UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

RODRIGO YASUHIRO UEDA

FRUTO URBANO: UMA APLICAÇÃO WEB E MOBILE UTILIZANDO TECNOLOGIAS DE GEOPROCESSAMENTO PARA ARBORIZAÇÃO URBANA

BAURU
2015

RODRIGO YASUHIRO UEDA

FRUTO URBANO: UMA APLICAÇÃO WEB E MOBILE UTILIZANDO TECNOLOGIAS DE GEOPROCESSAMENTO PARA ARBORIZAÇÃO URBANA

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP, Câmpus de Bauru.

Orientador: Prof. Dr. Eduardo Martins Morgado

Co-orientador: Luiz Miguel Axcar

BAURU
2015

RODRIGO YASUHIRO UEDA

FRUTO URBANO: UMA APLICAÇÃO WEB E MOBILE UTILIZANDO TECNOLOGIAS DE GEOPROCESSAMENTO PARA ARBORIZAÇÃO URBANA

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" – UNESP, Câmpus de Bauru.

Aprovado em ___/___/_____

BANCA EXAMINADORA

_____
Prof. Dr. Aparecido Nilceu Marana

_____
Prof. Dr. Eduardo Martins Morgado

_____
Prof. Dr. João Pedro Albino

BAURU
2015

Dedico este trabalho à minha namorada Camila que sempre esteve do meu lado me apoiando desde o início da minha gradução.

AGRADECIMENTOS

Agradeço aos meus pais por despertar a curiosidade em mim que me fez chegar até aqui e por todo suporte recebido durante toda a minha vida.

A minha namorada Camila, sem seu apoio e dedicação este trabalho não teria sido concluído.

Ao meu orientador, Prof. Dr. Eduardo Martins Morgado por todos os anos de experiência e conhecimento transmitidos.

Ao meu co-orientador Luiz Miguel Axcar pelo apoio durante o desenvolvimento desse trabalho e pela amizade.

A todos os meus amigos que fizeram parte da minha graduação.

Ao LTIA (Laboratório de Tecnologia da Informação Aplicada) pelos vários anos de aprendizado e conquistas.

RESUMO

As tecnologias de geoprocessamento são cada vez mais comuns no cotidiano das pessoas, demonstrando-se de grande utilidade. Além disso, pode se observar que as plataformas de desenvolvimento *web* e móvel têm se tornado muito populares na sociedade. Pensando nisso, esse trabalho propõe o desenvolvimento aplicação web e outra mobile que se comunicam com um *web service* a fim de se mapear pontos de vegetação, locais para plantio e podas drásticas em ambientes urbanos; além da elaboração de um relatório de desenvolvimento para auxiliar na manutenção e replicação de sistemas semelhantes para a espacialização de outros pontos de interesse.

Palavras chave: Geoprocessamento, Arborização, Aplicação, Mapas.

ABSTRACT

The geoprocessing technologies are increasingly common in the peoples lifes, and showing up to be very usefull. Besides, it can be notice that the web and mobile platforms has become very popular in between the society. Thinking about that, this project proposes a development of a web application as long as a mobile application that communicate with a web service in order to map vegetation, places to plant and drastic pruning at urban centers; in addition of a elaboration of a development report for the sake of help in the maintenance and replication of similar systems that focus on spacialize another differents points of interest.

Keywords: Geoprocessing, Afforestation, Application, Maps.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CLI	Command-Line Interface
CSS	*Cascading Style Sheets*
HTML	*Hyper Text Markup Language*
JSON	*JavaScript Object Notation*
NPM	*Node Package Manager*
MVC	*Model-View-Controller*
ORM	*Object-relational mapping*
REST	*Representational State Transfer*
RFC	*Request For Comments*
SGBD	Sistema de Gerenciamento de Banco de Dados
URI	*Uniform Resource Identifier*
XML	*eXtensible Markup Language*

SUMÁRIO

1. INTRODUÇÃO

O termo arborizar pode ser compreendido como plantar ou manter árvores em um determinado local, por sua vez, a arborização urbana caracteriza-se pelo plantio de árvores em locais como praças, parques, alamedas e calçadas de vias públicas e representa uma das mais relevantes atividades da gestão urbana (SANTOS, 2001).

Segundo Pivetta e Silva Filho (2002, p. 2) a vegetação urbana proporciona muitas vantagens ao homem que podem ser caracterizados desde efeitos abstratos como proporcionar bem estar e melhor efeito estético; até proteger e direcionar o vento, amortecer o som e diminuição da temperatura.

Tendo em vista a importância da presença das árvores no ambiente urbano, a legislação brasileira disponibilizou um artigo na Lei de Crimes Ambientais apenas para esse tópico:

> "Art. 49. Destruir, danificar, lesar ou maltratar, por qualquer modo ou meio, plantas de ornamentação de logradouros públicos ou em propriedade privada alheia:
> Pena - detenção, de três meses a um ano, ou multa, ou ambas as penas cumulativamente." (Lei de Crimes Ambientais – Lei nº 9605/98).

Em um plano paralelo, deve se notar a existência do geoprocessamento, definido pela disciplina do conhecimento que agrega técnicas computacionais e matemáticas e que é capaz de tratar a informação geográfica, ou espacial. São denominadas de Sistema de Informação Geográfica as ferramentas para geoprocessamento de origem da computação. (CÂMARA; DAVIS, [20--]).

Ainda segundo Câmara e Davis, a tecnologia de geoprocessamento pode ser muito útil para um país como o Brasil:

> "Num país de dimensão continental como o Brasil, com uma grande carência de informações adequadas para a tomada de decisões sobre os problemas urbanos, rurais e ambientais, o Geoprocessamento apresenta um enorme potencial, principalmente se baseado em tecnologias de custo relativamente baixo, em que o conhecimento seja adquirido localmente."

Esse projeto visa o desenvolvimento de uma aplicação web e mobile que seja capaz de mapear, com ajuda de uma base de usuários, todas as árvores bem como locais para seu possível plantio nos ambientes urbanos, utilizando da tecnologia de geoprocessamento, criando um Sistema de Informação Geográfico.

1.1 Problema

Em ambientes urbanos, existem diversos problemas relacionados a arborização, como a remoção ilegal de árvores e ocorrência de podas drásticas que podem acabar danificando permanentemente esses vegetais. Também pode se notar a ausência de árvores em determinados locais com potencial para seu crescimento.

Este projeto deseja fazer com que a responsabilidade de manter e plantar novas árvores possa ser diluída entre órgãos governamentais, como prefeituras municipais, e toda a população que desejar se engajar nesse processo, graças a uma ferramenta de geolocalização capaz de apontar irregularidades na arborização urbana e locais para o plantio de novas plantas.

Também deseja demonstrar a possibilidade do uso de variações da aplicação criada para atender outras carências sociais ou econômicas dos municípios brasileiros ou até de outros lugares do mundo.

Em adição, também foi produzido um relatório de desenvolvimento do sistema, que qualquer profissional, ou estudante da área de computação, que atenda os pré requisitos que serão estipulados, seja capaz adquirir o conhecimento necessário para reproduzir um sistema com características similares ao desenvolvido.

1.2 Justificativa

Com o aumento de árvores em ambientes urbanos, diversas vantagens a população, assim como exemplificadas na seção de introdução na referência ao trabalho de Pivetta e Silva Filho.

O engajamento da população ao sistema pode trazer um maior controle na auditoria de infrações relacionadas a arborização urbana, podendo também servir como uma ferramenta dos órgãos governamentais no suporte de tais atividades.

Com a identificação e rotulação das árvores existentes, também será possível tornar a plataforma informativa para seus usuários, obtendo mais informações sobre as características de cada vegetal, agregando assim, um aprendizado.

Como diversas carências da sociedade podem ser enquadradas em situações similares a proposta por este trabalho, o conhecimento técnico obtido através do

mesmo poderá ser útil para mais estudantes ou profissionais da área, principalmente com a existência do relatório de desenvolvimento, o que poderá diminuir o tempo de produção de trabalhos futuros.

1.3 Objetivo

Desenvolver e documentar uma aplicação utilizando tecnologias de geoprocessamento compatível com plataformas web e mobile para o mapeamento de árvores existentes no ambiente urbano, para sua preservação, bem como a identificação de novos locais para o plantio de novas plantas.

2. FUNDAMENTAÇÃO TEÓRICA

2.1. Protocolo HTTP

O protocolo HTTP ("*Hypertext Transfer Protocol"*) atualmente se encontra na versão 1.1, sendo descrita pela RFC 2616 de junho de 1999, e, posteriormente, sendo substituída pelas seis especificações, numeradas em ordem, RFC 7230 até RFC 7235 de junho de 2014. O HTTP pode ser definido como um protocolo de requisições e respostas da camada de aplicação para sistemas de informação hipermídia distribuídos e colaborativos.

Pode ser caracterizado como um protocolo sem estado de troca de informações em que um cliente estabelece uma conexão com um servidor como o objetivo de enviar uma ou mais requisições; e, por sua vez, o servidor é capaz de aceitar novas conexões, retornando respostas. Um mesmo nó de uma rede atuar tanto como um cliente, quanto um servidor, dependendo apenas de seu papel em uma determinada conexão.

A forma mais comum das comunicações no protocolo HTTP ocorre um retorno de dados de uma requisição representada por uma URI ("*Uniform Resource Identifier"*), um conjunto de caracteres que identifica um recurso da internet. De forma mais detalhada, deve se relatar que além da URI descrita anteriormente, uma requisição também deve conter dados sobre o método, versão do protocolo e um campos de cabeçalho, além de uma mensagem, caso exista.

Também existem oito métodos diferentes de se realizar uma requisição no protocolo descrito ("*GET*", "*HEAD*", "*POST*", "*PUT*", "*DELETE*", "*CONNECT*", "*OPTIONS*" e "*TRACE"*), indicando o objetivo de seu autor e qual o tipo de resposta esperado pelo mesmo. Em casos gerais, deve se optar pelos dois primeiros métodos citados, mantendo os demais como facultativos.

Quadro 01 - Descrição dos métodos de requisição do protocolo HTTP.

Método	Descrição
GET	Transfere a atual representação de um recurso alvo.
HEAD	Mesma funcionalidade do método *GET*, porém apenas transfere a linha e o cabeçalho.
POST	Realiza um processamento de um recurso específico com base na requisição.
PUT	Substitui todas as atuais representações de um recurso alvo com os dados da requisição.
DELETE	Remove todas as atuais representações de um recurso alvo.
CONNECT	Estabelece uma conexão com um servidor identificado pelo recurso alvo.
OPTIONS	Descreve opções de comunicação para o recurso alvo.
TRACE	Realiza uma mensagem de teste de retorno com o caminho do recurso alvo.

Fonte: RFC 7231, 2014.

Em resposta a uma requisição, o servidor deve sempre informar um código compostos por três digitos correspondente ao estado da resposta, sendo o primeiro digito responsável por identificar sua classe. A especificação RFC 7321 identifica e caracteriza quarenta e um códigos distintos, destacam se o código 200 (“*OK*”) que representa uma requisição bem sucedida; e o código 404 (“*Not Found*”) que indica que o servidor não encontrou uma representação para a requisição ou não deseja mostrar que a mesma existe.

Quadro 02 - Padrão de identificação de códigos de retorno do protocolo HTTP.

Padrão de código	**Classe da resposta**	**Descrição**
1XX	Informação	A requisição foi recebida, prosseguindo processo.
2XX	Sucesso	A requisição foi devidamente recebida, interpretada e aceita.
3XX	Redirecionamento	Ações adicionais devem ser tomadas para que a requisição seja concluída.
4XX	Erro no cliente	A requisição contém erro de sintaxe ou não pode ser realizada.
5XX	Erro no servidor	O servidor falhou em realizar uma requisição válida.

Fonte: RFC 7231, 2014.

2.2 Tecnologias web

Basicamente, quase todo o conteúdo web existente é constituído de uma tríade de tecnologia: o HTML (HyperText Markup Language), o CSS (Cascading Style Sheets) e o JavaScript. A primeira das tecnologias citadas é a responsável pela estrutura e conteúdo semântico de um site, enquanto as outras duas dispõem de ferramentas para a customização e controle desta. (MOZILLA, 2015b)

O CSS é uma linguagem de estilização do HTML sendo capaz de descrever como seus objetos ficarão disponibilizados e suas características como cor e tamanho. Essa ferramenta se encontra atualmente em sua terceira versão e é mantida pela comunidade internacional World Wide Web Consortium (W3C). (MOZILLA, 2015a)

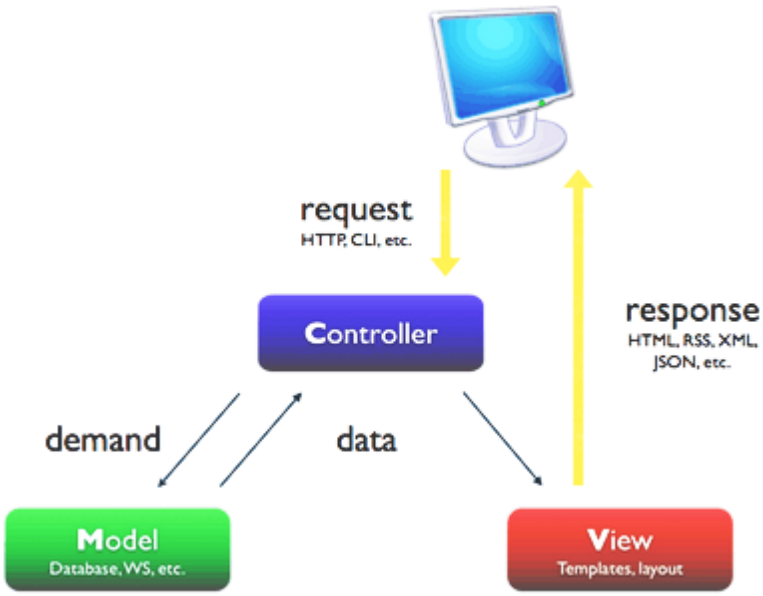
Por fim, o JavaScript é conhecido como uma linguagem de scripts para web, mas também pode ser usada em outros ambientes. Possui as características de ser uma linguagem pequena e leve, podendo ser conectada a objetos em seu ambiente de execução e prover controle sobre eles de forma dinâmica. (MOZILLA, 2015c)

2.3 Arquitetura MVC

A arquitetura MVC (“*Model-View-Controller*”) foi criada no início dos anos 80 e desde então se popularizou no desenvolvimento de aplicações *web*. A tecnologia funciona, assim como mostrado na figura 1, possuindouma camada “*Controller”*, responsável pelo controle de fluxo da aplicação, que recebe todas as requisições e envia todos os dados que precisam sofrer algum tipo de tratamento para a camada “*Model”.* Após processados, os dados retornam para o “*Controller*” que os envia para a camada de “*View”* para serem organizadas e exibidas.

De forma mais detalhada: o modelo (“*Model”)* tem como responsabilidade armazenar todas as regras de negócios e realizar a comunicação com um ou mais bancos de dados, caso existam; a visão (“*View*”) pode ser definida como uma estrutura que fornece a interface do usuário; o controlador (“*Controller*”) é aquele que transforma as interações de um usuário, na visão, em tarefas para o modelo. (LUCIANO; ALVES, 2011)

Figura 1 - Funcionamento da arquitetura MVC.



Fonte: OPENCLASSROOMS, 2015.

2.4 Node.js

O Node.js é uma plataforma desenvolvida através do motor de JavaScript do Google Chrome, é uma ferramenta que teve sua idéia concebida no ano de 2009 e tem como objetivo, através de uma máquina virtual, rodar códigos JavaScript diretamente de um servidor, podendo realizar diversas tarefas de forma leve e eficiente. (NODEBR, 2013)

Com o passar do tempo diversos pacotes compatíveis com o Node.js foram criados e disponibilizados pela comunidade de desenvolvedores. Entre eles, pode se salientar o *Gulp*, um *task runner* capaz de realizar tarefas como como concatenar e minificar scripts, além de compilar o pré-processadores de CSS: LESS e SASS.

Outro pacote que deve ser lembrado é o NPM, um gerenciador de pacotes CSS e JavaScript. Através de um repositório, ele é capaz de inserir em um projeto inúmeros *plugins*, bem como registrar suas respectivas versões, evitando gerar conflitos entre diferentes localidades do projeto.

2.5 Bancos de dados geográficos

Bancos de dados geográficos podem ser definidos como bancos de dados comuns que apresentam algumas funcionalidades extras para o tratamento de pontos espaciais sendo capazes de representar pontos, linhas, áreas e volume, além de serem capazes de organizar e manipular esse tipo de dados de forma rápida e otimizada. (Lisboa; Iochpe, 1996)

> “Um banco de Dados Geográfico é uma coleção de dados referenciados espacialmente, que funciona como um modelo da realidade”. (Lisboa; Iochpe, 1996)

Um exemplo de um banco de dados geográfico é o PostGIS, um extensor de tipos, funções, operadores e técnicas de indexação de dados do PostgreSQL, possibilitando a execução de consultas de dados espaciais diretamente com o uso da linguagem SQL. (POSTGIS, [2015?])

2.6 Web service

Web services são aplicações web capazes de criar interoperabilidade em entre sistemas heterogêneos. Através dessa tecnologia, é possível transmitir informações entre aparelhos com características completamente diferentes como é o caso de um computador e um celular; é transparente a linguagem de programação utilizada e pode usar a internet como veículo dos dados.

Para realizar essa comunicação, é necessário o uso de algum tipo de protocolo de dados, os mais comumente usados são o XML (eXtensible Markup Language) e o JSON (JavaScript Object Notation).

2.7 REST

REST é um conjunto de especificações para web services que definem como os recursos serão endereçados e transferidos utilizando o protocolo HTTP sendo compatível com aplicações independentemente da linguagem de progração utilizada em sua construção.

Os quatro principios de um *web service* que implementa o REST, recebendo o adjetivo *RESTful* são: utilizar explicitamente os métodos do protocolo HTTP; ser independente de estado; obedecer uma estrutura de URI específica; ser capaz de tranferir XML e/ou JSON.

Em relação a carcterística de ser independente do estado um *RESTful web service* não deve ser capaz de armazenar dados de interação de uma aplicação, por exemplo, quando requisitar dados paginados, a aplicação deve informar ao *web service* qual a página deve ser resgatada ao invés de requisições pela próxima página ou anterior.

3. TECNOLOGIAS

3.1 Laravel

O Laravel é um *Framework*, ou conjunto de bibliotecas, web de PHP que utiliza uma arquitetura MVC. A implementação de um *software* com o Laravel difere muito da tradicional utilizando somente a linguagem PHP, com essa framework é possível ter acesso a novas ferramentas e métodos que agilizam muito todo o processo de desenvolvimento, trazendo também, uma maior segurança para a aplicação. (LARAVEL, [2015?])

3.1.1 Models

Em relação aos *models* e ao acesso aos bancos de dados, podem ser destacadas três grandes bibliotecas do Laravel, ambas são genéricas a ponto de que, se a implementação de código corresponder aos padrões estipulados, é possível alterar de SGBD somente alterando os arquivos de configuração correspondente ao servidor, como usuário e senha.

3.1.1.1 *Migration*

Migrations são responsáveis por criar e alterar as tabelas em um banco de dados sem a necessidade de se usar linguagens específicas, desta forma, sem depender do SGBD escolhido para o sistema, todo o código correspondente será escrito na linguagem PHP.

As *migrations* também são responsáveis por ordenar cronologicamente e administrar todos os scripts que criam ou alteram as tabelas, desta forma, quando o sistema se encontra em status de produção, ou seja, apresentando dados nas tabelas que não podem ser deletados, é possível identificar quais alterações já foram realizadas, evitando que um script seja executado desnecessariamente mais de uma vez.

3.1.1.2 *Query Builder*

Assim como as *migrations*, o *query builder* é uma interface que, somente usando a linguagem de programação PHP, permite a execução de operações nos bancos de dados. Ela é utilizada em em ações de adição, consulta, edição e

remoção de dados nas tabelas existentes, possibilitando o suporte a todo tipo de ação nativa dos bancos de dados e, em casos específicos, permitindo que as operações possam ser realizadas parcial ou integralmente na linguagem nativa do SGBD.

3.1.1.3 *Eloquent*

Por fim, o *eloquent* pode ser caracterizado como um mapeamento objeto-relacional de um banco de dados, criando as regras de negócios de uma tabela do banco de dados, descrevendo, caso existente, sua relação com as demais tabelas. Em adição, esse módulo também automatiza funcionalidades muito utilizadas como a administração de colunas de data com informações de quando um registro for inserido e modificado; e a consulta de de registros atrvés de seu número de identificação.

3.1.2 *Controller*

A implementação de *controllers* do Laravel já conta com várias estruturas para facilitar o fluxo e navegação de páginas de um *website*, normalmente três componentes são envolvidos em uma requisição para uma página:

3.1.2.1 Arquivo de rotas

O arquivo de rotas é único para toda a aplicação e é responsável por associar uma URI com uma determinada lógica. Diferentemente de uma aplicação web sem uma *framework* onde as pastas e arquivos definem as rotas, é possível realizar atribuições customizadas dependendo da necessidade.

Apesar de ser capaz de fazê-lo, normalmente o arquivo de rotas, redireciona cada requisição, ou conjunto de de requisições, para um arquivo contendo a classe *Controller.*

3.1.2.2 Classe *Controller*

Essa classe é um conjunto de uma ou mais funções, comumente, cada função representa uma rota de um URI, sendo responsável por coletar todos os dados relevantes da requisição e, utilizá-los para interagir com o banco de dados ou

aplicando algum tipo de lógica sobre estes. Ao fim, a classe deve retornar o resultado de suas ações, podendo atribuir uma *view* para a impressão destes dados de forma mais amigável para um usuário, ou em algum protocolo de comunicação como o JSON.

3.1.2.3 *Middleware*

Os *middlewares* são classes que podem ser chamadas antes dos controller para com o objetivo de filtrar uma requisição recebida pela applicação. Seu uso mais comum seria para a validação de dados de um usuário como, por exemplo, em um sistema em que os dados são são acessíveis mediante senha, a classe em questão é capaz de verificar se o usuário já inseriu sua senha de acesso, e em caso negativo redirecioná-lo para uma página em que ele possa fornecê-la. Outras ações podem ser realizadas como a mudança de cabeçalhos HTTP da requisição.

3.1.3 *View*

Views são um mecanismo para a impressão de dados em uma página web de maneira amigável para um usuário, utilizando a linguagem de marcação HTML, juntamente com dados impressos dinamicamente pelo PHP, com seu estilo definido por um arquivo CSS e interações geradas por uma arquivo JavaScript.

O principal recurso do Laravel para a manipulação *views* são as *blades* que são modelos de como os dados devem ser impressos na página. Caso várias páginas possuam dados repetidos, através das *blades* é possível definir um único arquivo que contém esse dado repetido e resgatá-lo em qualquer outro arquivo.

Também pode se observar a implementação de estruturas de controle, diretivas que são suportadas pelo interpretador de PHP que facilitam ações como impressão de dados e implementação de estruturas de decisão e repetição.

3.1.4 *Artisan*

Artisan é uma CLI - interface de linha de comando - capaz de automatizar e agilizar processos durante a produção de uma aplicação Laravel. É capaz de criar modelos de arquivos de *models* e *controllers* evitando um esforço repetitivo do desenvolvedor, por exemplo, um arquivo possuindo uma classe de controller que

obedece os padrões *RESTful* com cerca de noventa linhas, carregando as bibliotecas necessárias e criando as funções utilizadas nas rotas de requisições.

Essa interface também é responsável pela administração dos scripts de *migrations*, criando e alterando as tabelas no banco de dados e registrando quais destes já foram executados, também sendo capaz de revertê-los, se necessário.

3.2. *OPEN LAYERS 3*

OpenLayers 3 é uma biblioteca da linguagem JavaScript para a visualização e manipulação de dados espaciais, toda sua arquitetura é baseada em camadas (*layers*) sendo capaz de criar representações de dados no formato de imagens em blocos (*tiles*) e vetores. (OPENLAYERS, [2015c?])

Figura 2 - Exemplo de um mapa construido usando a biblioteca *OpenLayers* 3

Fonte: O autor.

Todo mapa criado utilizando essa biblioteca possui quatro principais componentes:

3.2.1 *View*

Uma view é uma estrutura que define tudo o que pode ser observado em um mapa como sua posição central, em coordenadas; nível de zoom, contendo valores inteiros dentro do intervalo de um até vinte e dois; projeção, o sistema de

representação de coordenadas que esse componente deve utilizar para posicionar o mapa.

3.2.2 *Layer*

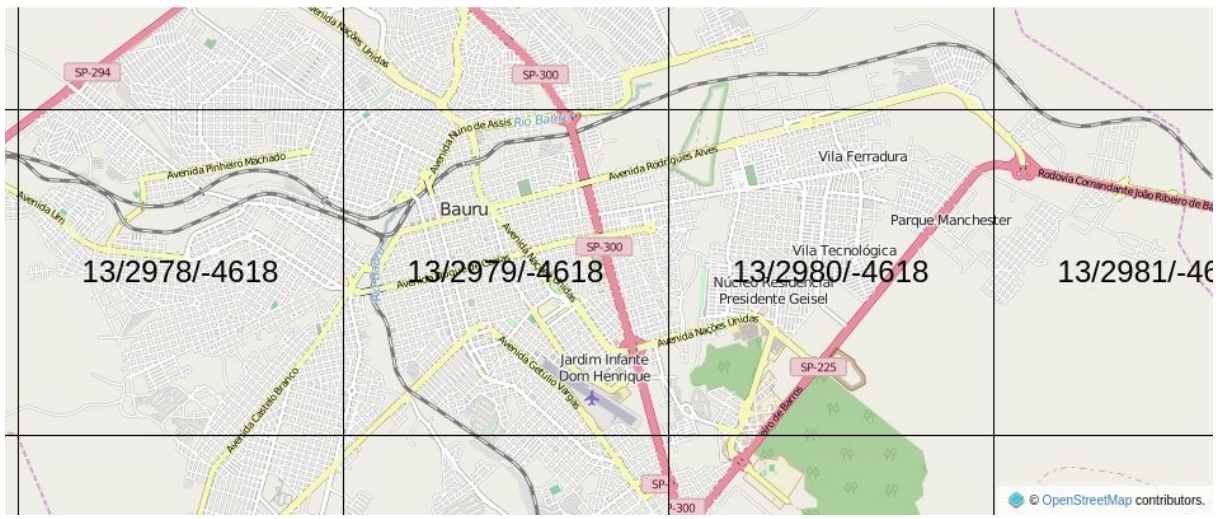
Como previamente mencionado, uma *layer*, ou camada, é responsável por informar quais são os dados que serão representados no mapa. Estes dados podem ser classificados em dois diferentes tipos, cada um com uma aplicação específica.

3.2.2.1 *Tiles*

Tiles podem ser definidos como um conjunto de imagens que, quando unidas se tornam um mapa, esse método é muito utilizado para a visualização de mapas temáticos como é o caso de imagens aéreas e de satélite, além de representações de vias de transporte.

Como pode ser observado na figura 03, a seguir, cada quadrado demarcado por linhas pretas é uma imagem de *tile,* sendo que cada nível de zoom diferente exige diferentes tiles. Para sua localização, cada imagem precisa ser organizada utilizando três coordenadas contendo o nível de zoom e sua posição em um plano de duas dimensões XY.

Figura 3 - Representação de imagens do tipo *tile*



Fonte: OpenLayers, [2015a?].

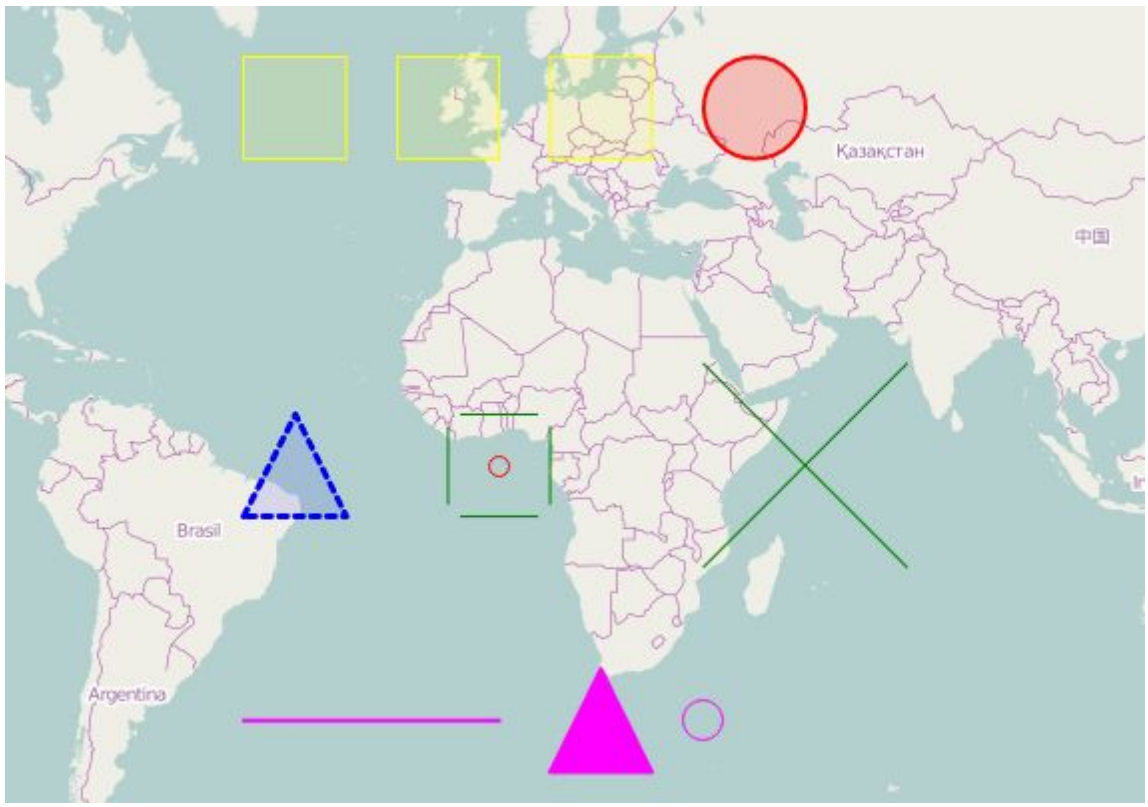
3.2.2.2 Vetores

Vetores são representações de informações espaciais dependentes de coordenadas geográficas, para isso o *OpenLayers* utiliza uma estrutura denominada *feature*. Uma *feature* contém um conjunto de um ou mais pares de coordenadas,

assim como, opcionalmente, dados tabulares associados a localização referenciada.

Uma feature pode ser do tipo ponto, linha ou polígono e cada um desses elementos deve possuir uma estrutura que descreve o seu estilo de impressão, diversas opções podem ser aplicadas como cor e grossura da linha e cor de preenchimento assim como pode ser notado na figura 04.

Figura 4 - Exemplos de uso e estilização de *features*.



Fonte: OpenLayers, [2015b?].

3.2.3 Interaction

Uma *interaction*, ou interação, é capaz de definir ações relacionadas as *features*. Através dela é possível se implementar regras para casos como a seleção, desenh, modificação ou sua remoção de *features*. Também podem ser relacionadas ao componente de *view* de mapa associando-se com eventos como o clique de botões do mouse e teclado.

3.2.4 *Controls*

São responsáveis por criar novas funcionalidades para o mapa, descrevendo o que deve ser mostrado para o usuário, como um novo botão, e, juntamente com esse elemento de interface, uma lógica associada. Por padrão a biblioteca já oferece controles comuns como botões para zoom, tela cheia e rotação.

3.3. APACHE CORDOVA

O Apache Cordova é uma *framework* de código aberto para o desenvolvimento de aplicações móveis utilizando as tecnologias *web* como o HTML5, CSS3 e o JavaScript.

Outra característica relevante dessa tecnologia, é que ela é multiplataforma, fazendo com que todo o código criado possa gerar aplicativos para as principais plataformas móveis como o *android*, *ios* e o *windows phone* sem a necessidade de se utilizar código nativo e, ainda assim, mantendo o acesso de bibliotecas nativas como de câmera, GPS e acelerômetro.

Todo o fluxo de compilação de geração de arquivos executáveis para cada plataforma com suporte pela framework pode ser realizado através do uso de linhas de comando (CLI). Também é possível emular esses arquivos no computador sem ao menos ter a necessidade de se possuir um aparelho compatível.

4. DESENVOLVIMENTO

O processo de desenvolvimento desse trabalho foi dividido em três etapas:

4.1 Planejamento

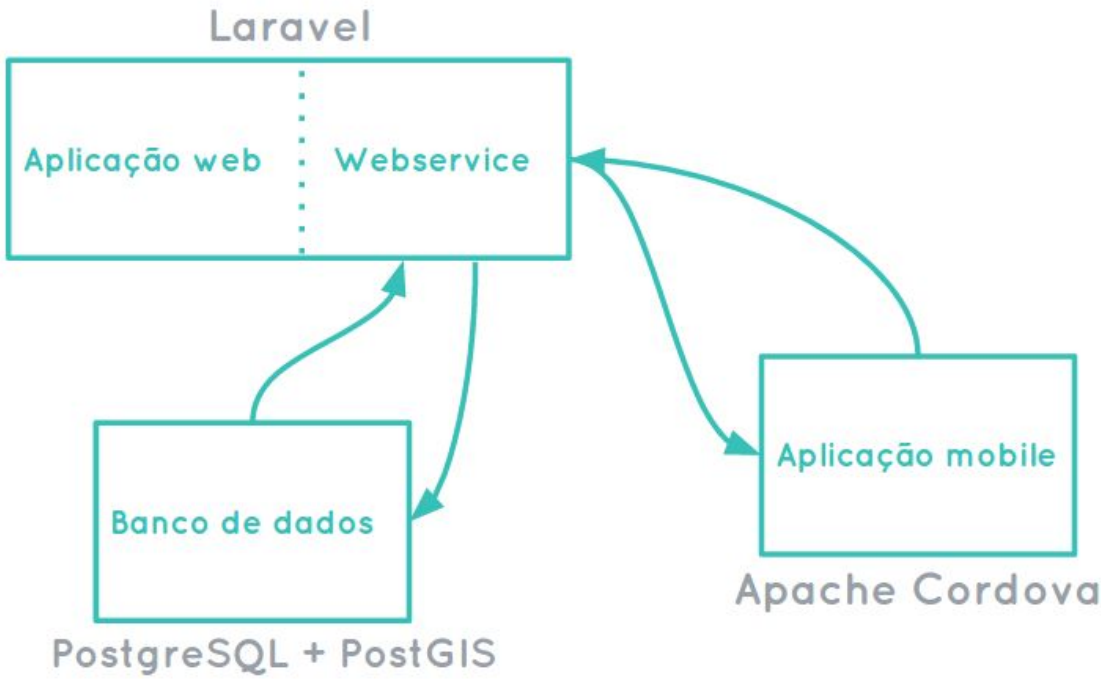
O planejamento, que pode ser definido como a etapa onde foram definidas todas as tecnologias que seriam utilizadas, além da criação de esboços de como a aplicação iria se comportar. Localizado no Apêndice A, encontra-se a modelagem de banco de dados que foi posteriormente implementado; objetivo dessa modelagem foi atingir um banco de dados mais enxuto e genérico, para que essa base possa também ser utilizada em diferentes aplicações.

4.2 Implementação

A etapa de implementação também pode ser subdividida em outras três menores: *website, web service* e aplicação móvel. Para as duas primeiras citadas, se utilizou o Laravel, atentando-se para que todas as informações que estariam presentes no website e na aplicação móvel, pudessem ser acessadas através do *web service* arquitetado segundo os padrões REST, dessa forma, é possível que as informações sobre os usuários e pontos espacias, juntamente com seus atributos, possam ser utilizados por qualquer plataforma sem a necessidade de que os dados sejam replicados.

A figura 5 mostra como é a transmissão de dados de todo o sistema. Somente o *webservice* é capaz de realizar consultas e editar o banco de dados e ambas as aplicações (*web* e *mobile*) fazem requisições para este servidor, informando que alterações desejam fazer.

Figura 5: Fluxo de dados entre o *webservice* e as aplicações



Fonte: O autor.

4.2.1 *Bing Maps*

O serviço externo bing maps foi utilizado pela aplicação em dois casos diferentes: obtenção de imagens no formato de *tiles*. Esses tiles são uma forma híbrida entre imagens de satélite e representação de vias, avaliou-se como a melhor forma de visualização e entendimento dos mapas por parte dos usuários.

Em adição foi necessário utilizar também um serviço para a tradução de endereços como nomes de ruas, bairros em cidades para coordenadas geográficas. Esse serviço demonstrou-se muito útil para que os usuários pudessem facilmente centralizar o mapa próximo a localidade escolhida.

4.2.2 Manipulação de dados

Para inserir, editar e remover pontos espaciais nos mapas, o usuário precisa realizar um login no sistema, dessa forma, somente este pode editar dados de pontos que adicionou.

Cada um dos pontos possui as seguintes informações: tipo do ponto, identificando se o ponto é uma árvore, ponto de poda drástica ou possível local para

plantio; caso exista uma árvore plantada, opcionalmente pode ser informada a sua espécie; uma foto da árvore ou local de plantio; um campo de observações onde o usuário possui liberdade para inserir informações adicionais e comentários. Na figura 6 pode se observar o formulário onde os dados citados podem ser inseridos.

Figura 6: Exemplo de cadastro de um novo ponto.

Fonte: O autor.

4.2.3 Dados geográficos

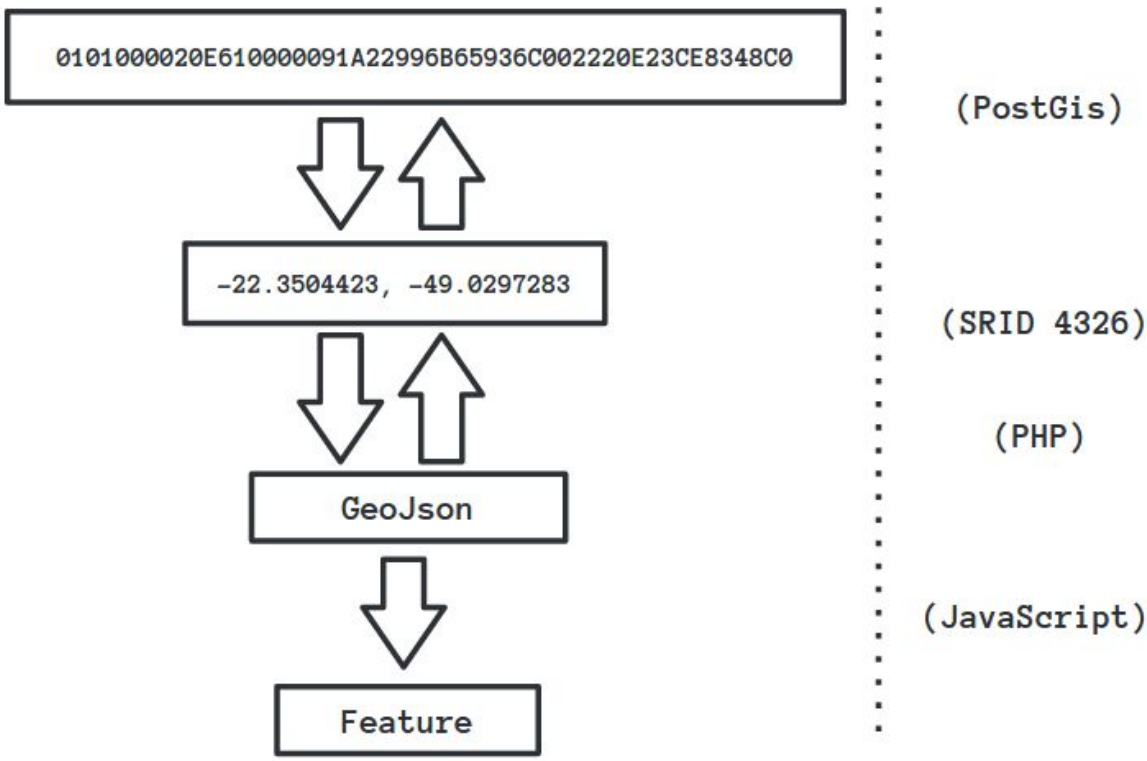
Informações de coordenadas geográficas sofrem transformações de representação desde o banco de dados até o seu o seu desenho em JavaScript. Quando se encontra como tipo de campo *geometry* nos PostGIS é representada como uma um valor hexadecimal. Em seguida, com uma requisição a este banco de dados, ele pode ser transmitido para coordenadas e recebido pelo PHP que ainda deve organizá-las em um formato denominado GeoJSON, que nada mais é do que

um objeto JSON que deve possuir propriedades específicas contendo a descrição da geometria, além de suas propriedades.

Por fim, a estrutura GeoJSON criada deve ser enviada para a biblioteca JavaScript OpenLayers 3 que, ao recebe-la irá transformá-la novamente, desta vez, em uma estrutura chamada *feature*, só assim a coordenada em questão pode ser desenhada no mapa.

Todo o processo descrito por essa subseção pode ser melhor visualizado na figura 7, utilizando dados de exemplo.

Figura 7: Fluxo de dados geográficos

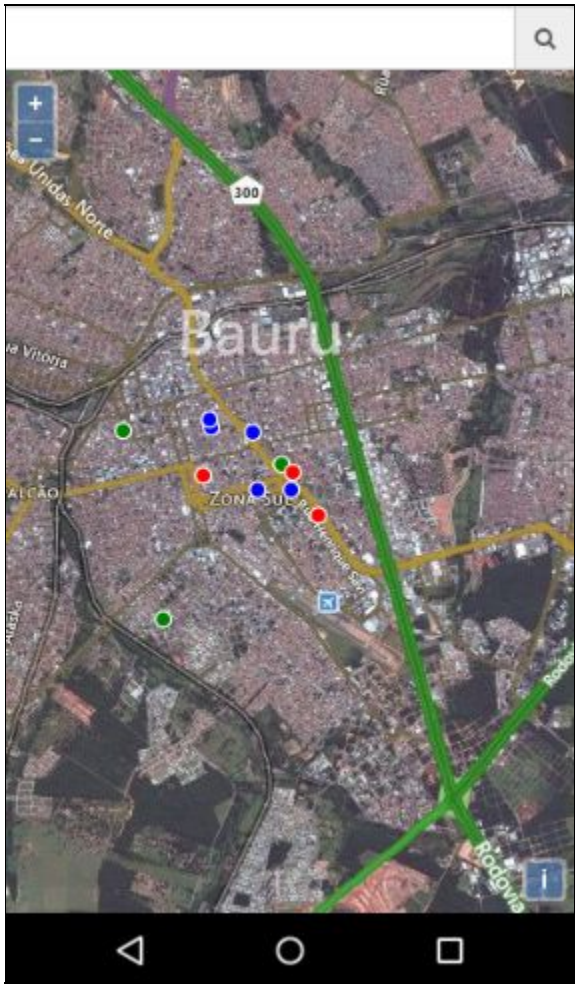


Fonte: O autor.

4.2.4 Aplicações

Basicamente as aplicações *web* e *mobile* constituem-se de uma única tela contendo um mapa que ocupando uma grande área. Como pode ser visto na figura 8, a palicação mobile somente é capaz de visualizar os pontos existentes na base de dados, com capacidade de navegação e um campo em sua área superior para localizar endereços.

Figura 8: Aplicação *mobile*



Fonte: O autor.

Por sua vez, a aplicação *web,* como pode ser visualizado na figura 9, encontra-se em um estado mais final, possuindo funcionalidades além da visualização, sua aba esquerda muda de acordo com a funcionalidade desejada (visualização e inserção de pontos), além de possuir um botão de *login* e, ao clicar com o botão direito do mouse em qualquer parte do mapa, oferece a opção para adicionar um novo ponto.

Figura 9: Aplicação *web*

Fonte: O autor.

4.3 Relatório

Por fim, o relatório de desenvolvimento busca ajudar outros desenvolvedores a compreender melhor as tecnologias utilizadas pela aplicação e auxiliar na replicação ou manutenção de sistemas semelhantes. É composto por um guia de instalação de todas as ferramentas necessárias para o sistema operacional Linux, distribuição Ubuntu, versão 14.04; links úteis para consulta durante o desenvolvimento; sugetões de uso de cada ferramenta, mostrando sucintamente como cada uma delas funciona. O relatório pode ser encontrado no Apêndice B desta monografia.

5. CONCLUSÃO

Através desse trabalho é possível concluir que é totalmente viável o desenvolvimento de uma aplicação *web* com características de geoprocessamento, observando-se que pode se manipular dados espaciais de maneira satisfatória e desempenho aceitável.

Em adição conclui-se que o mesmo modelo de aplicação pode ser utilizado para o mapeamento de outras informações, de maneira relativamente rápida e sem a necessidade de nenhum gasto com ferramentas, isto é, somente utilizando tecnologias cujo uso não gera nenhum custo.

Uma vez que o processo de instalação e abstração das tecnologias utilizadas pelo projeto consumiram uma significamente parte do desenvolvimento desse trabalho, pode se avaliar que o relatório criado pode se tornar muito útil para o suporte a novos desenvolvedores que planejam realizar alterações no projeto ou criar aplicações semelhantes.

Por fim, avalia-se que ao fim desse trabalho, uma grande e relevante quantidade de conhecimento pôde ser adquirida, ressaltando-se que se trata de uma tecnologia bem recente com uma provavel longa validade.

6. TRABALHOS FUTUROS

Espera-se que o desenvolvimento do sistema não se encerre com a conclusão desta monografia, existem planos de diversas novas funcionalidades com a intenção de publicar ambas as aplicações para acesso público até o ano seguinte da publicação deste trabalho. Podem ser destacadas a integração com redes sociais para fazer com que os usuários sejam capazes de ajudar na auditoria dos pontos inseridos; e a capacidade de construção de relatórios, podendo assim gerar conhecimento com todos os dados adquiridos.

Esses relatórios devem ser capazes de gerar dados relativos aos municípios, estados e o país, podendo assim descobrir quais são as áreas urbanas mais e menos arborizadas além de verificar as espécies de vegetação presentes em cada lugar.

Existe também a expectativa de parear todas as funcionalidades da aplicação *mobile* e aplicação *web,* além de, aproveitando das comodidades da tecnologia utilizada na produção da primeira citada, criar suporte para outras plataformas a serem definidas além do Android.

REFERÊNCIAS

BRASIL; **Lei n.º 9.605, de 12 de fevereiro de 1998.** Dispõe sobre as sanções penais e administrativas derivadas de condutas e atividades lesivas ao meio ambiente, e dá outras providências. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/l9605.htm> Acesso em: 24/05/2015.

CÂMARA, G.; DAVIS, C.; **Introdução à Ciência da Geoinformação,** [20--]. Disponível em: <http://www.dpi.inpe.br/gilberto/livro/introd/cap1-introducao.pdf>. Acesso em: 24/05/2015.

CORDOVA; **Documentation**, [2015?]. Disponível em: <https://cordova.apache.org/docs/en/latest/guide/overview/>. Acesso em: 10/01/2016.

LARAVEL; **Documentation,** [2015?]. Disponível em: <https://laravel.com/docs/5.2>. Acesso em: 29/12/2015.

LUCIANO, J.; ALVES, W. J. B.; **PADRÃO DE ARQUITETURA MVC: MODEL-VIEW-CONTROLLER**, 2011. Disponível em: <http://unifafibe.com.br/revistasonline/arquivos/revistaepeqfafibe/sumario/20/16112011142249.pdf>. Acesso em: 29/12/2015.

MOZILLA; **CSS**, 2015a. Disponível em: <https://developer.mozilla.org/ptBR/docs/Web/CSS> Acesso em: 27/05/2015.

MOZILLA; **HTML (HyperText Markup Language)**, 2015b. Disponível em: <https://developer.mozilla.org/ptBR/docs/Web/HTML> Acesso em: 27/05/2015.

MOZILLA; **JavaScript Guide**, 2015c. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide> Acesso em: 27/05/2015.

NODEBR; **O que é Node.js?**, 2013. Disponível em: <http://nodebr.com/o-que-e-node-js/> Acesso em: 28/05/2015.

OPENCLASSROOMS; **ITÉRATION 2 : PASSAGE À UNE ARCHITECTURE MVC**, 2015. Disponível em: <https://openclassrooms.com/courses/evoluez-vers-une-architecture-php-professionnelle/iteration-2-passage-a-une-architecture-mvc>. Acesso em: 04/01/2016.

OPENLAYERS; **Canvas Tiles**, [2015a?]. Disponível em: <http://openlayers.org/en/v3.12.1/examples/canvas-tiles.html>. Acesso em: 10/01/2016.

OPENLAYERS; **GeoJSON**, [2015b?]. Disponível em: <http://openlayers.org/en/v3.12.1/examples/geojson.html?q=features>. Acesso em: 10/01/2016.

OPENLAYERS; **API**, [2015c?]. Disponível em: <http://openlayers.org/en/v3.12.1/apidoc/index.html>. Acesso em: 10/01/2016.

PIVETTA, K. F. L.; SILVA FILHO, D.F.; **Arborização Urbana**. Jaboticabal: UNESP/FCAV/FUNEP, 2002 (Boletim Acadêmico). Disponível em: <http://www.uesb.br/flower/alunos/pdfs/arborizacao_urbana%20Khatia.pdf>. Acesso em: 24/05/2015.

POSTGIS; **Features**, [2015?]. Disponível em: <http://postgis.net/features> Acesso em: 25/05/2015.

RFC 2616. Request for Comments: 2616. **Hypertext Transfer Protocol -- HTTP/1.1**, 1999. Disponível em: <http://www.ietf.org/rfc/rfc2616.pdf>. Acesso em: 28/12/2015.

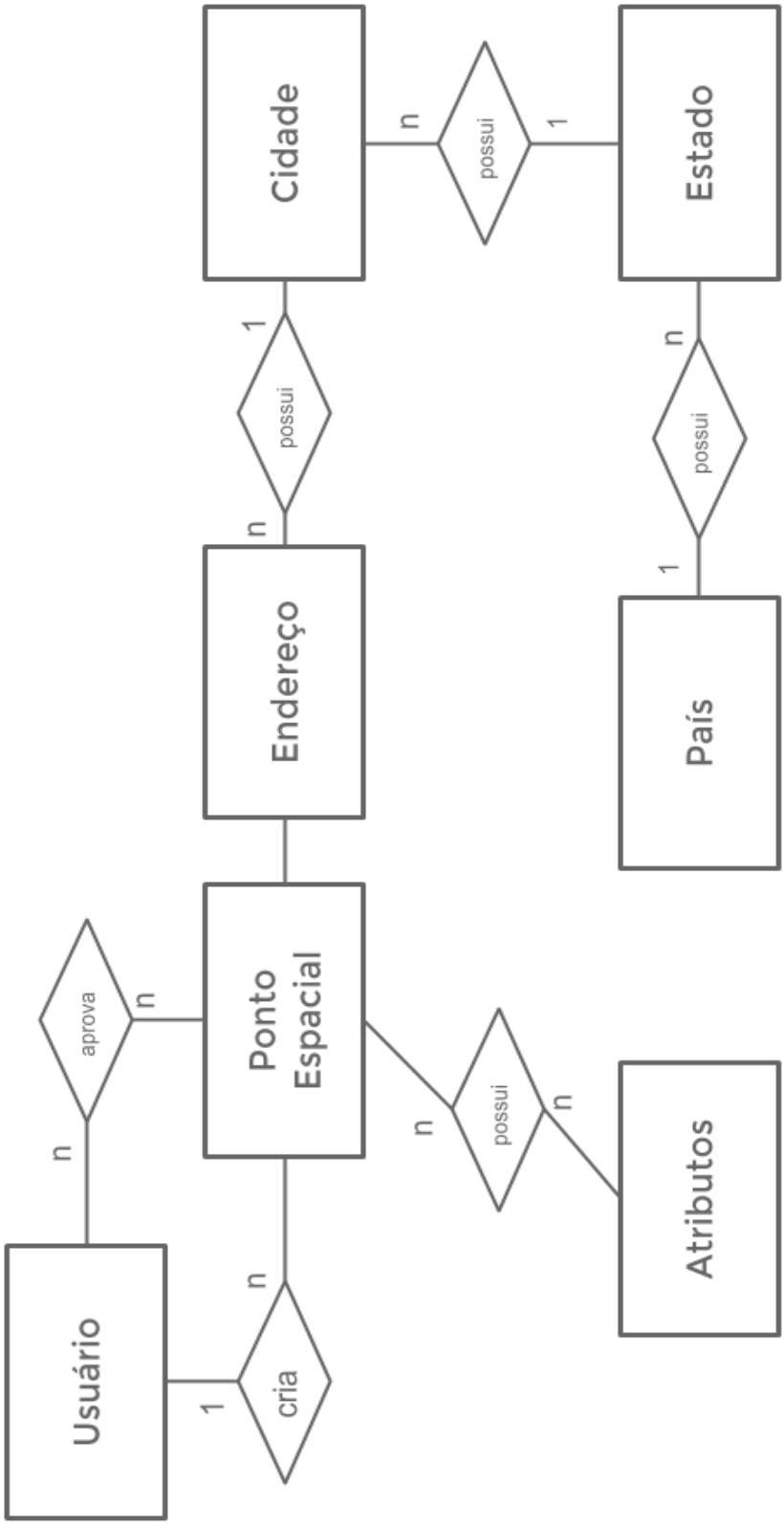
RFC 7231. Request for Comments: 7231. **Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**, 2014. Disponível em: <http://tools.ietf.org/html/rfc7230>. Acesso em: 28/12/2015.

RFC 7230. Request for Comments: 7230. **Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**, 2014. Disponível em: <https://tools.ietf.org/html/rfc7231>. Acesso em: 28/12/2015.

Rodriguez, A.; **RESTful Web services: The basics**, 2008. Disponível em <http://www.ibm.com/developerworks/library/ws-restful/>. Acesso em: 23/01/2015.

SANTOS, Antonio Silveira R. dos; **Arborização urbana: considerações**, 2011. Disponível em <http://www.aultimaarcadenoe.com.br/wp-content/uploads /2011/09/Arborizacao_urbana_consideracoes-ASilveira-3343.pdf> Acesso em: 24/05/2015.

APÊNDICE A - Modelagem do banco de dados



APÊNDICE B - Relatório de desenvolvimento

O documento

Este documento é uma coleção de guias de instalação, primeiros passos e dicas para o desenvolvimento de uma aplicação web e mobile utilizando tecnologias de geoprocessamento. Será pressuposto que o público-alvo é um usuário com conhecimento limitado ou nulo nas tecnologias abordadas. Diversas fontes citadas se encontram em inglês.

Sistema operacional

Todos os guias de instalação são previstos para serem compatíveis com o sistema operacional Linux, distribuição Ubuntu, versão 14.04 LTS (Long Term Support) que pode ser adquirido gratuitamente no link:

http://www.ubuntu.com/download/desktop

Guias de instalação

Procure entender os comandos antes de inseri-los no terminal, **todo texto entre chaves "{}" deve ser substituido de acordo com a ocasião.**

Apache

O apache é o software responsável por criar um web server.

```
sudo apt-get update
sudo apt-get install apache2
```

PHP

O PHP é uma linguagem de script para a criação de páginas dinâmicas.

```
sudo apt-get install php5 libapache2-mod-php5 php5-mcrypt
```

Editar o arquivo dir.conf:

```
sudo nano /etc/apache2/mods-enabled/dir.conf
```

Alterando para:

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.php index.xhtml index.htm
</IfModule>
```

A seguir um link com um ótimo ponto de começo para o PHP.

http://br.phptherightway.com/

*Mais detalhes sobre a instalação do PHP e do Apache, juntamente com a instalação do banco de dados MySQL podem ser encontrados em:

https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu

NodeJS e NPM

O NodeJS é uma plataforma para a execução de código de JavaScript do lado do servidor, diversas ferramentas para o desenvolvimento são construídas dessa forma e podem ser instaladas utilizando o NPM, um gerenciador de pacotes.

```
sudo apt-get update
sudo apt-get install nodejs
sudo apt-get install npm
```

Git

Gerenciador de versão de projetos.

```
sudo apt-get update
sudo apt-get install git
```

Composer

Gerenciador de pacotes e dependências para PHP.

```
sudo apt-get update
sudo apt-get install curl php5-cli
curl -sS https://getcomposer.org/installer | sudo php --
--install-dir=/usr/local/bin --filename=composer
```

Laravel

Framework de PHP utilizada para a construção da aplicação web e do web service.

```
cd /var/www/html/
composer create-project --prefer-dist laravel/laravel {nome_do_projeto}
cd {nome_do_projeto}
npm install
```

Outros detalhes em:

https://laravel.com/docs/5.2#installation

Repositório git

Para criar um repositório git do projeto:

```
git add .
git commit -m "{mensagem_de_descricao_das_mudancas}"
```

Caso possua um repositório remoto, como o Github, é possível sincronizar seu repositório local, copiando-o em um local fora de seu computador:

```
git remote add origin git@github.com:{nome_do_usuario}/{nome_do_repositorio}.git
git push -u origin master
```

PostgreSQL e PostGIS

Banco de dados que será utilizado juntamente com o plugin para bancos de dados geográficos.

```
sudo apt-get install postgresql postgresql-contrib phppgadmin
sudo sed -r -i 's/#.?allow from all/allow from all/'
/etc/apache2/conf.d/phppgadmin
sudo sed -i 's/\(all *\)peer/\1md5/' /etc/postgresql/9.3/main/pg_hba.conf
```

```
sudo apt-get install build-essential postgresql-9.3 postgresql-server-dev-9.3
libxml2-dev libproj-dev libjson0-dev libgeos-dev xsltproc docbook-xsl
docbook-mathml libgdal-dev

cd ~/
wget http://download.osgeo.org/postgis/source/postgis-2.0.7.tar.gz
tar -zxvf postgis-2.0.7.tar.gz
cd postgis-2.0.7
./configure
make
sudo make install
sudo ldconfig
sudo make comments-install
sudo ln -sf /usr/share/postgresql-common/pg_wrapper/usr/local/bin/shp2pgsql
sudo ln -sf /usr/share/postgresql-common/pg_wrapper/usr/local/bin/pgsql2shp
sudo ln -sf
/usr/share/postgresql-common/pg_wrapper/usr/local/bin/raster2pgsql
```

Criar um usuário e o banco de dados

```
sudo su postres
psql
CREATE USER {nome_do_usuario} WITH PASSWORD '{senha_do_usuario}';
CREATE DATABASE {nome_do_banco_de_dados} OWNER {nome_do_usuario};
\q
psql --dbname={nome_do_banco_de_dados} --command="CREATE EXTENSION
adminpack;CREATE EXTENSION postgis;CREATE EXTENSION postgis_topology;"
exit
```

Plugins para aplicação web

Na pasta do projeto (`/var/www/html/{nome_do_projeto}`) é possivel instalar os plugins através do npm com:

```
npm install {nome_do_pacote} --save
```

A seguir alguns pacotes que podem ser instalados:

Nome do pacote	Descrição
openlayers	Pacote para a visualização de mapas.
bootstrap	Framework de desenvolvimento front-end.
jquery	Biblioteca de JavaScript que agiliza o desenvolvimento.
font-awesome	Fonte contendo simbolos vetorizados.
sweetalert	Biblioteca para criação de alertas amigáveis ao usuário.

Após instalação, é aconselhável dedicar um tempo para o aprendizado dos pacotes.

http://getbootstrap.com/getting-started/
http://learn.jquery.com/
https://fortawesome.github.io/Font-Awesome/get-started/
http://t4t5.github.io/sweetalert/

Utilizando o PostGIS

O plugin acrescenta a um banco de dados PostgreSQL os seguintes tipos de campos:

- box2d
- box3d
- geometry
- geometry_dump
- geography

Para a representação de campos espacias (pontos, linhas e polígonos) é utilizado o campo geometry, a seguir um exemplo de como transformar uma coordenada em um ponto no banco:

```
ST_GeomFromText('POINT({longitude} {latitude})')
```

Diversas outras funções podem ser encontradas em:

http://www.postgis.us/downloads/postgis21_cheatsheet.pdf

SRID

Existem diversas formas de representar coordenadas geográficas, a mais comum é a **4326**, utilizada nos principais sites de mapa, a biblioteca OpenLayers 3 também utiliza a representação identificada pelo código **3857**, conversões entre elas podem ser realizadas através do banco de dados

```
ST_Transform({geometry}, {id_de_representacao_para_conversao})
```

ou através do OpenLayers:

```
ol.proj.transform({vetor_de_coordenadas}, {SRID_origem}, {SRID_destino})
```

Código	uso no OpenLayers
4326	“EPSG:4326”
3857	“EPSG:3857”

Desenvolvimento em OpenLayers 3

Essa biblioteca possui uma documentação extensa e detalhada, além de muitos exemplos, tudo pode ser encontrado em:

http://openlayers.org/en/master/apidoc/
http://openlayers.org/en/master/examples/

Uma estrutura básica de um mapa é composta por:

- ol.map
  - ol.view
  - ol.layer
    - ol.source
      - ol.layer
        - ol.layer.tile
        - ol.layer.vector
    - ol.style
  - ol.interaction
  - ol.control

Desenvolvimento em Laravel

Para o aprendizado de Laravel, é aconselhável o curso do site LaraCasts:

https://laracasts.com/series/laravel-5-fundamentals

Após compreender os 27 videos dessa série, é possível desenvolver tranquilamente, sem maiores dificuldades.

Também pode se destaca a documentação da framework e um documento que contém todos os principais comandos:

https://laravel.com/docs/5.2/quickstart
http://cheats.jesse-obrien.ca/

Gulp

O gulp é um automatizador de tarefas, por padrão, no Laravel ele é instalado assim que se utiliza o comando:

```
npm install
```

Sua configuração é através do arquivo **gulpfile.js,** para mais informações:

https://laravel.com/docs/5.2/elixir

Desenvolvimento móvel - Apache Cordova

No caso do desenvolvimento de aplicações móveis, uma ferramenta muito utilizada é o Apache Cordova:

```
npm install -g cordova
```

```
É possível criar e testar um novo projeto com o comando:
```

```
cordova create {nome_do_projeto}
cordova platform add {nome_da_plataforma}
cordova run {nome_da_plataforma}
```

Android

Para que o cordova seja capaz de criar aplicativos android, é necessário a instalação de sua SDK.

```
sudo apt-get update
sudo apt-get install openjdk-7-jre
sudo apt-get install openjdk-7-jdk

cd ~/
wget http://dl.google.com/android/android-sdk_r24.4.1-linux.tgz
tar zxvf android-sdk_r24.4.1-linux.tgz
mv android-sdk-linux /usr/local/.
sudo chmod -R ugo+rwx /usr/local/android-sdk-linux

export ANDROID_HOME="/usr/local/android-sdk-linux/"
export ANDROID_PLATFORM_TOOLS="/usr/local/android-sdk-linux/platform-tools"
export PATH="$PATH:$ANDROID_HOME:$ANDROID_PLATFORM_TOOLS"

android
```

Instale as ferramentas de desenvolvimento marcadas, juntamente com uma versão do android de sua escolha.

```
sudo apt-get update
sudo apt-get install lib32stdc++6
sudo apt-get install lib32z1
```

*Também é possível utilizar o OpenLayers no desenvolvimento de aplicações do Apache Cordova.