INSTITUTO FEDERAL DE CIÊNCIA E TECNOLOGIA DE PERNAMBUCO

Campus Recife

Curso Tecnólogo em Análise e Desenvolvimento de Sistemas

DAVI DA CRUZ GUIAR

JOSÉ JORGE DOS SANTOS NETO

SISTEMA WEB PARA LOCALIZAÇÃO *INDOOR* - ESTUDO DE CASO NO INSTITUTO FEDERAL DE PERNAMBUCO

Recife

2018

DAVI DA CRUZ AGUIAR
JOSÉ JORGE DOS SANTOS NETO

SISTEMA WEB PARA LOCALIZAÇÃO INDOOR - ESTUDO DE CASO NO INSTITUTO FEDERAL DE PERNAMBUCO

Monografia apresentada ao Curso Tecnólogo em Análise e Desenvolvimento de Sistemas do Instituto Federal de Pernambuco, como requisito parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientadora: Profa. Dra. Aida Araújo Ferreira

Recife
2018

Ficha elaborada pela bibliotecária Emmely Cristiny Lopes Silva CRB4/1876

A282s
2018 Aguiar, Davi da Cruz.
Sistema web para localização indoor – estudo de caso no Instituto Federal de Pernambuco / Davi da Cruz Aguiar; José Jorge dos Santos Neto. --- Recife: O autor, 2018.
46f. il. Color.

TCC (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Instituto Federal de Pernambuco, Departamento Acadêmico de Controle de Sistemas Eletrônicos - DASE, 2018.

Inclui Referências.

Orientadora: Professora Dra. Aida Araújo Ferreira.

1. SIGWeb. 2. Localização. 3. GPS. 4. Mobile. 5. Indoor. I. Título. II. Ferreira, Ainda Araújo (orientadora). III. Instituto Federal de Pernambuco.

CDD 004 (21ed.)

Trabalho de Conclusão de Curso apresentado pelos alunos **Davi da Cruz Aguiar e José Jorge dos Santos Neto** à coordenação de Análise e Desenvolvimento de Sistemas, do Instituto Federal de Pernambuco, sob o título de "***SISTEMA WEB PARA LOCALIZAÇÃO INDOOR - ESTUDO DE CASO NO INSTITUTO FEDERAL DE PERNAMBUCO***", orientado pelo Profa. **Dra. Aida Araújo Ferreira** e aprovado pela banca examinadora formada pelos professores:

Recife, 14 de Novembro de 2018.

Profa. Dra. Aida Araújo Ferreira
CTADS/DASE/IFPE

Prof. Ms. Henrique Correia Torres
CTADS/DASE/IFPE

Profa. Dra. Vania Soares de Carvalho
IFPE

Davi da Cruz Aguiar

José Jorge dos Santos Neto

RESUMO

Embora a utilização do GPS atenda grande parte das situações relacionadas a posicionamento e localização, esta tecnologia possui limitações dentro de ambientes fechados, causando o enfraquecimento do sinal, e assim diminuindo a precisão dos sistemas baseados em tal tecnologia. Portanto, poderíamos utilizar um sistema de localização indoor, como solução alternativa as que são baseadas em GPS. Diante deste problema, este projeto teve como principal finalidade, implementar uma aplicação de localização indoor, fazendo uso de dispositivos móveis e web, que irá auxiliar os alunos e colaboradores do instituto federal de Pernambuco (IFPE), a localizarem de forma simples e intuitiva, uma determinada sala da instituição. Foi realizado um estudo buscando as principais tecnologias utilizadas na criação de um Sistema de informações Geográficas (SIG). Através desse estudo, foi definido e implementado um SIG Web, em que o método de desenvolvimento do sistema foi baseado no padrão de projetos MVC, incrementado com a utilização de um banco de dados geográfico, Shapefiles e dados vetoriais. Ao final, o resultado obtido foi satisfatório, visto que o sistema funcionou bem com as aplicações clientes.

Palavras-chave: Sigweb. Localização. *GPS. Mobile. Indoor*.

ABSTRACT

Although the use of GPS addresses a large part of the situations related to positioning and location, it can also suffer serious obstructions indoors. Weakening signal, decreasing precision and accuracy of systems based on such technology. Therefore, it is necessary to use indoor location and mapping systems, alternative to GPS-based solutions. Based on this problem, the main purpose of this project was to implement an easily accessible indoor location system using mobile and web devices, helping students and employees of the federal institute of Pernambuco (IFPE) to locate a certain room of the institute in a simple way. For this, a study was carried out looking for the main technologies used in the creation of a Geographic Information System (GIS). Through this study, a Web GIS was defined and implemented, in which the method of development of the system took place in the use of a spatial database and its Shapefiles, and Vector Data. At the end the result was satisfactory, since the system worked well with the client applications.

Keywords: Sigweb. Location. *GPS. Mobile. Indoor*.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de localização têm se tornado cada vez mais populares ao passar dos anos. Sejam eles para plataformas *desktop, web* ou *mobile*. Geralmente eles são divididos em duas grandes frentes no mercado atual: Os sistemas *outdoor* (ambientes abertos) e *indoor* (ambientes fechados).

A palavra *indoor* é do inglês e se refere a algo que está situado no interior de um local (INDOOR, 2018), algo que foi feito para ser usado em um estabelecimento ou algo que ocorreu no interior de determinado ambiente. Temos como antônimo a palavra *outdoor*, que tem as definições similares às anteriores, trocando o ambiente interno para o exterior de estabelecimentos, áreas abertas. Logo, localização *indoor* pode ser definida como a localização em espaços fechados e a localização *outdoor* é a localização em áreas abertas. Contudo, enquanto o problema da localização em tempo real em ambientes *outdoor* já é bem resolvido pelo sistema *GPS*, existem inúmeras aplicações que fazem uso da tecnologia. Já os sistemas *indoors*, são mais escassos e ainda estão em crescimento. Uma forma bastante interessante e similar ao tema proposto neste trabalho, é a aplicação da localização *indoor* tendo como base as ondas de radiofrequência.

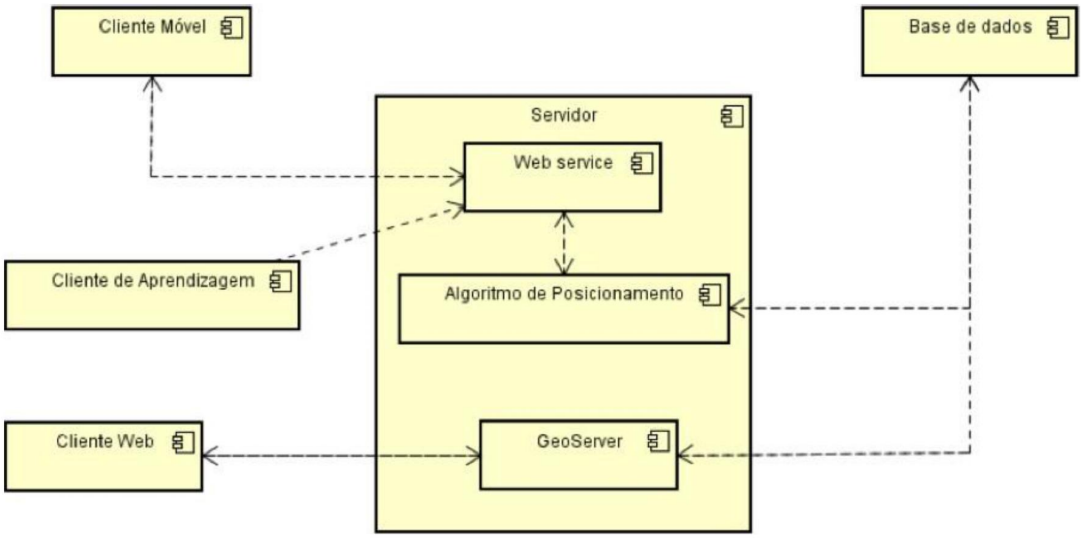
> É possível fazer uso de ondas de radiofrequência para localização *indoor*, através de mapeamento de ambiente prévio seguido da localização através de inteligência artificial como método knn, tendo como entrada os sinais de radiofrequência medidos pelo dispositivo a ser localizado e como saída as coordenadas da localização do dispositivo (LEITE, 2018)

A motivação deste projeto é facilitar a mobilidade do usuário dentro de um ambiente fechado, como por exemplo, o IFPE. Com isso, foi necessário o desenvolvimento de um sistema de localização *indoor*. Tendo como ideia básica, o uso da tecnologia de geolocalização em um sistema web que disponibilize informações de forma ágil sobre uma determinada área destacada no mapa.

Com isso em mente, um estudo de caso foi realizado no IFPE campus Recife, onde a equipe de pesquisa e desenvolvimento do projeto foi dividia em três módulos, a equipe era composta pelos alunos Marcus Lucas Abreu de Araujo Falcão (responsável pela implementação de um servidor em Java que implementa a técnica de posicionamento *indoor* RSSI *fingerprinting* utilizando o método de classificação KNN) (FALCÃO, 2018), Rodrigo

Venâncio de Lima Barbosa (responsável pela implementação de uma ferramenta que auxilia os usuários a se localizarem dentro de um ambiente fechado através da rede local) (BARBOSA, 2018), José Jorge dos Santos Neto e Davi da Cruz Aguiar (ambos responsáveis pelo módulo *web* apresentado neste trabalho). Todos são alunos do curso de Análise e Desenvolvimento de Sistemas, do IFPE, orientados pela professora Aida Araújo Ferreira (também professora deste curso). O diagrama do projeto como um todo pode ser observado na figura 1.

Figura 1 - Diagrama de componentes geral do projeto



Fonte: Os Autores (2018)

1.1 O PROJETO E SEUS MÓDULOS

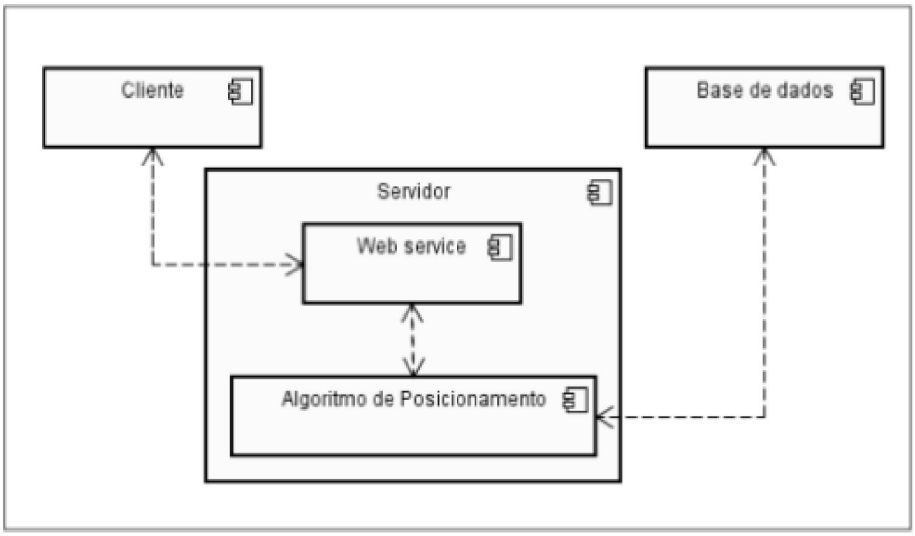
Para um total entendimento do projeto, esta seção apresenta os dois módulos que o complementam. São eles: o módulo servidor e o módulo Android para dispositivos móveis.

1.1.1 Módulo servidor

O servidor de posicionamento interno auxilia na localização de salas ou locais na instituição. Além da utilização de uma técnica de aprendizagem de máquina e todo o estudo

que permite a aplicação da mesma. Abaixo podemos ver a representação da estrutura do módulo servidor na figura 2.

Figura 2 - Estrutura geral do Servidor



Fonte: Os Autores (2018)

1.1.2 Módulo android

Com a popularização do sistema operacional Android, também apareceram muitas possibilidades e facilidades para os seus usuários. Uma delas é utilizar o dispositivo para localização em um lugar como um *shopping*, faculdade, supermercado, entre outros lugares, onde o usuário não os conhece e gostaria de se localizar rapidamente. Utilizando a rede sem fio do local, é possível localizar o dispositivo e consequentemente, guiar o usuário até o local que o mesmo deseja encontrar, como foi desenvolvido no módulo em questão (ver Figura 3). O Módulo Android do sistema, ou Absaber (BARBOSA, 2018), é uma aplicação Java baseada no protocolo *HTTP*, onde o cliente faz uma solicitação ao servidor e o mesmo responde. Para seu funcionamento foi utilizado um mapeamento dos *AP's (Acess Points)* existentes no campus e seus endereços *MAC (Media Acess Control)*. No IFPE existe uma rede *WI-FI* interna para visitantes onde existem diversos *AP's* espalhados no campus. Através da intensidade de sinal de cada *AP*, o aplicativo *Android* irá solicitar a localização ao servidor, que por sua vez tem guardada uma referência de todos os *AP's* e de suas tabelas de localização, onde o dispositivo Android do cliente, utilizando o aplicativo no celular, irá passar os valores de intensidade de

sinal encontradas atualmente e o local onde o usuário deseja chegar no campus. A aplicação utilizará o servidor (FALCÃO, 2018) por conta de possíveis mudanças de infraestrutura no campus e também por sua fase de aprendizagem. A mesma acontece quando o sistema é instalado ou precisa de calibração por conta de alguma mudança na infraestrutura de rede ou física do campus, ou até erros de localização que podem existir, fazendo com que as tabelas de localização no servidor sejam atualizadas, melhorando a localização.

Figura 3 - Sistema Absaber

Fonte: Barbosa (2018)

1.2 OBJETIVO GERAL

Disponibilizar um sistema *web* que poderá ser usado em empresas ou instituições, auxiliando os seus usuários a obterem, através de um dispositivo móvel ou web, as informações sobre um determinado setor à sua escolha. Exibindo sua localização dentro da instituição, facilitará o acesso e seu deslocamento através das rotas traçadas no mapa. Para este projeto utilizaremos como base o IFPE (Instituto Federal de Educação, Ciência e Tecnologia).

1.3 OBJETIVOS ESPECÍFICOS

1. Revisão bibliográfica sobre banco de dados espaciais, Javascript, GeoServer, *ShapeFiles*, Spring, Bootstrap;
2. Projetar a arquitetura do sistema de localização *indoor* na *web*;
3. Integrar arquitetura com a arquitetura do ABSABER (aplicação Java baseada no protocolo *HTTP*, onde o cliente faz uma solicitação ao servidor e o mesmo responde através do mapeamento dos *AP's* existentes no local) (BARBOSA, 2018);
4. Modelar o banco de dados;
5. Desenvolver o sistema;
6. Testar o sistema em diferentes navegadores Google Chrome, Safari, Internet Explorer;
7. Configurar o servidor de mapas;
8. Implantar o sistema.

1.4 JUSTIFICATIVA

A tecnologia *GPS* apesar de ser bastante poderosa para localizações geográficas, não é precisa o bastante em ambientes fechados, como *shopping centers*, universidades, etc. apresentando significativas taxas de erro. Os efeitos dos pequenos erros do *GPS* em um ambiente fechado podem ser gigantescos, ao ponto de que duas portas que se encontram a poucos metros de distância, podem ser confundidas causando uma instrução de direção errada. Tendo isso em mente, surgiu a ideia desse projeto, que tem como justificativa o seu uso em ambientes fechados.

2 REFERENCIAL TEÓRICO

2.1 TECNOLOGIAS UTILIZADAS

2.1.1 Sistemas de informação geográfica

Segundo Fitz (2008) e MEDEIROS (2016a), um sistema de informações geográficas (SIG) é um conjunto de funções automatizadas, que fornecem aos profissionais, capacidades avançadas de armazenamento, acesso, manipulação e visualização da Informação georreferenciada. Que permite e facilita a análise da informação geográfica. A utilização dos SIGs possibilitam um melhor gerenciamento e visualizações das informações. A importância de um SIG se deve principalmente na facilidade de trabalhar com uma grande quantidade de informação, tempo de resposta no apoio à decisão, fácil gestão e armazenamento dos dados geográficos e rapidez com que a informação pode ser atualizada (ver Figura 4). Atualmente a popularização de ferramentas SIG tem crescido cada vez mais. Trazendo o surgimento de novos e acessíveis recursos na área.

Figura 4 - SIG

Fonte: Medeiros (2016a)

2.1.2 Banco de dados geográficos

Um banco de dados geográfico é parte fundamental dos componentes de um SIG, pois é nele que estão armazenadas as referências da relação do dado com o mundo real (ver Figura 5),

os dados georreferenciados (MACHRY, 2018). Dados georreferenciados (ou geoespaciais) é o nome atribuído às informações manipuladas pelas aplicações de Geoprocessamento (FERREIRA, 2016). Esses dados recebem esta denominação por possuírem atributos relacionados à sua localização geográfica em um sistema de coordenadas. Por meio do banco de dados geográfico, é possível um SIG realizar processamentos geométricos, análise espacial e fazer relação entre dados convencionais e espaciais.

> No caso particular dos bancos de dados espaciais, questões referentes ao armazenamento da representação (geometria) dos dados espaciais, bem como dos relacionamentos entre eles (especialmente os topológicos) são relevantes e carecem de estruturas de dados específicas. A geometria diz respeito à forma geométrica utilizada para representar o dado espacial e está intimamente ligada à escala de trabalho e ao objetivo da aplicação (SILVA, 2002, p. 19).

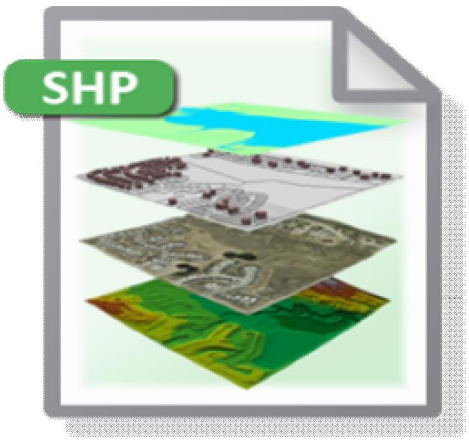
Figura 5 - Banco de dados geográficos

Fonte: Medeiros (2016b)

2.1.3 Arquivos shapefile e dados vetoriais

O *Shapefile* é um formato de arquivo contendo dados geoespaciais em forma de vetor (ESRI, 1998). As estruturas de dados vetoriais são utilizadas para representar as coordenadas das fronteiras de cada entidade geográfica através de três tipos básicos: ponto, linha e polígono. Devemos utilizar esses tipos de dados conforme a informação representada e a informação desejada. Um ponto pode representar um poste, pessoas, locais de ocorrências, as linhas podem representar riachos e estradas, enquanto que polígonos levam em consideração a distribuição espacial dos dados, podendo representar todo um país ou apenas um bairro. O *Shapefile* pode apresentar uma ou várias camadas para representação geográfica (ver Figura 6).

Figura 6 - Shapefile

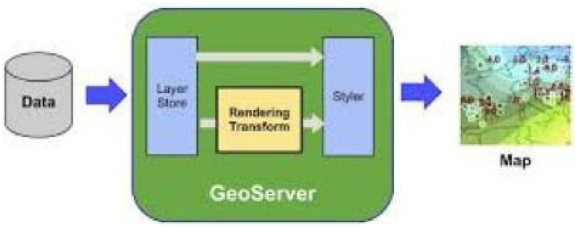


Fonte: ESRI (2016)

2.1.4 Servidor de mapas geoserver

Um servidor de mapas é o componente central de um SIG *web*, pois ele que estabelece a comunicação entre o cliente e o banco de dados geográficos ou arquivos *Shapefile*, retornando ao cliente as informações na forma de mapas (ver Figura 7). O GeoServer é um servidor de código aberto escrito em Java, permite que os usuários compartilhem e editem dados geoespaciais (GEOSERVER, 2016). Ele permite a implementação dos três componentes principais de um SIG web: Visualização do mapa, visualização das camadas e operações de Geoprocessamento.

Figura 7 - Geoserver

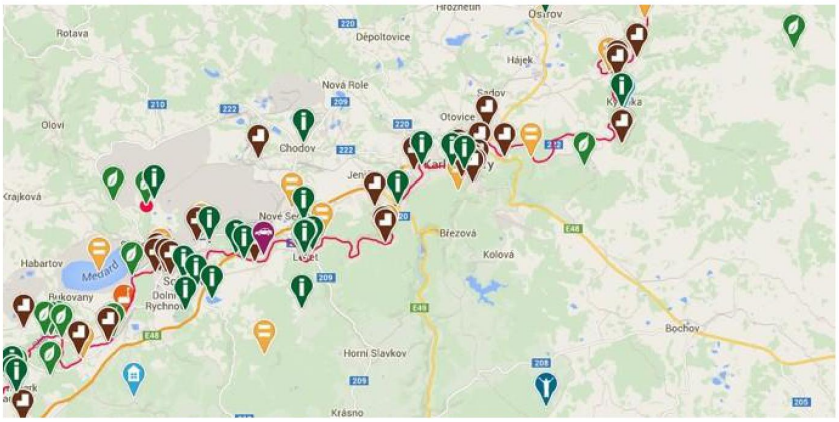


Fonte: Illek (2016)

2.1.5 Openlayers

O *OpenLayers* é uma biblioteca de código aberto desenvolvida em javascript para a visualização de mapas em navegadores (OPENLAYERS, 2016). O *OpenLayers* permite também a associação com diversos tipos de mapa base, no qual representam a visualização da superfície terrestre, sendo indispensáveis para um SIG *web*. Dentre os mapas suportados pelo *OpenLayers*, se destaca principalmente o mapa do Google Maps. O Google Maps fornece pesquisa e visualização de mapas e imagens de satélites que são atualizadas constantemente. A figura 8 mostra uma aplicação utilizando o *OpenLayers* e o *Google Maps* como mapa base.

Figura 8 - Uso do OpenLayers



Fonte: Illek (2016)

2.1.6 Javascript

O JavaScript inicialmente conhecido pelo nome LiveScript, foi criado por um programador da Netscape chamado Brendan Eich em setembro de 1995, como parte do projeto do navegador Netscape 2.0, e só então renomeado para o nome conhecido hoje em dia, em dezembro de 1995. Segundo Luiz (1998) a linguagem foi desenvolvida para que pudesse ser executada no lado do cliente e tivesse interação com o usuário, com a função de controlar o navegador, realizando comunicação assíncrona e alterando o conteúdo do site. Em novembro de 1996 a Netscape anunciou que tinha submetido o JavaScript para Ecma internacional (uma associação industrial fundada em 1961, dedicada a padronização de informação e comunicação de sistemas), como padrão industrial e o trabalho subsequente resultou na versão

padronizada chamada *ECMAScript*. Inicialmente a linguagem foi denegrida, porém com o advento do Ajax, o JavaScript teve sua popularidade de volta e recebeu mais atenção profissional. O resultado foi a proliferação de *frameworks*, bibliotecas, práticas de programação melhoradas e o aumento do uso fora de navegadores, bem como o uso de plataformas *server-side*.

2.1.6.1 Baseada em objetos

JavaScript é quase inteiramente baseada em objetos, onde eles são representados por *arrays* associativos, aumentados com protótipos. Propriedades e valores podem ser adicionados, modificados, ou deletados em tempo de execução. A maioria das propriedades de um objeto (e aqueles em sua cadeia de herança via protótipo) pode ser enumerada usando-se uma estrutura de repetição *for...in*.

2.1.6.2 Protótipos

JavaScript usa protótipos em vez de classes para o mecanismo de herança. É possível simular muitas características de orientação a objetos baseada em classes com protótipos.

2.1.6.3 Compatibilidade

Já que JavaScript roda em ambientes variados, uma parte importante do teste e depuração de seu código consiste na verificação de compatibilidade entre navegadores. As interfaces *DOM (Document Object Model)* para a manipulação de páginas *web* não são parte do padrão *ECMA*, ou do próprio JavaScript. Oficialmente, são definidas por um esforço de padronização da *W3C (World Wide Web Consortium)*. Na prática, implementações de navegadores diferem do padrão de uma para as outras, e nem todos navegadores executam JavaScript. Para lidar com essas diferenças, programadores JavaScript, com frequência tentam escrever códigos que conformam com o padrão comum a maioria dos navegadores, não sendo possível isso, tentam escrever de maneira *ad-hoc* um código que verifique a presença de certos recursos e que se comporte de maneiras diferentes. Os programadores podem ainda achar prático detectar qual navegador está rodando e mudar o comportamento de seus *scripts* para que possa

se adequar a cada um ou até mesmo usar bibliotecas ou ferramentas que abstraem tais diferenças entre navegadores.

2.1.7 Bootstrap

Bootstrap, originalmente chamado *Twitter Blueprint*, foi desenvolvido por Mark Otto e Jacob Thornton no Twitter como uma estrutura para padronizar as ferramentas internas. Antes do Bootstrap, várias bibliotecas foram utilizadas para desenvolvimento de interface, o que levou a inconsistências e gastos elevados de manutenção. Em 31 de Janeiro de 2012, Bootstrap 2 foi anunciado. Esta versão adicionou o layout de doze colunas de grade e componentes de design responsivo, bem como as alterações dos componentes existentes. A liberação do Bootstrap 3 foi anunciado em 19 de agosto de 2013, seguindo para uma primeira abordagem móvel e usando um *design flat*. Em 29 de Outubro, 2014, Mark Otto anunciou que o Bootstrap 4 estava em desenvolvimento. A primeira versão alfa do Bootstrap 4 foi implantada em 19 de agosto de 2015 (BOOTSTRAP, 2011).

2.1.7.1 Container

O *container* é nada mais que o comportamento de ocupar a largura máxima do *layout*, sua altura vai depender de como os componentes estão posicionados dentro do mesmo, todo componente dentro do *container* utiliza se do mesmo como base para se posicionar no *layout*.

2.1.7.2 Columns

Os *columns* dividem as *rows* na horizontal(colunas), desta forma obrigatoriamente toda *column* têm que está dentro de uma *row* para apresentar este comportamento.

As *columns* apresenta duas propriedades importantes primeiro col-(dispositivo)-(tamanho da coluna).

Figura 9 - Exemplo Columns

```
1  <div class = "container">
2      <div class="row">
3          <div class="col-sm-6">COLUNA 1</div>
4          <div class="col-sm-6">COLUNA 2</div>
5      </div>
6      <div class="row">
7          <div class="col-sm-6">COLUNA 1</div>
8          <div class="col-sm-6">COLUNA 2</div>
9      </div>
10 </div>
```

Fonte: Os autores (2018)

No exemplo acima resultará em um *layout* responsivo com 2 linhas, onde cada linha terá 2 colunas. Para aplicar o comportamento de coluna, é necessário adicionar a classe da *column* desejada com seu respectivo tamanho. Nas linhas 3, 4, 7 e 8 é adicionado colunas com tamanho 6 cada. Esse *layout* apenas responderá de forma responsivo para celulares e *tablets*.

Uma *row* comporta no máximo o somatório do tamanho das *columns* menor ou igual a 12, e o tamanho da coluna é uma variação de 1 à 12, ou seja, podemos descrever que a classe que descreve uma column têm a seguinte composição col-[dispositivo]-[tamanho desejado].

2.1.8 Spring Framework

O Spring é um *framework* open source para a plataforma Java criado por Rod Johnson e descrito em seu livro "Expert One-on-One: JEE Design e Development". É um *framework* não intrusivo, baseado nos padrões de projeto inversão de controle (IoC) e injeção de dependência.

No Spring o container se encarrega de "instanciar"classes de uma aplicação Java e definir as dependências entre elas através de um arquivo de configuração em formato XML, inferências do *framework*, o que é chamado de *auto-wiring* ou ainda anotações nas classes, métodos e propriedades. Dessa forma o Spring permite o baixo acoplamento entre classes de uma aplicação orientada a objetos.

O Spring possui uma arquitetura baseada em interfaces e POJOs (*Plain Old Java Objects*), oferecendo aos POJOs características como mecanismos de segurança e controle de

transações. Também facilita testes unitários e surge como uma alternativa à complexidade existente no uso de EJBs. Com Spring, pode-se ter um alto desempenho da aplicacao.

Esse *framework* oferece diversos módulos que podem ser utilizados de acordo com as necessidades do projeto, como módulos voltados para o desenvolvimento *web*; persistência, acesso remoto e programação orientada a aspectos (WIKIPEDIA, 2018).

2.1.8.1 Controller

Abaixo podemos visualizar um exemplo de implementação de um controller no *Spring Framework*.

Figura 10 - Exemplo Controller

```
1  @Controller
2  public class OlaMundoController{
3
4      @RequestMapping("/olaMundoSpring")
5      public String execute() {
6          System.out.println("Executando a lógica com Spring MVC");
7          return "ok";
8      }
9  }
```

Fonte: Caelum (2018)

2.1.8.2 View

A seguir podemos ver a implementação de uma *view*, que é uma página para exibição dos dados enviados através do controller.

Figura 11 - Exemplo View

```
1  <html>
2      <body>
3          <h2>Olá mundo com Spring MVC!</h2>
4      </body>
5  </html>
```

Fonte: Os autores (2018)

3 METODOLOGIA

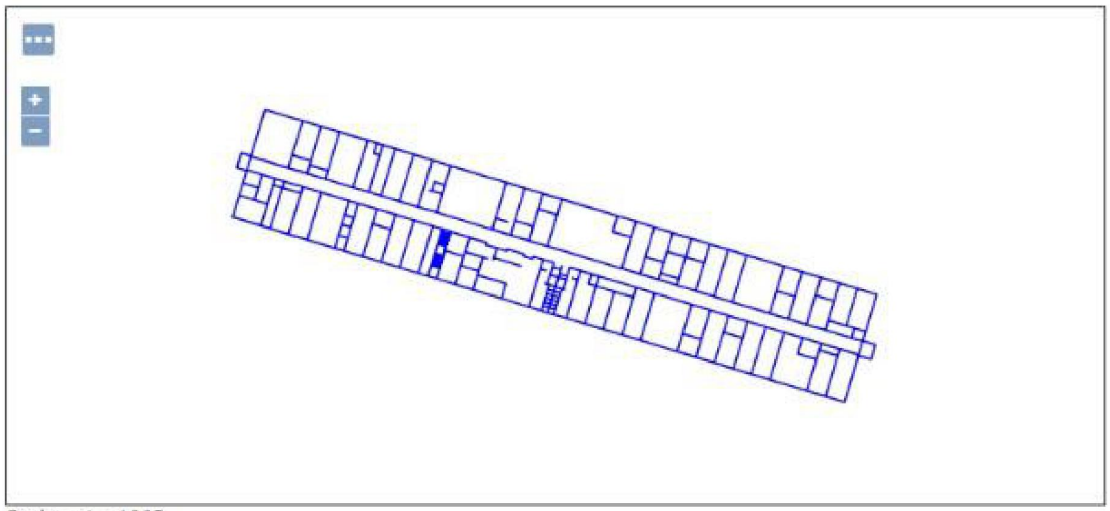
O processo de desenvolvimento desse projeto teve início com o levantamento de referências bibliográficas sobre o tema e informações em sites especializados, dando prioridade a fontes que apresentassem exemplos e que atendessem as especificações e o desenvolvimento da aplicação, visando alcançar o objetivo de implementar o nosso sistema web de geolocalização. Com isso, primeiro decidimos qual seria o servidor do banco de dados geográfico que seria utilizado no sistema. Através do estudo realizado, foi escolhido como este componente, o Geoserver que seria responsável pela comunicação entre cliente e o banco de dados geográfico do sistema. Após isso, definimos a integração do mesmo com o banco de dados cadastral e a comunicação com o cliente *web* da aplicação. Com os pilares principais da aplicação já definidos, tivemos que resolver o problema de como aplicar o conceito do GeoServer e seus *Shapefiles* ao IFPE. Para isso, o método utilizado foi customizar as plantas baixas do instituto (cedidas por alunos do curso de edificação para a realização do nosso trabalho) para a criação de *Shapefile* de pontos a partir da localização escolhida na planta. Após a customização da planta, o arquivo foi convertido para *Shapefile*, sendo possível sua utilização dentro do nosso banco de dados geográfico. Após a etapa da pesquisa e de banco de dados, partimos para a criação do cliente *web*, onde seria necessária uma tecnologia que atendesse alguns requisitos, como multiplataforma, com os *inputs* nativos Clark. Pois, a utilização do HTML5 com a definição de *inputs* com tipos específicos (Telefone, Data, Hora, etc.) garante a manipulação da informação, da forma que o usuário está mais acostumado. Para o desenvolvimento do sistema, era necessário verificar a existência de algum *framework* que adiantasse o processo do desenvolvimento e abstraisse toda a manipulação de *layout*, exigindo pouca experiência no desenvolvimento do módulo criado. Tendo isso em mente, o Bootstrap, junto ao padrão de desenvolvimento MVC, foram escolhidos para criação da *interface* das páginas, simplificando a criação dos componentes *web*.

3.1 RESULTADOS

A figura 12, mostra um *Shapefile* obtido após o resultado da customização e conversão de uma planta baixa do IFPE. Já na Figura 13, podemos ver a comunicação realizada entre os

principais componentes do sistema. Onde é verificado que objeto central o Geoserver, estabelece a comunicação entre o banco de dados e o cliente *web*, que por sua vez obedece ao padrão MVC adotado no desenvolvimento do sistema.

Figura 12 - Planta baixa após ser convertida para Shapefile



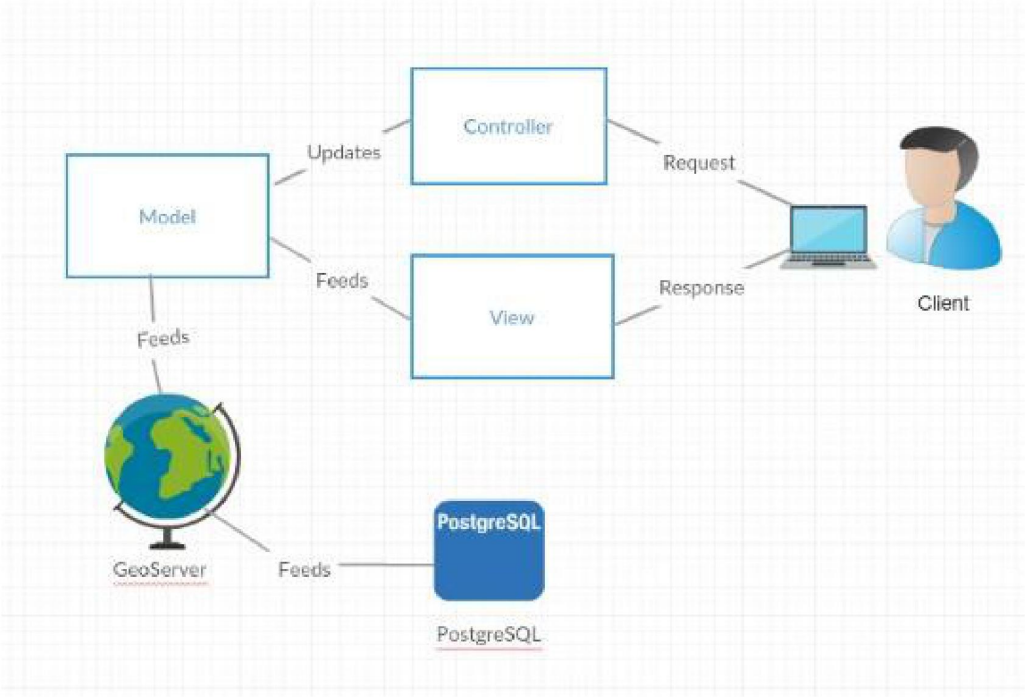
Scale = 1 : 1065

geo_bloco_a0

fid	fid_	entity	layer	color	linetype	elevation	linewt	refname
geo_bloco_a0.1	0			0		0E-11	0	

Fonte: Os Autores (2018).

Figura 13 - Comunicação entre os pilares do sistema



Fonte: Os autores (2018)

4 DESENVOLVIMENTO EARQUITETURA

Nesse capítulo será apresentado como foi o processo de desenvolvimento da aplicação, e quais foram as decisões tomadas que permitiram alcançar o objetivo principal.

4.1 REQUISITOS

Esta sessão especifica os requisitos do projeto. Seu propósito é organizar, analisar e definir as necessidades do cliente e as características que o sistema deve prover, focando nos requisitos técnicos identificados.

4.1.1 Visão geral

- **Requisitos funcionais**: lista os requisitos funcionais do sistema, especificando seus objetivos e prioridades.
- **Requisitos não funcionais:** especifica todos os requisitos não funcionais do sistema, divididos em requisitos de usabilidade, confiabilidade, desempenho, segurança, distribuição, adequação a padrões e requisitos de *hardware* e *software*.

4.1.2 Prioridade dos requisitos

- **Essencial:** requisito sem o qual o sistema não entra em funcionamento. Requisitos essenciais são requisitos imprescindíveis, devem ser implementados desde as primeiras implantações do sistema.
- **Importante:** requisito sem o qual o sistema entra em funcionamento, mas de forma não satisfatória. Requisitos importantes devem ser implantados o mais rápido possível, mas, se não forem, parte do sistema poderá ser implantada mesmo assim.
- **Desejável:** requisito que não compromete as funcionalidades básicas do sistema, isto é, o sistema pode funcionar de forma satisfatória sem ele. Requisitos desejáveis são requisitos que podem ser implantados por último, sem comprometer o funcionamento do sistema.

4.2 REQUISITOS FUNCIONAIS

- **Consultar Mapa Automaticamente -** Deve disponibilizar para o usuário, o mapa completo da instituição. Bem como, todos os setores que constituem o mesmo.

Prioridade: *Essencial*

- **Consultar Localização Manualmente -** Deve ser capaz de pesquisar um local solicitado e informado manualmente pelo usuário, desde que esse local esteja dentro da instituição.

Prioridade: *Essencial*

- **Disponibilizar informações -** Ao selecionar o setor pesquisado, o sistema deve ser capaz de mostrar as informações detalhadas do local ao usuário.

Prioridade: *Importante*

- **Informar posição de destino -** Deve ser capaz de mostrar no mapa a posição do local escolhido pelo usuário como destino final.

Prioridade: *Desejável*

- **Traçar rota do Ponto Inicial até Ponto Destino -** A aplicação deve traçar a rota de um local inicial que o cliente deseja, até o local de destino informado em pesquisa realizada.

Prioridade: *Desejável*

- **Cadastro de Usuário -** O sistema deve permitir o cadastro de novos usuários administrativos do sistema. Estes usuários são os que terão permissão de inclusão e edição de salas e blocos da instituição.

Prioridade: *Importante*

- **Cadastro de Bloco -** O sistema deve permitir o cadastro de novos blocos da instituição. Desde que esse cadastro seja realizado por um usuário logado no sistema.

Prioridade: *Importante*

- **Cadastro de Sala -** O sistema deve permitir o cadastro de novas salas da instituição. Desde que esse cadastro seja realizado por um usuário logado no sistema, e também que este usuário informe a localização geográfica das mesmas.

Prioridade: *Importante*

- **Mapa no cadastro de sala -** O sistema deve disponibilizar o mapa da instituição na tela de cadastro de novas salas da instituição. Este mapa servirá para que o usuário logado indique a localização geográfica da sala a ser cadastrada. **Prioridade:** *Importante*

4.3 REQUISITOS NÃO FUNCIONAIS

- **Desenvolvimento em cascata -** O desenvolvimento deve ser feito em modelo cascata. Em que a ideia principal, é que as diferentes etapas de desenvolvimento seguem uma sequência.

  Prioridade: *Essencial*

- **Linguagem de desenvolvimento -** O projeto deve ser implementado na linguagem de programação Java.

  Prioridade: *Essencial*

- **Interface -** A interface da aplicação deve ser simples e intuitiva.

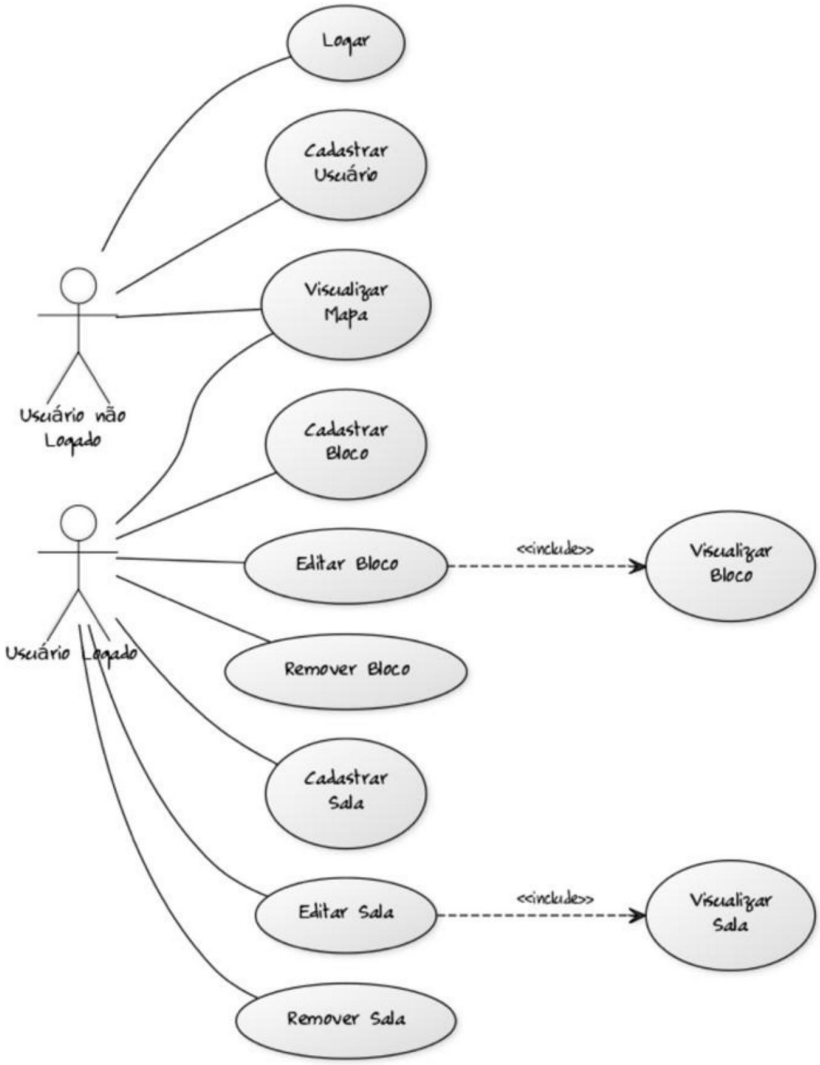
  Prioridade: *Importante*

- ***Case Insensitive* -** A pesquisa em rota do sistema não deve ser *Case-Sensitive*. Ou seja, não deve diferenciar letras maiúsculas de minúsculas, para melhor eficiência nas buscas.

  Prioridade: *Essencial*

- **Estatística de uso -** O sistema deve realizar estatísticas das rotas concluídas, canceladas e pesquisadas, realizadas pelo usuário.
- **Prioridade:** *Desejável*

Figura 14 - Casos de Uso



Fonte: Os autores (2018)

4.4 CASOS DE USO

4.4.1 Cadastrar bloco

- Após o usuário ter feito o *login*, clicar no menu Blocos
- Então a tela da listagem de blocos irá aparecer
- Clicar no botão Cadastrar Novo Bloco

- A tela de cadastrar bloco irá aparecer. Ao preencher todos os dados, clicando em cadastrar, irá salvar o novo bloco
-

4.4.2 Editar bloco

- Após ter feito o *login*, clicar no menu Blocos
- A tela de listagem dos blocos será exibida
- Encontrar o bloco que deseja editar na listagem, em seguida clicar em editar
- Ao abrir a nova tela, modificar os dados desejados, em seguida clicar em cadastrar

4.4.3 Visualizar bloco

- Após ter feito o *login*, clicar no menu Blocos
- Uma listagem com todos os blocos cadastrados será exibida, com os dados dos blocos em questão

4.4.4 Remover bloco

- Após ter feito o *login*, clicar no menu Blocos
- A tela de listagem dos blocos será exibida
- Encontrar o bloco que deseja remover na listagem, em seguida clicar em excluir

4.4.5 Cadastrar sala

- Após o usuário ter feito o *login*, clicar no menu Salas
- Então a tela da listagem de salas irá aparecer
- Clicar no botão Cadastrar Nova Sala
- A tela de cadastrar sala irá aparecer. Ao preencher todos os dados, clicando em cadastrar, irá salvar o novo bloco

4.4.6 Editar sala

- Após ter feito o *login*, clicar no menu Salas
- A tela de listagem das salas será exibida
- Encontrar a sala que deseja editar na listagem, em seguida clicar em editar
- Ao abrir a nova tela, modificar os dados desejados, em seguida clicar em cadastrar

4.4.7 Visualizar sala

- Após ter feito o *login*, clicar no menu Salas
- Uma listagem com todas as salas cadastradas será exibida, com os dados das salas em questão

4.4.8 Remover sala

- Após ter feito o *login*, clicar no menu Salas
- A tela de listagem das salas será exibida
- Encontrar a sala que deseja remover na listagem, em seguida clicar em excluir

4.4.9 Visualizar mapa

- Clicar em Mapa no menu superior
- O usuário terá acesso mesmo sem ter efetuado o *login* no sistema
- A tela com o mapa será exibida
- Após preencher o cadastro, clicar no botão cadastrar usuário

4.4.10 Cadastrar usuário

- Na tela de *login* do sistema existirá o botão cadastrar usuário
- Ao clicar em cadastrar usuário, a tela de cadastrar usuário será exibida
- Após preencher o cadastro, clicar no botão cadastrar usuário
- A tela de cadastrar sala irá aparecer. Ao preencher todos os dados, clicando em cadastrar, irá salvar a nova sala

4.4.11 Logar

- Na tela inicial do sistema, existirá um campo para usuário e outro para senha
- Após preencher os dados, clicar em *Login*
- O usuário será redirecionado à tela de visualizar o mapa no sistema

4.5 ARQUITETURA

A aplicação é um SIG *web*, que entre muitas outras possibilidades, permite que o usuário interaja continuamente com o sistema, expedindo instruções e recebendo respostas. Levando em conta a facilidade de acesso que se dá basicamente a partir de um simples browser. Aqui, nesta sessão, veremos quais são os principais componentes que formam a arquitetura deste projeto, e como eles se comunicam entre si.

4.5.1 Visão geral

Basicamente o sistema é dividido em cinco elementos, detalhados abaixo:

- O elemento cliente: Se refere à interface gráfica do sistema onde o usuário deve interagir diretamente através de um navegador *web* como o Google Chrome, Mozilla Firefox, dentre outros.
- O elemento servidor web: Se refere ao servidor utilizado em sites comuns como o Apache Tomcat, ou o Glassfish.
- O elemento servidor de mapas: Se refere ao software que por meio de requisições do cliente, faz a comunicação com os dados espaciais e reproduz as informações obtidas em forma de mapas como por exemplo o Geoserver.
- O banco de dados geográfico: Neste projeto é representado pelos arquivos *Shapefile*, arquivos que guardam estruturas geográficas.
- Liguagem de programação: A codificação implementada neste sistemas foi realizada por meio da linguagem Java *web* baseando-se no padrão Spring MVC.

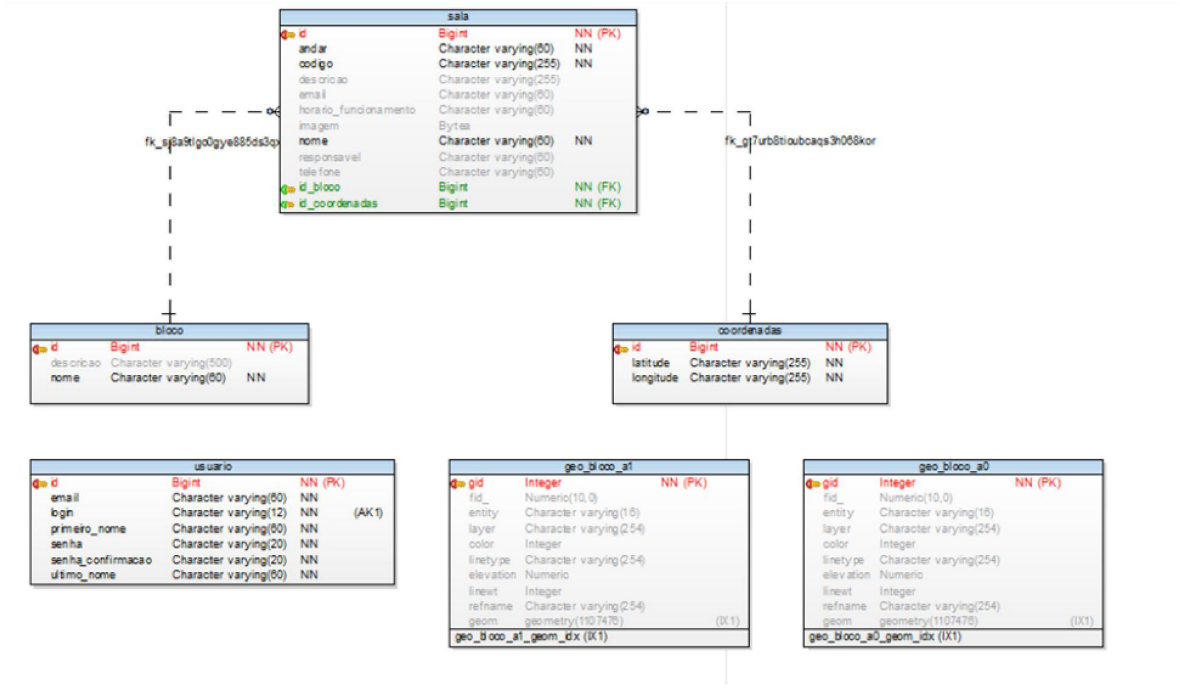
Figura 15 - Elementos da arquitetura do sistema

Fonte: Os autores (2018)

4.5.2 Diagrama do banco de dados

O banco de dados do sistema foi criado no Postgres, que é um sistema gerenciador de banco de dados objeto relacional PostgreSQL (2004), e ele foi modelado e estruturado em seis tabelas. Onde, duas tabelas são responsáveis por guardar os dados dos *shapes* do mapa da instituição (ou seja, a representação geográfica das plantas baixas da estrutura da instituição), uma tabela responsável por guardar as salas pertencentes à instituição, uma tabela responsável por guardar os blocos e outra responsável por guardar as informações relacionadas aos usuários como mostra a figura 16. A relação entre as tabelas bloco e sala, e a relação entre sala e coordenadas são ambas de um para vários. Ou seja, um bloco pode ter várias salas, mas uma sala só pertence a um bloco, da mesma forma que uma sala pode ter um par de coordenadas geográficas (latitude e longitude), mas a localização só pertence a uma sala. A tabela usuário não possui relacionamento com nenhuma outra tabela. Um *plugi*n do banco de dados recomendado para um SIG e utilizado neste projeto, é o PostGIS, uma extensão do Postgres que trabalha com dados espaciais. Com o PostGIS é possível guardar na mesma base todos os dados numéricos e textuais juntamente com os dados espaciais.

Figura 16 - Diagrama do bando de dados



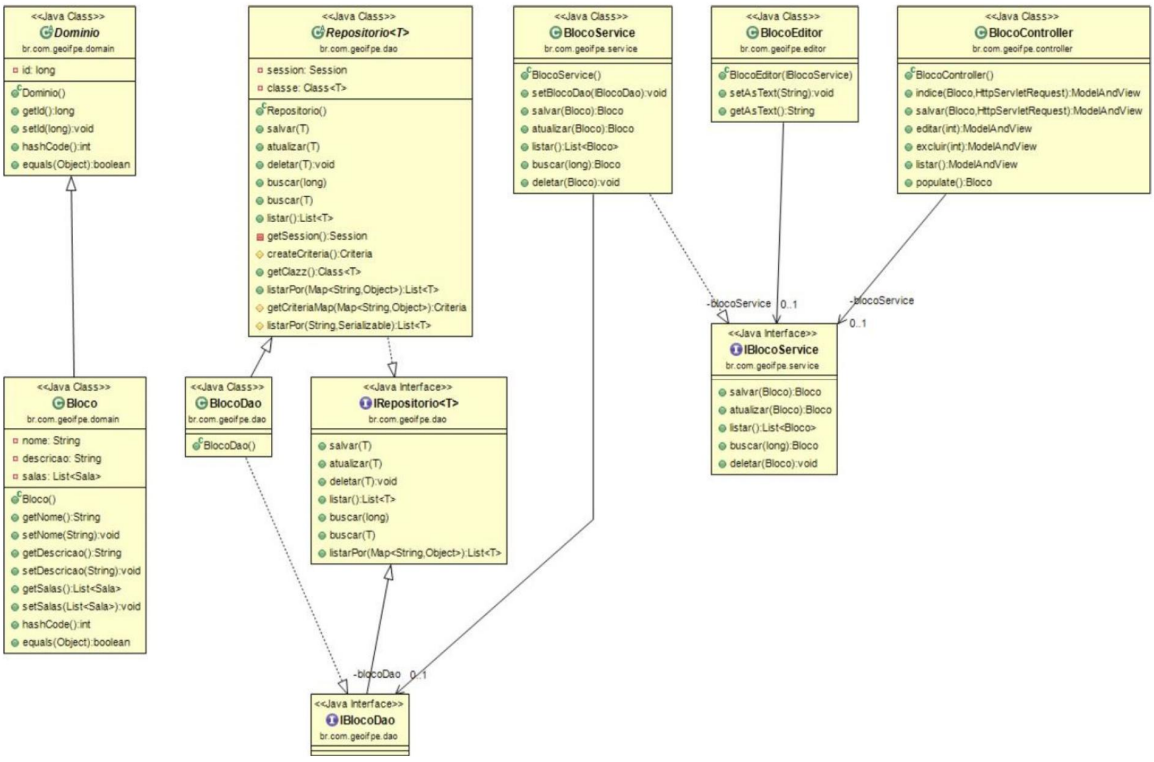
Fonte: Os autores (2018)

4.5.3 Diagrama de classes

Para definição do fluxo das classes e como elas conversariam entre si, primeiramente definimos a plataforma e como os componentes do sistema iriam se organizar. O padrão de projeto que escolhemos para este trabalho foi o modelo *Model-View-Controller* que proporciona aos desenvolvedores uma manutenção mais fácil e o possível reaproveitamento de classes. A utilização do padrão MVC traz como benefício isolar as regras de negócios da lógica de apresentação, e interface com o usuário (OMAR, 2004). Uma das características de um padrão de projeto é poder aplicá-lo em sistemas distintos. O padrão MVC pode ser utilizado em vários tipos de projetos como, por exemplo, *desktop, web e mobile*, que era nossa necessidade inicial. O *framework* Spring, foi o escolhido para a implementação do modelo MVC. Conforme a definição da própria iniciativa Spring, além do suporte para a arquitetura N camadas com o Spring MVC, fornece também uma implementação para os padrões de Injeção de Dependência. E a para a conversação com o banco de dados, adotamos o Hibernate que tem como objetivo tornar a associação entre os objetos da aplicação e a base de dados que irá persisti-los. Isso simplifica a consulta ao banco de dados, nos livrando da tarefa manual de escrever a relação entre objetos da aplicação e as tabelas, dispensando, por exemplo, o uso de

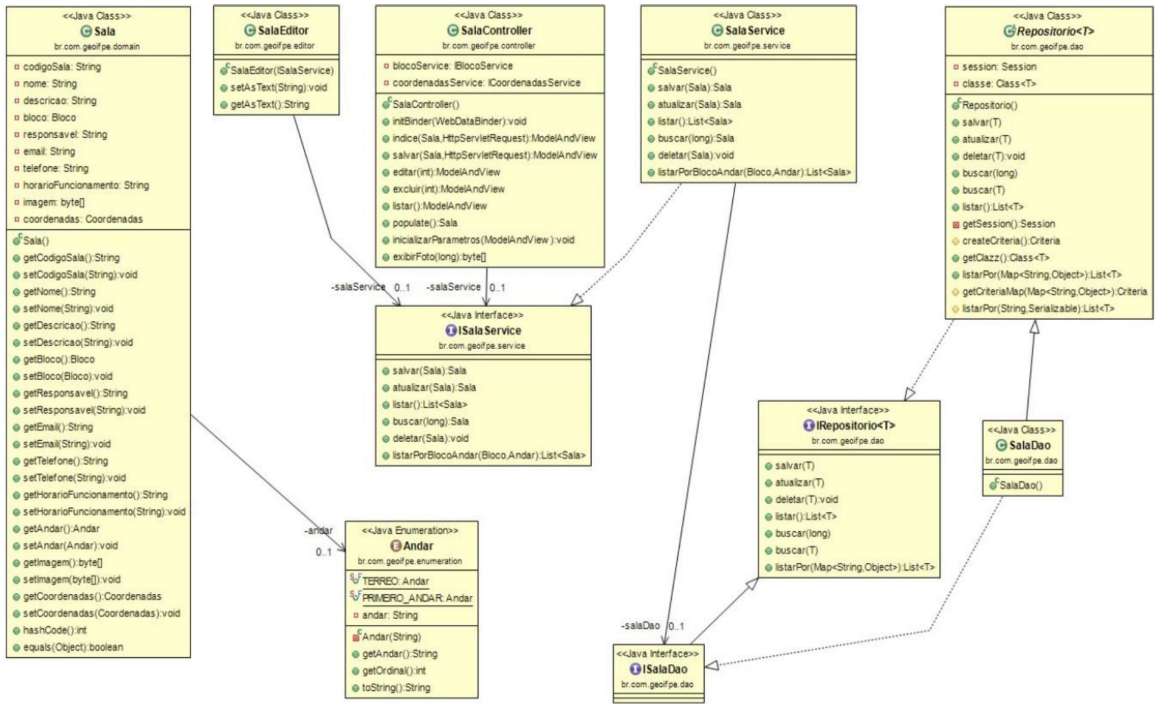
SQL. Para o tratamento das dependências do projeto, utilizamos o Maven que é uma ferramenta desenvolvida pela Apache, ela serve para gerenciar as dependências e automatizar seus *builds*. O Maven indica que você estará fazendo uso de uma determinada dependência em seu projeto, e faz o *download* dela e armazena em um repositório local no seu computador, se você possuir mais de um projeto fazendo uso da mesma dependência você só terá um único arquivo no seu computador, no seu repositório local organizado pelo Maven. O que é outra grande vantagem a dependência estar em um repositório local, e não na pasta do seu projeto. Por fim, após o uso dessas tecnologias e implementação das classes do projeto, as figuras mostram como o diagrama ficou. A Figura 17 exibe a estrutura das classes que são responsáveis pela parte de blocos. Enquanto a Figura 18 apresenta a estrutura das classes responsáveis pelas salas, a Figura 19 apresenta a estrutura que é responsável pelos mapas e a Figura 20 mostra como o *login* está estruturado.

Figura 17 - Diagrama de classes da estutura de blocos



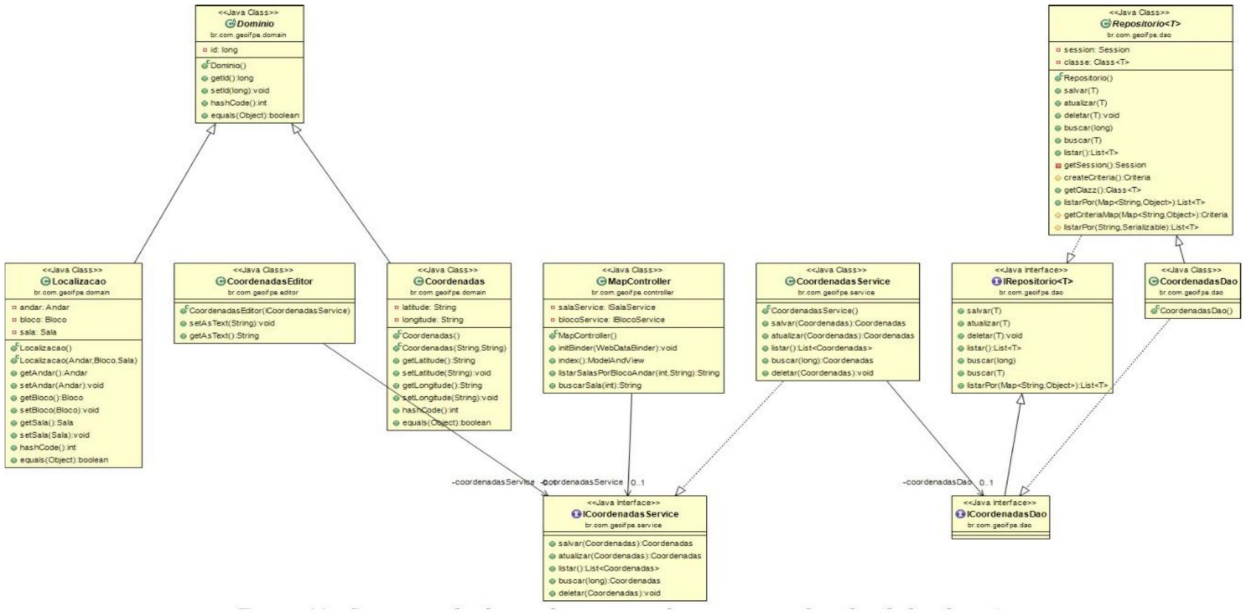
Fonte: Os autores (2018)

Figura 18 - Diagrama de classes da estrutura de salas



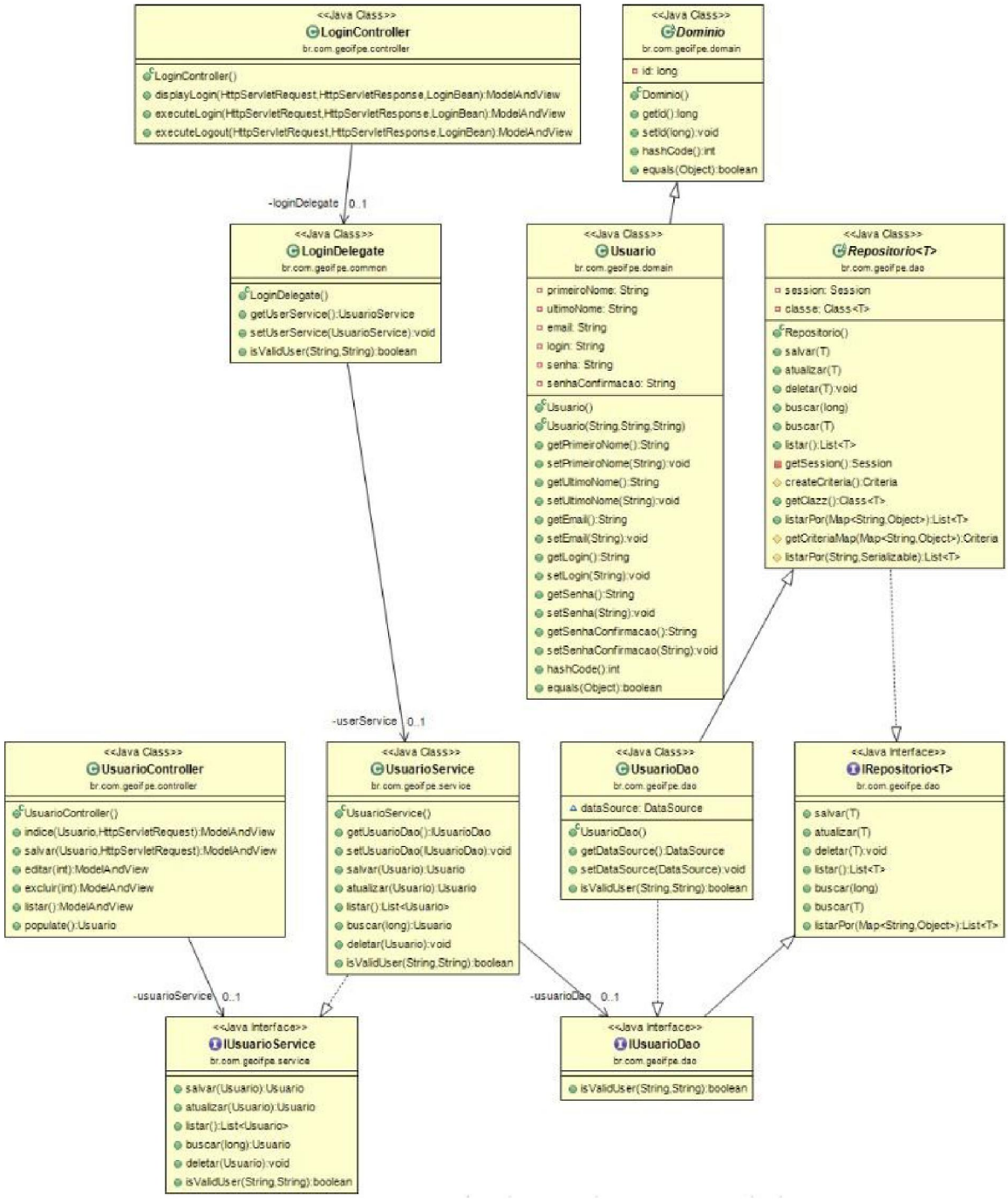
Fonte:Os autores (2018)

Figura 19 - Diagrama de classes da estrutura de mapa e coordenadas de localização



Fonte: Os autores (2018)

Figura 20 - Diagrama de classes da estrutura de login

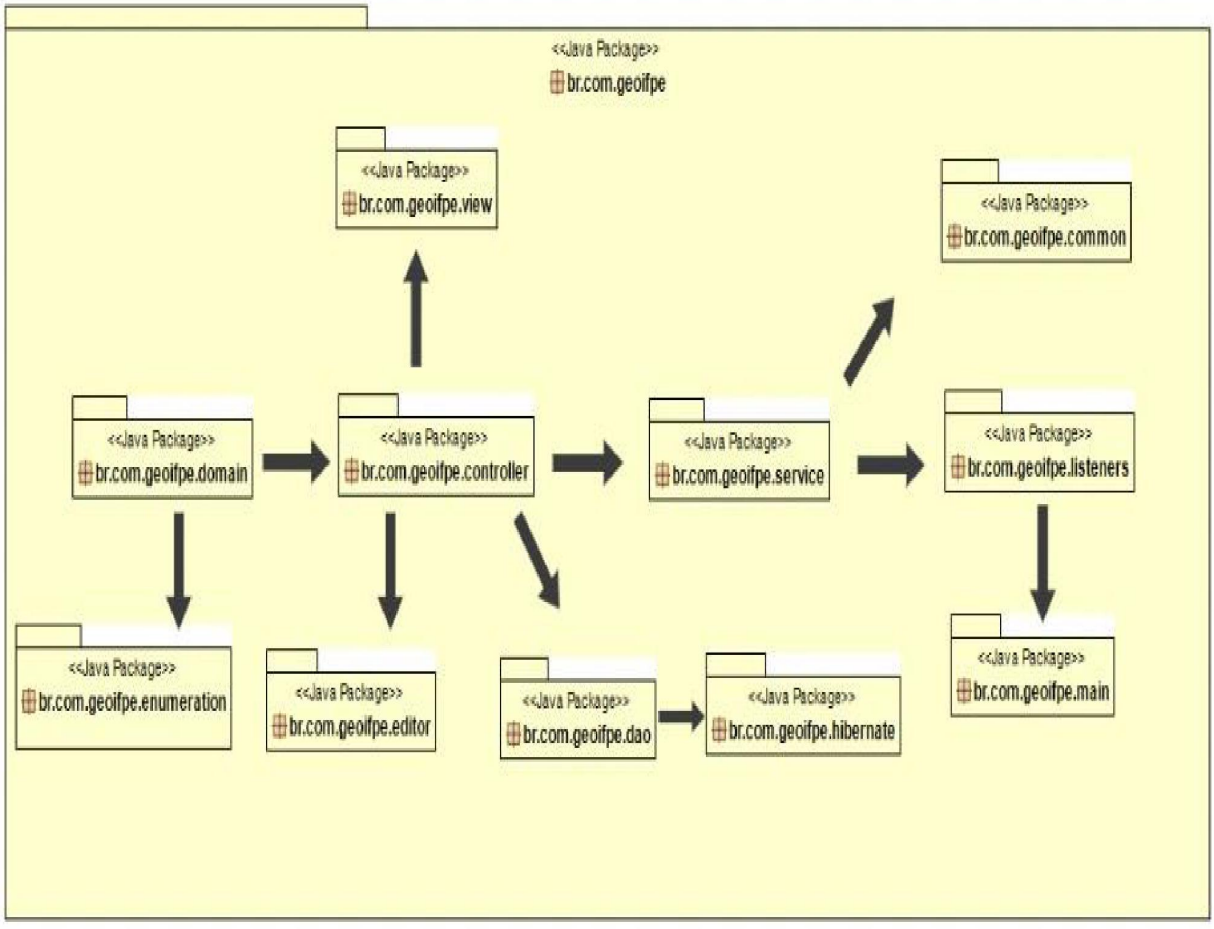


Fonte: Os autores (2018)

4.5.4 Diagrama de pacotes

Visto que o diagrama de classes pode ser grande para representar todo o sistema. Foi necessário deixar a representação do fluxo de classes da aplicação com uma menor complexidade visual, quebrando esse sistema em subpartes menores. Para isto criamos o diagrama de pacotes, também conhecido como diagramas de módulos. Nesse diagrama (ver Figura 21) mostramos como estão divididas as classes do projeto a nivel dos pacotes que as contém e a dependência entre eles (interação de pacotes).

Figura 21 - Diagrama de pacotes



Fonte: Os autores (2018)

5 SISTEMA

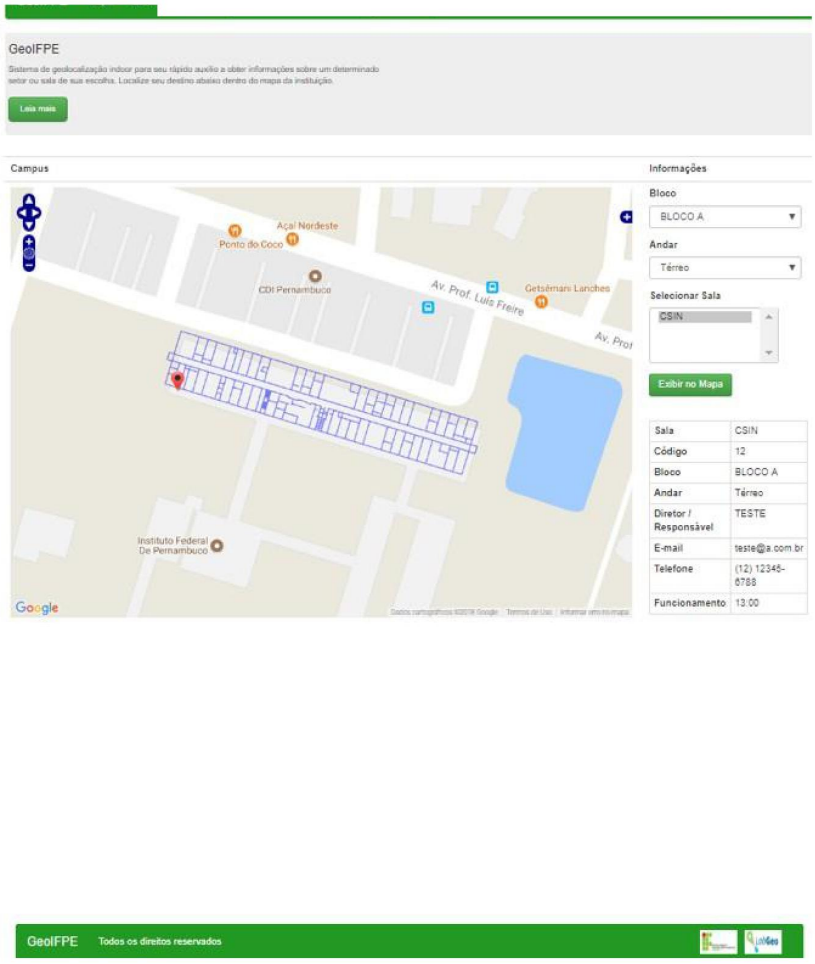
Este capítulo especifica os requisitos do projeto. Seu propósito é organizar, analisar e definir as estruturas do sistema.

5.1 ESTRUTURA DO SISTEMA

5.1.1 Mapa

A Figura 22 mostra a tela principal do sistema. Nesta tela o usuário pode visualizar o mapa da instituição. Onde é possível consultar a localização de cada sala e bloco do instituto e assim fazer a marcação do destino desejado.

Figura 22 - Tela principal do sistema

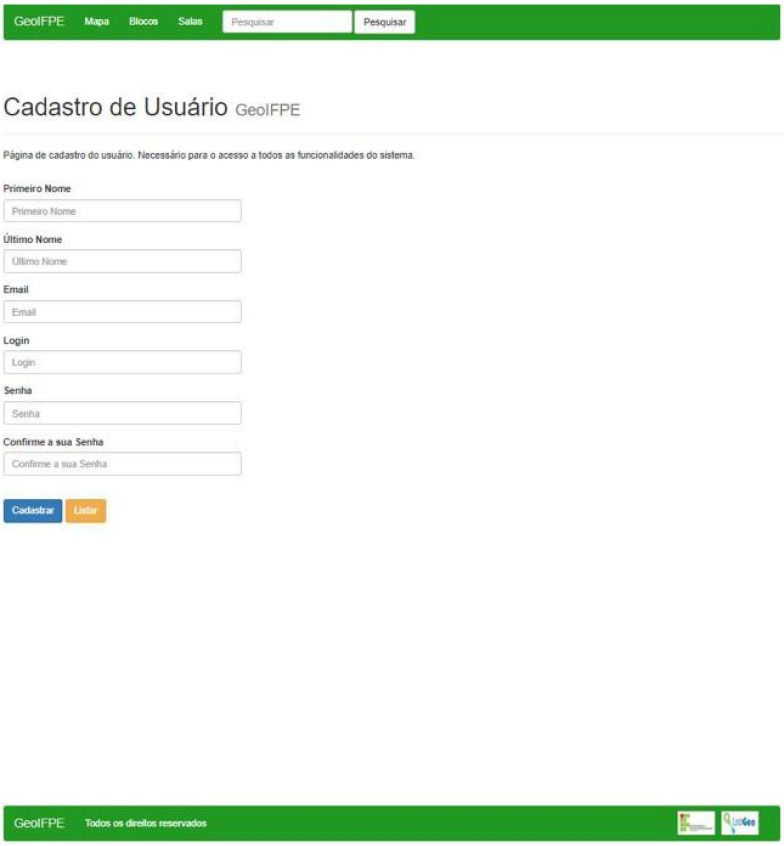


Fonte: Os autores (2018)

5.1.2 Usuário

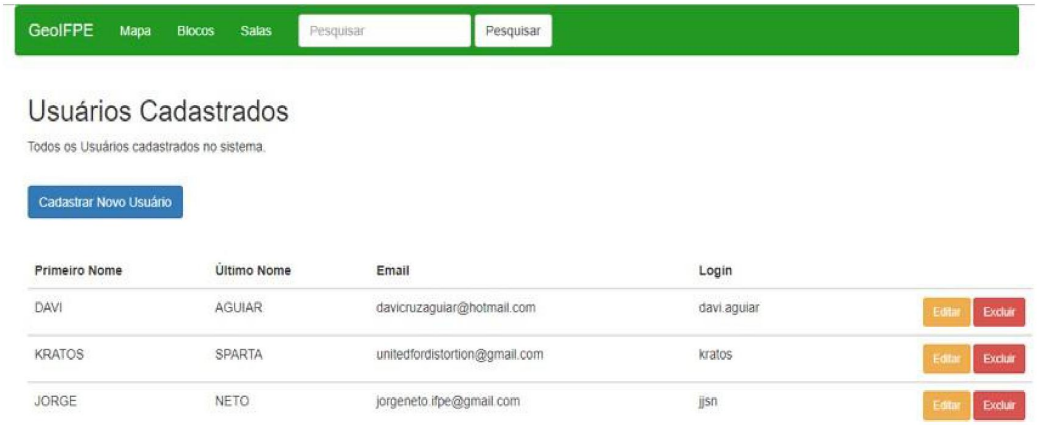
A Figura 23 mostra a tela de cadastro de usuários do sistema. Quando o usuário é cadastrado ele poderá fazer *login* na aplicação, ganhando assim, permissão de administrador do sistema, tendo acesso ao menu onde pode fazer o gerenciamento dos blocos e salas que compõem a instituição. Na Figura 24, podemos ver a tela de gerenciamento dos usuários já cadastrados no sistema.

Figura 23 - Tela de cadastro de usuário



Fonte: Os autores (2018)

Figura 24 - Tela de gerenciamento dos usuários cadastrados



Fonte: Os autores (2018)

5.1.3 Tela de login

A Figura 25 mostra a tela de *login* do sistema. Quando o usuário faz o *login*, ele ganha permissão de administrador do sistema, tendo acesso ao menu onde pode fazer o gerenciamento dos blocos e salas que compõem a instituição.

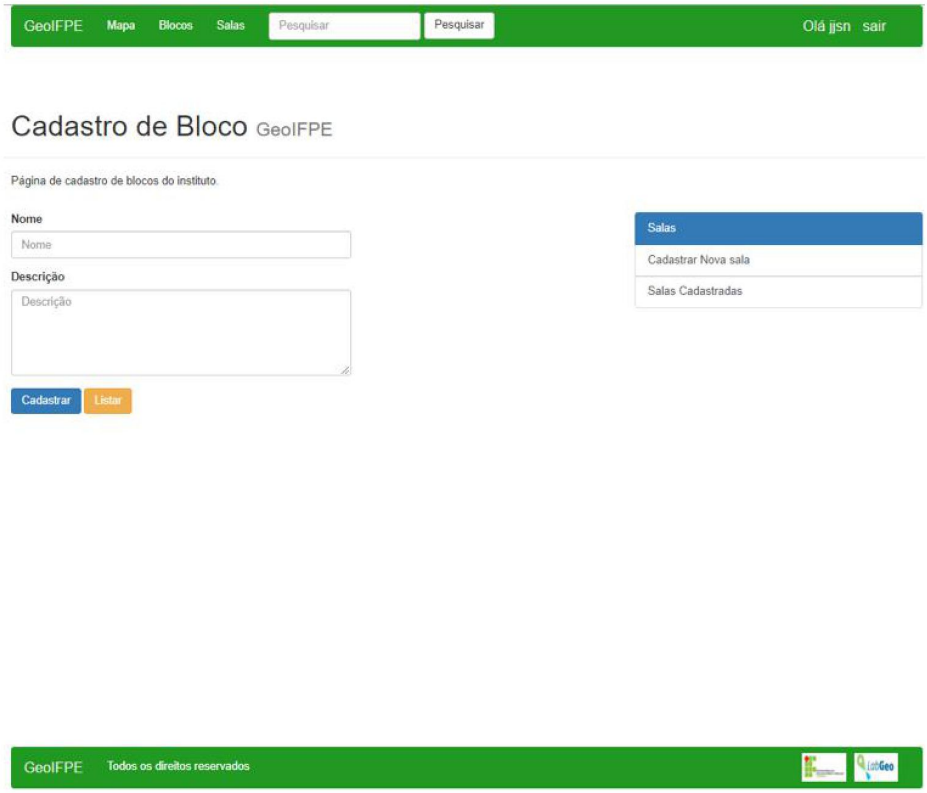
Figura 25 - Tela de login

Fonte: Os autores (2018)

5.1.4 Bloco

A Figura 26 mostra a tela de blocos da instituição. Quando o usuário faz o *login*, ele ganha permissão de administrador do sistema, tendo acesso ao menu onde pode fazer o gerenciamento dos blocos que compõem a instituição. Na Figura 27 podemos ver a tela dos blocos já cadastrados no sistema.

Figura 26 - Tela de cadastro de bloco



Fonte: Os autores (2018)

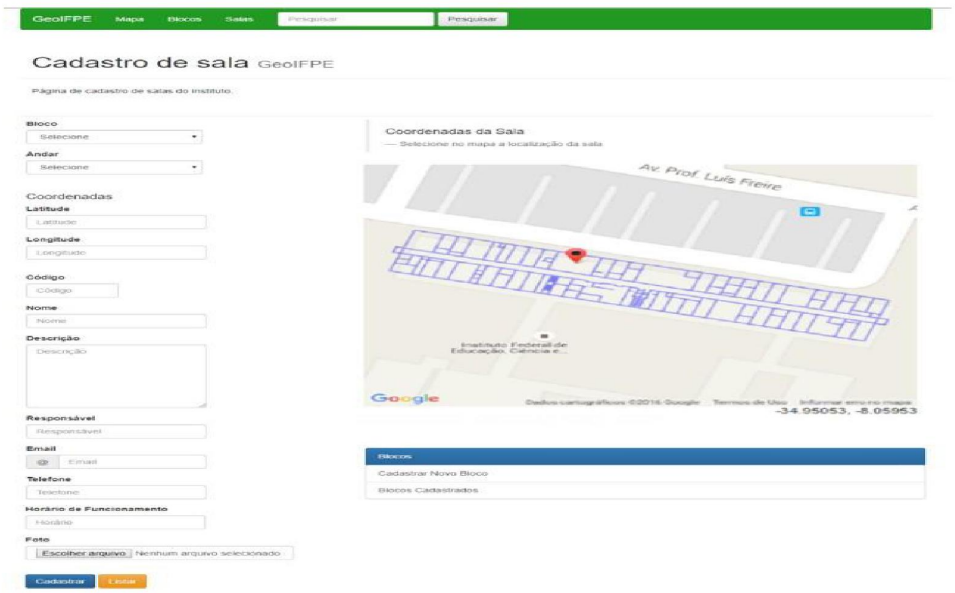
Figura 27 - Tela de cadastro de blocos

Fonte: Os autores (2018)

5.1.5 Sala

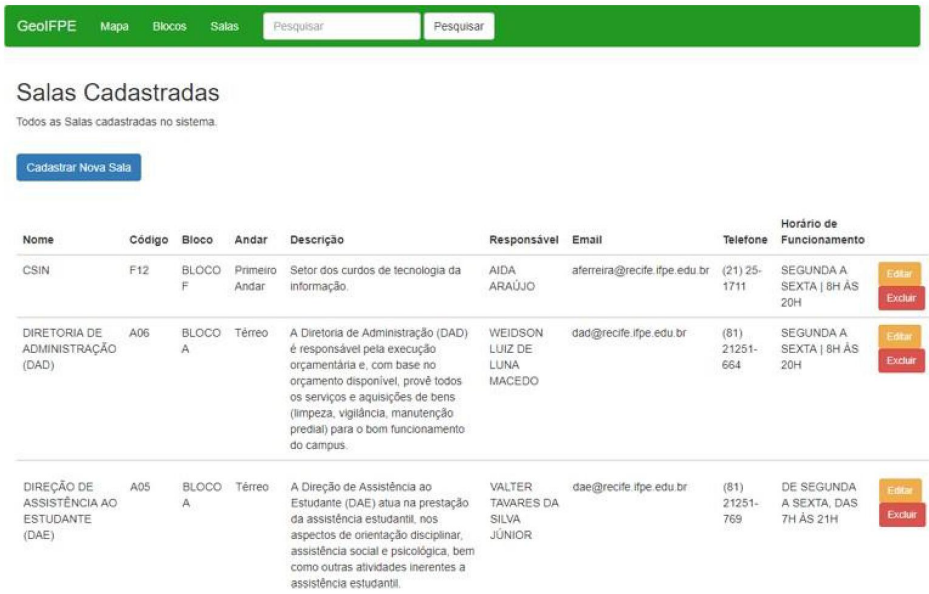
A figura 28 mostra a tela de cadastro de salas da instituição. Quando o usuário faz o login, ele ganha permissão de administrador do sistema, tendo acesso ao menu onde pode fazer o gerenciamento das salas que compõem a instituição. Na figura 29 podemos ver a tela de gerencimanto das salas já cadastradas no sistema.

Figura 28 - Tela de cadastro de sala



Fonte: Os autores (2018).

Figura 29 - Tela de gerenciamento das salas



Fonte:Os autores (2018)

6 CONCLUSÃO E TRABALHOS FUTUROS

Apresentamos aqui o estudo e desenvolvimento de uma aplicação SIG *web*, que demonstrou os benefícios que os alunos e colaboradores do Instituto Federal de Pernambuco teriam com o auxílio do sistema para localizar blocos e salas dentro da instituição. Aqui também podemos verificar que além deste trabalho apresentado, o projeto possui outros dois módulos que complementam este sistema. Que surgiram após um estudo de caso realizado no IFPE campus Recife, dividindo a equipe de pesquisa e desenvolvimento do projeto em três módulos: Módulo Servidor, Módulo *Android* e o Módulo *Web*, apresentado neste trabalho. Através deste trabalho, não só pudemos aplicar vários segmentos do conhecimento que adquirimos ao longo de nossa graduação, realizada no curso de análise e desenvolvimento de sistemas, como também expandimos esses conhecimentos. Aqui, neste trabalho, realizamos uma longa pesquisa a respeito de tecnologias em auxílio da geolocalização, como banco de dados geográfico, linguagens de programação e *frameworks* para pontos geográficos, entre outras tecnologias. Como pontos de melhorias para trabalhos futuros, deixamos a implementação do traçado em tempo real das rotas realizadas pelo usuário. E a alternância na perspectiva dos andares existentes dentro da instituição. Funcionalidades úteis que podem complementar o auxílio na geolocalização *indoor* desta aplicação.

REFERÊNCIAS

BARBOSA, Rodrigo Venâncio de Lima. **Utilização de Redes WIFI para Localização de Dispositivos Móveis**. 2018. Trabalho de Conclusão de Curso. Instituto Federal de Pernambuco. Departamento Acadêmico de Controle De Sistemas Eletro-Eletrônicos, Recife, PE, 2018.

BOOTSTRAP. **Framework de Template**. [S.l.], 2011. Disponível em: <http:/-/getbootstrap.com>. Acesso em: 23 jul. 2016.

CAELUM. **Spring Framework MVC.** 2018. Disponível em: <https://www.caelum.com.br/apostila-java-web/spring-mvc/exerccios-configurando-o-spring-mvc-e-testando- a-configurao>. Acesso em: Set. 2018.

ESRI. **Esri Shapefile techinical description.** 1998. Disponível em: <http://www.esricom/library/whitepapers/pdfs/shapefile.pdf>. Acesso em: Set. 2018.

ESRI. **ShapeFiles.** 2016. Disponível em: <https://doc.arcgis.com/pt-br/arcgis-online-/reference/shapefiles.htm>. Acesso em: 09 maio. 2016.

ILLEK, Petr. **How to Use Custom Markers for OpenLayers.** 2016. Disponível em: <https://www.morpht.com/blog/how-use-custom-markers-openlayers>. Acesso em: 09 maio. 2016.

INDOOR. *In:* OXFORD essential portuguese dictionary. Oxford, UK: Oxford University Press, 2018.

FALCÃO, Marcus Lucas Abreu de Araujo. **Implementação de Técnica knn e rest- ful webservice para auxílio de posicionamento indoor.** 2018. Trabalho de Conclusão de Curso. Instituto Federal de Pernambuco. Departamento Acadêmico de Controle De Sistemas Eletro-Eletrônicos, Recife, PE, 2018.

FERREIRA, Nilson Clementino. **Apostila de Sistemas de Informações Ge- ográficas.** 2016. Disponível em: <http://www.geolab.faed.udesc.br/sites disciplinas-/geoprocessamento_aplicado_ao_planejamento/docs/apostila sig5b15d.pdf>. Acesso em: 09 maio. 2016.

FITZ, Paulo Roberto. **Geoprocessamento sem complicação**. [S.l]: Oficina de textos, 2008.

GEOSERVER. **Geoserver.** 2016. Disponível em: <*http://geoserver.org*>. Acesso em: 09 maio. 2016.

LEITE, José Renê Santos. **Aplicação de Algoritmo de localização indoor baseado em sinais de radiofrequência**. 2018. Trabalho de Conclusão de Curso. Universidade Federal de Pernambuco. Centro de Informática, Recife, PE, 2018.

LUIZ, Andrey. **JavaScript 1** - Uma breve história da linguagem**.** 1998. Disponível em: <http://shipit.resultadosdigitais.com.br/blog/javascript-1-uma-breve-historia- da-linguagem/>. Acesso em: Set. 2018.

MACHRY, Márcio Fernando. **Banco de Dados Geográficos.** 2018. Disponível em: <http://www.inf.unioeste.br/~olguin/4463-semin/g3-apresentacao.pdf>. Acesso em: Set. 2018.

MEDEIROS, Anderson. **Como Desenvolver um GIS:** Parte 1**.** 2016a. Disponível em:< http://www.andersonmedeiros.com/como-desenvolver-um-gisl/>. Acesso em: 09 maio 2016.

MEDEIROS, Anderson. **O Geoprocessamento e Suas Tecnologias:** Parte 2. 2016b. Disponível em: <http://www.andersonmedeiros.com/geotecnologias-parte2/>. Acesso: 09 maio 2016.

OMAR, Omar. **Desenvolvimento de Aplicações Web utilizando o MVC Design Pattern**. 2004. Trabalho de Conclusão de Curso. Universidade Federal de Santa Catarina, Florianópolis, SC, 2004.

OPENLAYERS. **OpenLayers.** 2016. Disponível em: <http://openlayers.org/>. Acesso em: 09 maio. 2016.

POSTGRESQL. **PostgreSQL:** The World's Most Advanced Open Source Re- lational Database**.** 2004. Disponível em: <https://www.postgresql.org/>. Acesso em: Set. 2018.

SILVA, Rosângela. **Bancos de dados geográficos:** uma análise das arquiteturas dual (spring) e integrada (oracle spatial). 2002. Dissertação (Mestrado em Engenharia de Transportes). Universidade de São Paulo, São Paulo, SP, 2002.

WIKIPEDIA. **Spring Framework.** 2018. Disponível em: <https://pt.wikipedia.org-/wiki/Spring Framework/>. Acesso em: Set. 2018.