UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO
DE SISTEMAS

PETERSON RICARDO MAIER SCHMITT

APLICAÇÃO WEB UTILIZANDO API GOOGLE MAPS

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2013

PETERSON RICARDO MAIER SCHMITT

APLICAÇÃO WEB UTILIZANDO API GOOGLE MAPS

Trabalho de diplomação apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Claudio L. Bazzi

MEDIANEIRA

2013

Ministério da Educação

Universidade Tecnológica Federal do Paraná

Diretoria de Graduação e Educação Profissional

Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

Aplicação web utilizando API GOOGLE MAPS

Por

Peterson Ricardo Maier Schmitt

Este Trabalho de Diplomação (TD) foi apresentado às 15:40 h do dia 25 de março de 2012 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, *Campus* Medianeira. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado com louvor e mérito.

Prof. Dr. Cláudio Leones Bazzi
UTFPR – *Campus* Medianeira
(Orientador)

Prof. Dr. Neylor Michel
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Msc. Fernando Schütz
UTFPR – *Campus* Medianeira
(Convidado)

Prof. Msc. Juliano Rodrigo Lamb
UTFPR – *Campus* Medianeira
(Responsável pelas atividades de TCC)

“Nem tudo na vida se resolve com dois cliques.”

Wendel Henrique Ferreira

RESUMO

SCHMITT, Peterson Ricardo Maier. Aplicação web utilizando API Google Maps. Trabalho de Conclusão de Curso (Tecnologia em Análise e Desenvolvimento de Sistemas), Universidade Tecnológica Federal do Paraná. Medianeira 2013.

Este trabalho apresenta um estudo bem como desenvolvimento baseado sobre a API do Google Maps para desenvolvimento WEB, mostrando um breve histórico conceitos de banco de dados relacionais, banco de dados geográficos, API googlemaps, PHP, *Postgre.* Ao final mostrar exemplos de uma aplicação, utilizando os recursos anteriormente citados, que visam facilidade em diversas pesquisas de localidades com maior êxito e confiabilidade.

Palavras-chave: PHP, JSON, JQUERY, JAVASCRIPT, DOM*, Mapas, PostgreSQL*, Postgis.

ABSTRACT

SCHMITT, Peterson Ricardo Maier. Web application using Google Maps API. Trabalho de Conclusão de Curso (Technology Analysis and Systems Development), Federal Technological University of Paraná. Medianeira 2013.

This paper presents a study and development based on Google Maps API for Web development, showing a brief historical concepts of relational databases, geographic database, googlemaps API, PHP, Postgre. At the end show examples of an application using the resources mentioned above, aimed at ease in various research locations with greater success and reliability.

Keywords: PHP, JSON, JQUERY, JAVASCRIPT, DOM*, Mapas, PostgreSQL*, Postgis.

LISTA DE FIGURAS

LISTA DE SIGLAS

AJAX	Asynchronous JavaScript + XML
API	Application Programming Interface
BD	Banco de dados
BSD	Berkeley Software Distribution
CSS	Cascading Style Sheets
DOM	Document Object Model
GIS	Geographic Information System
GNU	General Public License
GPS	Sistemas de Posicionamento Global
HTML	HyperText Markup Language
JS	JavaScript
JSON	JavaScript Object Notation
MER	Modelo Entidade-Relacionamento
MVC	Model View Controller
PHP	Personal Hypertext Preprocessor
SGBD	Sistema de Gerenciamento de Banco de Dados
SGML	Standard Generalized Markup Language
SIG	Sistemas de Informação Geográficas
SQL	Structured Query Language
UML	Unified Modeling Language
UTFPR	Universidade Tecnológica Federal do Paraná
WKT	Well-Known Text
WWW	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

Para que haja um aumento de produtividade considerável em ambientes de desenvolvimento web, é preciso que os desenvolvedores web, busquem inúmeras ferramentas e softwares visando rapidez, confiabilidade e êxito. Para que isso ocorra surgem as API's, que estão cada vez mais acessíveis para os programadores, permitindo a utilização do mesmo em softwares, como o uso da API do Google Maps para desenvolvimento web.

Atualmente uma verdadeira evolução está acontecendo. Pessoas que até então não tinham qualquer contato com ferramentas GIS, de uma hora para outra podem ter acesso à qualquer parte do planeta por meio de aplicações que misturam imagens de satélite, modelos 3D e GPS, sendo que o usuário necessita apenas ter conexão à Internet. Essas tecnologias de geoprocessamento tornaram-se indispensáveis para profissionais de diversas áreas. Neste contexto que surge a necessidade de utilizar e desenvolver produtos de localização geográfica e análise espacial para o ambiente web.

A cada dia fica mais comum estar em contato com tecnologias relacionadas a geoprocessamento, e a maioria das vezes os usuários não sabem que as mesmas estão de alguma forma sendo utilizadas.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Realizar um estudo sobre geoprocessamento fazendo uso da API Google Maps, apresentando exemplos das principais funcionalidades desta API, mostrando como a mesma pode ser utilizada em aplicações web.

1.1.2 Objetivos Específicos

- Construir um referencial teórico sobre tecnologias voltadas a programação para geoprocessamento;
- Desenvolver a análise e projeto de uma aplicação que contemplem o uso de tecnologias de geoprocessamento, utilizando linguagem UML;
- Desenvolver uma aplicação propostas fazendo uso da linguagem de programação PHP, e tecnologias como *Javascript, JSON, JQuery e Google Maps API v3*;

1.2 JUSTIFICATIVA

Uma API (*Application Programming Interface*) é uma interface que pode estar conectada a diferentes sistemas e aplicativos, no entanto, para o usuário isto é imperceptível pelo fato de estar rodando por trás de tudo, enquanto o usuário usufrui de um aplicativo ou site, a sua API pode estar conectada a diversos outros sistemas e aplicativos, sem que o usuário perceba.

Cada vez mais o uso de SIG (Sistema de Informações Geográficas) tem se tornado mais visível e comum em sistemas e sites com o uso da API do Google Maps, permitindo assim a criação de mapas com localização definida, controle de zoom, tipos de mapa, geração de rotas, pesquisa por estabelecimentos entre outros.

Usar essa interface é de suma importância, pois através da mesma pode-se ter facilidade e comodidade em encontrar inúmeros locais, com maior rapidez e eficácia.

A cada dia o usuário torna-se mais exigente e procura maior êxito em suas pesquisas, utilizando a API Google Maps estas buscas se tornam mais acessível satisfazendo assim o usuário.

Este trabalho tem como objetivo a realização de um estudo das linguagens, PHP, HTML, CSS, *JavaScript* e o banco de dados PostgreSQL, aplicando-as em um estudo experimental voltado ao desenvolvimento de uma aplicação web para

mapeamento geográfico como o Google Maps, onde o usuário poderá cadastrar pontos e polígonos no mapa, cadastrando as principais características de cada um.

2 REFERENCIAL TEÓRICO

2.1 PHP (Hypertext Preprocessor)

A linguagem PHP surgiu por volta de 1994, o criador foi Rasmus Lerdorf, foi criada para suprir as necessidades do desenvolvedor, e aperfeiçoada para se adequar às necessidades de sua crescente comunidade, Segundo Darlan (2007), as primeiras versões não foram disponibilizadas, tendo sido utilizadas na home - page apenas para que pudessem ter informações sobre as visitas que estavam sendo feitas.

Para Silveira (2010), com o crescimento da popularidade do PHP, um grupo de desenvolvedores criou uma API para ele transformando-o no PHP3. Para melhorar a sua performance, o *scripts* foi completamente reescrito, surgindo dessa forma o PHP4, muito mais veloz do que o PHP3. Essa versão passou a incluir suporte a gerenciamento de sessões, uma característica antes presente apenas no ASP.

De acordo com Milani (2010), ao longo do tempo o PHP teve um grande crescimento e aumento de popularidade, sendo que em junho de 2004 foi lançada a versão 5 do PHP, introduzindo um novo modelo de orientação a objetos, incluindo a reformulação dos construtores e adição de destrutores, visibilidade de acesso, abstração de objeto e interfaces de objetos.

Para Freitas (2006), as principais vantagens para a utilização da linguagem PHP são:

- linguagem de fácil aprendizado;
- performance e estabilidade excelentes;
- código aberto;
- suporte nos principais servidores do mercado;
- suporta conexão com os principais bancos de dados do mercado;
- é multiplataforma;

- suporta grande variedade de protocolos; e
- não precisa ser compilado, por ser uma linguagem interpretada.

Segundo Freitas (2006), o PHP difere-se de outras linguagens, pois seu código é escrito embutido a um arquivo HTML. O que diferencia o PHP do *JavaScript* no lado do cliente é que o cliente recebe somente a resposta, não tendo acesso ao código que são interpretados no servidor.

Freitas (2006), apresentam as principais tarefas da linguagem PHP, são elas:

- Funções de correio eletrônico: pode-se enviar um e-mail a uma pessoa ou uma lista parametrizando toda uma série de aspectos, tais como assunto, pessoa a responder e outras funções;
- Gestão de bases de dados: oferece interface para o acesso a maioria de dados comerciais e todas as bases possíveis em sistemas Microsoft, podendo editar o conteúdo do site;
- Gestão de arquivos: pode realizar qualquer tipo de operação como criar, modificar, mover, apagar a partir das funções para gestão de arquivos.
- Tratamento de Imagens: uniformizar em tamanho e formatos várias imagens recebidas automaticamente através do PHP e pode-se também criar botões que realizam uma única chamada a uma função.

2.2 HTML (Hyper Text Markup Language)

Para Fernandes (2008), através da *World Wide Web* é possível acessar informações armazenadas em documentos escritos usando-se uma linguagem chamada *Hyper Text Markup Language* (HTML). Esta linguagem, inventada por Tim Berners-Lee no Laboratório CERN na Suíça, é composta por comandos através dos quais é possível definir a aparência e a estrutura de documentos. Albuquerque (2004), destaca que a linguagem HTML é baseada na linguagem Standard *Generalized Markup Language* (SGML).

Segundo Ferreira (2012), desde o lançamento do HTML4, o W3C alertou os desenvolvedores sobre boas práticas que deveriam ser seguidas no

desenvolvimento dos códigos. Contudo, o HTML4 não trazia diferencial e também não facilitava a manipulação de elementos JavaScript e CSS.

Para Oliveira (2011), o HTML 5 (*Hypertext Markup Language*) é a quinta versão da linguagem HTML. Esta nova versão traz consigo importantes mudanças quanto ao papel do HTML no mundo da *web*, trazendo novas funcionalidades como semântica e acessibilidade, com novos recursos antes só possíveis por meio de outras tecnologias, e trazendo uma importante disseminação dentre todos os novos navegadores de internet, tornando-o dessa forma mais universal.

Segundo Ferreira e Eis (2010), com a evolução da linguagem padrão para web pôde-se eliminar a necessidade de *plug-ins* para aplicações multimídia em navegadores. Críticos consideram a tecnologia como um forte concorrente ao Flash, da Adobe, ao Silverlight, da Microsoft, e ao recente JavaFX, da Sun (Oracle).

De acordo com Martins (2010), após dez anos sem atualizações, a forma como se escreve páginas na Internet passa por uma significativa transformação. O HTML5 oferece uma experiência web totalmente diferente para usuários tornando a navegação mais rápida, simples e melhorando a performance de uma página web, embora exista um longo caminho para ser finalizado, muitos navegadores importantes já implementaram grandes partes da linguagem, incluindo tags de vídeo e suporte à tecnologia Canvas.

2.3 JAVASCRIPT

Para Majour (2008), *JavaScript* é uma linguagem de S*cript* projetada principalmente para adicionar interatividade a uma página *web* além da criação de aplicativos. Essa linguagem foi implementada pela *Netscape Communications* em 1995. A linguagem de S*cript* é usada em milhões de páginas web e aplicativos de servidor em todo o mundo.

Embora essa linguagem compartilha muitas características e estruturas da linguagem Java, foi desenvolvida de forma independente. *JavaScript* pode interagir com o código HTML, permitindo que autores da Web possam deixar seus sites mais robustos com conteúdos mais dinâmicos. *JavaScript* é apoiado por uma série de empresas de *software* e é uma linguagem aberta que qualquer um pode utilizar sem precisar adquirir uma licença.

2.4 JQUERY

De acordo com Belem (2010), JQuery é uma biblioteca *JavaScript*, que simplifica o desenvolvimento de aplicações web. JQuery auxilia os programadores a manter o código simples e legível além de serem reutilizáveis. JQuery simplifica o processos em *JavaScript*, como chamadas de *AJAX* e manipulação de *DOM (Document Object Model).*

Para Silva (2008), as principais vantagens do JQuery sobre *JavaScript* são:

- acesso direto a qualquer componente do DOM, ou seja, não há necessidade de várias linhas de código para acessar determinados pontos no DOM;
- o uso de regras de estilo não sofre qualquer tipo de limitação devido as inconsistências dos navegadores. Mesmo os seletores CSS3 podem ser usados sem qualquer restrição;
- manipulação de conteúdos, sem limitações, com algumas poucas linhas de código;
- suporte para toda a gama de eventos de interação com o usuário sem limitações impostas pelos navegadores;
- possibilidade de inserir uma grande variedade de efeitos de animação com uma simples linha de código;
- uso simplificado e sem restrições com AJAX e linguagens de programação, como PHP e ASP;
- simplificação na criação de scripts; e
- emprego *cross-browser*.

De acordo com Rigoni (2009), JQuery está em conformidade com todos os padrões *web* estipulados pela W3C, ela oferece total suporte a CSS3, é uma biblioteca compatível com qualquer navegador (*cross browser*). JQuery visa incrementar de forma progressiva e não obstrutiva a usabilidade e acessibilidade, as principais características do JQuery são:

- Utiliza seletores CSS para busca de elementos DOM na árvore HTML;
- Arquitetura simples para instalação de *plugins* e criação de *plugins*;
- Não é necessário a criação de loops para busca de elementos na árvore DOM;

- Programação encadeada, pois todo método retorna um objeto;
- Extensível, permite o usuário estender a própria biblioteca e customizar a seu modo.

Para Alvarez (2009), outra forte característica do *JQuery* é que ela torna mais fácil para escrever *JavaScript* que funciona em muitos navegadores diferentes. Incompatibilidades entre os navegadores populares como o IE (*Internet Explorer)* e *Firefox* significa que muitas vezes o usuário precisa escrever os diferentes partes de código *JavaScript* para cada navegador. Com *JQuery*, no entanto, é apenas chamar a função *JQuery* apropriado e deixar que *JQuery* contorna o código executando em diferentes navegadores.

2.5 AJAX

AJAX (*Asynchronous JavaScript + XML)*, termo criado por Jesse James Garrett em 2005, que descreve uma "nova" abordagem de como usar uma série de tecnologias em conjunto, incluindo: *HTML* ou *XHTML*, *CSS*, *JavaScript*, o *DOM*, *XML*, *XSLT* e, o mais importante objeto *XMLHttpRequest*.

Para Goldbach (2009), quando essas tecnologias são combinadas no modelo AJAX, as aplicações web são capazes de fazer rápidas atualizações incrementais para a interface do usuário sem recarregar a página inteira do navegador. Isso torna o aplicativo mais rápido e mais sensível às ações do usuário.

2.6 JSON

JSON (*JavaScript Object Notation*) é uma estrutura, baseada em texto para armazenar e transmitir dados estruturados. Ao usar uma sintaxe simples, o usuário pode facilmente armazenar qualquer coisa a partir de um único número por meio de *Strings*, matrizes e objetos usando nada além de uma seqüência de texto simples. O usuário também pode alinhar matrizes e objetos, o que lhe permite criar estruturas complexas de dados.

Uma vez criada a *String* JSON, pode-se enviá-la para outra aplicação ou computador, porque se trata de texto simples.

As principais vantagens do JSON são:

- é compacto;
- é de fácil entendimento para leitura como para desenvolvimento;
- ele mapeia muito facilmente as estruturas de dados usadas por muitas linguagens de programação (números, strings, booleanos, nulos, matrizes e matrizes associativas); e
- quase todas as linguagens de programação contem funções ou bibliotecas que podem ler e escrever estruturas JSON.

2.7 DOM

Para Balduino (2012), a representação interna de uma página é chamada de DOM, que significa *Document Object Model*, que é criada automaticamente pelo browser toda vez que carregamos um arquivo XML ou HTML válido. Esse arquivo é chamado de Documento, e cada item dentro dele (textos, imagens, botões, caixas de texto) é chamado genericamente de Elemento. Quando for utilizado um *JavaScript* para ler ou escrever dados em numa página HTML, estamos na verdade manipulando um elemento DOM.

2.8 BANCOS DE DADOS RELACIONAL

De acordo com Heuse (1998), os sistemas de gerência de banco de dados (SGBD) surgiram no início da década de 70 com o objetivo de facilitar a programação de aplicações de banco de dados (BD). Os primeiros sistemas eram caros e difíceis de usar, requerendo especialistas treinados para usar o SGBD específico.

O modelo de banco de dados Relacional introduzido por Coddem 1970, é o mais simples, com estrutura de dados uniforme, e também o mais formal.

Segundo Otavio (2003), enfatiza que no modelo de dados relacional as informações são organizados e agrupados em tabelas (relacionais). Essas tabelas

guardam estes dados e podem possuir referência a outra tabela. Assim o banco todo não passa de uma série de tabelas que se referenciam.

Para Otavio (2003), o modelo relacional é uma teoria matemática para descrever como as bases de dados devem funcionar. Embora esta teoria seja a base para o software de banco de dados relacionais, poucos sistemas de gestão de bases de dados seguem o modelo de forma restrita, e todos têm funcionalidades que violam a teoria, desta forma variando a complexidade e o poder.

De acordo com Dantas (2002), a arquitetura ANSI / SPARC, os bancos de dados relacionais consistem de três componentes:

- uma coleção de estruturas de dados, formalmente chamadas de relações, ou informalmente tabelas, compondo a parte do nível conceitual;

- uma coleção dos operadores, a álgebra e o cálculo relacionais, que constituem a base da linguagem SQL; e
- uma coleção de restrições da integridade, definindo o conjunto consistente de estados de base de dados e de alterações de estados. As restrições de integridade podem ser do tipo domínio, atributo, relvar (variável de relacionamento) e restrições de base de dados.

2.9 BANCO DE DADOS GEOGRÁFICO

Segundo Ferreira (2006), os dados geográficos são aqueles que possuem uma dimensão espacial, ou uma localização, diretamente ligada ao mundo geográfico real, como por exemplo, imagens de satélites de sensoriamento remoto. Bancos de dados Geográficos (BDG) são coleções de dados geo referenciados, manipulados por Sistemas de Informação Geográficas (SIG). Dentre as finalidades e possibilidades que as bases de dados com geometria oferecem, pode-se citar as de análise e consultas espaciais. É possível calcular por exemplo, áreas, distâncias e centróides, além de realizar a geração de buffers e outras operações entre as geometrias.

De acordo com Ferreira (2006), a evolução científica e tecnológica dos últimos anos, impulsionada principalmente pelas necessidades de padronização de dados e a interoperabilidade entre os programas de SIG, fez surgir o conceito de

bancos de dados geográficos. Em um banco de dados geográficos, as geometrias e as descrições dos elementos que representam as características do mundo real que são armazenadas, gerenciadas e processadas em um único ambiente computacional, o Sistema Gerenciador de Bancos de Dados Relacional (SGBDR). Muitos desses SGBDR's suportam dados geográficos a partir da utilização de drivers específicos, entre eles se pode destacar o *PostgreSQL* com o driver *PostGIS*, o Oracle com os *drivers Spatial* e SDE. O driver tem a função de realizar conversão, inserção, recuperação e extração de dados geográficos junto ao SGBD.

Para Júnior (2004), com às restrições existentes nos SGBD's convencionais no instante do tratamento de tipos de dados complexos, mais exatamente de dados espaciais, iniciaram estudos para integração dos dados espaciais, que armazenam as informações da localização do objeto, e dos dados alfanuméricos, que contem informações que descrevem o objeto.

Segundo Júnior (2004), o Modelo Objeto-Relacional, que integra as funcionalidades do Modelo Relacional com o da Orientação a Objetos, permite a definição de Tipos Abstratos de Dados e a Manipulação de Objetos Complexos. Essas características fazem com que este modelo consiga atender, em grande parte, as exigências impostas pelos SIG's.

2.9.1 Sistema gerenciadores de banco de dados

De acordo com Câmara e Queiros (2005), um SGBD é um *software* ou uma coleção de programas que ajudarão no gerenciamento do Banco de Dados. O SGBD serve para facilitar o processo de definição, construção e manipulação do Banco.

Para Kugler (2010), a principal diferença entre os SIG's é a forma como os dados geográficos são gerenciados. Segundo Kugler há basicamente três diferentes arquiteturas de SIG que utilizam recursos de um SGBD: a) dual, b) integrada baseada em SGBDs relacionais, e c) integrada baseada em extensões espaciais sobre SGBDs objeto-relacionais.

2.10 SISTEMA DE INFORMAÇÃO GEOGRÁFICA

Segundo Câmara (1999), um Sistema de Informação Geográfica ou *Geographic Information System* (GIS) é um termo designado para programas que realizam a manipulação e tratamento computacional de dados geográficos, não se limitando a recuperar apenas dados alfanuméricos, mas também sua localização espacial, disponibilizando ao usuário uma visão mais detalhada. No entanto, se faz necessário que os atributos dos dados e a geometria estejam referenciados geograficamente (dados georreferenciados) e representados em uma projeção cartográfica.

Para Pitz (2001), o avanço da tecnologia e da ciência nos propiciou novas ferramentas, equipamentos de alta precisão e tecnologia como, satélites, sistemas de posicionamento global (GPS), radares e fotografias aéreas, que nos fornecem informações instantâneas. Nesse sentido a adoção de Sistemas de Informação Geográfica são fundamentais para uma rápida e precisa interpretação destas informações.

2.11 POSTGRESQL

Segundo Milani (2008), o PostgreSQL é um SGBDR, que é utilizado para armazenar informações de soluções de informática em todas as áreas de negócios existentes, bem como administrar o acesso a estas informações.

Para o autor o PostgreSQL é um excelente banco de dados com todas as características e propriedades necessárias para atender aos mais exigentes padrões de aplicações do mundo da informática.

De acordo com Milani (2008), tanto o PostgreSQL quanto a licença BSD (*Berkeley Software Distribution*) tiveram origem no mesmo local, a Universidade de Berkeley, na Califórnia. Para o autor esse é o primeiro fator que faz com que o PostgreSQL utilize esta licença, pois os interesses iniciais da ferramenta e da licença tinham algo em comum. Mesmo com o código adquirido mais tarde, sua licença BSD (*Berkeley Software Distribution*) foi mantida e é utilizada até hoje, sendo atualizada e revisada periodicamente. Diferente de muitos *softwares* livres existentes

no mercado, o PostgreSQL não utiliza a licença *GNU* (*General Public License*) para regularizar a sua utilização, mas sim, a licença BSD.

A licença BSD possui menos restrições do que as impostas pela licença GNU, tornando o código muito mais acessível para diversos tipos de atualizações, incluindo a livre utilização da ferramenta até mesmo para fins comerciais.

O *PostgreSQL* supera algumas características do SGBD's no que diz respeito a compatibilidade de sistemas operacionais, a linguagem de programação, a plataforma de desenvolvimento e a versão de SQL utilizada (POSTGRESQL, 2009).

De acordo com Milani (2008), há bibliotecas e drivers de conexão para o PostgreSQL para as principais plataformas e linguagens utilizadas, podendo citar: PHP, C/C++, Java/JSP, ASP, .NET, Perl, Python, Ruby, TCl e Driver ODBC, entre outros.

O PostgreSQL é uma ferramenta extremamente portável, disponibilizando instalação para diversos sistemas operacionais, como por exemplo (linux, unix, mac OS e windows).

O PostgreSQL é um banco de dados que contém as principais características desejadas em um banco de dados:

- recuperação automática após crash de sistema (WAL);
- MVCC (controle de concorrência de multi - versão;
- Logging de transações;
- Commit / Rollback / Checkpoints;
- Triggers / Stored Procedures;
- Constraints / Foreign Keys;
- Backup On-line;
- tamanho ilimitado de registro; e
- múltiplos tipos de Índice;

O PostgreSQL oferece o mais baixo custo total de propriedade (TCO), reduzindo de forma significativa seus custos de administração, suporte e licenciamento e, ao mesmo tempo, fornecendo alta performance, confiabilidade e escalabilidade.

2.11.1 Postgis

Segundo Queiros (2004), o PostGIS é uma extensão do PostgreSQL que fornece suporte de banco de dados espacial para o PostgreSQL. Isso significa que ele otimiza PostgreSQL para armazenar e consultar dados relacionados a objetos e adicionando suporte para as três características: tipos espaciais, índices e funções.

O PostGIS trabalha com objetos espaciais dos tipos: *Polygon*, *Multipolygon*, *Point*, *Multipoint*, *Linestring*, *Multilinestring* e *Geometrycollection*.

Na Figura 1, podem ser visualizados os tipos de dados espaciais suportados pela extensão espacial PostGIS.

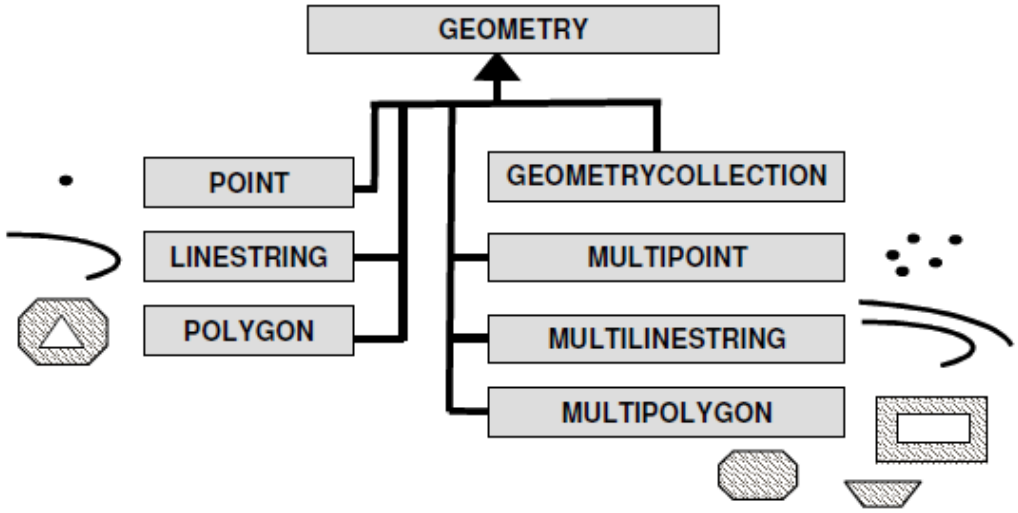

Figura 1 - Tipos de Dados Geométricos
Fonte – Queiros, 2004

Segundo Queiros (2004), um dos padrões utilizados para representar objetos espaciais, é a forma *Well-Known Text* (WKT), que inclui informações sobre o tipo de objeto e suas coordenadas, os quais determinam os valores utilizados nas colunas das tabelas, Os principais tipos de objetos são listados a seguir junto com sua representação na forma WKT:

- point: POINT(-20.1929 -44.11221);

- linestring: LINESTRING(0 0,1 1,1 2);

- polygon :POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1));

- multipoint: MULTIPOINT(0 0,1 2);

- multilinestring : MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4));

- multipolygon : MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)));

- geometrycollection: GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4));

Para Andrade(2011), o PostGIS possui várias funções espaciais que equivalem às operações de agregação e junção num banco de dados relacional. Elas são baseadas em relacionamentos espaciais como: determinação de topologia entre dois objetos, aritmética de polígonos, cálculo de área e etc.

Algumas das principais funções de análises espaciais no PostGIS são:

- ST_Disjoint(obj1, obj2): Analisa se dois objetos possuem pontos em comum e retorna verdadeiro em caso positivo;

- ST_Intersects(obj1, obj2): Analisa se dois objetos possuem alguma intersecção e retorna verdadeiro caso haja;

- ST_Within(obj1, obj2): Analisa se um objeto está completamente dentro do segundo objeto;

- ST_Crosses(obj1, obj2): Analisa se dois objetos se cruzam e retorna verdadeiro caso ocorra; e

- ST_asText(Geometria): Retorna a representação WKT(*Well-Known Text*) da geometria.

2.12 GOOGLE MAPS

Para Erle e Gibson (2006), Google Maps é um serviço do Google que oferece uma poderosa tecnologia de mapas amigáveis e informações de locais, incluindo a localização, informações de contatos e direções de condução.

Segundo Erle e Gibson (2006), Google Maps foi desenvolvida inicialmente por dois irmãos, Lars e Jens Rasmussen, co-fundadores de *Where* 2 *Technologies* uma empresa dedicada a criação de soluções de mapeamento. A empresa foi comprada pela Google em outubro de 2004, e logo depois os dois irmão criaram Google Maps.

Antes de que tivesse uma API pública, alguns desenvolvedores descobriram uma maneira de hackear Google Maps para incorporar os mapas ao seus próprios sites, Isso levou a Google a conclusão que havia a necessidade de uma API pública, e no início de 2005 nas principais localidades dos EUA e posteriormente se expandiu e passou a servir de referência para a busca de endereços e pontos de interesse nos demais centros urbanos de outras nações e continentes - inclusive cobrindo várias cidades brasileiras (GOOGLE, 2012).

Com o passar do tempo adicionou novas funcionalidades aos usuários, gerando comodidade e facilidades nunca antes oferecidas, que vão desde o cálculo de rotas, visualização 3D de ruas e edificações, até informações sobre tráfego e sobre o transporte público.

Para Purvis e Sambells (2006), o grande sucesso e aceitação dos usuários, pouco depois do lançamento oficial do Maps, foi lançada a sua API (Application Programming Interface), que permite aos usuários inserir mapas em suas páginas web, contando com a possibilidade de personalização e customização dos mapas como bem entenderem.

Para Erle e Gibson (2006), a funcionalidade principal do Google Maps é a exibição de um mapa no website, partindo de uma coordenada que é exibida centralizada na tela. Só isso já basta para usuários que buscam ajuda para localização de ruas e regiões aos redores do endereço fornecido. No entanto a

ferramenta não se resume somente a isso, e para a aplicação pretendida nesse projeto nos utilizaremos também marcadores, que podem inclusive ser carregados a partir de um banco de dados e exibidos de forma fixa no mapa a fim de marcar a posição de pontos de interesse do usuário. Como meio de facilitar o entendimento por parte do usuário a visualização do mapa pode ser feita tanto do modo cartográfico - onde aparecem as ilustrações das ruas e quadras - como do modo satélite que exibe uma imagem aérea da área selecionada.

A API do Google Maps é disponibilizada à todos os sites que tem acesso gratuito para qualquer usuário, mas também pode ser utilizada por websites comerciais ou sem fins lucrativos desde que de acordo com os Termos de Serviço estabelecidos pela Google. Ainda há a possibilidade de uso por sites que cobram por serviços e utilizem o Google Maps, ou sites disponibilizados somente em redes restritas e intranets.

Segundo Erle e Gibson (2006), alguns dos benefícios básicos de mapas do Google é que ela é uma importante fonte de visitantes para as empresas geograficamente específicas que tradicional recebem uma grande quantidade de seu costume de diretórios locais. Ao adicionar os mapas do Google para o seu site, ele permite que os usuários acessem o conteúdo interativo, dando altamente responsiva representação visual de seu local de negócios a fim de obter o seu interesse. É também fácil de usar, com ele os usuários podem atingir vários locais desejados devido à sua capacidade de obter direções com base em modo de usuário de viagem e lhes permite adicionar um novo destino para suas rotas com apenas um clique, sempre que o usuário precisa de indicações, que incluem várias paradas. Google maps simplicidade é o principal benefício que não pode ser negligenciado, pois garante apenas coisas úteis sem confundir os usuários com sinais indesejados.

Google Maps é o aplicativo de serviço livre e tecnologia para mapeamento web fornecido pela empresa Google. Antes do Google Maps, era difícil de pesquisar ou planejar uma viajem por meio de a pé, carro ou ônibus. Mas o Google Maps torna mais fácil, oferecendo os mapas de ruas para viajar a pé, de carro ou transporte público, fornece três visualizações diferentes. Existe uma visualização do mapa normal, uma vista de imagem por satélite e uma vista terra (Google Earth) para visualizar imagens e terrenos em 3D para poder obter uma vista panorâmica dessas imagens e incliná-las, dependendo da necessidade do utilizador. Ela não só fornece

altamente receptivo, interface de mapeamento intuitiva com dados detalhados de rua incorporados, mas além disso, oferece aos usuários mapas controles embutidos nos produtos, para ter total controle sobre a exibição de rua e mapa de navegação. Google Maps combina todos os dados geográficos em um único sistema rápido, de fácil utilização acessado por todos os usuários de todos os departamentos.

3 MATERIAL E MÉTODOS

Neste capítulo estão descritos os materiais e métodos utilizados para o desenvolvimento da aplicação, do estudo experimental, para ilustrar a integração das tecnologias mostradas anteriormente. Essa aplicação consiste de um sistema onde o usuário pode marcar pontos e polígonos em um mapa e cadastrar os dados referente ao ponto ou polígono.

A aplicação foi desenvolvida utilizando as tecnologias PHP, HTML, CSS, JavaScript, JQuery e JSON, foram desenvolvidas através da ferramenta Netbeans na versão 7.0, o banco utilizado foi PostgreSQL na versão 8.4 com a utilização da ferramenta PgAdmin III na versão 1.10.5, que é a parte administrativa para o banco PostgreSQL.

Para o ambiente de teste da aplicação nesse estudo foi utilizado um servidor local através do *wamp server* 2.0, um pacote de distribuição Apache contendo PHP, POSTGRESQL, entre outros. Nesta versão estão disponíveis o Apache 2.2.11, PHP 5.3.0 e PostgreSQL 8.4 porem o PostgreSQL é instalado separadamente e é ativado a extensão do mesmo no *wamp server*.

3.2 ESTRUTURA DA APLICAÇÃO

Durante o desenvolvimento da aplicação como citado anteriormente foi utilizado a ferramenta Netbeans na versão 7.0 e para realização dos teste o servidor local *Wamp Server* 2.0.

3.2.1 Conexão com o banco

Para visualização e cadastro da aplicação, precisa ser estabelecida uma conexão com o banco, para obter os dados no caso da visualização ou inserir em caso de cadastro.

Por meio da Figura 2, pode-se visualizar o código de conexão com a base de dados:

```php
<?php

    $SERVER ="localhost";
    $BANCO  ="TCC";
    $PORTA  ="5432";
    $USER   ="postgres";
    $SENHA  ="peterson";

    $BD_CON = "host=$SERVER port=$PORTA dbname=$BANCO user=$USER password=$SENHA";
    $CONNECT = pg_connect($BD_CON);
    if(!$CONNECT) die("Erro na Conexão!!");

?>
```

Figura 2 – Parte do código para conexão com o banco de dados

A função **pg_connect()** e responsável pela conexão com o banco de dados para a realização da conexão é necessário informar os parâmetros como servidor, banco, porta para a conexão, usuário e senha.

pg_connect() abre uma conexão com um servidor de banco de dados PostgreSQL especificado por **connection_string**. Retorna um recurso (resource) de conexão em caso de sucesso. Retorna FALSE se a conexão não pôde ser estabelecida. **connection_string** deve ser uma string entre aspas duplas (PHP, 2013).

3.2.2 Integração com Google Maps API v3

Para a criação e visualização dos pontos e polígonos, foram criados três arquivos JavaScript: polygon.js, marker.js e main.js. Esses arquivos implementa todas as funções que foram utilizadas no desenvolvimento da aplicação e é carregado junto com a página principal da aplicação. Para ter um melhor entendimento da proposta do trabalho será mostrado partes dos código que referem-se às funções do *JavaScript.*

A figura 3 mostra uma parte do código da função **initialize()** que é chamado no momento em que a pagina é carregada e executa as funções que criam o mapa e exibe o mesmo na tela.

Nas linhas 13 a 17 é criado uma variável **mapOptions** onde será armazenadas as opções do mapa para configurar a aparência. Contendo zoom inicial, ponto central e tipo do mapa. Na linha 14 representa o nível de aproximação

do mapa, na linha 15 a opção **center** recebe um objeto da classe ***google.maps.LatLng***, onde os parâmetros são valores de latitude e longitude que representam a localização inicial do mapa, na linha 16 a opção **mapTypeId** exibe o tipo de mapa que são blocos 2D padrão do Google Maps, na linha 19 define uma variável **map** e atribui essa variável a um novo objeto da classe ***google.maps.Map***, também são enviadas as opções definidas para o mapa na variável **mapOptions**. Essas opções serão usadas para iniciar as propriedades do mapa.

```
11  function initialize() {
12
13    var mapOptions = {
14      zoom: 13,
15      center: new google.maps.LatLng(-24.953923,-53.458112),
16      mapTypeId: google.maps.MapTypeId.ROADMAP
17    }
18
19    var map = new google.maps.Map(document.getElementById("mapa"), mapOptions);
20
```

Figura 3 - Parte do código para inicializar o mapa

Um componente do Google Maps utilizado é o **infowindows** que não foi somente utilizado para exibir informações, no nosso caso também foi utilizado para carregar um formulário onde será enviada informações para serem persistidas, conforme é mostrado na figura 4.

Na linha 30 a linha 51 é criado o formulário com informações básicas para o cadastro, que serão utilizadas para o cadastro de pontos e polígonos, na linha 53 a 55 o formulário é adicionado a **infowindows**, para posteriormente ser exibida quando for adicionado um ponto ou um polígono.

```
30        var html  =  "<form id=\"dialog\" method=\"get\" action=\"gravaPontos.php\">"+
31                     "<p class=\"endereco\"><span><label for=\"endereco\">Endereço:</label></span><input type=\"text\" name=\"endereco\" id=\"endereco\" />
32                     "<p class=\"bairro\"><label for=\"bairro\">Bairro:</label> <input type=\"text\" name=\"bairro\" id=\"bairro\"/></p>"+
33
34                               "<label for=\"tipo\">Tipo:</label>"+
35                               "<select id='tipo'>"+
36                                 "<option value='selecione' SELECTED>--Selecione--</option>"+
37                                 <?php while ($row = pg_fetch_row($resultado_tipoimovel)) { ?>
38                                 "<option value='<?php echo $row[0]?>'> <?php echo $row[1] ?> </option>"+
39                                 <?php }?>
40                               "</select>"+
41
42                               "<label for=\"finalidade\">Finalidade:</label>"+
43                              "<select id='finalidade'>"+
44                                "<option value='selecione' SELECTED>--Selecione--</option>"+
45                                <?php while ($row = pg_fetch_row($resultado_finalidade)) { ?>
46                                "<option value='<?php echo $row[0]?>'> <?php echo $row[1] ?> </option>"+
47                                <?php }?>
48                              "</select>"+
49                            "<p class=\"submit\"><input type='button' id='btnGravar' value='Adicionar' onclick='salvarDados()' /></p>"+
50                            "<div class=\"resposta\"></div> "+
51                            "</form>";
52       //recebe o formulario html
53    infowindow = new google.maps.InfoWindow({
54     content: html
55    });
```

Figura 4 - Parte do código onde é criado o formulário

A função que permite criar pontos no mapa é mostrada na figura 5, entre a linha 57 e 67 é definido um evento “click” no mapa, onde a linha 59 é o ponto onde foi clicado no mapa, na linha 60 define em qual mapa será adicionado o ponto, na linha 61 é o ícone que irá aparecer que no nosso caso foi alterado não é o ícone padrão. Entre a linha 64 e 66 outro evento “click” porem não e mais no mapa e sim no ponto em que acabamos de adicionar, que ao ser clicado ira abrir a **infowindows** com o formulário que foi criado conforme a figura 5, na linha 65 é executado o função **infowindows.open** que abre um balão com o formulário de cadastro.

```
57          google.maps.event.addListener(map, "click", function(event) {
58              marker = new google.maps.Marker({
59                position: event.latLng,
60                map: map,
61                icon: image
62              });
63
64              google.maps.event.addListener(marker, "click", function() {
65                 infowindow.open(map, marker);
66              });
67          });
```

Figura 5 - Função para criar pontos e exibir infowindows

Na figura 4 apresentada anteriormente na linha 49 na ação “onClick” do botão a chamada da função **salvarDados()**, o código dessa função e mostrado na figura 9, entre as linhas 72 e 76 recupera os valores dos inputs através do **name** e é

adicionado em uma variável , na linha 77 para recuperar a latitude e longitude e executa a função **marker.getPosition()**, no intervalo das linhas 79 a 91 é criado uma função *Ajax* onde os parâmetros são passados pelo método **get** na url **gravarPontos.php** e atribui os valores recuperados entre as linhas 73 e 77 para que possam ser persistidos.

```
71      function salvarDados(){
72
73        var bairro = $("input[name=bairro]").val();
74        var endereco = $("input[name=endereco]").val();
75        var tipo = $("#tipo").val();
76        var finalidade = $("#finalidade").val();
77        var latlng = marker.getPosition();
78
79                    $.ajax({
80                            type: "GET",
81                            url: "gravarPontos.php",
82                            data:  "endereco=" + endereco +
83                                   "&bairro="  + bairro   +
84                                   "&tipo=" + tipo +
85                                   "&finalidade=" + finalidade +
86                                   "&lat="+ latlng.lat() +
87                                   "&lng="+ latlng.lng(),
88                            success: function (data) {
89                                alert("Gravado com sucesso!");
90                            }
91                    });
92
93      }
```

Figura 6 - Método onde contem a função ajax para persistir os dados

Na figura 7 mostra parte do código do arquivo **gravarPontos.php**, na linha 14 cria uma String com a SQL para gravar o ponto no banco, lembrando que a coluna da tabela pontos é do tipo *Geometrycollection* que aceita qualquer tipo de objeto desde que seja do tipo ponto ou polígono que é as duas formas geométricas que serão persistidos nessa tabela, e desde que estejam no formato WKT que incluem informações sobre o tipo de objeto e suas coordenadas.

Como o registro a ser persistido é do tipo ponto é adicionado um objeto do tipo **point** passando a latitude e longitude, na linha 17 chama a função **pg_query()** para persistir os dados.

```
14  $sql_pontos = "INSERT INTO pontos(localizacao) VALUES('POINT($lat $lng)')";
15
16  //Chamando a função, e passando como parâmetro a String de SQL
17  $query_pontos = pg_query($sql_pontos) or die(pg_last_error());
18
```

Figura 7 - Parte do código para gravar pontos

Para desenhar um polígono no mapa os procedimentos são semelhantes, no lugar de criar um objeto do tipo **google.maps.Marker** é criado um objeto do tipo **google.maps.Polygon** conforme e mostrado na figura 8, onde na linha 31 **strokeWeight** é a largura da linha do polígono, na linha 32 **fillColor** é a cor do interior do polígono e a linha 33 **strokeColor** é a cor da linha do polígono.

```
29              poly = new google.maps.Polygon(
30              {
31                  strokeWeight: 1,
32                  fillColor: '#3B00FF',
33                  strokeColor: '#3B00FF'
34              });
```

Figura 8 - Parte do código usada para criar um polígono

A função **salvarDados()** do polígono também é semelhante a função para salvar os pontos, única diferença que no lugar de passar somente uma latitude e uma longitude é passada uma *array* com todas as latitudes e longitudes do polígono criado, a figura 9 mostra o trecho de código para criar um *array* com as coordenadas de um polígono, na linha 116 é criado um *array* para adicionar os pontos do polígono, entre as linhas 117 e 120 é percorrido o for com a quantidades de pontos que contem o polígono e é adicionado em cada posição a latitude e longitude de cada ponto e na linha 122 é adicionado o primeiro ponto para fechar o polígono para deixar no formato WKT, na linha 126 a *url* não é mais **gravarPontos.php** e sim **gravarPoligono.php** e nos dados na linha 128 é passado uma *array* com as coordenadas.

```
116                 var latlng = new Array();
117                 for(i=0; i<markers.length; i++)
118                 {
119                   latlng[i] = markers[i].getPosition().lat() + " " + markers[i].getPosition().lng();
120                 }
121                 // fecha o polígono
122                 latlng[markers.length] = markers[0].getPosition().lat() + " " + markers[0].getPosition().lng();
123 
124                                 $.ajax({
125                                         type: "GET",
126                                         url: "gravarPoligono.php",
127                                         data:  "endereco=" + endereco + "&bairro="+ bairro  + "&tipo="
128                                                 + tipo + "&finalidade=" + finalidade + "&pontos[]="+ latlng,
129                                         success: function (data) {
130                                             alert("Gravado com sucesso!");
131                                              infowindow.close();
132                                         }
133                                 });
134             }
```

Figura 9 - Criar um *array* de um polígono e chamada ajax para persistir os dados

4 RESULTADOS E DISCUSSÃO

Este capítulo apresenta os resultados obtidos durante o desenvolvimento da aplicação *Web* utilizando a API do Google Maps.

4.2 ESTRUTURA DO BANCO DE DADOS

Para o desenvolvimento da aplicação *Web* optou-se por utilizar o banco de dados PostgreSQL, pelo fato do mesmo ser frequentemente utilizado em aplicações que utilizam geoprocessamento. A estrutura do banco de dados da aplicação consiste em quatro tabelas que armazenarão os dados do imóvel, tipo do imóvel, finalidade e localização.

Segundo PONSONI (2009), o MER (Modelo Entidade-Relacionamento) tem o objetivo de representar as estruturas de dados da forma mais próxima dos negócios, onde existem três conceitos:

- Entidade: são os objetos;
- Atributos: as características dos objetos; e
- Relacionamentos: é a relação entre os objetos.

A figura 10, representa o diagrama de entidade e relacionamento, onde contem as tabelas do banco de dados.

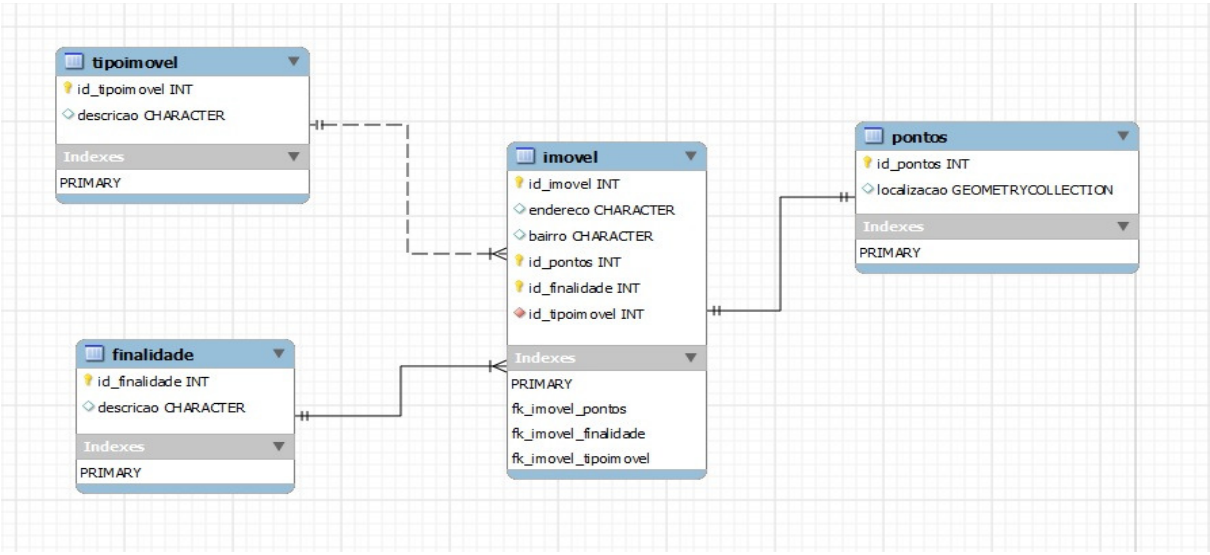

Figura 10 - Diagrama de entidade e relacionamento

4.3 ANÁLISE E PROJETO

A partir do diagrama de seqüência é possível conhecer as funcionalidades da aplicação. Para conhecer melhor o processo e a ordem que as mesmas são executadas foram levantados alguns requisitos funcionais como:

- o sistema deve permitir que o usuário crie pontos no mapa; e
- o sistema deve permitir que o usuário crie polígonos no mapa.

4.3.1 Caso de uso: Adicionar ponto

Esse caso de uso refere-se a opção de marcar um pontos no mapa, nesse caso de uso o ator é o usuário que utiliza o sistema.

O fluxo de eventos primários respeita a seguinte ordem:

1. Usuário seleciona o menu "Incluir ponto";
2. Usuário clica sobre o mapa para adicionar um ponto;
3. Sistema posiciona a imagem de um ícone sobre o ponto salvo no mapa;
    3.1. O usuário clica sobre o ponto adicionado;
    3.2. Sistema abre um formulário;
4. Usuário preenche os campos do formulário;
5. Usuário submete os dados do formulário;
    5.1. Sistema valida as informações;
6. Sistema salva os dados no banco;

O diagrama de seqüência da figura 11 mostra a interação do usuário no processo de marcar um ponto e cadastrar as informações.

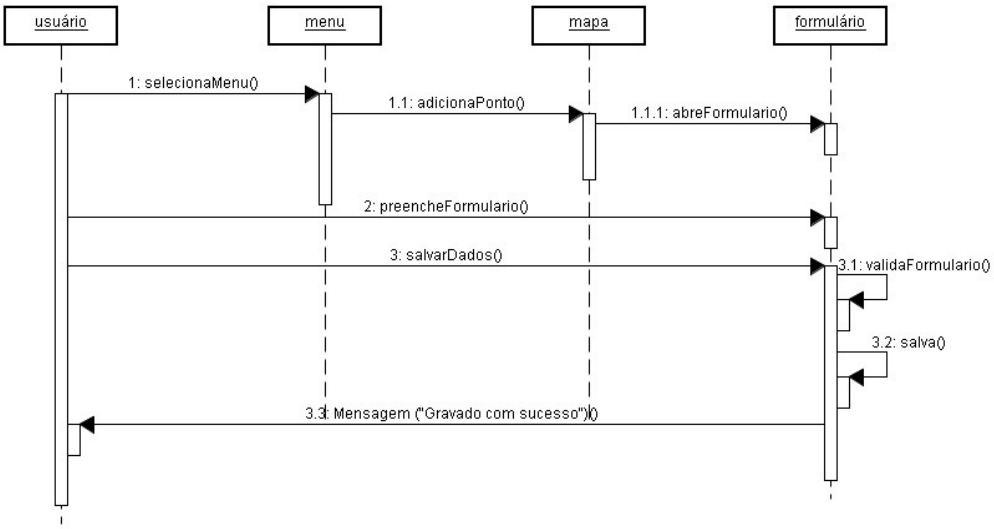

Figura 11 - Diagrama de seqüência adicionar ponto

4.3.2 Caso de uso: Adicionar polígono

Esse caso de uso refere-se a opção de marcar um polígono no mapa, nesse caso de uso o ator é o usuário que utiliza o sistema.

O fluxo de eventos primários respeita a seguinte ordem:

7. Usuário seleciona o menu "Incluir polígono";
8. Usuário clica sobre o mapa para adicionar um polígono;
9. Sistema posiciona polígono salvo no mapa;
    9.1. O usuário clica sobre o polígono adicionado;
    9.2. Sistema abre um formulário;
10. Usuário preenche os campos do formulário;
11. Usuário submete os dados do formulário;
    11.1. Sistema valida as informações;
12. Sistema salva os dados no banco;

O diagrama de seqüência da figura 12 mostra a interação do usuário no processo de marcar um polígono e cadastrar as informações.

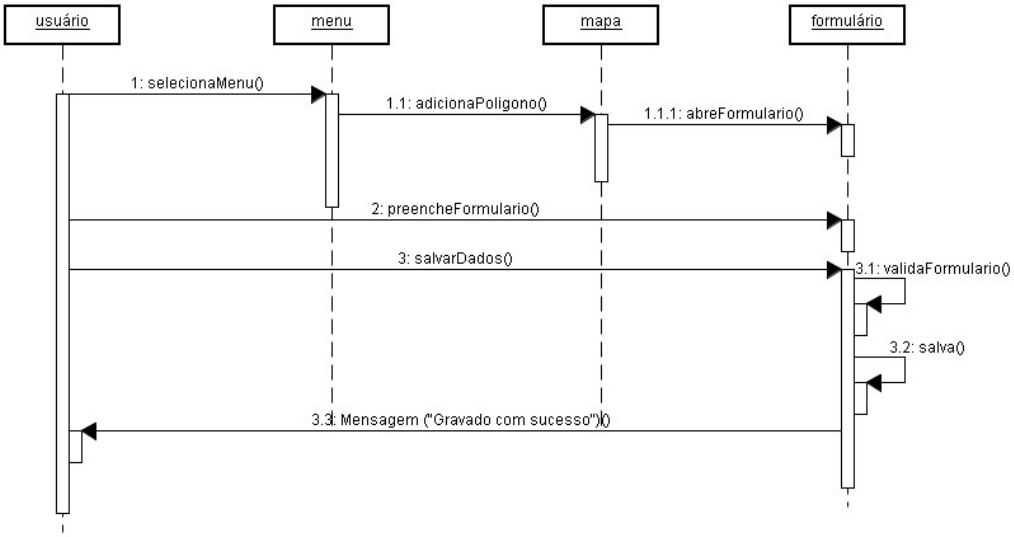

Figura 12 - Diagrama de seqüência adicionar polígono

4.4 APLICAÇÃO WEB

A aplicação possui um layout simples e intuitivo. Buscou-se utilizar cores no tom de azul, para não tornar a mesma cansativa e desagradável aos olhos do usuário. A figura 13 apresenta a página inicial da aplicação onde contem todos os pontos e polígonos já cadastrados.

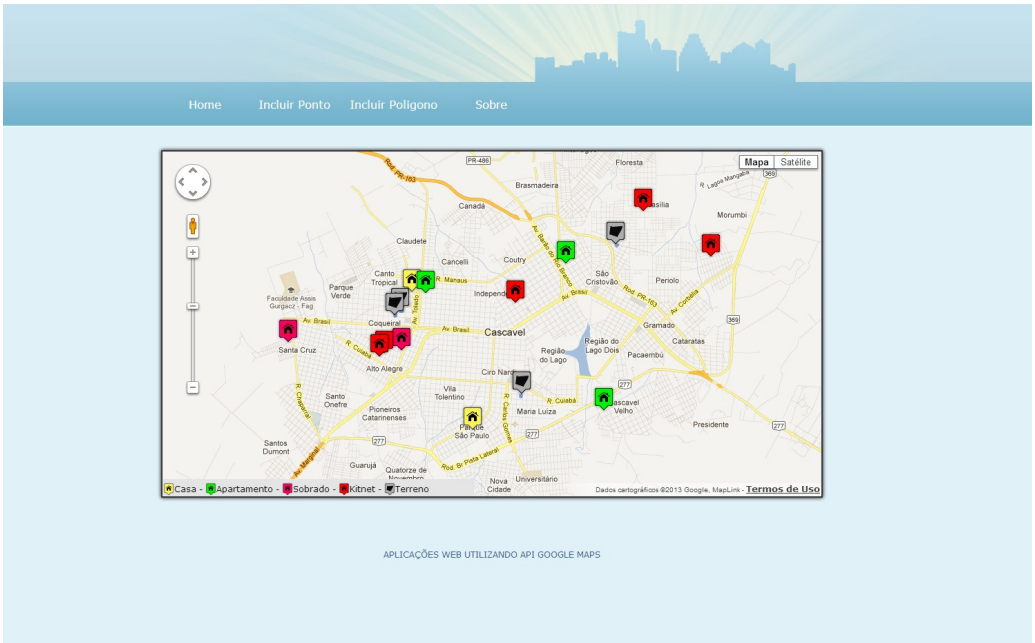

Figura 13 - Página inicial

Ná figura 14 onde apresenta todos os pontos e polígonos cadastrados, ao clicar sobre um icone de imovel (ponto) ou terreno (polígono), é aberto um *infowindows* com as informações do imovel, a figura 16 mostra detalhes de quando é clicado sobre um icone de um imovel, os imoveis são divididos por categorias e as mesmas podem ser vistas com detalhes na figura 16, onde na parte destacada em vermelho, que cada categoria é separada por uma cor para melhor visualização dos imoveis no mapa.

Figura 14 - *Infowindows* quando é clicado no ponto ou polígono

Para exibição dos pontos é feito uma função ajax conforme figura 15, onde na linha 94 chama a url **recuperarPontos.php,** e a linha 100 é o retorno com o resultado obtido.

```
92              $.ajax({
93                  "type": "GET",
94                  'url': 'recuperarPontos.php',
95                  'dataType': "json",
96                  'success': function (data) {
97                      json = data;
98                  }
99              });
100             return json;
101         });
```

Figura 15 - Função ajax para buscar os pontos

Na figura 16 é mostrado um trecho do codigo para criar o JSON, com os pontos e os dados de cada imóvel. Na linha 25 é adicionado em um array os dados retornados na consulta.

```
25          $dados[] = array(
26                "endereco"=>"$linha[0]",
27                "bairro"=>"$linha[1]",
28                "finalidade"=>"$linha[2]",
29                "tipoImovel"=>"$linha[3]",
30                "lat"=>"$lat",
31                "lng"=>"$lng"
32           );
```

Figura 16 - Parte do código usada para montar um json de pontos

A figura 17 retorna a string contendo a representação JSON criada no código representado na figura 18.

```
37          echo json_encode($dados);
```

Figura 17 - Código utilizado para codificar um JSON

Já na figura 18 é o resultado no formato *string* da função **json_encode** que está presente a partir da versão 5.2 do PHP que retorna uma representação JSON de um valor, esta função só funciona com dados codificados em UTF-8. O retorno para um dado não codificado é *undefined*.

```
[
    {
        "endereco": "Nereu ramos 2570",
        "bairro": "Centro",
        "finalidade": "venda",
        "tipoImovel": "casa",
        "lat": "-24.9468416570702",
        "lng": "-53.4748739004135"
    },
    {
        "endereco": "R. rui naipi",
        "bairro": "Centro",
        "finalidade": "Locação",
        "tipoImovel": "Apartamento",
        "lat": "-24.9465303661085",
        "lng": "-53.4783500432968"
    }
]
```

Figura 18 - Representação JSON de pontos

O trecho de código da figura 19 mostra como é adicionado os pontos no mapa apartir do JSON que foi retornado pela função ajax. Na linha 131 é o inicio do for que percorre todos os locais, na linha 132 a variavel **imovel** recebe a latitude e longitude e adiciona num objeto **google.maps.LatLng**, entre o intervalo das linhas 143 e 140 é criado o **infowindows** para mostrar os detalhes do imóvel, na linha 144 é criado um objeto do tipo **google.maps.Marker**, onde é adicionado a latitude e longitude que contem na variavel **latLng**, seta o mapa, o icone referente ao tipo de imovel, e um titulo quando o mouse passar sobre o ponto que mostra sé o imovel é para venda ou locação.

```
131             for (var i = 0; i <= locais.length; i++) {
132               var imovel = locais[i];
133               var latLng = new google.maps.LatLng(imovel.lat, imovel.lng);
134               var html =   "<table><tr><td rowspan=\"5\"> <img src=\"images/casa_padrao.png\"/></td>"+
135                            "<td style=\"font-weight:bold\">Endereço</td><td>"+imovel.endereco+"</td></tr>"+
136                            "<tr><td style=\"font-weight:bold\">Bairro:</td><td>"+imovel.bairro+"</td></tr>"+
137                            "<tr><td style=\"font-weight:bold\">Tipo imóvel:</td><td>"+imovel.tipoImovel+"</td></tr>"+
138                            "<tr><td style=\"font-weight:bold\">Finalidade:</td><td>"+imovel.finalidade+"</td></tr>"+
139                            "<tr><td></td><td colspan=\"2\" style=\"text-decoration: none; float: right;\">"+
140                            "<a href=\"#\" class=\"detalhes\">Detalhes</a></td></tr>";
141
142
143               var icon = icones[imovel.tipoImovel] || image;
144               var marker = new google.maps.Marker({
145                   position: latLng,
146                   map: map,
147                   icon: icon.icon,
148                   title: imovel.finalidade
149               });
150               bindInfoWindow(marker, map, infoWindow, html);
151             }
```

Figura 19 - Adicionando pontos e criando *infowindows*

Para adicionar um polígono e semelhante, a função que faz a requização ajax para gerar o JSON é semelhante a da figura 15 unica difereça é que a url é **recuperarPoligono.php**, um trecho de código para gerar um JSON de polígono é mostrado na figura 20, a diferença é que não passa mais a latitude e longitude em uma *string* **lat** e **lng** é sim uma string **coordenadas** onde recebe um array com todas as coordenadas do polígono.

```
44              $dados[] = array(
45                    "endereco"=>"$linha[0]",
46                    "bairro"=>"$linha[1]",
47                    "finalidade"=>"$linha[2]",
48                    "tipoImovel"=>"$linha[3]",
49                    "coordenadas" => $pontos
50                );
```

Figura 20 - Parte do código usada para montar um JSON de polígono

Na figura 21 é a representação JSON de polígono.

```
{
    "endereco": "R. Taroba",
    "bairro": "Centro",
    "finalidade": "Venda",
    "tipoImovel": "lote",
    "coordenadas": [
        {
            "lat": "-24.9695443197608",
            "lng": "-53.4510015696287"
        },
        {
            "lat": "-24.9695151416479",
            "lng": "-53.4514575451612"
        },
        {
            "lat": "-24.9699625386176",
            "lng": "-53.4514897316694"
        },
        {
            "lat": "-24.9699965796248",
            "lng": "-53.4510337561369"
        },
        {
            "lat": "-24.9695443197608",
            "lng": "-53.4510015696287"
        }
    ]
}
```

Figura 21 - Representação JSON de polígono

Na figura 22 dentro do laço for é percorrida a variável **coordenadas.length** e é criado um objeto do tipo **google.maps.LatLng** passando a latitude e longitude e adicionando em uma variável, na linha 51 é utilizado o método **push** que adiciona a coordenadas na ultima posição.

```
47                  var coordenadas = imovel[i].coordenadas;
48                  var path = [];
49                  for (var j = 0; j < coordenadas.length; j++) {
50                       var latlng = new google.maps.LatLng(coordenadas[j].lat,coordenadas[j].lng);
51                        path.push(latlng);
52                  }
```

Figura 22 - Adicionando as vértices do polígono

Ao fim da sequência de repetição, é executado o código da figura 23, onde na linha 58, o polígono recebe a coleção de coordenadas que servirão de vértices para o polígono. E por fim, a linha 67 representa a parte do código que faz o polígono aparecer no mapa.

```
56                  paths.push(path);
57                  var opcoesPolygono  = {
58                      paths: paths,
59                      strokeColor: '#7694D1',
60                      strokeOpacity: 0.8,
61                      strokeWeight: 3,
62                      fillColor: "#7694D1",
63                      fillOpacity: 0.35
64                  };
65
66                   var poly = new google.maps.Polygon(opcoesPolygono);
67                   poly.setMap(map);
```

Figura 23 - Parte do código para adicionar polígono no mapa

Na parte de cadastros de pontos e polígonos, quando é adicionado um ponto no mapa ao clicar sobre esse ponto é aberto uma *infowindows* com um formulário onde podem ser adicionadas informações para o cadastro, o resultado pode ser visualizado conforme Figura 24.

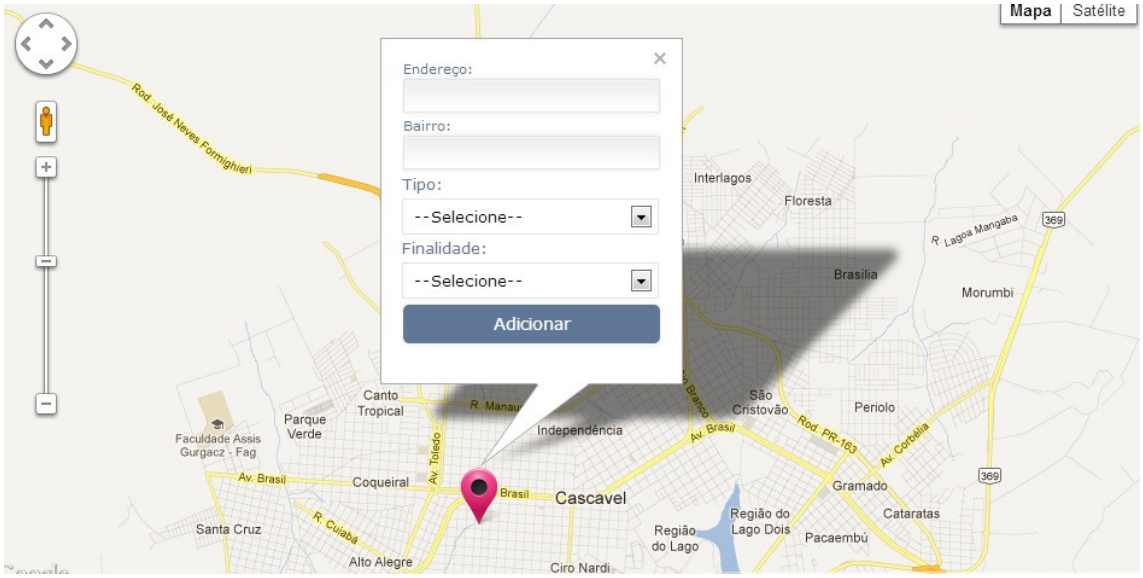

Figura 24 - Formulário para adicionar um ponto

O mesmo procedimento ocorre quando é adicionado um polígono no mapa, quando ao clicar sobre o polígono é aberta uma *infowindows* com um formulário conforme Figura 25.

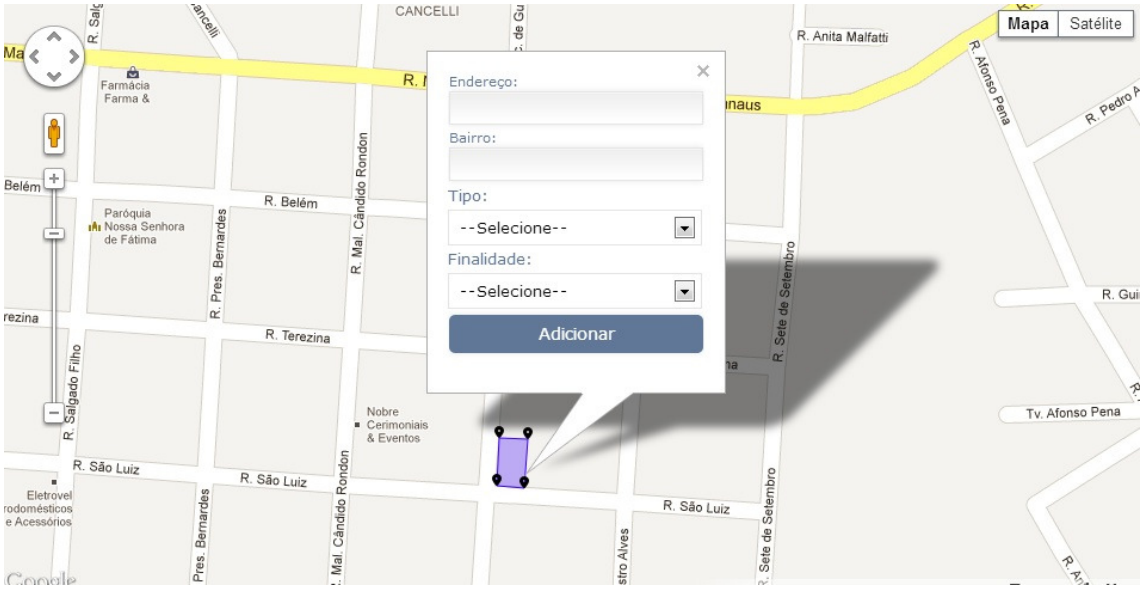

Figura 25 - Formulário para adicionar um polígono

5 CONCLUSÃO

A aplicação desenvolvida nesse trabalho mostra como é simples o uso de API's como a API Google Maps. É importante ressaltar a experiência que o desenvolvedor adquire ao trabalhar em projetos que utilizem serviços estáveis e popularizados como o da Google.

Também demonstrou a eficiência do uso do banco de dados PostgreSQL com php visto que o mesmo é mais utilizado com a linguagem MySQL, e para que o mesmo funcionasse teve que ativar as extensões do PostgreSQL.

Outras tecnologias também foram úteis para agilizar o processo de desenvolvimento do projeto, como AJAX, JSON e JQuery, onde conseguiu-se através dessas tecnologias uma agilidade no tempo de desenvolvimento.

Durante o processo de desenvolvimento foi gerado uma chave, que é disponibilizada pela própria Google para qualquer usuário que tenha uma conta, com a chave em mãos qualquer usuário pode ter acesso a API e um total funcionamento do modulo de mapas.

Esse trabalho permitiu verificar que pode ser desenvolvido um sistema que use serviços disponíveis gratuitamente na Internet, deixando a aplicação com bom desempenho e de fácil manutenção.

5.2 TRABALHOS FUTUROS

Com esse trabalho, surgiu a idéia de implementar uma aplicação mais robusta e com mais funcionalidades para o ramo imobiliário, com design responsivo, que além da aplicação WEB também contenha uma aplicação móvel desenvolvida em ANDROID que faça uso da API Google Maps, que além do conteúdos tenha uma opção de rota onde irá traçar uma rota do local onde se encontra até o ponto do imóvel de interesse, mostrando detalhes de distancia e melhor caminho, para ter acesso a esse recurso o sistema irá interagir com o Google Maps Navigator.

6 REFERÊNCIAS BIBLIOGRÁFICAS

AJAX. Disponível em: <http://api.jquery.com/jQuery.ajax/>. Acessado em 01/01/2013.

CÂMARA, Gilberto, QUEROZ, Gilberto Ribeiro de. **Arquitetura de Sistemas de Informação Geográfica**. Disponível em: <http://www.dpi.inpe.br/gilberto/livro/introd/cap3-arquitetura.pdf> Acesso dia: 10/12/2012.

CARLOS, João. Diagramas: Sequência e Atividades, 2005. Disponível em: <http://imasters.com.br/artigo/3004/uml/diagramas-sequencia-e-atividades> Acesso em 26/01/2013.

CSS 3 Preview. Disponível em:<www.css3.info/preview>. Acesso em 10/09/2012.

DANTAS, Mário. Tecnologias de Redes de Comunicação e Computadores 2002. Acessado em 10/12/2012.

DALL'OGLIO ,Pablo. PHP **Programando com Orientação a Objetos.** Novatec. 2009.

DARLAN, Diego. **O que é PHP**, 2007. Disponível em: <http://www.oficinadanet.com.br/artigo/659/o_que_e_php> Acesso em: 20/12/2012.

ERLE, Schuyler; GIBSON, Rich. Google Maps Hacks. 2006. Editora O'Reilly. Acessado em 10/12/2012

FERREIRA, Nilson. Apostila de sistemas de informação geográficas, 2006. Acessado em 10/12/2012.

GOOGLE. Google Maps Api. Disponível em <http://code.google.com/intl/pt-BR/apis/maps/documentation/>. Acessado em 25/10/2012.

JSON. Disponível em: < http://www.json.org/> Acessado em 12/01/2013.

JSON/PHP. Disponível em: < http://php.net/manual/en/book.json.php> Acessado em 12/01/2013.

JQUERY. Disponível em: <http://jquery.com/> Acessado em 19/01/2013.

MANUAL DE PHP. Disponível em: <http://www.php.net/manual/en/preface.php>. Acessado em 25/10/2012.

MELONI, Julie C. **Fundamentos de PHP**. Rio de Janeiro: Ciência Moderna, 2000.

PITZ, Jean. Sistemas de informação geográfica, 2001. Acessado em 10/12/2012.

PONSONI, Viviane. MER e DER, 2009. Disponível em: <http://www.devmedia.com.br/mer-e-der/14332> Acessado em 26/01/2013.

POSTGIS DOC. PostGIS 1.5.0 Manual. Disponível em : <http://postgis.refractions.net/docs/> Acesso dia 10/12/2012.

POSTGRESQL. Disponível em: <http://www.postgresql.org/docs/8.2> Acessado em 10/12/2012.

PURVIS, Michael; SAMBELLS, Jeffrey; TURNER, Cameron**. Beginning Google Maps Applications with PHP and Ajax: From Novice to Professional**. 2006. Editora Appress. Acessado em : 10/12/2012

QUEIROS, Juliano. SGBD: O que é?. Disponível em: <http://espacoinfo.net/o-que-e-sgbd-bd-ii/> Acessado em: 10/12/2012.

W3 Schools – CSS. Disponível em: <www.w3schools.com/css/> Acesso em 10/09/2012.

W3 Schools – PHP. Disponível em:<www.w3schools.com/php>. Acesso em 0810/2012.

W3C. Disponível em: <www.w3c.org>. Acessado em: 08/10/2012.