CENTRO UNIVERSITÁRIO UNIVATES
CENTRO DE CIÊNCIAS EXATAS E TECNOLÓGICAS
CURSO DE ENGENHARIA DA COMPUTAÇÃO

SISTEMA GERADOR DE ROTAS ATRAVÉS DE MAPAS DE CALOR

Rafael Leonardo Danieli

Lajeado, novembro de 2015

Rafael Leonardo Danieli

SISTEMA GERADOR DE ROTAS ATRAVÉS DE MAPAS DE CALOR

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Tecnológicas do Centro Universitário UNIVATES, como parte dos requisitos para a obtenção do título de bacharel em Engenharia da Computação.
Área de concentração: segurança pública, geoprocessamento, sistemas móveis.

ORIENTADOR: Prof. Alexandre Stürmer Wolf

Lajeado, novembro de 2015

Rafael Leonardo Danieli

SISTEMA GERADOR DE ROTAS ATRAVÉS DE MAPAS DE CALOR

Este trabalho foi julgado adequado para a obtenção do título de bacharel em Engenharia da Computação do CETEC e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Alexandre Stürmer Wolf, UNIVATES

Mestre pela PUC-Rio – Rio de Janeiro, Brasil

Banca Examinadora:

Prof. Alexandre Stürmer Wolf, UNIVATES

Mestre pela PUC-Rio – Rio de Janeiro, Brasil

Prof. Mouriac Halen Diemer, UNIVATES

Mestre pela UFRGS – Porto Alegre, Brasil

Prof. Paulo Roberto Mallmann, UNIVATES

Mestre pela UNISINOS – São Leopoldo, Brasil

Coordenador do Curso de Engenharia da Computação:_____

Prof. Marcelo de Gomensoro Malheiros

Lajeado, novembro de 2015.

AGRADECIMENTOS

Ao Professor Alexandre Stürmer Wolf, com quem compartilhei as primeiras ideias daquilo que veio a ser esse trabalho. Nossas conversas e trocas de conhecimento foram fundamentais, sem o seu conhecimento acadêmico nada disto seria possível.

À minha família pelo carinho e incentivo nos momentos difíceis desta trajetória. Aos meus pais e ao meu irmão por me apoiarem do início ao fim deste sonho, auxiliando de todas as formas possíveis para torna-lo realidade.

À minha noiva Clediane, quero agradecer pela paciência, pelo incentivo, pela força e principalmente pelo carinho e lhe dizer que valeu a pena todo o nosso esforço. Hoje, estamos colhendo juntos os frutos do nosso empenho e dedicação.

RESUMO

Ao longo dos anos muito é discutido sobre segurança pública e estratégias para a redução dos índices de criminalidade. A falta de ferramentas de tecnologia da informação que permitem a realização de análises detalhadas da criminalidade, para minimizar a situação de insegurança vivenciada pelos cidadãos, não é realidade no cotidiano e planejamento do trabalho policial. Este projeto visa a criação de uma aplicação móvel multiplataforma, voltada para os órgãos de segurança pública do Estado do Rio Grande do Sul, com objetivo de auxiliar no policiamento preventivo da cidade de Porto Alegre. O desenvolvimento deste Sistema de Informação Geográfica permitirá o direcionamento de viaturas para locais específicos dentro da área de abrangência de cada batalhão da polícia com base em dados históricos de criminalidade, os quais estão exibidos em mapa de calor no aplicativo. O sistema utiliza ferramentas de geoprocessamento auxiliadas pelo Google Maps e mecanismos de Banco de Dados geográficos para que as ocorrências policiais registradas em delegacias sejam mapeadas para fornecer informações de rotas *online* para o patrulhamento preventivo.

Palavras-chave: Geoprocessamento, Segurança Pública, Aplicativo Móvel, Mapa de Calor.

ABSTRACT

Over the years much is discussed about public safety and strategies to reduce crime rates. The lack of information technology tools that allow you to perform detailed analyzes of crime, to minimize the situation of insecurity experienced by the citizens, is not reality in daily life and planning of police work. This project aims to create a mobile application platform, aimed at the public security organs of the State of Rio Grande do Sul, in order to assist in the preventive policing of the city of Porto Alegre. The development of Geographic Information System will allow the targeting of vehicles to specific locations within the coverage area of each Police Battalion based on historical crime data, which are mapped by heat maps in the application. The system uses geoprocessing tools aided by Google Maps and geographic database mechanisms for law enforcement incidents recorded in police stations are mapped to provide online route information for preventive patrolling.

Keywords: Geoprocessing, Public Safety, Mobile App, Heatmap.

LISTA DE FIGURAS

Figura 1 - Taxas de Registro de Crimes Violentos contra ao Patrimônio 17
Figura 2 - Ilustração processo preventivo e reativo..... 18
Figura 3 - Estrutura geral de um SIG 23
Figura 4 - Três abstrações básicas: ponto, linha e região. 26
Figura 5 - Estrutura matricial e estrutura vetorial..... 27
Figura 6 - Arquitetura do PhoneGap 37
Figura 7 - Arquitetura MVC do AngularJS..... 39
Figura 8 - *Data binding* com AngularJS 39
Figura 9 - Interações com o usuário 43
Figura 10 – Diagrama ER do protótipo 44
Figura 11 - Arquitetura da aplicação proposta 48
Figura 12 – Instrução SQL que lista as ocorrências válidas..... 52
Figura 13 - Código fonte responsável por converter endereço em ponto geográfico..... 53
Figura 14 - Funcionamento da extração de dados..... 53
Figura 15 - Compilação para o sistema operacional Android 54
Figura 16 - Resultado da compilação 54
Figura 17 - Tela de *login* do aplicativo..... 55
Figura 18 - *View* da tela de *login*..... 56
Figura 19 - *Controller* da tela de *login*..... 56
Figura 20 - *Service* da tela de *login*..... 57
Figura 21 - Código que gera o *token* e atualiza no BD..... 58
Figura 22 - Exemplo de comunicação com *token* no cabeçalho da requisição 58
Figura 23 - *Token* gerado no BD 58
Figura 24 - Código para geração do mapa de calor e os gradientes utilizados..... 59
Figura 25 - Visualização do mapa de calor 60
Figura 26 - Tela com o filtro para o mapa de calor 61
Figura 27 - Ocorrências do bairro centro de Porto Alegre/RS 61
Figura 28 - Consulta aos registros do bairro centro..... 62
Figura 29 - Filtragem para crime de furto de veículos no centro de Porto Alegre..... 62
Figura 30 - Código que realiza requisições de rotas..... 63
Figura 31 - Solicitação de rota..... 64
Figura 32 - Consultas pra o cálculo de rotas 65
Figura 33 - Exemplo de geração de rota para usuário logado 66

LISTA DE CÓDIGOS

Listagem 1 - Inserção de um registro em uma tabela espacial 36
Listagem 2 - Seleção em uma tabela espacial 36

LISTA DE TABELAS

Tabela 1 - *Ranking* criminal por munícipio 19
Tabela 2 - Tabela de metadado do sistema de coordenadas 35
Tabela 3 - Tabela de metadado das tabelas com colunas espaciais 35
Tabela 4 - Tabela de usuários 44
Tabela 5 - Tabela de batalhões 45
Tabela 6 - Tabela de ocorrências 45
Tabela 7 - Tabela de bairros 46

LISTA DE ABREVIATURAS

AJAX	-	Asynchronous Javascript and XML
ANSI	-	American National Standard Institute
API	-	Application Programming Interface
ASF	-	Apache Software Foundation
BD	-	Banco de Dados
BDG	-	Banco de Dados Geográfico
BO	-	Boletim de Ocorrência
CDMA	-	Code Division Multiple Access
CSS	-	Cascading Style Sheets
EJB	-	Enterprise Java Beans
EPSG	-	European Petroleum Survey Group
GiST	-	Generalized Search Trees
GPS	-	Sistemas de Posicionamento Global
GUI	-	Graphical User Interface
HTML	-	Hypertext Markup Language
HTTP	-	Hypertext Transfer Protocol
JSON	-	JavaScript Object Notation
MVC	-	Model-View-Controller
NASA	-	National Aeronautics and Space Administration
OGC	-	Open Geospatial Consortium
PPS	-	Precise Positioning Service
REST	-	Representational State Transfer
RMPA	-	Região Metropolitana de Porto Alegre
RPC	-	Remote Procedure Call
SGBD	-	Sistemas Gerador de Banco de Dados
SIG	-	Sistema de Informações Geográfica
SOAP	-	Simple Object Access Protocol

SPS	-	Standard Positioning Service
SQL	-	Structured Query Language
SRID	-	Spatial Reference System Identifier
SSP-RS	-	Secretaria de Segurança Pública do Rio Grande do Sul
TOA	-	Time Of Arrival
URI	-	Uniform Resource Identifier
UTM	-	Projeção Universal de Mercator
WKB	-	Well-Known Binary
WKT	-	Well-Known Text
XML	-	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO 13

2 SEGURANÇA PÚBLICA 16
2.1 Classificação de políticas de segurança pública: preventiva e reativa 17
2.2 Criminalidade na Região Metropolitana de Porto Alegre/RS 19

3 REFERENCIAL TEÓRICO 21
3.1 Geoprocessamento 21
3.2 Sistema de Informação Geográfica - SIG 22
3.3 Geocodificação 25
3.4 Banco de Dados Geográfico ou Espacial 26
3.4.1 Operadores Espaciais 28
3.4.2 Índices GiST e R-Tree 30
3.4.3 SRID - Reference System Identifier 30
3.4.4 OGC e OSGeo 31

4 TECNOLOGIAS ENVOLVIDAS 32
4.1 PostgreSQL 32
4.2 PostGIS 34
4.3 PhoneGap 36
4.4 AngularJS 38
4.5 Ionic Framework 40
4.6 Web Services REST 40

5 PROPOSTA 42
5.1 Visão Geral 42
5.2 Modelo relacional 44
5.3 Aplicativo Mapoli 47

6 RESULTADOS 51
6.1 Conversão de dados 51
6.2 *Build* da aplicação móvel 54
6.3 Controle de Acesso 54
6.4 Mapa de calor 59
6.5 Geração de rotas 63

7 CONSIDERAÇÕES FINAIS 67

1 INTRODUÇÃO

A capacidade de fornecer informações corretas e precisas para a tomada de decisões é um dos itens mais importantes para o desenvolvimento do país e dos Estados, logo, estas informações podem ser utilizadas como um dos recursos estratégicos mais preciosos que os órgãos de segurança pública possuem e devem usar ao seu favor para o combate à criminalidade.

O mapeamento do crime pode desempenhar um papel importante no policiamento e redução de ocorrências policiais. Inicialmente, com a coleta de dados por meio do monitoramento e avaliação de perguntas específicas que agem como um mecanismo crucial, fornecendo uma prevenção ao crime e ajudando a criar iniciativas bem sucedidas no combate ao problema criminal (CHAINEY, RATCLIFFE, 2005).

Cada vez mais órgãos públicos de segurança têm à sua disposição uma quantidade significativa de tecnologias capazes de auxiliar no cotidiano policial. O geoprocessamento, que faz parte dessas tecnologias, vem sendo utilizado como importante mecanismo de otimização para o alcance das ações dos governos federais, estaduais e municipais por se tratar de um conjunto de técnicas que visa responder questionamentos sobre “onde?” determinado fato acontece em uma localização espacial.

Os Sistemas de Informação Geográfica (SIG) existem para fornecer ferramentas de apoio à segurança pública visando a segurança do cidadão, a redução de índices de criminalidade, o combate e prevenção do tráfico de drogas e o consumo de entorpecentes, propiciando uma relação de confiança e cooperação entre polícia e cidadão. A utilização intensiva de SIGs tem promovido uma verdadeira revolução nas polícias do mundo todo (REULAND, 1997).

O acréscimo do número de aplicações *web* que utilizam tecnologias derivadas de SIGs disseminou-se com o passar dos anos. Entre as tecnologias, ferramentas como Google

Maps, somaram em 2008 cerca de 50.000 *websites* com recursos geográficos. Aplicações SIG podem apresentar o mapa do mundo, de uma maneira simples ao usuário final, porém, com uma complexa análise espacial de distribuições e processos envolvendo alguma área de negócio em específico. Essas aplicações utilizam muitas tecnologias e plataformas de base de dados, as quais, podem impactar a performance das aplicações web (ADNAN *et al*., 2010).

Na realidade atual de Porto Alegre, os órgãos competentes que mantém os índices de criminalidades regulares não utilizam boa parte das informações de registros de ocorrências para agir preventivamente. Visando essa lacuna, imprópria para os dias atuais, o presente projeto tem como objetivo a criação de um sistema de informação geográfica para a segurança pública, atuando sobre os registros de boletins de ocorrências (BO), cadastrados no Banco de Dados (BD) da Polícia Civil do Estado. Através destes dados, criar mapas de calor que auxiliem no melhor planejamento de rotas de rondas policiais, obtendo conhecimento sobre onde localizam-se as maiores incidências de violência na cidade.

Com base nos dados da violência que ocorrem na cidade, a ferramenta irá auxiliar no planejamento de rondas preventivas, onde poderá ser definido a distância desejada para a viatura percorrer na abrangência de seu batalhão, criando rotas dinâmicas (baseadas na localização atual do usuário) e disponibilizando informações *online* para as viaturas realizarem a ronda nesses locais, assim, pode-se garantir que o veículo passará por todas as ocorrências onde foi gerada a rota.

Este projeto visa abastecer a iniciativa de policiamento preventivo na cidade de Porto Alegre, como um projeto piloto, e mais adiante ser implementado para todas as cidades do Estado do Rio Grande do Sul, assim, oferecendo a população um policiamento mais inteligente, agindo na prevenção ao invés de agir na causa. Além disso, estudar tecnologias para manipular dados geográficos e como elas podem auxiliar no objetivo do projeto e também, intensificar a inciativa do desenvolvimento de aplicações hibridas e fomentar a busca por soluções de entrega rápida e que possuam desempenho igual ou superior a tecnologias nativas de cada plataforma.

Para elucidar este trabalho, o mesmo está organizado em capítulos. No Capítulo 2 será apresentada uma visão geral sobre segurança pública no Brasil e no Rio Grande do Sul, com um cenário voltado para a criminalidade e as diferenças de políticas preventivas e reativas, além disso, exibir informações estatísticas da criminalidade nas cidades do Estado

do Rio grande do Sul o que mostram a justificativa para a escolha da cidade de Porto Alegre como implantação piloto.

No Capítulo 3 será apresentado um referencial teórico sobre as tecnologias e conceitos envolvidos para o desenvolvimento do sistema proposto, contendo as técnicas fundamentais de Geoprocessamento, Sistema de informação geográfica e Banco de Dados Espaciais.

No Capítulo 4, será apresentado as ferramentas envolvidas para a criação do protótipo, incluindo as tecnologias encarregadas pela criação do *front-end* da aplicação. No Capítulo 5, o sistema proposto é descrito, detalhando todas as tecnologias utilizadas no *back-end*, sendo feita uma ilustração da comunicação entre o *front-end* e o *back-end.*

No Capítulo 6 está composto com os resultados que puderam ser obtidos após o desenvolvimento do sistema móvel visando manter uma relação com as limitações de escopo determinadas da proposta de desenvolvimento.

E por último, o Capítulo 7, são feitas as considerações finais que puderam ser obtidas com a realização do presente trabalho.

2 SEGURANÇA PÚBLICA

Segundo o artigo 144 da Constituição da República Federativa do Brasil (1988):

> A segurança pública, dever do Estado, direito e responsabilidade de todos, é exercida para a preservação da ordem pública e da incolumidade das pessoas e do patrimônio [...].

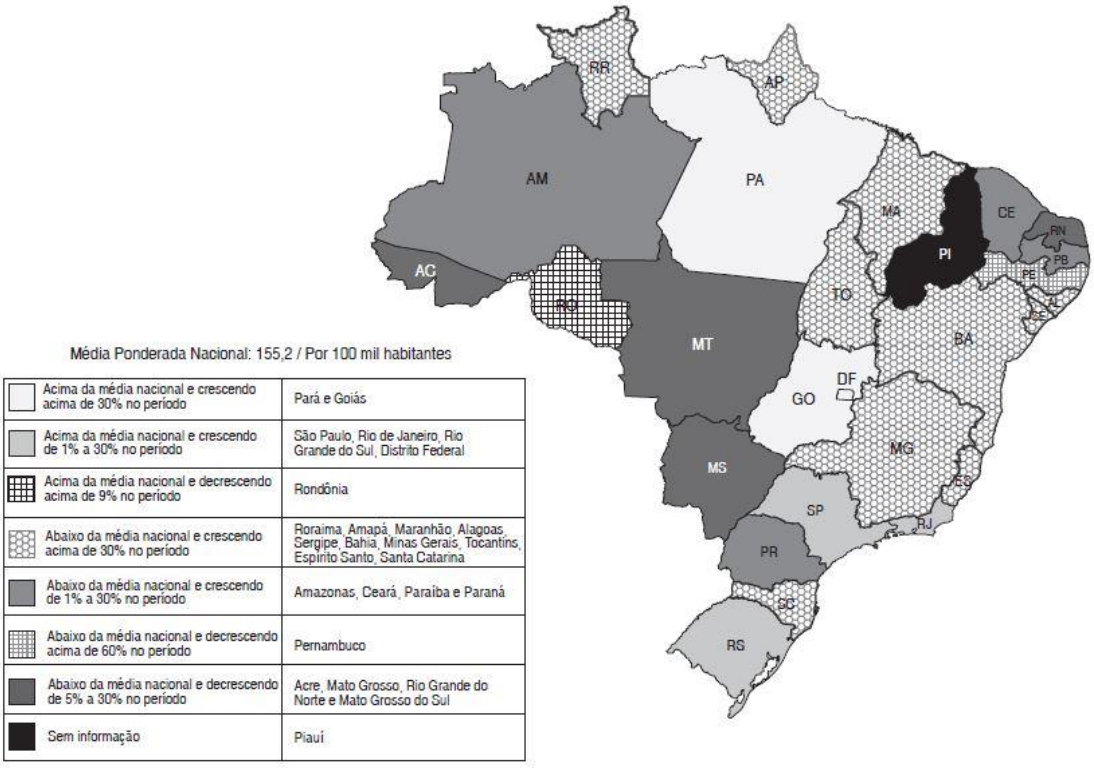
A violência faz parte do cotidiano brasileiro, onde vivenciamos cenas de frustração no tocante à fragilidade da segurança pública no país. Demasiadas são as notícias e imagens que chegam através do noticiário de TV, Internet e outros meios de comunicação sobre a violência e insegurança sofridas pelos brasileiros.

No Brasil, o suporte da polícia teve início antes da independência formal, com a transferência da família real portuguesa para as terras brasileiras levando a criação da "Intendência Geral da Polícia da Corte e do Estado do Brasil, em 10 de maio de 1808" (HOLLOWAY, 1997).

Conforme informações da Secretaria de Segurança Pública do Rio Grande do Sul (SSP-RS), após longos anos e várias descentralizações, a divisão de Justiça e Polícia foi permitida pelo Decreto n.º 4824 de 22 de novembro de 1871, sancionado pela lei n.º 2033 de 20 de setembro de 1871, tais quais, estão em vigor até os dias atuais sem nenhuma alteração no seu decreto original.

A análise da violência e criminalidade no Brasil intensificou-se a partir dos anos 70, debates sobre metodologias de análise espacial e temporal da criminalidade ganharam força dando sentido aos dados que estão relacionados com dimensões de espaço e tempo. Com esta evolução, propostas de metodologias que descrevam as reais diferenças regionais do crime e violência foram desenvolvidas. Na Figura 1, os dados no Brasil de ocorrências policiais registradas entre o primeiro semestre de 2001 e o primeiro semestre de 2003 mostra a situação vivenciada de cada Estado em relação a crimes contra o patrimônio daquela época (PEIXOTO *et al*., 2004).

Figura 1 - Taxas de Registro de Crimes Violentos contra ao Patrimônio



Média Ponderada Nacional: 155,2 / Por 100 mil habitantes

Acima da média nacional e crescendo acima de 30% no período	Pará e Goiás
Acima da média nacional e crescendo de 1% a 30% no período	São Paulo, Rio de Janeiro, Rio Grande do Sul, Distrito Federal
Acima da média nacional e decrescendo acima de 9% no período	Rondônia
Abaixo da média nacional e crescendo acima de 30% no período	Roraima, Amapá, Maranhão, Alagoas, Sergipe, Bahia, Minas Gerais, Tocantins, Espírito Santo, Santa Catarina
Abaixo da média nacional e crescendo de 1% a 30% no período	Amazonas, Ceará, Paraíba e Paraná
Abaixo da média nacional e decrescendo acima de 60% no período	Pernambuco
Abaixo da média nacional e decrescendo de 5% a 30% no período	Acre, Mato Grosso, Rio Grande do Norte e Mato Grosso do Sul
Sem informação	Piauí

Fonte: Peixoto *et al.* (2004).

Na Seção 2.1, será apresentada a classificação das políticas utilizadas pelos órgãos de segurança que visam reduzir a criminalidade potencial.

2.1 Classificação de políticas de segurança pública: preventiva e reativa

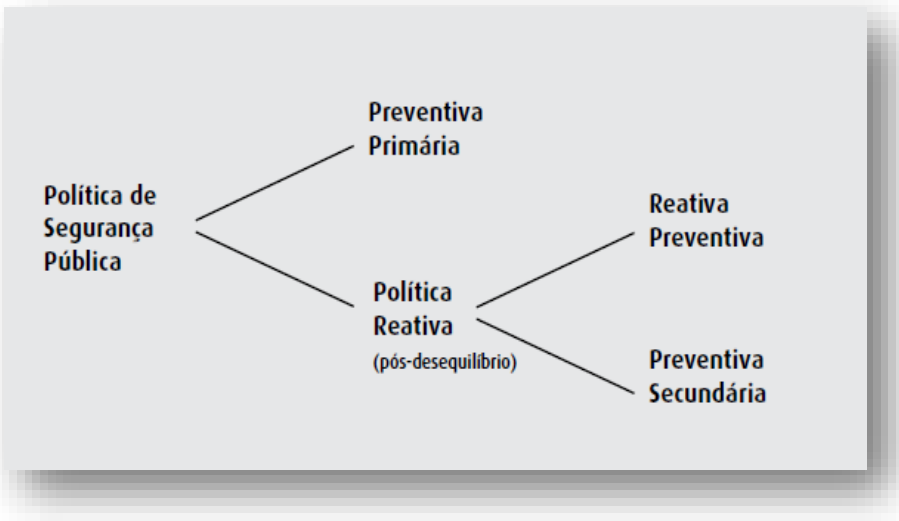
As políticas de segurança pública podem ser entendidas de duas maneiras: políticas preventivas e reativas, porém, não podem ser confundidas com prevenção e repressão do crime. O objetivo da política de segurança pública é manter a ordem pública, a criminalidade compatível com a estabilidade social (FILOCRE, 2009).

> Denomina-se política de segurança pública preventiva primária aquela elaborada com o objetivo de manter a criminalidade em nível condizente com a estabilidade social. Caso ocorra o desequilíbrio, faz-se então presente a política de segurança pública reativa, que,

> por sua vez, se subdivide em: repressiva, que visa retornar a criminalidade ao patamar desejado; e preventiva secundária, que evita que os índices de criminalidade novamente ultrapassem o nível de estabilidade (FILOCRE, 2009).

A Figura 2 ilustra como estão organizados os tipos de políticas de segurança pública pela ordem da sequência de ações.

Figura 2 - Ilustração processo preventivo e reativo



Fonte: Filocre (2009).

Políticas de prevenção e controle do crime são uma preocupação sobre os aspectos da criminologia, pois atuam como uma solução imediata nos problemas de violência nas grandes cidades (AZEVEDO, 2003). Para o autor, o resultado de um policiamento preventivo é o desencorajamento ou desestimulo do criminoso na prática do crime, reduzindo as oportunidades. Esta redução, acaba forçando uma mudança de comportamento na comunidade criminosa.

> Certo é que a política de segurança pública reativa não é necessariamente repressiva, enquanto a preventiva secundária resulta de situações de desequilíbrio anterior. Na secundária, são previstos mecanismos de atuação especial sobre as causas do desequilíbrio preexistente, de forma tal que a criminalidade fica contida no desejado. A preventiva secundária decorre do fato que o desequilíbrio social provoca desequilíbrio na criminalidade tanto quanto a criminalidade provoca desequilíbrio social (FILOCRE, 2009).

Azevedo (2003) mantém uma ideia de que a polícia controla o crime, policiais agem para identificar e prender culpados, desmantelar comportamentos indevidos e impor a ordem de acordo com a lei. Assim, a perspectiva é fundamentalmente punitiva e as ações sobre os

indivíduos são reativas, ou seja, policiamento reativo, que espera a ocorrência do crime para entrar em ação.

Compreendendo as definições de políticas preventivas e reativas, entende-se que existe uma forma adequada para cada momento. Prevenir para evitar crimes e diminuir as oportunidades de indivíduos criminosos e reagir após um desequilíbrio de determinado acontecimento.

A Seção 2.2 visa expor a realidade criminológica da região metropolitana de Porto Alegre/RS.

2.2 Criminalidade na Região Metropolitana de Porto Alegre/RS

A Região Metropolitana de Porto Alegre (RMPA) foi criada em 1968 e implantada pela Lei Complementar nº 14, de 8 de junho de 1973 (BRASIL, 1973), decreto que também criou as regiões de São Paulo, Belo Horizonte, Recife, Salvador, Curitiba, Belém e Fortaleza. A RMPA conta com 32 municípios que se fortaleceram a partir de 1989 com a inclusão do Complexo Petroquímico de Triunfo (METROPLAN, 2015).

Dados de 2014, obtidos junto à SSP-RS, correspondem aos delitos consumados e registrados através de Boletins de Ocorrências nas delegacias de Polícia de cada munícipio.

A Tabela 1 exibe o *ranking* dos valores das 12 primeiras cidades onde aconteceram maiores delitos. Nos valores listados estão crimes contra o patrimônio e crimes contra a pessoa.

Tabela 1 - *Ranking* criminal por munícipio

Municípios	Homicídio Doloso	Furtos	Furto de Veículo	Roubos	Roubo de Veículo	Estelionato
Porto Alegre	571	36.781	4.062	24.057	6.903	4.654
Canoas	107	6.006	753	3.199	922	449
Pelotas	65	4.566	800	2.794	205	435
Caxias do Sul	92	6.054	1.635	2.764	667	606

Alvorada	157	2.209	338	2.256	337	296
Novo Hamburgo	87	3.836	1.112	2.131	793	486
Viamão	93	2.701	500	1.937	378	251
Gravataí	93	3.255	612	1.837	583	500
São Leopoldo	113	3.154	680	1.768	466	357
Rio Grande	55	3.410	331	1.612	35	298
Sapucaia do Sul	48	1.998	321	1.394	191	177
Santa Maria	42	4.563	212	1.372	34	522
Cachoeirinha	35	1.900	290	1.140	312	225

Fonte: SSP-RS (2014).

O motivo para a escolha da cidade de Porto Alegre é que ela ocupa a primeira posição no *ranking* com o maior número de ocorrências registradas. Logo, estes registros podem ser utilizados para mapear, através de mapas de calor, localizações com maiores índices de violência.

Para o combate para estes números elevados de criminalidade uma possível solução é agir na causa dos problemas, investir na criação de técnicas preventivas, pois a maioria das instituições do país mantém uma visão ultrapassada, que tem como objetivo apenas capturar criminosos sem a devida capacidade de prevenir (JUCÁ, 2002).

No Capítulo 3, será abordado o referencial teórico sobre tecnologias de geoprocessamento que auxiliam no combate ao crime utilizando métodos preventivos.

3 REFERENCIAL TEÓRICO

Neste capítulo será abordado de forma mais detalhada os conceitos sobre Geoprocessamento e suas características, bem como o conceito de Sistemas de Informação Geográfica e o Banco de Dados Relacional Espacial.

3.1 Geoprocessamento

A necessidade de manter informações é essencial para qualquer operação policial. Portanto, utilizar uma tecnologia que melhor atenda as demandas da polícia é indispensável e mostra que além de viaturas e armamentos modernos, o efetivo também precisa de recursos tecnológicos que caminhem paralelos com a evolução social, cabendo as geotecnologias assumirem um papel na luta do combate à criminalidade (FREITAS, VIEIRA, 2007).

Segundo Máximo (2004), o geoprocessamento constitui-se em um método informatizado no qual são inseridos dados relacionados à cartografia digital[1]. Desta forma, é possível obter uma análise acurada dos acontecimentos criminais ao se definir estratégias preventivas para ações policiais. Contudo, estas ações acabam necessitando de um tratamento cauteloso, pois, a aplicação de ferramentas de apoio analítico dependem da informação acurada dos dados e de uma coleta eficiente.

Furtado (2002) define geoprocessamento como “Um conjunto de técnicas de coleta, tratamento, manipulação e apresentação de informações espaciais”. As técnicas mais relevantes são:

---

[1] Representação de dados para os processos que ocorrem no espaço geográfico (MÁXIMO, 2004).

- Cartografia automatizada;
- Processamento de imagens de satélite;
- Digitalização de mapas;
- SIG.

Para o autor, o geoprocessamento é considerado a geoinformação que indica uma ou mais informações em um ponto geográfico especifico, referenciado através de coordenadas (latitude, longitude e altitude), em algum ponto da Terra. Nesse sentido, o tratamento desta geoinformação permite manipular e planejar diversas áreas do conhecimento, dentre elas, o controle do meio ambiente e o planejamento de cidades.

Da Silva (2009) salienta que o conceito sobre geoprocessamento evoluiu com o passar dos anos e com a utilização dos seus métodos e técnicas. Para o autor, quem possui o benefício pelo uso do geoprocessamento são os ambientes técnico-científicos do planejamento e gestão de lugares.

Da Silva (2009) discorda que Geoprocessamento seja definido como “um conjunto de geotecnologias incluindo Sensoriamento Remoto, Cartografia, Sistemas de Posicionamento Global (GPS), entre outros ramos”, para ele, o correto é afirmar que Geoprocessamento é “um conjunto de técnicas computacionais que opera sobre bases de dados georreferenciadas visando a transformação em informações”. No geoprocessamento, são atribuídas análises de sínteses da gestão territorial e planejamento ambiental, devido à grande quantidade de dados que é tratada. Para realizar esta análise, se faz necessário a utilização de técnicas computacionais eficientes.

3.2 Sistema de Informação Geográfica - SIG

Um Sistema de Informação Geográfica, ou simplesmente SIG, é um sistema de computador criado para mapear dados espaciais. A palavra Geográfica significa que as localizações dos dados são conhecidas, ou podem ser calculadas e transformadas em coordenadas geográficas (latitude, longitude). A palavra Informação define que os dados são organizados para produzir conhecimento, frequentemente vistos em mapas de calor e

imagens, mas também com gráficos estatísticos e tabelas. A palavra Sistema informa que um SIG é criado através de severas interações ligadas por componentes com diferentes funções.

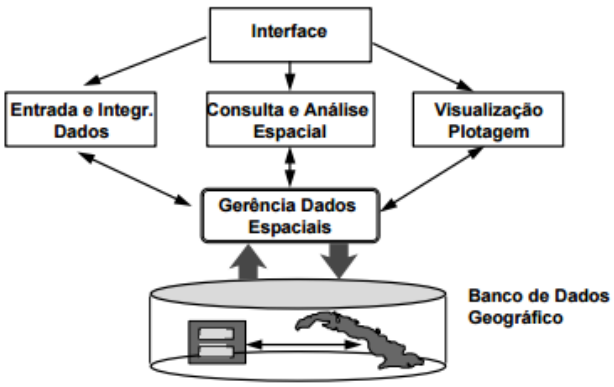
Os SIGs consistem de um pacote de programas de computador com uma interface para o usuário que provem acesso às funções particulares. O usuário interage com o SIG através de uma interface comumente chamada na computação de Graphical User Interface (GUI), ou por linha de comando que consiste em trechos de programas ditando a sequência e os tipos de operações desejadas (BONHAM-CARTER, 1994).

De acordo com o Instituto de Pesquisa de Sistemas Ambientais (ESRI, 1990): "Um SIG é um conjunto organizado de hardware, software, dados geográficos e pessoal destinados a eficientemente obter, armazenar, atualizar, manipular, analisar e exibir todas as formas de informação geograficamente referenciadas".

> O propósito de um SIG tradicional é primeiramente e acima de tudo a análise espacial. Portanto, a captura dos dados e produção cartográfica pode ser limitada. Capacidade de análises tipicamente apoiam a tomada de decisão para projetos específicos e/ou áreas geográficas limitadas. As características da base de dados cartográficos (exatidão, continuidade, completitude, etc.) são tipicamente apropriados para produção de mapas em pequena escala. (HUXHOLD, 1991).

Os principais componentes de um SIG, que podem ser vistos através da Figura 3, são organizados de forma hierárquica, no nível mais próximo ao usuário pela interface até o mais baixo nível pelo banco de dados geográfico. Pode-se observar que o nível intermediário fica encarregado do processamento de dados espaciais (DAVIS, NETTO, 2001).

Figura 3 - Estrutura geral de um SIG



Fonte: Davis e Neto (2001).

Ainda pode ser adicionado como um componente essencial para o funcionamento de um SIG o hardware a ser utilizado.

Segundo Chainey e Ratclifee (2005), o hardware pode ser um computador com capacidade suficiente de processamento e memória para atender a demanda do SIG proposto. Para sistemas mais robustos são consideradas estações de trabalho com sistemas operacionais de alto desempenho e alta escalabilidade (*clusters*[2]). Estes computadores devem estar conectados a uma rede local ou servidor. Existem soluções de SIGs prontas e disponíveis no mercado, dentre eles, ArcGIS, ArcView, MapInfo, GeoMedia e Northgate blue8.

Uma importante parte de toda aplicação *web,* baseada em tecnologia SIG é o seu mapeamento ou tecnologia de visualização, a qual, torna possível expor dados em forma de mapas. Esta visualização vem se tornando popular em *websites* que apresentam informações georreferenciadas, o que torna a aplicação mais amigável. Todavia, antes de realizar o trabalho de visualização é necessário alcançar uma análise eficaz das informações (ADNAN *et al*., 2010).

Para Moura (2014), SIG é "um instrumento de elaboração eletrônica que permite coleta, gestão, análise e representação automatizada de dados georreferenciados". Logo, não existe uma definição padronizada e universalmente aceita. Ao referenciar um SIG entende-se por informações espacialmente localizadas e que permitem o controle e gestão do território exibido.

Esta falta de definição dá-se a dois fatores: primeiro, as potencialidades da informática ainda não estão completamente exploradas e previstas, e segundo, a tendência de que os conceitos de geografia sejam associados ao quadro teórico, enquanto, o instrumento operacional para os estudos espaciais seja associado à cartografia.

O requisito para manter a geometria dos objetos geográficos e atributos representa uma característica básica para SIGs, pois, para cada atributo é necessário armazenar suas inúmeras representações gráficas associadas. Desta forma, são três os segmentos de maior importância para utilização de SIGs:

---

[2] Definição de *cluster*: sistema que relaciona dois ou mais computadores ligados através de uma rede e trabalhando de maneira conjunta no intuito de processar uma tarefa, processando-as mais rápido que um computador convencional e podendo ser facilmente expandido (ADNAN *et al*., 2010).

- Ferramenta de produção de mapas;
- Suporte na análise espacial de fenômenos;
- Banco de dados geográfico com funções de armazenamento e recuperação de informações espacial.

Dentro destas três aplicações de SIGs reflete-se a importância do tratamento da informação geográfica (DAVIS, NETTO, 2001).

3.3 Geocodificação

Geocodificação é o processo de conversão de uma determinada descrição textual em um dado geoespacial válido (latitude, longitude). Logo, este endereço geoespacial pode ser georreferenciado em um mapa terrestre, caracterizando-o em uma informação espacial explicita ou informação geográfica.

Caso a informação seja apenas geocoficada e não for georreferenciada será conhecida como informação não-geográfica ou informação espacial implícita (HILL, 2006).

> Geocodificação é o processo de encontrar coordenadas geográficas associadas (normalmente expressa como latitude e longitude) por outros dados geográficos, tais como endereços residenciais ou códigos postais (CEP). Com as coordenadas geográficas, os elementos podem ser mapeados e incorporados a Sistemas de Informação Geográfica, ou as coordenadas podem ser incorporadas a mídias como fotografias digitais (SKABA, 2009).

Geocodificação refere-se à atribuição de coordenadas geográficas a uma entrada textual e sua finalidade é identificar determinada localização sobre a superfície da terra, este trabalho pode ser feito por aplicações Desktop, bibliotecas de programação ou web services disponíveis na Internet, como Google Geocoding Application Programming Interface[3] (API), Yahoo! Geocoding API, Bing Geocoding API.

[3] Portal Geocoding API Google: https://developers.google.com/maps/documentation/geocoding/?hl=pt-br

Desta forma, para que a geocodificação funcione corretamente, é necessário que a entrada do dado a ser geocodificado não possua erros de grafia, abreviações inconsistentes, CEP incorreto, endereço incompleto ou inexistente e falta do número do logradouro. Caso estas regras não sejam obedecidas o processo de geocodificação pode não funcionar ou retornar resultados diferentes dos quais está sendo solicitado (GRUBESIC, MURRAY, 2004).

3.4 Banco de Dados Geográfico ou Espacial

Um banco de dados espacial ou geográfico (BDG) permite a utilização de tipos de dados espaciais e o armazenamento em tabelas de banco de dados relacional.

BDG disponibilizam funções espaciais e índices para consulta e manipulação de dados utilizando Structured Query Language (SQL). Por conta disso, um BDG possui particularidades que permitem encontrar dados por um determinado raio, distância entre pontos e caminho mais curto, além de disponibilizar uma série de funções que podem auxiliar na resolução de problemas de caráter geográfico (OBE, HSU, 2011).

Segundo Güting e Schneide (2005), as principais aplicações que utilizam BDG são as aplicações SIG. Desta forma, para que o BDG possa entregar a uma aplicação SIG informações geográficas, são aplicadas abstrações em objetos individuais. A Figura 4 exibe as três abstrações básicas utilizadas.

Figura 4 - Três abstrações básicas: ponto, linha e região.

Fonte: Güting, Schneide (2005).

Estas abstrações são definidas como:

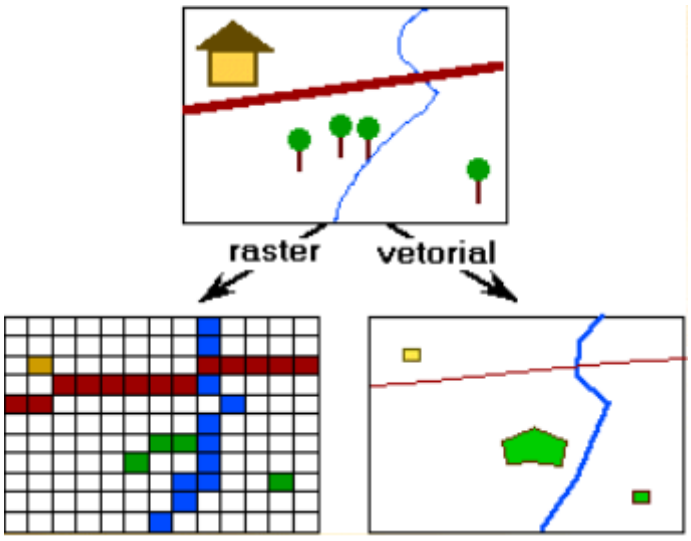
- Ponto: Em seu aspecto geométrico, representa um objeto no qual apenas um local está definido no espaço, por exemplo, uma cidade pode ser representada como um ponto em um modelo que descreve sua grande área geográfica;

- Linha: Neste contexto sempre pode ser entendido como uma curva no espaço, usualmente, representado por uma polilinha. É a abstração básica para facilitar a movimentação ou conexões no espaço (ruas, rios, cabos de telefone, eletricidade, etc.);

- Região: É a abstração para algo no espaço tendo uma extensão em duas dimensões, por exemplo, um parque ou uma cidade. Uma região pode conter um segmento não preenchido, mas pode ser constituído de várias regiões distintas.

Uma das características que um BDG possui é o seu mecanismo de armazenamento de informações geográficas. Pode ser estruturado de diversas formas, mas, as duas abordagens amplamente utilizadas são conhecidas como: estrutura matricial (*raster*) e estrutura vetorial. A Figura 5 ilustra a diferença entre esses dois tipos de estruturas.

Figura 5 - Estrutura matricial e estrutura vetorial



Fonte: Filho, 2000.

Segundo Filho (2000), na estrutura matricial, pode-se considerar uma grade de células em formato retangular onde a posição da célula é determinada pela coluna e pela linha que está localizada na grade. Portanto, cada célula armazena um valor que corresponde ao tipo de entidade que é encontrada naquela posição e as camadas dessas posições ficam totalmente preenchidas. Assim, cada célula irá corresponder a uma porção do espaço sendo representado.

Na estrutura vetorial, cada fenômeno geográfico é representado no banco de dados por um objeto, possuindo identificação própria e representação espacial do tipo ponto, linha,

polígono ou um objeto complexo. Desta forma, a posição de cada objeto é definida por sua localização no espaço, de acordo com um sistema de coordenadas a ser utilizado.

Ao longo do tempo, muitos Sistemas Geradores de Banco de Dados (SGBD) tiveram que se adequar a necessidade de contemplar informações espaciais em suas bases. Em virtude disso, o PostgreSQL, maior banco de dados de código aberto do mundo, criou a extensão PostGIS para dar suporte a esta necessidade.

3.4.1 Operadores Espaciais

Outro grande destaque dos BDG é o grande número de operadores espaciais disponíveis, os quais, podem ser separados em sete categorias (CASANOVA *et al*., 2005):

- Operadores topológicos:
    - *Equals* (*geometry*, *geometry*): retorna 1 se as geometrias passadas como parâmetros são iguais espacialmente;
    - *Disjoint* (*geometry*, *geometry*): retorna 1 se as duas geometrias são disjuntas entre si, ou seja, se não possuírem nenhum ponto em comum;
    - *Intersects* (*geometry*, *geometry*): retorna 1 se uma geometria intercepta espacialmente a outra;
    - *Touches* (*geometry*, *geometry*): retorna 1 se uma geometria toca espacialmente a outra;
    - *Crosses* (*geometry*, *geometry*): retorna 1 se uma geometria cruza a outra;
    - *Within* (*geometry*, *geometry*): retorna 1 se a primeira geometria está completamente dentro da segunda;
    - *Overlaps* (*geometry*, *geometry*): retorna 1 se uma geometria sobrepõe espacialmente a outra;

- *Contains* (*geometry*, *geometry*): retorna 1 se a primeira geometria contiver completamente a segunda.

- Operador de construção de mapas de distância:
  - *Buffer* (*geometry*, *double*, [*integer*]): retorna uma geometria que representa todos os pontos cujo as distâncias da geometria sejam menores ou iguais a distância informada.

- Operador para construção do fecho convexo:
  - *Convexhull* (*geometry*): retorna uma geometria convexa a partir do objeto geográfico passado como parâmetro.

- Operadores de conjunto:
  - *Intersection* (*geometry*, *geometry*): retorna uma geometria que representa a interseção das geometrias passadas como parâmetro;
  - *GeomUnion* (*geometry*, *geometry*): retorna uma geometria que representa a união das geometrias passadas como parâmetro;
  - *Difference* (*geometry*, *geometry*): retorna uma geometria que representa a diferença entre as geometrias passadas como parâmetro;
  - *Symdifference* (*geometry*, *geometry*): retorna uma geometria que representa a porção entre A e B que não se intersectam.

- Operadores métricos:
  - *Distance* (*geometry*, *geometry*): retorna a distância cartesiana entre duas geometrias;
  - *Area* (*geometry*): retorna a área se a geometria passada como parâmetro for um polígono ou multi-polígono.

- Centróide de geometrias:
  - *Centroid* (*geometry*): retorna o ponto que identifica o centro da geometria passada como parâmetro.

- Validação:
    - *IsSimple* (*geometry*): retorna 1 se a geometria passada como parâmetro não possuir nenhum ponto anormal como interseção com ela mesma ou tangência própria.

3.4.2 Índices GiST e R-Tree

A implementação da R-Tree nativa possui limitação de 8 Kbytes. Na prática é muito comum um SIG manipular dados representados por polígonos maiores de 8 Kbytes cada, o que torna a utilização deste índice inviável.

O método de indexação chamado Generalized Search Trees (GiST) não possui restrições de tamanho e consiste em um índice de árvore balanceada fornecendo funcionalidades de uma B-Tree e R-Tree. A principal vantagem deste índice é a possibilidade de definição de seu comportamento (CASANOVA *et al*., 2005).

3.4.3 SRID - Reference System Identifier

Uma importante configuração de um BDG é o sistema de referência utilizado para manter e gerenciar o dado da posição GPS. No BDG, este sistema de referência é identificado com um Spatial Reference System Identifier (SRID) e mais especificamente, a implementação deste SRID é definida pelo European Petroleum Survey Group (EPSG).

Cada sistema de referência é associado com um código único. Coordenadas GPS são, usualmente, interpretadas de sensores de longitude/latitude, utilizando o WGS84 *geodetic datum* (coordenadas geográficas).

WGS84 é um sistema utilizado globalmente, aonde suas coordenadas angulares estão relacionadas a uma elipsoide muito próxima do desenho da terra. Como consequência, não é correto aplicar funções que são desenvolvidas para trabalhar no espaço Euclidiano^2,

porque uma elipsoide possui caminho mais próximo entre dois pontos e não é uma linha perfeita, mas sim um arco.

Portanto, muitas das camadas disponíveis para uma determinada área são projetadas em um sistema de referência plano (UTM). Neste sentido, o PostGIS possui dois grupos principais de tipos de dados de vetores espaciais: geométrico, o qual trabalha com qualquer tipo de referência espacial e geográfico, o qual, é especificado por coordenadas geográficas (longitude/latitude WGS84) (URBANO, CAGNACCI, 2014).

3.4.4 OGC e OSGeo

Open Geospatial Consortium (OGC) é o órgão que existe para padronizar a maneira como os dados espaciais e a geografia são acessados e distribuídos. Sua missão é impor especificações de como os dados espaciais podem ser acessados de serviços *web*, o formato de entrega e como é feita a consulta dessas informações.

Open Source Geospatial Fundation (OSGeo) é o órgão responsável por financiar, dar suporte e criar estratégias de vendas para as ferramentas SIG de código livre. Ambos os órgãos são capacitados a entregar ferramentas e dados geográficos para todos, e estes, são os principais interessados por padrões de código livre (OBE, HSU, 2011).

No Capítulo 4 serão apresentadas todas as ferramentas e tecnologias que estão envolvidas no desenvolvimento do projeto.

4 TECNOLOGIAS ENVOLVIDAS

Neste capítulo serão abordadas as principais características do SGDB PostgreSQL juntamente com a extensão PostGIS e suas funções para manipulação de dados espaciais. Além disso, também serão descritas as tecnologias *web* e *mobile* envolvidas para o desenvolvimento do aplicativo híbrido, bem como bibliotecas JavaScript, Phonegap e Ionic Framework.

4.1 PostgreSQL

O sistema de gerenciamento de banco de dados objeto-relacional (SGBDOR) PostgreSQL é fruto de um projeto da Universidade de Berkeley na Califórnia (EUA), a qual deu origem a sua licença de uso gratuita, a licença BSD (Berkeley Software Distribution), em meados da década de 80.

Porém, o SGDB PostgreSQL se popularizou somente com o passar dos anos e foi em 1994 que sofreu uma grande modificação que deu origem ao programa Postgres95, que otimizou alguns recursos e incorporou a linguagem SQL substituindo a então PostQUEL. Nesta versão, o programa foi totalmente compatibilizado com o padrão American National Standard Institute (ANSI) podendo ser adaptado a mais de uma plataforma (MILANI, 2008).

Com o passar dos anos, o PostgreSQL ganhou robustez por prover uma arquitetura estável, com forte reputação, confiabilidade e integridade dos dados armazenados. Além disso, o SGBD é compatível com os principais sistemas operacionais atuais, Linux, UNIX e Windows.

Fornece suporte ACID[4], além de chaves estrangeiras, *joins*, *views*, *triggers* e *stored procedures*, incluindo todos os tipos de dados concebidos pela SQL 2008. Disponibiliza interface de comunicação com as principais linguagens de programação, entre estas, C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC (POSTGRESQL, 2015).

O PostgreSQL é um SGBD que possui todas as características necessárias para ser considerado robusto:

- Recuperação automática após *crash* de sistema (WAL);
- MVCC – Controle de concorrência;
- *Logging* de transações;
- *Commit, rollback, checkpoints*;
- *Triggers;*
- *Stored procedures;*
- *Constraints;*
- *Foreing keys;*
- *Backup online*;
- Múltiplos tipos de Índices;
- Escalabilidade.

Estas características são determinantes para colocar o PostgreSQL entre os dez SGBD mais populares da atualidade. Dados da DB-Engines[5] (Maio de 2015), o apontam como o 5º SGBD mais popular da atualidade, ficando atrás de marcas como Oracle, MySQL, Microsoft SQL Server e MongoDB.

---

[4] O termo ACID é descrito como: Atomicidade: Todas as atualizações feitas por uma transação são efetivadas no BD ou nenhuma delas (tudo ou nada); Consistência: A execução de uma transação deve garantir que o BD passe de um estado consistente para outro; Isolamento: Eventos dentro de uma transação devem ser transparentes para outras transações executando concorrentemente (sincronização de transações); Durabilidade: sempre que uma transação é executada com sucesso, o SGBD deve garantir que o seu resultado sobreviva a qualquer falha subsequente (RAMAKRISHNAN, GEHRKE, 2008).

[5] Portal *Ranking* DB-Engines: http://db-engines.com/en/ranking

4.2 PostGIS

PostGIS é um módulo de extensão espacial gratuito e de código aberto que permite adicionar entidades geográficas ao PostgreSQL. Este módulo apresenta um expressivo número de funções espaciais e topológicas que estendem o SQL do SGDBOR, ou seja, permite o uso de objetos GIS, os quais, podem ser manipulados no banco de dados.

Entre as características do PostGIS está a adoção de índices espaciais GiST e R-Tree, além de funções para análise básica (CHAWDHARY, 2014). A extensão disponibiliza cerca de 300 operadores espaciais, funções espaciais, tipos de dados espaciais e índices espaciais (OBE, HSU, 2011).

Sobre a modelagem de dados espaciais, existem extensões físicas em vários SGBD baseado nas especificações do OpenGIS. Porém, existem variações relevantes entre os modelos internos de dados, semântica dos operadores espaciais, mecanismos de indexação e esquema de sintaxe da SQL estendida com tipos espaciais. No caso do PostGIS, sua implementação está certificada pela OpenGIS para o perfil de tipos e funções (FERREIRA *et al.*, 2002).

Os objetos GIS suportados pelo PostGIS são um conjunto dos elementos definidos pela OpenGIS, a qual, define duas formas padrão para expressar dados espaciais: a Well-Known Text (WKT) e a Well-Known Binary (WKB). Exemplos da representação WKT de objetos espaciais são apresentadas a seguir (CASANOVA *et al*., 2005):

- *POINT* (0 0) - Um ponto. São necessárias duas coordenadas (x, y) para definir um ponto;
- *LINESTRING* (0 0,1 1,1 2) - Uma linha. São necessários no mínimo quatro pontos, sendo que dois são referentes ao ponto de partida da linha e outros dois referentes ao ponto de chegada;
- *POLYGON* ((0 0,4 0,4 4,0 4,0 0), (1 1, 2 1, 2 2, 1 2,1 1)) - Um polígono. Pode possuir vários ângulos, só deve se ter atenção pois a última coordenada deve ser idêntica a primeira, para que o polígono possa fechar no mesmo ponto que se iniciou;

- *MULTIPOINT* (0 0,1 2) - Vários pontos. É composto por vários pontos cada uma contendo coordenadas x e y;

- *MULTILINESTRING* ((0 0,1 1,1 2), (2 3,3 2,5 4)) - Várias linhas. É composta por várias linhas cada uma contendo quatro coordenadas, sendo duas do início da linha e duas do final da linha;

- *MULTIPOLYGON* (((0 0,4 0,4 4,0 4,0 0), (1 1,2 1,2 2,1 2,1 1)), ((-1 -1,-1-2,-2 -2,-2 -1,-1 -1))) - Vários polígonos. É composto por vários polígonos cada um com vários ângulos;

- *GEOMETRYCOLLECTION* (POINT (2 3), LINESTRING (2 3,3 4)) - Várias geometrias, sendo estas as citadas acima.

A criação de uma tabela em um banco de dados espacial é constituída de duas etapas. Na primeira, são definidos os atributos básicos (alfanuméricos). Na segunda etapa, é necessário a utilização de uma função especifica para a criação de uma coluna do tipo espacial, conhecida no PostGIS como *AddGeometryColumn*. Essa função, implementada no PostGIS e especificada pelo OpenGIS, realiza todo o trabalho de preenchimento da tabela de metadado *geometry_columns* (CASANOVA *et al*., 2005).

As tabelas de metadados do PostGIS são representadas pelas seguintes estruturas (Tabela 2 e Tabela 3):

Tabela 2 - Tabela de metadado do sistema de coordenadas

spatial_ref_sys		
Atributo	Tipo	Modificador
srid	INTEGER	PK
auth_name	VARCHAR (256)	
auth_srid	INTEGER	
srtext	VARCHAR(2048)	
proj4text	VARCHAR(2048)	

Fonte: Casanova *et al*. (2005).

Tabela 3 - Tabela de metadado das tabelas com colunas espaciais

geometry_columns		
Atributo	Tipo	Modificador
f_table_catalog	VARCHAR (256)	PK

geometry_columns		
Atributo	Tipo	Modificador
f_table_schema	VARCHAR (256)	PK
f_table_name	VARCHAR (256)	PK
f_geometry_column	VARCHAR (256)	PK
coord_dimension	INTEGER	
srid	INTEGER	PK
type	VARCHAR (30)	

Fonte: Casanova *et al*. (2005).

Após criar as tabelas de dados, é possível inserir as informações usando o comando SQL INSERT. Para isso, utiliza-se a representação textual das geometrias em conjunto com a função *GeometryFromText* que recebe a representação textual e mais o sistema de coordenadas em que se encontra a geometria. Exemplo:

Listagem 1 - Inserção de um registro em uma tabela espacial

```
INSERT INTO bairropoa (bairro, spatial_data)
VALUES('JARDIM DOS EUCALIPTOS', GeometryFromText('POINT(321588.628426 7351166.969244)', -1));
```

Fonte: Do autor.

Ao recuperar essas informações do banco de dados, uma seleção é feita utilizando a função espacial *AsText*, que é responsável por transformar a informação espacial da tabela, em uma informação textual.

Listagem 2 - Seleção em uma tabela espacial

```
SELECT bairro,    AsText(spatial_data) geom FROM bairropoa
```

bairro character varying(200)	geom text
JARDIM DOS EUCALIPTOS	POINT(321588.628426 7351166.969244)

Fonte: Do autor.

4.3 PhoneGap

PhoneGap é um *framework* de código aberto para desenvolvimento de aplicações *mobile*, que surgiu do projeto Cordova através do Apache Incubator, caminho de entrada

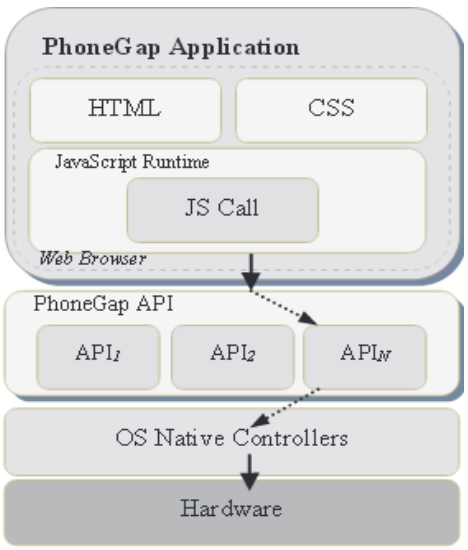
para o Apache Software Foundation (ASF). O objetivo do PhoneGap é permitir usar o navegador como um nível intermediário de abstração para implementar a camada lógica, que é baseada em JavaScript, CSS3 e HTML5 (CORRAL *et al*., 2012).

O PhoneGap permite a criação de aplicações baseadas em *scripts*, porém, somente com JavaScript não é possível explorar todas as funcionalidades de um aparelho. Para resolver este problema, a solução adotada pelo PhoneGap foi disponibilizar um conjunto de APIs que permitem a manipulação de componentes de baixo nível, como o gerenciamento do telefone, hardware e notificações através de uma *engine* nativa. Estas APIs são acessíveis ao JavaScript depois de serem expostas ao navegador por meio da PhoneGap JavaScript *engine* (CORRAL *et al*., 2012).

Desta forma, desenvolvedores preocupam-se somente com o desenvolvimento *web*, uma vez que a camada lógica irá contar com extensões necessárias e interfaces para acessar recursos por meio de métodos de sua própria API (Figura 6). Esta arquitetura simplifica a criação de aplicações e podem ser compiladas para vários sistemas operacionais mobile (Apple iOS, Android, Blackberry OS 6+, Windows Phone 7 e 8, Ubuntu e Firefox OS) (CORRAL *et al*., 2012).

Uma das características principais e determinante para a popularização do Phonegap é a opção de *build in Cloud*, a qual permite a geração do aplicativo nativo para as plataformas listadas.

Figura 6 - Arquitetura do PhoneGap



Fonte: Corral *et al*. (2011).

4.4 AngularJS

AngularJS é um *framework* estrutural, de código aberto, para aplicativos *web* dinâmicos. Permite utilizar HTML como *template* e fornece mecanismos que estendem a sintaxe HTML padrão, visando expressar componentes claros e sucintos. AngularJS utiliza os conceitos de *data binding* e injeção de dependências, além de ser um *framework* do tipo *Model-View-Controller* (MVC). Estes conceitos permitem eliminar boa parte do código a ser desenvolvido e isso tudo acontece em um navegador, tornando-o um parceiro ideal para qualquer tecnologia que irá rodar no servidor (ANGULARJS, 2015).

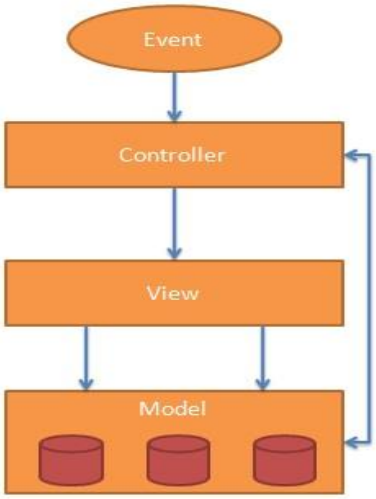
Segundo Seshadri e Green (2014), as tecnologias mais importantes envolvidas em uma aplicação com AngularJS são: Single Page Application (SPA), c*lient-side templates*, MVC, *data binding*, Injeção de dependências e Diretivas.

SPA – Utiliza o conceito de aplicação de página única utilizando Asynchronous Javascript and XML (AJAX), o qual reduz consideravelmente a quantidade de dados trafegados entre cliente e servidor.

Client-side templates – Pelo fato de utilizar SPA, o AngularJS só transita dados que realmente estão modificando na página do cliente. Estes dados não precisam ser processados em um servidor e por padrão transitam no formato JSON e são renderizadas pelo AngularJS e suas *templates*. Desta forma, o AngularJS consegue obter uma performance adequada em relação a outros *frameworks*.

MVC – A camada de modelo é responsável por gerenciar os dados da aplicação, respondendo a requisições da visualização e instruções do controlador para ser atualizada. A camada de visualização apresenta os dados em um formato HTML5 e CSS3 para o usuário final e responde a decisões do controlador para apresentar estas informações. O controlador recebe a entrada de dados do usuário e interage com o modelo indiretamente. O controlador é responsável por validar as informações de entrada e executar operações de negócio que irão alterar o estado da camada modelo (Figura 7).

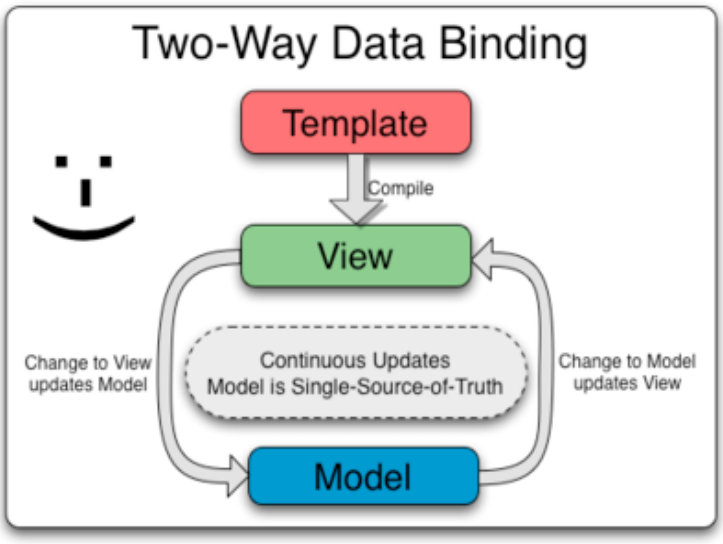
Figura 7 - Arquitetura MVC do AngularJS



Fonte: Seshadri e Green (2014).

Data binding – Em aplicações que utilizam o AngularJS, significa que existe uma sincronização automática de dados entre a camada de modelo e os componentes de visualização. A maneira que o Angular implementa a forma de *data binding* permite ao desenvolvedor tratar o modelo como uma fonte única de dados na aplicação. A visualização é uma projeção do modelo em todos os momentos, quando o modelo altera, a visualização reflete automaticamente estas alterações e vice-versa (Figura 8).

Figura 8 - *Data binding* com AngularJS



Fonte: AngularJS (2015).

A injeção de dependências consiste em fornecer recursos extras necessários na aplicação, de modo que o desenvolvedor somente deverá solicitar um recurso que será injetado pelo framework e ficará disponível para uso.

Através de diretivas é possível criar elementos específicos que serão aplicados ao documento HTML em forma de atributo, elemento, comentário ou classe CSS. Para isto, ao utilizar-se diretivas o compilador HTML do Angular insere um especifico comportamento ao documento HTML para a diretiva acrescentada.

4.5 Ionic Framework

Como visto na seção 4.4, AngularJS é um *framework* que permite a utilização de várias tecnologias para o desenvolvimento de softwares, trazendo melhorias de performance e uma nova visão sobre o conceito de desenvolvimento de aplicações com o *data binding*.

Ionic é um potente framework para o desenvolvimento de aplicativos híbridos que utiliza HTML, CSS e JavaScript e roda na maioria das plataformas móveis disponíveis no mercado. Possui uma excelente suíte de componentes que dão a impressão do usuário final estar utilizando um aplicativo nativo, e este é o seu foco, no design do *front-end* das aplicações. É uma das ferramentas que estão mudando a maneira como desenvolvedores constroem aplicativos móveis, pois assim como o PhoneGap, qualquer pessoa com conhecimento de desenvolvimento *front-end* pode começar a criar aplicativos para plataformas móveis (IONIC, 2015). Além disso, Ionic é gratuito e de código aberto, foi construído e otimizado para utilizar AngularJS.

4.6 Web Services REST

Web service é um software modelado para suportar a interação entre sistemas através de uma rede, na maioria das vezes escritos em linguagens e até sistemas operacionais diferentes. Assim, para que esta interação seja possível, é necessário ter um padrão na sua comunicação, ou seja, um web service é um aplicativo servidor que disponibiliza um ou mais

serviços para seus clientes sendo acessados via *web*, semelhante a um site, que responde a requisições de um Uniform Resource Identifier (URI).

A comunicação entre sistemas já existentes é sempre uma questão problemática. Ao imaginar um cenário onde tem-se dois ou mais softwares que precisam se comunicar, alguns pontos devem ser levados em consideração, como linguagem utilizada, banco de dados, infraestrutura, etc. Como uma solução para isto, são utilizados web services para realizar a comunicação entre dois ou mais sistemas.

Representational State Transfer (REST) não é a especificação de um novo protocolo. Usualmente, REST funciona com os recursos que o HTTP disponibiliza, e surgiu como uma alternativa popular ao uso de tecnologias baseadas em web services SOAP[6], por ser mais leve e ter a capacidade de transmitir dados via HTTP (BURKE, 2009).

Por conta disso, existe uma equivalência entre os métodos padrões utilizados pelo protocolo HTTP para a representação de serviços REST. Desta forma, o HTTP tem um papel fundamental nesta "interface uniforme" que é disponibilizada tendo como principais e mais comuns métodos utilizados o POST, GET, PUT e DELETE. Os quais correspondem a CREATE, READ, UPDATE e DELETE (CRUD) operações que são frequentemente utilizadas por serviços REST (FREDRICH, 2012).

No Capítulo 5 será apresentado a proposta de desenvolvimento do aplicativo e os resultados obtidos com o aplicativo, visando aplicar os conceitos vistos nos capítulos anteriores.

[6] SOAP é um protocolo de comunicação para troca de mensagens que permite a comunicação entre sistemas operacionais distintos através da internet baseado em XML, utiliza *Remote Procedure Call* (RPC) ou HTTP para negociação e troca de mensagens. Quando criado, surgiu como forte alternativa para protocolos proprietários como CORBA e DCOM (BURKE, 2009).

5 PROPOSTA

Neste capítulo, será apresentado o desenvolvimento do aplicativo móvel que utiliza geolocalização para mapear as áreas com maiores índices de criminalidade bem como crimes contra o patrimônio na cidade de Porto Alegre/RS utilizando mapas de calor. Após a criação deste mapa, foi abordado a criação de rotas para que as viaturas possam trafegar pelos locais onde estes crimes aconteceram, com o objetivo de reprimir o acontecimento de novos incidentes. Também serão exibidas as telas com os resultados obtidos pelo aplicativo.

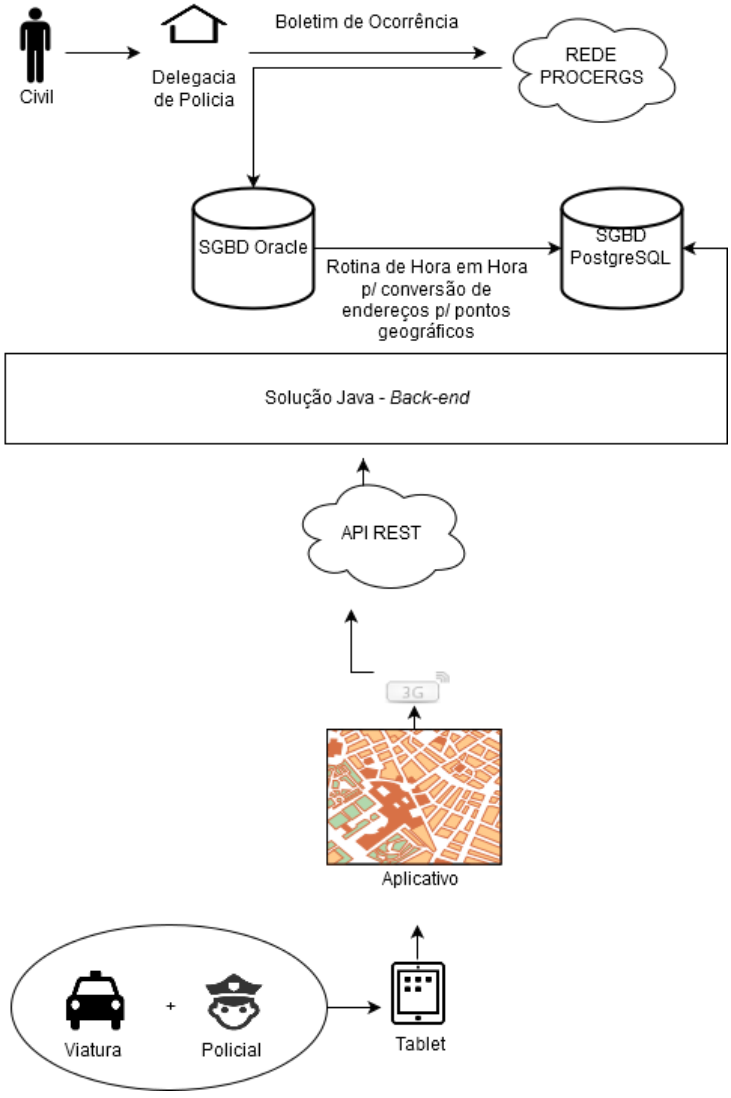
5.1 Visão Geral

O aplicativo permite aos usuários das viaturas (policiais) saber aonde se encontram os maiores índices de criminalidade da cidade de Porto Alegre/RS e também obter sugestões de rotas criadas com base nos registros policiais que as pessoas realizam em delegacias, na categoria de crime contra o patrimônio. Por conta disso, a proposta do aplicativo é que o seu uso seja restrito à Brigada Militar e órgãos da segurança pública do Estado, possibilitando aos usuários ter um controle maior da situação vivenciada pela cidade, trazendo o benefício da informação e o melhor direcionamento de efetivo, bem como políticas de prevenção de crimes, como visto no Capítulo 2.

A Figura 9 exibe o objetivo da utilização prática do aplicativo, como ele será relacionado ao dia-a-dia de uma viatura policial, é necessário que o Tablet que será utilizado possua conexão com a internet para o seu funcionamento, pois, a ideia principal deste produto é que ele se torne parte do planejamento diário de rondas e direcionamento de efetivo para áreas com maiores incidências de acontecimentos criminosos, logo, para que isso seja possível, é necessário que as informações exibidas no aplicativo sejam online. O mapa de calor por sua vez, exibe uma projeção macro das áreas de risco permitindo ao usuário aplicar

critérios de filtragem para qualquer data informada, desde que, os registros estejam convertidos para dentro do SGDB PostgreSQL. Na geração de rotas, o objetivo é permitir ao policial obter uma rota com base em acontecimentos passados informando a distância que ele deseja percorrer dentro da área de abrangência de seu batalhão, neste sentido, o aplicativo considerará somente às 24 horas anteriores de registros para que não sobrecarregue as requisições efetuadas a API do Google e também evitar direcionar efetivo para acontecimentos com datas defasadas.

Figura 9 - Interações com o usuário

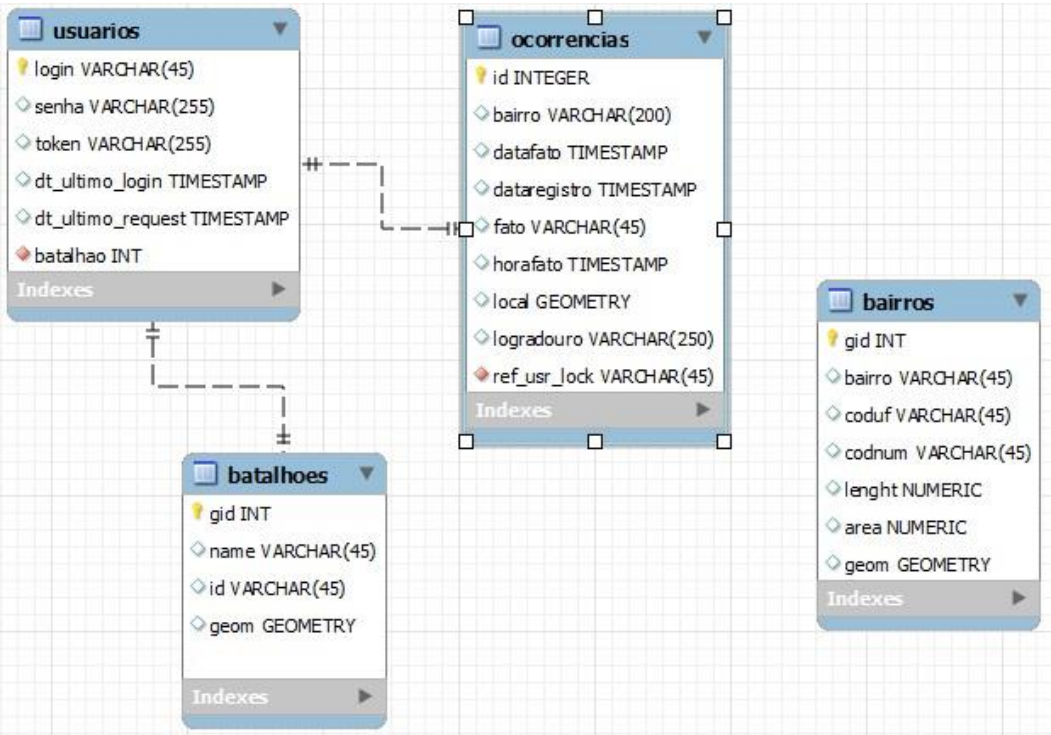


Fonte: Do autor.

5.2 Modelo relacional

Para melhor exemplificar a estrutura de tabelas utilizada no SGDB PostgreSQL do protótipo, a Figura 10 ilustra quais os tipos de informações foram armazenados nas colunas do BD e quais os relacionamentos que as tabelas possuem.

Figura 10 – Diagrama ER do protótipo



Fonte: Do autor.

As Tabelas 4 a 7 dispõem de uma breve descrição das tabelas, colunas e campos usados no protótipo.

Tabela 4 - Tabela de usuários

Tabela	Usuários			
Descrição	Armazena as informações dos usuários do sistema			
Campos/Atributos				
Chave	Nome	Tipo	Nulo	Descrição
PK	login	INT	não	Armazena o login do usuário no sistema.

	senha	TEXT	não	Armazena a senha do usuário no sistema.
	token	TEXT	sim	Usado para controlar as requisições REST, ao fazer o primeiro login o sistema passa a conversar somente com um token de acesso que expira em 24h.
	dt_ultimo_login	TIMESTAMP	sim	Armazena o último login do usuário para renovar o token após cada acesso.
	dt_ultimo_request	TIMESTAMP	sim	Armazena o último *request* que é realizado ao servidor.
FK	batalhão	INT	não	Armazena a qual batalhão o usuário pertence, para que na geração das rotas o sistema saiba para onde enviar o veículo que o usuário está utilizando.

Fonte: Do autor.

Tabela 5 - Tabela de batalhões

Tabela	batalhões			
Descrição	Armazena as informações dos batalhões da brigada militar			
Campos/Atributos				
Chave	Nome	Tipo	Nulo	Descrição
PK	gid	INT	não	Chave primária.
	name	TEXT	sim	Armazena a descrição do batalhão.
	id	TEXT	sim	Armazena o código do batalhão.
	geom	MULTIPOLYGON	sim	Armazena o multi-polígono que representa espacialmente a área de atuação do batalhão.

Fonte: Do autor.

Tabela 6 - Tabela de ocorrências

Tabela	ocorrências

Descrição	Armazena as informações das ocorrências			
Campos/Atributos				
Chave	Nome	Tipo	Nulo	Descrição
PK	id	INT	não	Chave primária.
	bairro	TEXT	sim	Armazena o registro do bairro aonde aconteceu a ocorrência.
	datafato	TIMESTAMP	sim	Armazena a data do acontecimento do fato.
	dataregistro	TIMESTAMP	sim	Armazena a data em que o indivíduo foi até uma delegacia registrar o BO.
	fato	TEXT	sim	Armazena a descrição do fato acontecido.
	horafato	TIMESTAMP	não	Armazena a hora do acontecimento do fato.
	local	POINT	não	Armazena a informação geográfica após convertido para coordenadas geográficas.
	logradouro	TEXT	não	Armazena o logradouro do fato
FK	ref_usr_lock	TEXT	não	Ao gerar uma rota é necessário atrelar a ocorrência a um determinado usuário para que a ocorrência não seja disponibilizada novamente para outras gerações de rotas.

Fonte: Do autor.

Tabela 7 - Tabela de bairros

Tabela	bairros			
Descrição	Armazena as informações dos bairros da cidade de Porto Alegre (Tabela importada de um sistema terceiro)			
Campos/Atributos				
Chave	Nome	Tipo	Nulo	Descrição
PK	gid	INT	não	Chave primária.
	bairro	TEXT	sim	Armazena o nome do bairro.

	coduf	TEXT	sim	Armazena o código uf, neste caso, sempre será RS pois os registros são somente de Porto Alegre.
	codnum	TEXT	sim	Não utilizado.
	lenght	NUMERIC	sim	Não utilizado.
	area	NUMERIC	sim	Não utilizado.
	geom	MULTIPOLYGON	sim	Geometria que contém o multi-polígono com as determinadas abrangências de cada bairro.

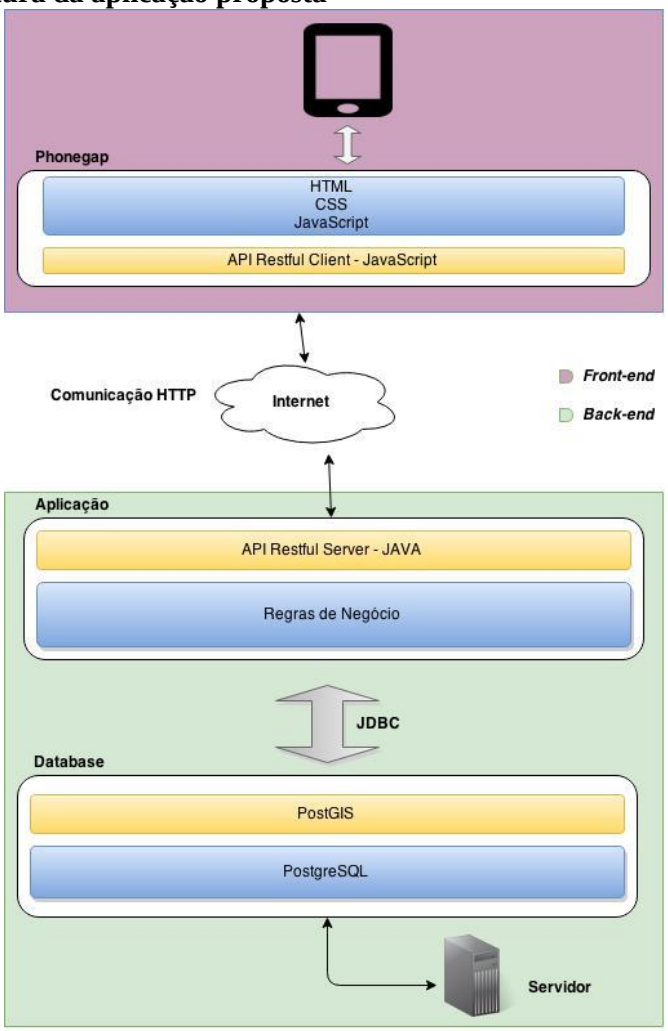
Fonte: Do autor.

5.3 Aplicativo Mapoli

A opção por tecnologias inovadoras e atuais traz uma motivação a mais para o desenvolvimento deste projeto, visto que, a alguns anos atrás só era possível desenvolver aplicações para dispositivos móveis utilizando linguagens nativas, o que tornava o processo do desenvolvimento mais lento e complicado. Após o contato com a ferramenta PhoneGap, vista na Seção 4.3, e as demais tecnologias envolvidas do Capítulo 4, verificou-se que é possível desenvolver aplicações *mobile* utilizando HTML, CSS e JavaScript, o que torna o processo de codificação mais rápido e eficaz.

O desenvolvimento está separado em duas interfaces. A parte visual e *mobile* da aplicação, denominada *front-end*, e a parte de regras de negócio e acesso a banco de dados, denominada *back-end*. A comunicação destas interfaces foi feita através do protocolo HTTP com o uso da tecnologia REST, como vista na Seção 4.6. Para o desenvolvimento da aplicação *front-end* o aplicativo foi desenvolvido utilizando o *framework* AngularJS, visto na Seção 4.4 juntamente com o *framework* Ionic, citado na Seção 4.5, que é responsável pela parte visual do aplicativo e compatibilidade dos componentes visuais com outros dispositivos. A Figura 11 ilustra toda a arquitetura *back-end* e *front-end* envolvida no projeto.

Figura 11 - Arquitetura da aplicação proposta



Fonte: Do autor.

Os requisitos funcionais necessários para a aplicação, serão:

a) Visualização do mapa de calor;

b) Filtro do mapa por: data e hora, tipo de ocorrência e categoria de crime;

c) Sugestão de rota a partir da localização do veículo;

d) Possibilidade de informar a distância da rota que deseja percorrer;

e) Controle de Acesso para usuários;

f) Controle de geração de rotas por viatura: Não permitir a geração de rotas duplicadas;

A necessidade de ferramentas e mecanismos que permitissem que o protótipo funcionasse com banco de dados geográfico, uma das decisões deste projeto foi optar pelo uso do Java no *back-end* juntamente com uma biblioteca conhecida como Hibernate Spatial, a qual, disponibiliza uma série de funcionalidades para que o desenvolvedor não necessite manipular diretamente os SQLs de inserção, atualização e remoção de registros no BD. Também, como justificativa a não utilização de outra tecnologia no *back-end* está a experiência do autor deste projeto com a tecnologia Java, sendo que já utiliza por mais de 5 anos. Todavia, outras tecnologias como Javascript no *back-end* através da ferramenta Node.js[7] seria ideal caso a solução não utilizasse um BD com estrutura tradicional e necessitasse de suporte geográfico.

A aplicação *back-end* foi desenvolvida com base na arquitetura *Service-Oriented Architecture*[8] (SOA), que visa interoperabilidade de sistemas por meio de um conjunto de interfaces de serviço fracamente acoplados, onde os serviços não necessitam de detalhes técnicos da plataforma para a troca de informações entre os sistemas. Neste projeto os serviços foram desenvolvidos utilizando a API REST, vista na Seção 4.6 utilizando a linguagem de programação Java com a API JAX-RS. Abaixo, estão listadas as tecnologias que serão utilizadas no *back-end*:

a) JBoss AS[9]: Servidor de aplicação que possui suporte a todas as especificações da API Java E, incluindo a interface entre a comunicação HTTP com o código Java escrito em um serviço REST (KURNIAWAN, 2002);

b) Maven[10]: É responsável por gerenciar dependências, controlar versão de artefatos, gerar relatórios de produtividade, garantir execução de testes, manter nível de qualidade do código e executar construção de projetos (MAVEN, 2015);

c) Enterprise Java Beans[11] (EJB): tem como principal função fornecer um desenvolvimento rápido e simplificado das aplicações Java, além de um container de injeção de dependências para classes (KURNIAWAN, 2002);

---

7 Portal Node.js: https://nodejs.org/en/about/
8 Portal Arquitetura SOA: http://www.oracle.com/br/products/middleware/soa/overview/index.html
9 Portal JBoss AS: http://jbossas.jboss.org/
10 Portal Maven: https://maven.apache.org/
11 Portal Java API EJB: http://www.oracle.com/technetwork/java/javaee/ejb/index.html

d) Java Persistence API[12] (JPA): Fornece aos Plain Old Java Objects (POJOs) um modelo de persistência para o mapeamento objeto-relacional.

Também foi utilizado a IDE de desenvolvimento conhecida como Eclipse[13], que facilita a manipulação de classes e objetos do Java no *back-end* da aplicação.

[12] Portal Java API JPA: http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html

[13] Portal IDE Eclipse: https://eclipse.org/

6 RESULTADOS

Neste Capítulo serão apresentados os resultados que foram encontrados após a criação do aplicativo descrito na proposta e uma visão geral de como funciona o *build* para o sistema operacional Android com o Phonegap. Além disso, será visto como ficou o funcionamento da conversão de registros de um BD para outro.

6.1 Conversão de dados

Para que as informações exibidas tenham confiabilidade, em uma versão em produção, serão coletadas do Banco de Dados Oracle que se encontra na Cia. de Processamento de dados do Rio Grande do Sul (PROCERGS) as ocorrências que possuem fatos definidos onde os logradouros estejam informados, que sejam da cidade de Porto Alegre/RS, estejam a cargo da Polícia Civil e sejam crimes contra o patrimônio, porém, para este protótipo os registros que foram selecionados pertencem ao Banco de Dados de desenvolvimento devido a estas informações não estão abertas para toda a sociedade. Após esta coleta, foi necessário converter os endereços dos registros de ocorrência para pontos geográficos dentro do sistema de coordenadas utilizando a técnica de geocodificação, descrita na Seção 3.3. Só foram considerados os endereços passíveis de conversão, aqueles que possuem a seguinte estrutura, ex: Rua Bento Gonçalves, 100 – Porto Alegre. Para determinar os registros de ocorrências que se encaixam neste critério foi executado o comando SQL exibido na Figura 12.

Figura 12 – Instrução SQL que lista as ocorrências válidas

```
SELECT trim(initcap(L.K944_TIPO_RUA || ' ' || decode( L.
K944_ADJETIVO_RUA, ' ', '', L.K944_ADJETIVO_RUA || '. ') || L.
K944_NOME_RUA)) || ', ' ||
      to_number(decode(REGEXP_INSTR (O.K66_NRO, '[^[:digit:]]'),0,O.
      K66_NRO,0)) Logradouro,
      initcap(decode(instr(decode(instr(L.K944_BAIRRO_INIC,'-'), 0, L
      .K944_BAIRRO_INIC),'/'), 0, L.K944_BAIRRO_INIC)) Bairro,
      initcap(F.K65_NOME_COMPLETO) Fato,
      to_date(lpad(O.K66_DD_REGISTRO,2,0) ||'/'|| lpad(O.
      K66_MM_REGISTRO,2,0) ||'/'|| lpad(O.K66_AA_REGISTRO,4))
      DataRegistro,
      to_date(O.K66_AAMD_INI_FATO,'YYYYMMDD') DataFato,
      to_char(to_date(lpad(o.k66_hms_registro,6,0),'HH24MISS'),
      'HH24:MI:SS') horaFato
FROM KICC66D O,
     KICC942D CID,
     KICC944D L,
     KICC65D F
WHERE
     --JOIN OCORRENCIA COM CIDADE
     O.K66_CEP = CID.K942_NRO_MUNIC AND

     --JOIN COM LONGRADOURO
     L.K944_NRO_MUNIC = O.K66_CEP AND
     L.K944_NRO_INT_LOGR = O.K66_LOGR AND
     L.K944_NOME_RUA <> '-' AND

     --JOIN COM FATOS
     F.K65_CODIGO_FATO = O.K66_CODIGO_FATO AND
     F.K65_COMPL_FATO = O.K66_COMPL_FATO AND

     O.K66_LOGR <> 1000 AND -- Somente os que informaram longradouros
     O.K66_CEP = 90000 AND --Cidade de Porto Alegre
     O.K66_PROCED = 1 AND -- Registros já na base da polícia civil
     F.K65_SUBGRUPO_FATO = 35 AND --CRIMES CONTRA O PATRIMONIO
     ( L.K944_TIPO_RUA = ' ' or
     to_number(decode(REGEXP_INSTR (O.K66_NRO, '[^[:digit:]]'),0,O.
     K66_NRO,0)) > 0) -- somente endereços com números
ORDER BY O.K66_AAMD_INI_FATO desc, o.k66_hms_registro desc
```

Fonte: Do autor.

Logo, para realizar a conversão destas informações foi necessário escrever um código Java no servidor que coleta as ocorrências da base de dados e executa a chamada à Google API passando como parâmetro o seu endereço. Após isso, o retorno obtido é a posição geográfica do endereço com a latitude e longitude de onde aconteceu o fato descrito na ocorrência. A Figura 13 mostra o código que realiza esta conversão e que atualiza o ponto geográfico no BD.

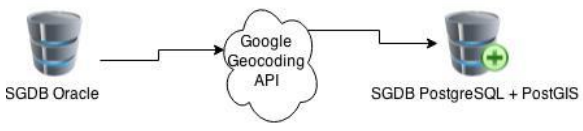
Figura 13 - Código fonte responsável por converter endereço em ponto geográfico

```
@Override
public String convertPontosGeo() {
  final Geocoder geocoder = new Geocoder();
  WKTReader reader = new WKTReader();

  List<Ocorrencia> list = findAll(new WhereClause<Ocorrencia>() {
    @SuppressWarnings({ "rawtypes", "unchecked" })
    @Override
    public void buildWhere(CriteriaQuery cq, CriteriaBuilder cb, Root root, Ocorrencia entity) {
      cq.where(cb.isNull(root.get("local")));

    }
  }, 200);

  Session session = entityManager.unwrap(Session.class);
  int count = 0;

  for (Ocorrencia i : list) {
    GeocoderRequest geocoderRequest = new GeocoderRequestBuilder()
    .setAddress(i.getLogradouro() + ", Porto Alegre - RS").getGeocoderRequest();

    try {
      GeocodeResponse geocoderResponse = geocoder.geocode(geocoderRequest);
      if (geocoderResponse.getResults().isEmpty()) {
        System.out.println(i.getSequence() + " - " + i.getLogradouro());
        continue;
      }
      LatLng v = geocoderResponse.getResults().get(0).getGeometry().getLocation();

      Point p = (Point) reader.read("POINT("+v.getLng()+" "+v.getLat()+")");
      p.setSRID(4326);

      i.setLocal(p);

      session.update(i);
      if (++count % 50 == 0) {
        session.flush();
        session.clear();
      }

      if (count == 200) {
        break;
      }
    } catch (IOException | ParseException e) {
      System.out.println("Não converteu: - " + i.getSequence());
      e.printStackTrace();
    }
  }
```

Fonte: Do autor.

A Figura 14 ilustra a arquitetura da aplicação que irá realizar o trabalho de exportação, importação e conversão dos registros.

Figura 14 - Funcionamento da extração de dados



Fonte: Do autor.

Após a conversão desses registros eles serão armazenados no SGDB PostgreSQL descrito na Seção 4.1, pois os registros serão manipulados pelas funções de um BDG, descritas na Seção 3.4.1. Todavia, esta extração do SGBD Oracle será realizada de 30 em 30 minutos.

6.2 *Build* da aplicação móvel

Através do PhoneGap, visto na Seção 4.3, foi possível utilizar o serviço de geolocalização do aparelho, capturar informações das redes, tanto da rede wireless como 3g e gerar um pacote de instalação, neste caso, como os tablets da Brigada Militar rodam o sistema operacional Android na versão 4.2.2, para este protótipo só foi necessário realizar a compilação para esta plataforma, ilustrado nas Figura 15 e Figura 16. Porém, no início desde projeto, não era conhecido o modelo do aparelho e qual o sistema operacional que o mesmo utilizava, devido a isto, optou-se por uma solução genérica.

Figura 15 - Compilação para o sistema operacional Android

```
D:\dev\workspaces\gps\apphm>cordova build android
```

Fonte: Do autor.

Figura 16 - Resultado da compilação

```
BUILD SUCCESSFUL

Total time: 40.414 secs
Built the following apk(s):
    D:\dev\workspaces\gps\apphm\platforms\android\build\outputs\apk\android-debug.apk

D:\dev\workspaces\gps\apphm>
```

Fonte: Do autor.

6.3 Controle de Acesso

O principal ganho ao optar pela arquitetura REST está na transmissão das informações entre as camadas *front-end* e *back-end*, o tamanho da troca de mensagens entre estas camadas é considerado pequeno com o uso de REST, visto na Seção 4.6. Porém, para garantir a segurança de serviços REST deve-se ter um cuidado a mais ao expor a API na internet.

Neste protótipo, foi utilizado o controle de autenticação de usuários baseado em *token*, no qual, o usuário informa as suas credenciais no aplicativo e as envia para o servidor por meio de uma conexão segura (HTTPS), ao receber estas credenciais no servidor é

realizado a autenticação do usuário e caso tenha sucesso ao autenticar será gerado um *token* que fica válido para o usuário que realizou a autenticação por 15 minutos, este tempo é renovado quando o usuário realiza a próxima requisição ao servidor. Ao efetuar o *login* pela primeira vez o sistema guarda este *token* na tabela de usuários e após isto o cliente só comunica-se com a aplicação por meio do *token* gerado, enviando-o no cabeçalho das requisições.

A Figura 17 mostra a tela de *login* do sistema e as Figura 18, Figura 19 e Figura 20 mostram o código *front-end* em AngularJS utilizando a arquitetura MVC relatada na Seção 4.4.

Figura 17 - Tela de *login* do aplicativo

Fonte: Do autor.

Figura 18 - *View* da tela de *login*

```
<ion-pane>
  <ion-header-bar>
      <h1 class="title">Seja Bem vindo!</h1>
  </ion-header-bar>
  <ion-content>
    <form ng-submit="doLogin()">
      <div class="list">
        <label class="item item-input">
          <span class="input-label">Usuário</span>
          <input type="text" ng-model="loginData.username" />
        </label>
        <label class="item item-input">
          <span class="input-label">Senha</span>
          <input type="password" ng-model="loginData.password">
        </label>
        <label class="item">
          <button class="button button-block button-positive"
                  type="submit">Entrar</button>
        </label>
        <p align="center">
          <button ng-show="msgErr != null"
                  class="button button-clear button-assertive">
            {{msgErr}}
          </button>
        </p>
      </div>
    </form>
  </ion-content>
</ion-pane>
```

Fonte: Do autor.

Figura 19 - *Controller* da tela de *login*

```
.controller('LoginCtrl', function($scope, $location, $ionicLoading, Auth) {
    // Form data for the login modal
    $scope.loginData = {};
    Auth.clearUser();

    $scope.doLogin = function() {
        $ionicLoading.show({
            content: 'Entrando...',
            animation: 'fade-in',
            showBackdrop: true,
            maxWidth: 200,
            showDelay: 0
        });

        Auth.doLogin($scope.loginData, function(descErro) {
            $scope.msgErr = null;
            try{
                if (!Auth.hasUser()) {
                    $location.path('/');
                    $scope.msgErr = "Usuário ou senha incorretos";
                } else {
                    $location.path('/map');
                }
                $scope.loginData = {};
            }catch(e){
               $scope.msgErr = descErro;
            } finally {
               $ionicLoading.hide();
            }
        });
    }
})
```

Fonte: Do autor.

Figura 20 - *Service* da tela de *login*

```
.service('Auth', function($http, $q, $window, $base64, ApiEndpoint) {
    return {
        hasUser: function() {
            return angular.fromJson($window.sessionStorage.currentUser)
                          .token != null;
        },
        getUser: function() {
            return angular.fromJson($window.sessionStorage.currentUser);
        },
        clearUser: function() {
            $window.sessionStorage.currentUser = null;
        },
        doLogin: function(credentials, callback) {
            $window.sessionStorage.currentUser = null

            if (credentials.username && credentials.password) {
                $http.post(ApiEndpoint.url + '/rest/usuarios/auth', '', {
                    headers: {
                    'usr': $base64.encode(credentials.username.toLowerCase()),
                    'pwd': $base64.encode(credentials.password)
                }}).
                then(function(response) {
                    $window.sessionStorage.currentUser = angular.toJson(response.data);
                    callback();
                }, function(response) {
                    callback("Erro ao autenticar usuário");
                });
            } else {
                callback('Informe usuário e senha!');
            }
        }
    }
})
```

Fonte: Do autor.

A Figura 21 mostra o código responsável pela geração do *token* e a regra de negócio que salva no BD. A Figura 22 exibe como é feita a comunicação após o *login* com o *token* no cabeçalho das requisições.

Figura 21 - Código que gera o *token* e atualiza no BD

```
@Override
public Usuario auth(@Context HttpServletRequest request,
                    @HeaderParam("usr") String usr,
                    @HeaderParam("pwd") String pwd) {
  Usuario usuario = null;

  try {
    if ((usuario = usuarioRn.login(new UsuarioBuilder(new String(Base64.decode(usr)),
                                                      new String(Base64.decode(pwd))).build()))
                                                      != null) {
      SecureRandom random = new SecureRandom();
      byte bytes[] = new byte[64];
      random.nextBytes(bytes);
      usuario.setToken(Base64.encodeBytes(bytes));

      usuario.setUltimoLogin(Calendar.getInstance());
      usuario.setUltimoRequest(Calendar.getInstance());
      usuarioRn.persistir(usuario);

      usuario.setSenha(null);
    }
  } catch (Exception e) {
    throw new IllegalArgumentException(e.getMessage());
  }
  return usuario;
}
```

Fonte: Do autor.

Figura 22 - Exemplo de comunicação com *token* no cabeçalho da requisição

```
pontosRotaSol: function(user, lat, lng, distance, callback) {
    $http.post(ApiEndpoint.url +
               '/rest/ocorrencias/pontosBatalhaoUsuario', {
        'lat':lat,
        'lng':lng,
        'distance':distance
    },
    {headers: {
        'login': $base64.encode(user.login),
        'token': $base64.encode(user.token)
    }}).
    then(function(response) {
        callback(response.data);
    }, function(response) {
        callback('Erro ao gerar rota!');
    });
}
```

Fonte: Do autor.

Figura 23 - *Token* gerado no BD

login character varying(255)	token character varying(255)
rdanieli	i6/cNioxhlWBHVIlzf9xy4Kvph5VkeFGxXRAQCLYLNMInVCBE/t9/3YTH9S0TmEyFhQsFYfCVZZxDx/+ZOh+9A==

Fonte: Do autor.

6.4 Mapa de calor

O mapa de calor[14] que foi utilizado faz parte da Google Maps API JavaScript v3 e funciona como uma camada que é colocada em cima do mapa padrão. Ao gerar o mapa de calor, é possível visualizar a criticidade da criminalidade nas regiões de Porto Alegre/RS.

Representar a densidade geográfica de pontos num mapa utilizando áreas coloridas é a solução quando existe um vasto número de elementos próximos uns aos outros. Neste sentido, a equação abaixo exibe a fórmula que é utilizada para determinar o valor na matriz de gradientes, ou seja, retorna os valores convertidos dentro da matriz de gradientes das ocorrências com base no raio informado em *pixels*, assim, define-se qual gradiente será aplicado no mapa para cada ponto (GOOGLE, 2015).

$$gradient = \frac{sum(points\ in\ radiusCircle)}{radius^2 \times \pi}$$

A Figura 24 exibe como foi feito a utilização do mapa de calor da API do Google para este protótipo.

Figura 24 - Código para geração do mapa de calor e os gradientes utilizados

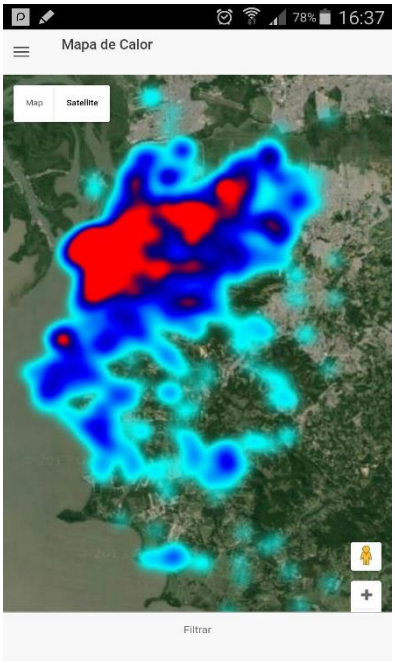
```
var reloadMap = function(data) {
    var map = MapFactory.init();
    var points = [];
    for (var i = 0; i < data.length; i++) {
        var cor = JSON.parse(data[i].jsonLocal).coordinates;
        var latlng = new google.maps.LatLng(cor[1], cor[0]);
        points.push(latlng);
    };
    var pointArray = new google.maps.MVCArray(points);
    heatmap = new google.maps.visualization.HeatmapLayer({
        data: pointArray
    });

    var gradient = [
        'rgba(0, 255, 255, 0)','rgba(0, 255, 255, 1)','rgba(0, 191, 255, 1)',
        'rgba(0, 127, 255, 1)','rgba(0, 63, 255, 1)','rgba(0, 0, 255, 1)',
        'rgba(0, 0, 223, 1)','rgba(0, 0, 191, 1)','rgba(0, 0, 159, 1)','rgba(0, 0, 127, 1)',
        'rgba(63, 0, 91, 1)','rgba(127, 0, 63, 1)','rgba(191, 0, 31, 1)','rgba(255, 0, 0, 1)'
    ];
    heatmap.setOptions({
        dissipating: true,
        maxIntensity: 30,
        radius: 20,
        opacity: 1,
        gradient: gradient
    });
    heatmap.setMap(map);
    google.maps.event.addDomListener($window, 'load', map);
};
```

Fonte: Do autor.

[14] Portal API Heatmap Google: https://developers.google.com/maps/documentation/javascript/examples/layer-heatmap?hl=pt-br

É possível informar parâmetros de filtragem para o mapa que está sendo exibido. A Figura 25 exibe o comportamento padrão do mapa após o usuário efetuar o *login* na aplicação. Um diferencial deste sistema é que a geração do mapa de calor é feita de maneira dinâmica, toda a informação que é adicionada no filtro é computada novamente pelo servidor.

Figura 25 - Visualização do mapa de calor



Fonte: Do autor.

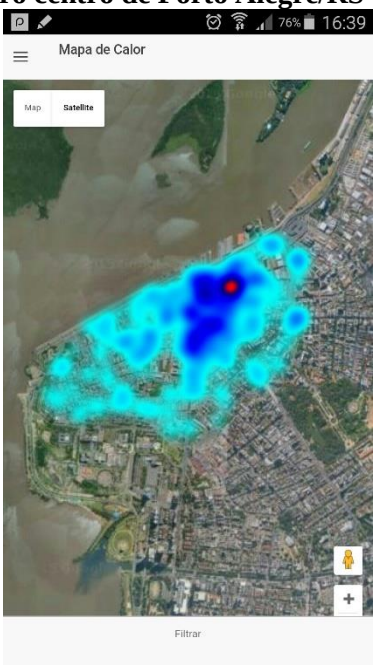
Como decisão de projeto foi optado pela geração do mapa de calor somente no *front-end*, ou seja, através da API REST o usuário busca as informações pelos critérios do filtro e a API devolve as ocorrências e então o mapa é gerado, a Figura 26 exibe a tela para informar os critérios de filtragem.

Figura 26 - Tela com o filtro para o mapa de calor

Fonte: Do autor.

Neste caso, foi aplicado o critério de filtro somente para registros do centro da cidade de Porto Alegre/RS e o mapa exibido pode ser acompanhado pela Figura 27.

Figura 27 - Ocorrências do bairro centro de Porto Alegre/RS



Fonte: Do autor.

Para realizar esta filtragem foi necessário executar um *select* geométrico no BDG, o qual, pode ser acompanhado pela Figura 28, aonde foram omitidos os pontos que definem o multi-polígono que determina a região do centro da cidade para que a imagem ficasse visível.

Figura 28 - Consulta aos registros do bairro centro

```
select
    o.id as "sequence",
    o.logradouro as "logradouro",
    o.bairro as "bairro",
    o.fato as "fato",
    o.dataregistro as "dataRegistro",
    o.datafato as "dataFato",
    o.horafato as "horaFato",
    o.ref_usr_lock as "usuarioResp",
    st_asgeojson(o.local) as jsonLocal
from
    ocorrencias o
where
    1=1
    and  ST_CONTAINS( ST_GeomFromText('SRID=4326;MULTIPOLYGON (((...)))'), o.local)
and o.local is not null
```

Fonte: Do autor.

É possível realizar a combinação do filtro de bairro com a descrição do fato informado na ocorrência. Para isto, foi realizado uma consulta utilizando o fato de furto de veículo e o centro da cidade como área de abrangência, o resultado é ilustrado pela Figura 29.

Figura 29 - Filtragem para crime de furto de veículos no centro de Porto Alegre

Fonte: Do autor.

6.5 Geração de rotas

Para a criação de rotas foi utilizado a Google JavaScript Directions API[15] que permite criar rotas dentro da malha viária do mapa. Através desta API foi possível calcular as direções com base em uma origem, destino e pontos de passagem (*waypoints*). Devido a utilização alto nível desta API é transparente ao desenvolvedor os cálculos realizados por ela na criação das rotas, pois são considerados diversos fatores como os sentidos das ruas e outras informações não reveladas pela Google. A utilização dá-se por meio de requisições onde é informado o caminho desejado e como retorno um *array* de pontos geográficos que determina a rota, após isto, é necessário a criação de polilinhas para representar a rota desejada pelo usuário.

Porém, para que fosse possível a utilização desta API teve-se o desafio de controlar a quantidade de requisições gratuitas que são realizados ao Google, pois ele restringe a quantidade de chamadas a 10 *waypoints* por requisição entre origem e destino, ou seja, para executar este controle via JavaScript foi necessário utilizar uma abordagem que atendesse as restrições da API e como solução a isto as requisições foram feitas de maneira segmentada. A Figura 30 ilustra o código que foi utilizado para fazer este controle.

Figura 30 - Código que realiza requisições de rotas

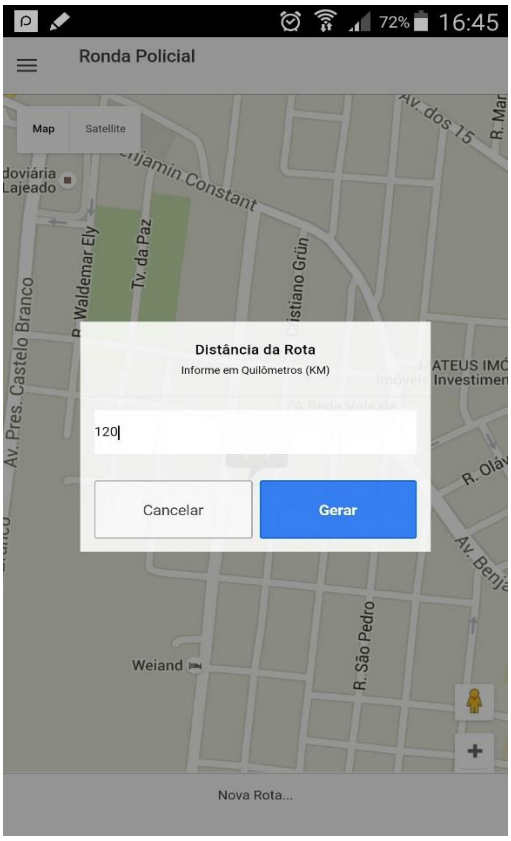
```
for (; i < waypts.length; i+=9) {
    var src  = waypts[i].location;
    var dest = waypts[Math.min(i+9, waypts.length-1)].location;
    var way = waypts.slice(i+1,Math.min(i+9, waypts.length-1));

    directionsService.route({
        origin: src,
        destination: dest,
        waypoints: way,
        travelMode: google.maps.TravelMode.DRIVING
    }, function(response, status) {
        if (status === google.maps.DirectionsStatus.OK) {
            var polyline = []

            var legs = response.routes[0].legs;
            for (i=0;i<legs.length;i++) {
              var steps = legs[i].steps;
              for (j=0;j<steps.length;j++) {
                var nextSegment = steps[j].path;
                for (k=0;k<nextSegment.length;k++) {
                  polyline.push(nextSegment[k]);
                }
              }
            }

            cb(polyline);
        } else {
            window.alert('Directions request failed due to ' + status);
        }
    });
}
```

[15] Portal Google Javascript Directions API: https://developers.google.com/maps/documentation/javascript/directions

Fonte: Do autor.

Quando o usuário entra na opção de “Ronda policial” é solicitado a utilização do GPS do aparelho para que ele possa obter a sua localização atual, após isso, o usuário informa a distância da rota que deseja percorrer, como visto na Figura 31, e então o sistema irá verificar através de funções espaciais qual a área de atuação do seu batalhão e devolver as ocorrências encontradas com base nos acontecimentos de crimes de até 24 horas anteriores ou mostrar uma mensagem informando que não existem ocorrências para a distância informada.

Figura 31 - Solicitação de rota



Fonte: Do autor.

No lado do servidor, uma das funções geométricas utilizadas para obter as ocorrências a serem renderizadas no mapa foi a função *Contains*, vista na Seção 3.4.1, com está função é possível obter todas as ocorrências disponíveis na área de abrangência do batalhão do usuário, usando como parâmetro o polígono que define esta área, também usando a função *Distance* no *order by* do SQL estas ocorrências são ordenadas pela distância decrescente do ponto de partida da viatura. Mas, para que o cálculo da rota seja facilitado ao *front-end* da aplicação, o sistema percorre por todas as ocorrências verificando através da função *Distance* a distância de cada uma e retorna somente as ocorrências pertencentes a

distância solicitada pela viatura. O código responsável por realizar este controle pode ser visto na Figura 32. Também foi necessário utilizar uma coluna no BD para relacionar as ocorrências que já tinham sido geradas para um determinado usuário, ou seja, quando o usuário solicitasse uma rota as ocorrências que pertencessem a esta rota não ficassem disponíveis para outros usuários, neste caso, em nenhum momento será gerado uma mesma rota para duas viaturas.

Figura 32 - Consultas pra o cálculo de rotas

```
while (result <= (distance * 1000) && !ids.isEmpty()) {
  StringBuilder sb = new StringBuilder
  ("SELECT ST_AsGeojson(o.local) as jsonLocal, " +
   "       ST_AsText(o.local) fut, " +
   "       ST_Distance_Sphere(o.local,'SRID=4326;" + wktPoint + "'), " +
   "       o.id " +
   "FROM ocorrencias o WHERE ");
  sb.append(" o.id in( ").append(ids.toString()
                                    .substring(1, ids.toString().length() - 1))
                                    .append(") ");
  sb.append(" and o.local is not null ");
  sb.append(" and o.ref_usr_lock is null ");
  sb.append(" ORDER BY ST_Distance_Sphere(o.local,'SRID=4326;" + wktPoint + "') LIMIT 1");

  Object[] obj = (Object[]) entityManager.createNativeQuery(sb.toString())
                                         .getSingleResult();

  wktPoint = obj[1].toString();

  result += new Double(obj[2].toString());
  if(result <= (distance * 1000)) {
    Ocorrencia oco = new Ocorrencia();
    oco.setJsonLocal(obj[0].toString());
    results.add(oco);
  }

  for (Iterator<Integer> iterator = ids.iterator(); iterator.hasNext();) {
    Integer i = (Integer) iterator.next();
    if (new Integer(obj[3].toString()).equals(i)) {
      lockedOcorrencias.add(i);
      iterator.remove();
      break;
    }
  }
}

if(!results.isEmpty()){
  atualizaRespoOcorrencia(lockedOcorrencias, usuario);
}

return results;
```

Fonte: Do autor.

Ao testar o aplicativo para verificar o correto funcionamento da geração de rotas, a Figura 31 exibe a tela de funcionamento do aplicativo após o usuário solicitar uma rota de 120km, neste caso, o usuário encontra-se na cidade de Lajeado/RS e pertence ao batalhão que atende a região da zona sul de Porto Alegre/RS, logo, a rota sugerida ao usuário foi a exibida na Figura 33 a, b e c.

Figura 33 - Exemplo de geração de rota para usuário logado

Fonte: Do autor.

Na rota exibida para o usuário estão apenas as ocorrências que pertence ao seu batalhão e mostra a rota proposta somente com ocorrências da abrangência do seu batalhão, ou seja, ocorrências que estão fora da área da zona sul de Porto Alegre, não serão consideradas para o cálculo da rota. Todavia, ao solicitar uma nova rota a mesma será apresentada, salvo o caso em que um outro usuário do mesmo batalhão já tenha efetuado o bloqueio das ocorrências.

No Capítulo 6 serão apresentadas as considerações finais em relação ao trabalho e quais os trabalhos futuros que podem ser aplicados com base nesta solução.

7 CONSIDERAÇÕES FINAIS

Ao buscar a inovação na área da segurança pública e estudar a abrangência e modelos de trabalhos da polícia, sejam eles preventivos ou reativos, foi possível desenvolver satisfatoriamente este projeto e atingir o objetivo principal que é a criação de uma ferramenta de apoio a segurança pública para melhorar a vida das pessoas. Contudo, o projeto visa agora preencher a lacuna que existia antes do desenvolvimento do aplicativo.

A utilização de geotecnologias bem como a criação deste SIG utilizando um SGDB-geográfico, expos uma solução de arquitetura de sistema pouco utilizada, porém, a ferramenta PostgreSQL juntamente com o *plugin* PostGIS para trabalhos espaciais mostraram características usuais ao aplicar consultas espaciais diretamente nos pontos geográficos das ocorrências, sem a utilização desta arquitetura não seria possível o desenvolvimento deste projeto pois não existem dados geográficos no BD atual da Polícia. Para resolver este problema, foi necessário a extração dos endereços textuais e conversão para pontos geográficos para que o processo de manipulação destas informações fosse possível utilizando os recursos do PostGIS.

As funções geográficas do PostGIS mais abordadas no projeto foram:

- *Contains* (*geometry*, *geometry*): retorna 1 se a primeira geometria contiver completamente a segunda;
- *Within* (*geometry*, *geometry*): retorna 1 se a primeira geometria está completamente dentro da segunda;
- *Distance_Sphere* (*geometry*, *geometry*): retorna a distância esférica entre duas geometrias;
- *GeoJSON* (*geometry*): retorna a geometria em formato JSON.

- *GeoFromText* (*SRID*; *textGeometry*): retorna convertido um texto em formato geográfico;

Demais manipulações realizadas pelo BD PostgreSQL são inerentes ao uso habitual de um BD relacional.

Devido ao autor não conhecer o aparelho que iria ser utilizado dentro das viaturas com o aplicativo desenvolvido, foi necessário a busca por uma tecnologia que abrangesse todos ou os mais vendidos *Smartphones* do mercado para que a solução proposta fosse genérica e transparente. Desta forma, foi realmente surpreendente a capacidade da ferramenta Phonegap/Cordova com seu modelo de *plugins* que acessam os recursos disponíveis no aparelho e expõe as informações por meio de uma API Javascript tornando mais simples a manipulação de recursos de *hardware* do dispositivo em questão.

O Aplicativo Mapoli, que encontrasse publicamente no GitHub[16], consegue informar quais as áreas precisam de mais atenção pelos policiais através de mapas de calor e permite filtros por determinado fato da ocorrência, bairro do acontecimento e também um período específico, tudo isso funcionando *online* com informações precisas e atualizadas, ainda permite ao policial a geração de rotas informando a distância que deseja percorrer, assim, o sistema automaticamente encontrará a melhor rota dentro da área de abrangência do seu batalhão.

Contudo, verificou-se que o mapeamento criminal permite a polícia saber quais as regiões possuem maiores índices de criminalidades, sejam de menor ou maior significância, o aplicativo ajudará os batalhões da polícia no planejamento e direcionamento de seu efetivo para áreas específicas. Além disso, como futuras melhorias deste projeto vale ressaltar as seguintes informações:

a) Manter informações dos sentidos das ruas no banco de dados e não utilizar a API do Google para a geração de rotas. Devido a limitação do escopo e tempo deste projeto não foi incluído o mapeamento do sentido das ruas das cidades, visto que, é um trabalho muito exaustivo e consumiria um tempo excessivo para este trabalho. Entretanto, com essa informação seria possível efetuar a geração de rotas somente com a utilização do banco de dados geográfico e cálculos no back-

16 Códigos fontes no GitHub: https://github.com/rdanieli

end. Além de que, não seria necessário efetuar todos os controles as requisições ao Google e também não haveria custos para a organização ao implantar este sistema, visto que, caso se torne crescente a utilização do aplicativo é necessário pagar ao Google por estes serviços;

b) Propiciar filtros mais elaborados ao mapa de calor para todo o Rio Grande do Sul e disponibilizar o acesso para todas as instituições de segurança do Estado, incluindo órgãos ligados a ações populares de conscientização e cultura da não violência nas cidades;

c) Efetuar a navegação pela ronda elaborada pelo sistema;

d) Controlar para que a rota gerada passe somente uma vez por cada rua, caso seja necessário passar novamente para ir até outro acontecimento que ela pegue outra rua, mesmo não existindo ocorrências;

e) Integrar a uma interface administrativa web já existe na Brigada Militar, conhecida como SISP (Sistema integrado de Segurança Pública), para que seja realizada a manutenção dos dados geográficos e dos usuários da aplicação móvel;

f) Criar regras por permissões de acesso na aplicação.

Dessa forma, pode-se concluir que o presente trabalho cumpriu com os objetivos e o escopo proposto, bem como os estudos bibliográficos e a teorização dos conceitos abordados aplicados na prática.

REFERÊNCIAS

ADNAN, M.; SINGLETON, A. D.; LONGLEY, P. A. **Developing Efficient Web -based GIS Applications. Working Papers Series.** 2010. Disponível em <http://discovery.ucl.ac.uk/19247/1/19247.pdf>. Acesso em: 22 abr. 2015.

ANGULARJS, **Guide to AngularJS Documentation.** Google Inc. 2015. Disponível em <https://docs.angularjs.org/guide/>. Acesso em: 17 setembro 2015.

AZEVEDO, Marco A. de. **Concepts on criminality and policing models.** Psicol. cienc. prof., Brasília, v. 23, n. 3, p. 18-25, set. 2003. Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1414-98932003000300004&lng=pt&nrm=iso>. Acesso em: 01 maio 2015.

BONHAM-CARTER, Graeme F. **Geographic information systems for geoscientists: modelling with GIS. Pergamon**, Ontario, 1994. Disponível em <https://books.google.com.br>. Acesso em: 15 maio de 2015.

BRASIL. Constituição (1988). **Constituição da República Federativa do Brasil.** Disponível em <http://www.planalto.gov.br/ccivil_03/Constituicao/Constituiçao.htm>. Acesso em: 15 maio de 2015.

BRASIL. Lei Complementar nº 14, de 08 de junho de 1973. **Presidência da República.** Subchefia para Assuntos Jurídicos. Disponível em <http://www.planalto.gov.br/CCiVil_03/Leis/LCP/Lcp14.htm>. Acesso em: 01 maio de 2015.

BURKE, Bill. **RESTful Java with JAX-RS.** O'Reilly Media Inc, Sebastopol, 2010. Disponível em: <https://books.google.com.br/books?id=cFBptKRXrk4C>. Acesso em: 11 maio 2015.

CASANOVA, Marco A.; CÂMARA, Gilberto; JUNIOR, Clodoveu A. D.; VINHAS, Lúbia; QUEIROZ, Gilberto Ribeiro. **Bancos de Dados Geográficos.** Editora Mundogeo, Curitiba

– PR, 2005. Disponível em <http://www.inf.puc-rio.br/~casanova/Publications/Books/2005-BDG.pdf>. Acesso em: 10 abr. 2015.

CHAINEY, Spencer; RATCLIFFE, Jerry. **GIS and Crime Mapping.** John Wiley & Sons Inc, Chichester, 2005. Disponível em < http://www.wiley.com/WileyCDA/WileyTitle/productCd-0470860995.html>. Acesso em 19 setembro 2015.

CHAWDHARY**,** Zeeshan. **PostGIS QuickStart: Introduction to PostGIS.** 2014. Disponível em: <https://books.google.com.br/books?id=AgyAAwAAQBAJ>. Acesso em: 13 maio 2015.

CORRAL, Luis; SILLITTI, Alberto; SUCCI, Giancarlo. **Mobile multiplatform development: An experiment for performance analysis.** 9th International Conference on Mobile Web Information Systems (MobiWIS), University of Bozen-Bolzano, Bozen-Bolzano, Italy, 2011. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877050912004516>. Acesso em: 11 maio de 2015.

DA SILVA, Jorge. X. **O que é Geoprocessamento?** Revista do CREA-RJ, Rio de Janeiro, v.79, p. 42 – 44, out. 2009. Disponível em < http://www.ufrrj.br/lga/tiagomarino/artigos/oqueegeoprocessamento.pdf>. Acesso em: 10 maio 2015.

DAVIS, Clodoveu; NETTO, Gilberto C. **Introdução a ciência da geoinformação.** INPE, São José dos Campos, 2001. Disponível em <http://mtc-m12.sid.inpe.br/attachment.cgi/sid.inpe.br/sergio/2004/04.22.07.43/doc/publicacao.pdf>. Acesso em: 27 abr. de 2015.

FILHO, Jugurta L. **Projeto de Banco de Dados para Sistemas de Informação Geográfica.** 2000, Escola de Informática da SBC- Sul, Universidade Federal de Viçosa, Viçosa – MG. Disponível em <http://www.ufpa.br/sampaio/curso_de_sbd/semin_bd_para_sig/eri-norte.pdf> Acesso em: 5 mar. 2015.

FILOCRE, D'Aquino. **Classificações de políticas de segurança pública. Revista Brasileira de Segurança Pública.** v 3, n 5, 2009. Disponível em

<http://revista.forumseguranca.org.br/index.php/rbsp/article/viewFile/57/55> Acesso em: 01 maio de 2015.

FREDRICH, Todd. **RESTful Service Best Practices: Recommendations for Creating Web Services.** 2012. Disponível em: <http://www.restapitutorial.com/media/RESTful_Best_Practices-v1_1.pdf>. Acesso em: 11 maio 2015.

FREITAS, C. E. R.; VIEIRA, V. C. B. **Uso do geoprocessamento para auxiliar a segurança pública no mapeamento da criminalidade em Teresina – PI.** II Congresso de Pesquisa e Inovação da Rede Norte Nordeste de Educação Tecnológica, 2007, João Pessoa. Disponível em <http://www.redenet.edu.br/publicacoes/arquivos/20080922_104353_GEOM-017.pdf>. Acesso em: 15 maio 2015.

FURTADO, Vasco. **Tecnologia e Gestão da Informação em Segurança Pública.** Rio de Janeiro: Garamond. 2002.

GOOGLE, Inc. **Heatmap Layer Documentation.** . Disponível em <https://developers.google.com/maps/documentation/javascript/heatmaplayer>. Acesso em: 22 abr. 2015.

GRUBESIC, Tony H.; MURRAY, Alan T. **Assessing positional uncertainty in geocoded data.** 2004, Unversity of Cincinnati. Disponível em <http://www.tonygrubesic.net/geocode.pdf> Acesso em: 26 abr. 2015.

GÜTING, Ralf H.; SCHNEIDE, Markus. **Moving Objects Databases.** 2005, Morgan Kaufmann Publishers. Disponível em <http://dna.fernuni-hagen.de/Lehre-offen/Kurse/1675/KE1.pdf> Acesso em: 5 mar. 2015.

HILL, Linda L. Georeferencing: **The Geographic Associations of Information. The MIT Press.** 2006. Disponível em <http://mitpress.mit.edu/books/georeferencing>. Acesso em: 22 abr. 2015.

HOLLOWAY, Tomas H. **Polícia no Rio de Janeiro: repressão e resistência numa cidade do século XIX.** Rio de Janeiro: Editora Fundação Getúlio Vargas, 1997.

HUXHOLD, William E. **Information in the Organization and Applications of Urban Geographic Information Systems.** Chaps. 1 e 3 em An Introduction to Urban Geographic Information Systems. New York: Oxford University Press, 1991. Disponível em <http://ukcatalogue.oup.com/product/9780195065350.do>. Acesso em: 15 maio de 2015.

IONIC, **Ionic Documentation.** Drifty Co. 2015. Disponível em <http://ionicframework.com/docs/>. Acesso em: 17 setembro 2015.

JUCÁ, Roberta Laena Costa. **O papel da sociedade na política de segurança pública**. Jus Navigandi, Teresina, v. 7, n. 60, nov. 2002. Disponível em <http://jus.com.br/artigos/3525/o-papel-da-sociedade-na-politica-de-seguranca-publica> Acesso em: 01 maio de 2015.

KURNIAWAN, Budi. **Java para a web com servlets, JSP e EJB.** Rio de Janeiro: Ciência Moderna, 2002

MAVEN, **Maven Documentation.** The Apache Software Foundation. 2015. Disponível em <http://maven.apache.org/guides/>. Acesso em: 17 setembro 2015.

MÁXIMO, Alexandre A. M. **A importância do mapeamento da criminalidade utilizando-se tecnologia de sistema de informação geográfica para auxiliar a segurança pública no combate à violência.** 2004, Dissertação (Mestrado) – Programa de Pós-graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Centro Tecnológico, Florianópolis, 2004. Disponível em < http://repositorio.ufsc.br/xmlui/handle/123456789/86752>. Acesso em: 10 de maio de 2015.

METROPLAN. **Fundação Estadual de Planejamento Metropolitano e Regional.** Disponível em <http://www.metroplan.rs.gov.br/conteudo/1598/?A_Metroplan>. Acesso em: 01 maio de 2015

MILANI, André. **PostgreSQL - Guia do Programador.** Novatec, 2008. Disponível em <https://www.novatec.com.br/livros/postgre/capitulo9788575221570.pdf> Acesso em: 20 abr. 2015.

MOURA, Ana Clara Mourão. **Geoprocessamento na gestão e planejamento urbano.** 3. ed. Rio de Janeiro: Interciência, 2014.

OBE, Regina O.; HSU, Leo S. **PostGIS in Action.** 2011, Manning Publications. Disponível em <http://www.manning.com/obe/PostGISiA_sampleCh01.pdf> Acesso em: 5 mar. 2015.

PEIXOTO, Betânia Totino; LIMA, Renato Sérgio de; DURANTE, Marcelo Ottoni. **Metodologias e criminalidade violenta no Brasil.** São Paulo Perspec., São Paulo, v. 18, n. 1, p. 13-21, 2004. Disponível em <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0102-88392004000100003&lng=en&nrm=iso>. Acesso em: 01 maio 2015.

POSTGRESQL. **PostgreSQL: About.** 2015, Disponível em <http://www.postgresql.org/about/> Acesso em: 14 mar. 2015.

RAMAKRISHNAN, Raghu; GEHRKE, Johannes. Sistemas de gerenciamento de banco de dados. São Paulo: McGraw-Hill, 2008.

REULAND, Melissa M. **Information Management and Crime Analysis : Practitioners' Recipes for Success.** Police Executive Research Forum, Washington, DC, 1997.

RIO GRANDE DO SUL (Estado). Secretária de Segurança Pública (SSP). **Indicadores Criminais SSP-RS 2014.** 2014. Disponível em < http://www.ssp.rs.gov.br/?model=conteudo&menu=300>. Acesso em: 12 maio 2015.

SESHADRI, Shyam; GREEN, Brad. **Desenvolvendo com AngularJS: aumento de produtividade com aplicações web estruturadas.** Novatec, 2008. Disponível em <https://books.google.com.br/books?id=ZEgcBQAAQBAJ>. Acesso em: 20 abr. 2015.

SKABA, Daniel A. **Metodologias de Geocodificação dos Dados da Saúde.** 2009. Tese (Doutorado) – Escola Nacional de Saúde Pública Sergio Arouca, Fundação Oswaldo Cruz, Rio de Janeiro, 2010. Disponível em <http://bvssp.icict.fiocruz.br/pdf/Skabadad.pdf>. Acesso em: 26 abr. 2015.

URBANO, Ferdinando; CAGNACCI, Francesca. **Spatial Database for GPS Wildlife Tracking Data: A Pratical Guide to create a data management system with postgresql/postgis and R.** Springer International Publishing, Switzerland., 2014. Disponível em <http://www.springer.com/br/book/9783319037424>. Acesso em: 26 abr. 2015.