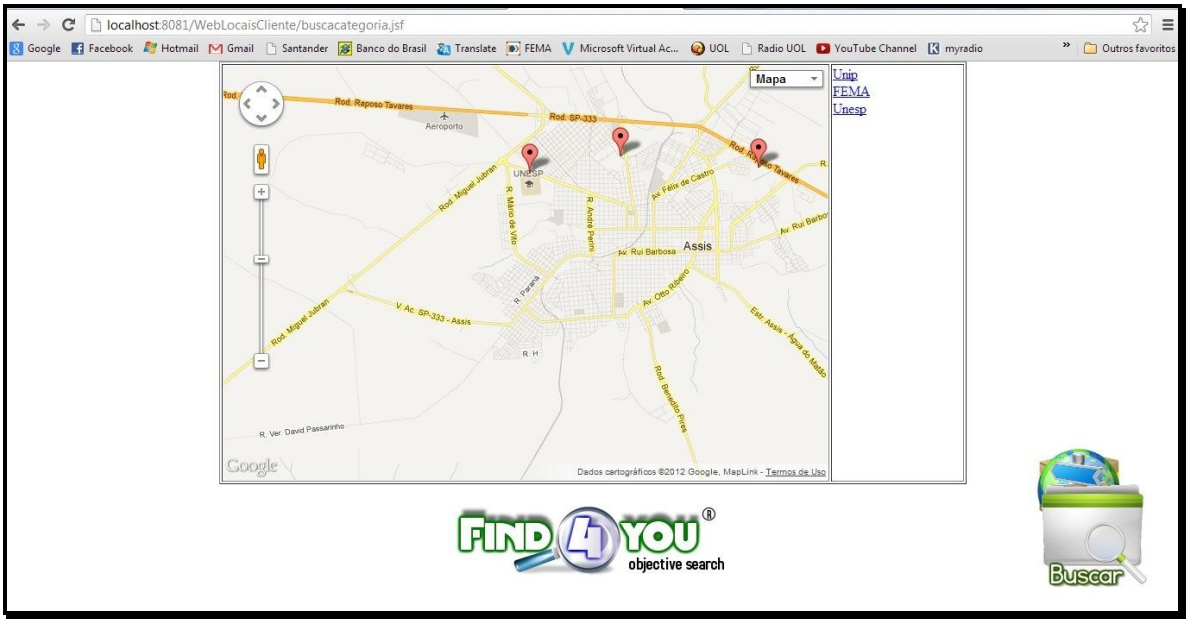


Figura 47 - Visualização dos "markers" no modo "map"

A API Google Maps V.3 permite a utilização dos recursos do *Street View*, que torna o aplicativo muito útil no caso de buscas por locais e se pode até mesmo fazer uma prévia visualização do local a qual procurado. A figura 48 mostra a exploração desse recurso na busca por categoria com filtro hotel, e se pode ver com clareza a fachada do Hotel Ônix da cidade de Assis-SP.

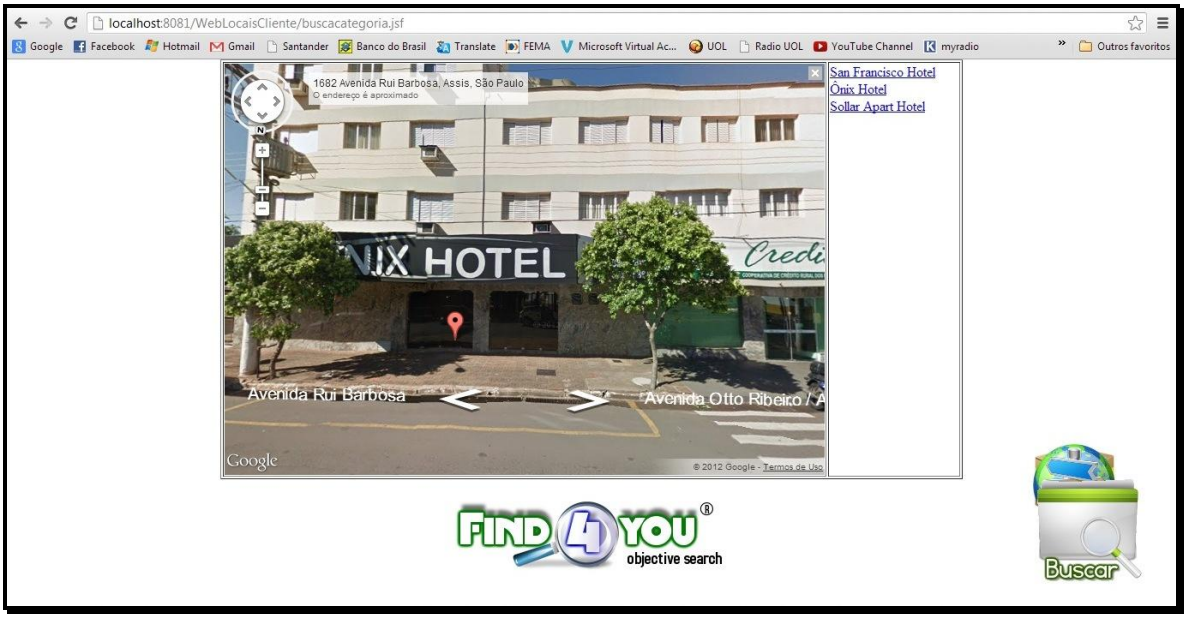


Figura 48 - Utilização do recurso *Street View* na busca por locais

5 CONCLUSÃO

Esse Projeto de Iniciação Científica que teve como objetivo o desenvolvimento de um aplicativo com uma arquitetura de software baseada em *web services*, alcançou seus propósitos.

A conclusão desse projeto disponibiliza uma boa revisão literária que com certeza será um excelente material para estudos futuros, os itens básicos e necessários para que possam ser disponibilizados e consumidos *web services* foram explanados durante o projeto. A construção dos aplicativos *web* resultantes dos estudos efetuados reforça a teoria, e desmistifica a dificuldade da criação e

consumo de *web services*, além é claro de tornar-se um ótimo material de base para a construção de novos aplicativos.

Durante o desenvolvimento desse projeto vários conhecimentos foram adquiridos, além de que novas tecnologias foram testadas e aprovadas. O mercado para desenvolvedores de *web services* com certeza cresce à passos largos e esse projeto é um bom material de apoio para aqueles que demonstrarem interesse.

É sábio ressaltar a importância desse Projeto de Iniciação Científica no amadurecimento acadêmico e também nos conhecimentos adquiridos e extraídos que certamente serão usados futuramente no mercado de trabalho, sendo com certeza um grande diferencial.

Enfim deseja-se que os estudos aqui realizados e que os aplicativos desenvolvidos sirvam de base para estudos futuros, e que possam auxiliar os interessados no desenvolvimento de software com arquitetura baseada em *web services*.

REFERÊNCIAS

ALONSO et al., **Web Services**, Springer-Verlag, 2004.

AYALA, D. et al. **Professional Open Source Web Services**. UK: Wrox Press Ltda, 2002.

BARALE, R.F., **Desenvolvimento de Um Sistema de Vendas na Web Utilizando JSP**, Trabalho de Conclusão de Curso, Faculdade de Ciências Aplicadas de Minas, 2007.

CESCONETI, D. e GLAZAR, J.E., **Construção de Aplicações Distribuídas Utilizando Serviços Web**, Revista Educação Tecnológica, Ano 1, Número 2, 2006.

CHAPPELL, D. e JEWELL, T. **Java Web Services**. First. [S.l.]: O'Reilly, 2002.

DE ASSIS, D.P. e NITTO, M.A., **Integração de Um Aplicativo Web em Java Com o Google Maps API**, IV Fórum de Ciência e Tecnologia, Fema-Imesa, Assis, SP, 2011.

DE ASSIS, D.P., **Integração de Um Aplicativo Web em Java Com o Google Maps API**, Relatório Final do Programa de Iniciação Científica, Fema-Imesa, Assis, SP, 2011.

DEITEL, H.M., **Java Como Programar**, trad. Carlos Arthur Lang Lisboa, 4ed. Bookman, Porto Alegre, 2003.

ENDEL A: **Web Services Description Language (WSDL) 1.1**. Disponível em: <http://www.w3.org/TR/wsdl> Acesso em abril de 2012.

ENDEL B: **Hypertext Transfer Protocol -HTTP/1.1**, The Internet Society, Network Working Group, RFC: 2616. Disponível em: <ftp://ftp.rfc-editor.org/in-notes/rfc1945.txt> Acesso em maio de 2012.

ENDEL C: **Web Services**, Tutorial Point. Disponível em: <www.tutorialspoint. com/webservices/>. Acesso em junho de 2012.

ENDEL D: **Guia de Orientação para Implementação de Web Services,** i3gov. Disponível em:<www.governoeletronico.gov.br/anexos/guia-de-orientacao-para-implementacao-de-web-services/download>. Acesso em junho de 2012.

ENDEL E: **Extensible Markup Language (XML)**, W3C. Disponível em: <http://www.w3.org/XML/> Acesso em maio de 2012.

ENDEL F: **XML Tutorial**, W3SCHOOLS. Disponível em: <http://www.w3schools. com/ xml/>. Acesso em abril de 2012.

ENDEL G: **Soap Tutorial**, w3Schools. Disponível em: <www.w3schools.com/ soap/> Acesso em maio de 2012.

ENDEL H: **UDDI Tutorial**, Tutorials Point. Disponível em: <http://www.tutorial spoint.com/uddi/> Acesso em junho de 2012.

ENDEL I: **Conheça o Apache Tomcat,** DevMedia. Disponível em : < http://www.devmedia.com.br/articles/viewcomp.asp?comp=4546> Acesso em agosto de 2012.

ENDEL J: **Sobre o PostgreSQL,** PostgreSQL.Disponível em: <http://www.postgresql.org.br/sobre>. Acesso em agosto de 2012.

ENDEL K: **Web Services Architecture**, W3C. Disponível em:< http://www.w3.org/TR/ws-arch/>. Acesso em Abril de 2012.

ENDEL L : FURTADO JR, M.B. e DUARTE, O.C.M.: **XML - Extensible Markup Language**. Disponível em: <http://www.gta.ufrj.br/grad/00_1/miguel/index.html> Acesso em Maio de 2012.

ENDEL M: RECKZIEGEL, Mauricio: **Descrevendo um Web Service – WSDL**. Disponível em: < http://imasters.com.br/artigo/4422/webservices/descrevendo_um_web_ service_wsdl/> Acesso em outubro de 2012.

ENDEL N :OMG. **Common Object Request Broker Architecture Specification**. 2005. Disponível em: <http://www.omg.org>. Acesso em outubro de 2012.

ENDEL O : SUMRA, R.: **Developing JAX-RPC-Based Web Services Using Axis and SOAP**, Disponivel em <http://www.developer.com/print.php/2237251>. Acesso em agosto de 2012.

ENDEL P: UPnP Forum. **UPnP Device Architecture**. 2000. Disponível em: <http://www.upnp.org>. Acesso em novembro de 2011.

ENDEL Q: BOOTH, David et al. **Web Services Architecture**. 2004. Disponível em: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/>. Acesso em dezembro de 2011.

ENDEL R: COSTA et al., **Utilização de Aspectos no Desenvolvimento de Aplicações Baseadas em Serviços Web**. WASP'2004 – I Workshop Brasileiro de Desenvolvimento Orientado a Aspectos. Brasília, 2004. Disponível em: <http://www.dcc.ufmg.br/~mcosta/>. Acesso em dezembro de 2011.

ENDEL S: CUNHA David: **Web Services, SOAP e Aplicações Web**. Disponível em:< http://devedge-temp.mozilla.org/viewsource/2002/soap-verview/index_pt_br.html> Acesso em setembro de 2012.

ENDEL T: DRJT.: **DeveloperResources for Java Technology**. Disponível em: <http://java.sun.com/>. Acesso em setembro de 2011.

FRONDANA, G, **ARQUITETURA ORIENTADA A SERVIÇOS PARA GESTÃO DE PROCESSOS ACADÊMICOS NA WEB,** Instituto Militar De Engenharia, 2009

GINIGE, A., MURUGESAN, S., **Web Engineering: an Introduction**, IEEE Multimedia, Vol. 8, Issue: 1, 2001.

GRAHAM et al., **Building Web Services With Java: Making Sense of XML, SOAP, WSDL and UDDI**, Sams Publishing, 2001.

GREGO, M. **De Carona Com Web Services**. Revista INFO Corporate, São Paulo: Abril, n.1, 2002.

HANSEN, M.D., **SOA Using Java Web Services**, Pearson Education, Inc, 2007.

LEMAY, L. e PERKINS, C.L., **Teach Yourself Java in 21 Days**, Sams.net Publishing, 1996.

MCGOVERN et al., **Java Web Services Architecture**, Morgan Kaufmann Publishers, 2003.

MOREIRA, J.J.M., **wsQL-Uma Arquitetura de Software Baseada em Web Service**, Tese de Mestrado, Universidade do Porto, Portugal, 2005.

OASIS, **Modelo de Referência para Arquitetura Orientada a Serviço 1.0**, 2006.

PRESSMAN, R.**Engenharia de Software**, Rio de Janeiro, McGraw Hill, 2002.

RODRIGUES, J.E.F., **Web Services e Arquitetura Orientada a Serviços Como Estratégia Para Integração de Sistemas**, Trabalho de Conclusão de Curso, Universidade Federal do Pará, 2008.

SILVA, E. e SIROTHEAU, L.F., **Conceito de SOA**, Senac, DF, 2006.

SILVA, R.O., **Estudo de Web Services**, Trabalho de Conclusão de Curso, Faculdade Integradas de Mineiros, 2007.

SILVEIRA, P. E. A. e COSENTINO, R. A.: **FJ-21 Java para desenvolvimento web. Caelum Ensino e Soluções em Java**, 2009.

VOGELS, Werner. **Web Services Are Not Distributed Objects**. IEEE Internet Computing, v. 7, n. 6, 2003.

WIEHLER, G., **Web Services And Service Oriented Architectures-The Impact on Business Applications**, SBS SYS, 2004