UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

GABRIEL HENRIQUE DA SILVA STEPIEN

Uma Aplicação para o Monitoramento da Disseminação do Mosquito *Aedes aegypti*

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Weverton Cordeiro
Co-orientador: Prof. Dra. Mariana Recamonde-Mendoza

Porto Alegre
2021

RESUMO

A dengue é uma doença que utiliza o mosquito *Aedes aegypti* como vetor de transmissão. Apesar de ter um padrão conhecido de condições que favorecem a multiplicação e disseminação do seu transmissor, ela continua fazendo milhares de vítimas a cada ano no mundo. Segundo o boletim epidemiológico Vol. 52 de Janeiro de 2021 publicado pela Secretaria de Vigilância em Saúde, foram registrados, em 2020, 987.173 possíveis casos da doença no território brasileiro até a semana epidemiológica 53. A utilização de ferramentas para a prevenção e identificação de possíveis surtos pode ser útil para entidades responsáveis tomarem decisões sobre campanhas de prevenção. Além disso, a centralização dos dados de identificações do mosquito pode estimular a população a tomar medidas preventivas de eliminação dos pontos focais em suas residências. Este trabalho de conclusão de curso propõe uma plataforma para agregar os dados de identificações do mosquito *Aedes aegypti* feitas a partir de dispositivos móveis e dados meteorológicos para gerar visualizações e *insights* com potencial de ajudar na prevenção e controle da proliferação dos mosquitos e, consequentemente, de doenças como a dengue. Será apresentada a modelagem de um sistema que tornará possível a integração de aplicativos que fazem a identificação do mosquito *Aedes aegypti* através de aprendizado de máquina. Por proporcionar uma integração em tempo real com os dados das identificações e pelas visualizações implementadas, o sistema desenvolvido neste trabalho mostra um potencial para ser utilizado como ferramenta principal no planejamento de ações de controle do mosquito. Também, foi validada a possibilidade de integração de um modelo preditivo para prever infestações do *A. aegypti* com base em identificações históricas e dados climáticos.

Palavras-chave: Plataforma de visualização. dengue. *Aedes aegypti*. monitoramento. controle. *crowdsourcing*. aprendizado de máquina. previsões.

ABSTRACT

dengue fever is a disease that uses *Aedes Aegypti* mosquito as its transmission vector. Although it has a known set of conditions that favor the multiplication and spread of its transmitter, it continues making millions of victims every year around the world. According to the epidemiological bulletin Vol. 52 of January 2021 published by *"Secretaria de Vigilância em Saúde"*, in 2020, 987.173 possible cases of dengue were registered in Brazilian's territory by the epidemiological week number 53. The use of tools for prevention and identification of possible outbreaks can be useful for entities that are responsible for making decisions about prevention and combat campaigns. In addition to that, centralizing mosquito identification can encourage the population to take preventive measures to eliminate focal points in their homes and neighborhood. This work proposes a system to aggregate data from *A. aegypti* mosquito identifications made by mobile devices and meteorological data to generate visualizations that can lead to insights with the potential to help in the prevention and control of dengue. The modeling of the system that will make it possible to integrate with devices that make the identifications will be presented. We also validated the integration of a predictive model to predict outbreaks of *A. aegypti* based on previous identifications and weather data.

Keywords: Visualization platform, dengue, *Aedes aegypti*, monitoring, control, crowding, spread, machine learning, prediction.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SMP	Symmetric Multi-Processor
NUMA	Non-Uniform Memory Access
SIMD	Single Instruction Multiple Data
SPMD	Single Program Multiple Data
ABNT	Associação Brasileira de Normas Técnicas
UI	User Interface
API	Aplication Programming Interface
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
MSE	Mean Squared Error
RNNs	Recurrent Neural Networks
CNNs	*Convolutional Neural Networks*
LSTM	*Long Short Term Memory*
SQL	*Structured Query Language*
OSI	*Open System Interconnection*
JPA	*Java Persistence API*
SGBD	*Sistema de Gerenciamento de Banco de Dados*
ER	*Entity Relationship*

SUMÁRIO

1 INTRODUÇÃO

Doenças transmitidas por vetores (como malária, dengue, chikungunya) são responsáveis por uma grande carga de mortes na população mundial, principalmente entre os menos favorecidos economicamente (WILSON et al., 2020). A principal forma de combate desse tipo de doença é por meio do controle do vetor.

A dengue é causada pelo vírus da dengue (DENV) que, por sua vez, utiliza o mosquito *Aedes aegypti* como vetor de transmissão. No Brasil, de acordo com o boletim da semana epidemiológica (SE) 53 de 2020 (ONLINE; NET, 2021), entre a SE 1 e SE 53 foram relatados 987.173 casos prováveis de dengue e confirmadas 554 mortes.

Atualmente, existem diversos esforços para combater o mosquito da dengue. Dentre eles, podemos citar a conscientização da população, aplicação de repelentes ambientais (inseticidas) ou vistoria de terrenos na procura de nascedouros. Além disso, existem cidades que criaram malhas de armadilhas que proporcionam dados sobre identificação do mosquito. Plataformas como a "Onde está o Aedes" (ECOVEC, 2021) são alimentadas com os dados dessas armadilhas através de um processo manual de coleta e registro dos dados, o que possibilita visualizações e um certo planejamento.

Como o processo manual existente exige o deslocamento dos agentes até as armadilhas para a realização da inspeção e contagem dos mosquitos, cobrir mais território exige um maior investimento em pessoas para realizar essa tarefa. Como monitorar a área de uma cidade inteira é um processo custoso, as leituras são feitas semanalmente. Perde-se, portanto, qualidade na informação das detecções dos mosquitos pois não há maneira de estimar em que dia o mosquito foi capturado desde a última verificação. O intuito deste trabalho é especificar e desenvolver uma ferramenta para facilitar o processamento de dados oriundos de dispositivos móveis capazes de identificar o mosquito *A. aegypti*. As identificações são consideradas como o resultado do esforço da implementação de técnicas já idealizadas em outros trabalhos (FERNANDES; CORDEIRO; RECAMONDE-MENDOZA, 2021; NUNES, 2020). O objetivo também é fornecer visualizações e previsões sobre a proliferação de mosquitos que poderão ser utilizadas pelos agentes responsáveis pelas ações de combate ao mosquito. No contexto de previsão de proliferação de mosquitos, este trabalho também desenvolve um modelo baseado em técnicas de aprendizado de máquina a fim de validar o uso do sistema para este propósito.

Foi desenvolvido um sistema composto por um *backend* em Java feito para ser executado na nuvem, *frontend* em Javascript e módulos para pré-processamento de dados

em Python. Além da comunicação e disposição dos dados necessários para a interface com o usuário, o *backend* expõe interfaces para a sua alimentação com dados sobre as identificações. O sistema mostra um potencial para facilitar a tomada de decisões com maior precisão, além de possibilitar a contribuição da comunidade para o combate à dengue.

O trabalho está dividido em nove capítulos, incluindo o presente de caráter introdutório. No Capítulo 2, a fundamentação teórica sobre as questões biológicas e métodos computacionais são introduzidas. No Capítulo 3 é discutida a bibliografia utilizada como referência para questões de monitoramento do mosquito *A. aegypti*. O Capítulo 4 tem como objetivo abordar o projeto de software considerado para o desenvolvimento do sistema protótipo. Na sequência, o Capítulo 5 é utilizado para apresentar a interface gráfica do sistema que se manifesta como uma plataforma web. Definições dos serviços de coleta e computação dos dados são feitas no Capítulo 6. O treinamento do modelo preditivo utilizado como prova de conceito é apresentado no Capítulo 7. Por fim, o Capítulo 8 traz as conclusões finais.

2 FUNDAMENTAÇÃO TEÓRICA

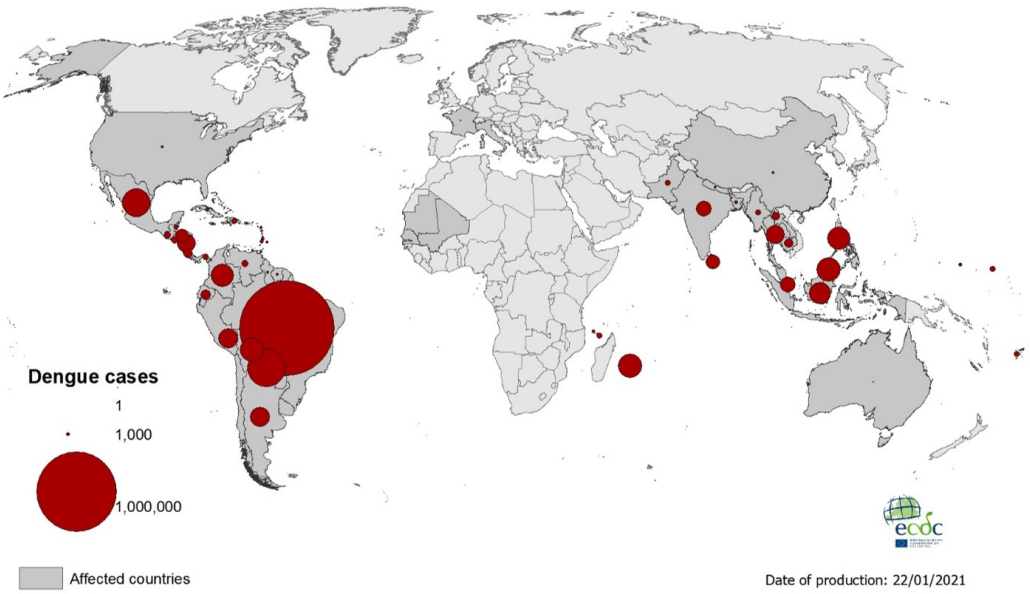
Este capítulo faz referência à fundamentação teórica necessária para o desenvolvimento do trabalho.

2.1 Dengue

Segundo a Organização Mundial da Saúde, a dengue é uma doença transmitida pela picada do mosquito *A.aegypti* infectado, chamado de vetor. Existem quatro sorotipos conhecidos do vírus responsável pela doença, são eles: DENV-1, DENV-2, DENV-3 e DENV4. A doença pode causar complicações na pessoa infectada, evoluindo o quadro para dengue severa que, por sua vez, é potencialmente letal.

Os números dessa doença são estimados em 50 a 100 milhões de casos no mundo inteiro por ano. Existem também os casos assintomáticos ou de diagnóstico errôneo. Existe um modelo que estima que há 390 milhões de casos de dengue por ano (BHATT et al., 2013). Para tomarmos a dimensão que a dengue representa para o mundo, basta observarmos o resultado do estudo que aponta que 3.9 bilhões de pessoas estão em regiões de risco ao contágio com o vírus da doença (BRADY et al., 2012).

Figura 2.1: Distribuição dos casos de dengue no mundo em 2020



Fonte: European Centre for Disease Prevention and Control

A Figura 2.1 mostra que a maioria dos casos estão localizados na América do

Sul, mais especificamente no Brasil, o que mostra a necessidade de tomarmos medidas de controle para a doença.

Quanto à sua transmissão, o vírus se encontra no sangue de um hospedeiro. Se uma fêmea do mosquito *A. aegypti* pica uma pessoa infectada, ela passará a ser uma transmissora que, ao se alimentar do sangue de uma pessoa sadia, poderá deixar o vírus no seu organismo. Portanto, existem dois fatores importantes que devem ser considerados para a propagação da dengue: mobilidade das pessoas infectadas e a população do mosquito *A. aegypti*. Ações direcionadas à trabalhar esses dois tópicos são as mais eficientes no controle da doença.

2.2 *Aedes aegypti*

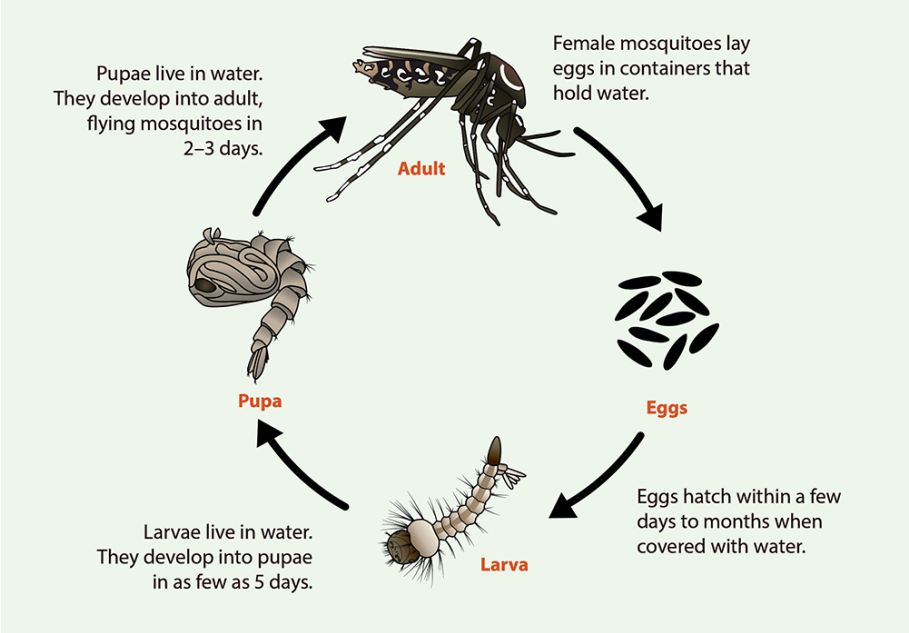
O mosquito *Aedes aegypti* além de ser vetor da dengue, também é responsável por transmitir as doenças chikungunya, zika e febre amarela. No contexto deste trabalho, é necessário entender um pouco sobre o ciclo de vida do mosquito e também sobre as condições favoráveis para a sua proliferação e mobilidade urbana.

O período de desenvolvimento de um ovo para um mosquito adulto é de aproximadamente 7 a 10 dias. Durante o seu ciclo de vida, o *Aedes aegypti* transita entre os estágios de ovo, larva, pupa e vida adulta (CONTROL; PREVENTION, 2020). Os ovos são colocados por uma fêmea da espécie em recipientes com água parada. A quantidade de água necessária para que esse processo aconteça é pequena, então qualquer descuido pode criar um ambiente favorável para a reprodução do mosquito. Os mosquitos podem permanecer no estágio de ovo por até 8 meses até encontrarem as condições favoráveis de temperatura para avançar para o próximo estágio.

Nos estágios de lava e pupa, os indivíduos vivem na água e podem ter suas atividades observadas. O mosquito adulto emerge da pupa e voa do ninho. As fêmeas da espécie precisam de sangue para gerar os ovos e iniciar o ciclo de reprodução. Os mosquitos utilizam humanos e animais como fonte de sangue. Após capturarem o sangue, as fêmeas procuram por uma fonte de água para colocarem os ovos. A Figura 2.2 retirada do CDC ilustra o ciclo de vida do mosquito.

O mosquito é comum em áreas urbanas, as infestações geralmente ocorrem em áreas com um grande volume populacional e com ocupações desordenadas. Nesse tipo de ambiente, as fêmeas tem mais oportunidades para se alimentarem, possuem locais para

Figura 2.2: Ciclo de vida do mosquito *Aedes aegypti*



Centers for Disease Control and Prevention

depositarem os ovos e usarem como criadouros.

A elevação das temperaturas e maiores taxas de chuva levam à um ambiente favorável para o desenvolvimento do mosquito, por esse motivo as infestações são mais intensas no verão. No entanto ações devem ser tomadas ao longo do ano para o controle do vetor e evitar infestações durante os períodos favoráveis. Como o habitat do mosquito é junto ao homem, a população deve ser engajada no seu combate.

2.3 Monitoramento e Controle

Nesta seção será discutido sobre o controle e monitoramento que são utilizados hoje em dia. O monitoramento e controle do mosquito *A. aegypti* é uma das principais formas de combate à doenças que ele é tido como vetor, como é o caso da dengue. Para a execução de medidas efetivas de combate, um dos pilares fundamentais a serem considerados é o monitoramento do mesmo. Um monitoramento preciso e sistemático, traria um aumento na efetividade das ações de combate e reduziria, consequentemente, os custos agregados do processo geral de controle.

O trabalho feito hoje em dia pelas autoridades é composto pela utilização de armadilhas com água parada para atrair a fêmea da espécie *A. aegypti* (Figura 2.3). O processo de verificação das armadilhas é rudimentar e manual onde os agentes precisam se deslo-

Figura 2.3: Armadilha de Monitoramento de Mosquito

Fonte: Prefeitura de Porto Alegre

car semanalmente até cada uma das armadilhas, coletar o material e realizar a limpeza do equipamento. O sistema como um todo tem um alto custo que compreende, além de outras aspectos, o combustível para o deslocamento dos agentes até cada uma das armadilhas e os seus salários. Além do custo, o sistema em si depende da ação humana em todas as etapas, o que pode resultar em algum erro inserido por falta de atenção. A automatização das etapas do sistema atual podem levar a redução de custos e erros, além de permitir a ampliação das áreas de monitoramento. Essa automatização é viabilizada pelos dispositivos que fazem a detecção por meio de técnicas de aprendizado de máquina como citado na Seção 3.1. Havendo a possibilidade de se ter os dados mais frequentes sobre as identificações, pode-se pensar na utilização de aprendizado de máquina para termos uma noção de previsões de infestações.

2.4 Aprendizado de Máquina

Nesta seção serão apresentadas os principais métodos computacionais que motivaram este trabalho a considerar a utilização de técnicas de aprendizado de máquina na abordagem do problema do controle e monitoramento do mosquito da espécie *A. aegypti*.

Aprendizado de máquina consiste em programar computadores para otimizarem

uma determinada métrica com base em exemplos do passada (ALPAYDIN, 2020). Nesse sentido, o recurso computacional é alimentado com exemplos utilizados para *treinar* um modelo capaz de inferir um algoritmo para realizar uma tarefa. Há uma geração de "conhecimento" através dos dados. Métodos de aprendizado de máquina podem ser considerados no contexto do monitoramento e no combate ao mosquito *A. aegypti* com o objetivo de melhorar a aplicação dos recursos para eliminar os focos de infestações. Por exemplo, se for possível ter um método que antecede onde seriam as regiões com maior número de mosquitos, então os agentes de saúde poderiam concentrar maiores esforços nessas regiões indicadas, o que aumentaria a efetividade das medidas tomadas.

Em aprendizado de máquina existe um conjunto de algoritmo que necessitam de instâncias de exemplo (dados rotulados), essa classe de aprendizado é chamada de aprendizado supervisionado. Os valores esperados são passados com o modelo juntamente com os valores de de entrada (as *features*). Assim, o algoritmo pode se adaptar durante as iterações para tentar se aproximar da função que mapeia as valorações das *features* de cada instância de treinamento para o valor de saída dada como exemplo. Essa tarefa é feita através da otimização de uma determinada função de avaliação, geralmente representada pelo erro onde busca-se minimizar o erro global. Na Seção 2.4.2 entramos em mais detalhes sobre o processo de validação de modelos de regressão.

2.4.1 *Recurrent Neural Networks*

Recurrent neural networks (RNN) (RUMELHART; HINTON; WILLIAMS, 1985) é um tipo de arquitetura de redes neurais que permite trabalhar com séries temporais. Podem ser usadas no contexto de processamento de áudio, discurso, detecção de anomalias e também para se trabalhar com series temporais históricas. No entanto, o contexto da informação pode estar distante em relação ao que estamos tentando prever, por esta razão, RNNs podem ser ineficientes ao se trabalhar com dados que podem depender de informações que estão em um passado distante.

Long short-term memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) é um tipo especial de arquitetura RNNs (*recurrent neural network*) projetada para modelar series temporais e suas relações a longo prazo com maior precisão. Com a utilização desse tipo de arquitetura, pode-se carregar informações do contexto de longo prazo, assim, dados que estão em um passado distante podem ser considerados nas predições, o que deixa o modelo mais eficiente no tratamento de dados históricos.

Podem existir diferentes camadas em uma rede neural, uma delas é a camada do tipo *dense* que é completamente conectada com todos os neurônios da camada anterior. Outro tipo de camada é a de *dropout* que serve para, de forma aleatória, desconsiderar alguns nodos durante o treinamento, essa abordagem serve para reduzir o *overfitting* da rede que é discutido na Seção 2.4.3.

2.4.2 Validação de Problemas de Regressão

Problemas de regressão precisam ser resolvidos por modelos que tentam explicar funções que mapeiam um conjunto de valores de determinadas *features* para um valor numérico. O problema estudado neste trabalho possui esta característica ao passo que tenta prever um número de identificações do futuro como explicado no Capítulo 7. Existem formas especificas para se validar esse tipo de modelo, elas variam de acordo com a função de erro utilizada.

A função de erro torna possível quantificar o quanto os dados preditos pelo modelo estão distantes dos dados reais sendo utilizados no treinamento/validação. Uma forma de se medir o erro é pela função do erro médio quadrático, do inglês *mean squared error* (MSE), essa medida representa o quanto uma determinada regressão está próxima de um conjunto de pontos, sua equação é $MSE=\frac{1}{n}\sum_{t=1}^{n} e_t^2$ onde n representa o número de pontos e e representa o erro de um determinado ponto t em n, quanto maior o número, mais distantes estão os pontos preditos e reais.

Durante o processo de treinamento, parte dos dados são separados para validação. O conjunto de validação é utilizado para verificação de como o modelo está performando para os dados gerais, ou seja, suas instâncias são como novas instâncias a serem apresentadas para o modelo. A função de erro é aplicada sobre o resultado da execução do modelo com os dados de validação resultando em uma estimativa mais assertiva de como seria a sua performance instâncias reais em uma tarefa de predição.

2.4.3 Overfitting

No contexto de aprendizado de máquina, o *overfitting* é um fenômeno que acontece quando o modelo resultante do treinamento consegue explicar bem os dados utilizados durante o treinamento, porém sua performance é baixa para novos dados. Ele

geralmente acontece quando os dados de treinamento não são suficientes para explicar as características dos dados reais. Esse fenômeno foi encontrado na base de dados utilizada para estudar o problema e que utilizamos para a concepção de um modelo como citado na Seção 7.7.3.

2.4.4 Visualização em Mapa

O trabalho faz uso de alguns componentes que proporcionam visualizações em mapa. São eles: marcadores, polígonos e *heatmaps*. Todos eles utilizam pontos geográficos (latitude e longitude) para que possam ser renderizados na região de interesse do mapa. Marcadores são utilizados para destacar pontos de coordenada de forma isolada, assim pode-se destacar uma região onde ocorreu um evento específico onde se sabe a posição exata da coordenada. Um polígono é composto por uma lista de coordenadas que são interligadas de acordo com a ordem dos pontos, o seu objetivo é destacar uma região de interesse conhecida que está dentro das fronteiras da região construída. Já *heatmaps* são compostos também são compostos por simples pontos no mapa, porém possuem um raio para destacar uma área em torno da coordenada informada.

3 TRABALHOS RELACIONADOS

Este capítulo tem como objetivo elencar os trabalhos relacionados que motivaram a concepção deste projeto. O sistema desenvolvido possui dependências de algumas metodologias e processos desenvolvidos em referências que aqui são apresentadas.

3.1 Identificação do Mosquito

A sub-notificação das identificações do mosquito é um dos desafios encontrados por quem faz o monitoramento desse vetor. Trabalhos como o de Arthur et al. (2014) realizam estudos sobre o bater de asas de indivíduos machos e fêmeas do mosquito utilizando gravações. Esse tipo de estudo permitiu observar padrões de frequências e de comportamento que se mostraram úteis para o processo de automatização das detecções. Tais observações serviram de base para outros trabalhos que motivaram o desenvolvimento do presente trabalho.

A partir da observação de padrões na frequência do bater de asas do mosquito, o trabalho de Fernandes, Cordeiro and Recamonde-Mendoza (2021) estuda a utilização de *convolutional neural networks* (CNNs) para a realização da detecção do *Aedes aegypti* a partir de amostras de som obtidas com dispositivos móveis. Esse trabalho compara diferentes tipos de redes para a tarefa de identificação do mosquito *A. aegypti* e sugere uma arquitetura capaz de fazer a identificação com um certo nível de acurácia.

Além de métodos que utilizam o som do bater de asas do mosquito para fazer a identificação, existem estudos que se baseiam na imagem do mosquito. Um exemplo é o trabalho conduzido por Ong et al. (2021) que idealiza um modelo que utiliza CNNs para a realização da classificação dos mosquitos do tipo *Aedes aegypti*. Os resultados chegaram a uma acurácia de 98%, o que se aproxima da efetividade dos especialistas responsáveis por fazer esse tipo de classificação manualmente.

Nunes (2020) implementa uma arquitetura de rede neural em dispositivos Android, tornando possível a identificação de mosquitos em tempo real a partir do som. Tal implementação da rede neural em dispositivos móveis valida a possibilidade de outros dispositivos, como armadilhas autônomas com microfones acoplados, a desempenharem essa tarefa. Se esse tipo de dispositivo tiver uma interface com a internet e implementarem algum protocolo de comunicação como HTTP, é possível criar um sistema que não tenha a necessidade da interação humana para a contagem e registro das identificações.

O que potencialmente reduziria o custo de manter programas de controle do vetor e possibilitaria o aumento da área monitorada, fazendo com o que os dados das identificações sejam mais confiáveis e possam contribuir para mais estudos de controle de infestações.

3.2 Monitoramento da Proliferação

No trabalho de Li and Chen (2016), foi estudada a difusão do mosquito utilizando dados de microrregiões. Nesse estudo foram utilizadas variáveis climáticas para identificar os principais fatores que elevam o risco de surtos de dengue e aumento da população do mosquito. Além de variáveis climáticas, também foram adicionados índices utilizados para classificar a condição sócio econômica da região. Na análise das variáveis, os resultados desse estudo mostraram que as principais variáveis para a classificação do risco são as relacionadas à dados socio econômicos. Em uma análise feita nas correlações entre os fatores estudados a quantidade de casos de dengue, a variável climática que mais teve correlação com o número de casos de dengue foi temperatura.

O trabalho de Luza et al. (2021) utiliza um modelo probabilísticos e se baseia nos dados dos boletins epidemiológicos para estimar a quantidade de detecções do mosquito *Aedes aegypty* em cidades do Rio Grande do Sul. O modelo conseguiu explicar os dados conforme o relatórios oficiais apenas para duas regiões do estado (oeste e noroeste). O trabalho de Bee, Lye and Yean (2009) estuda os impactos das mudanças climáticas que estão acontecendo a nível global e sugere que haverá mudanças significativas na distribuição da população do mosquito, fazendo com que o *A. aegypti* se prolifere por regiões que antes eram consideradas com livre de risco de infestações.

O trabalho de Mores et al. (2020) investiga o padrão sazonal da população do mosquito em uma cidade subtropical brasileira para inferir quais seriam as melhores épocas do ano para a execução de medidas de controle para o mosquito. Seus resultados mostram que as medidas de controle são mais eficazes em períodos como outono e início do inverto, momento em que a população do mosquito se encontra em declínio por conta do padrão climático.

3.2.1 Pontos em Aberto

Durante a revisão da literatura, não foram encontradas definições de sistemas projetados para trabalhar com uma nova família de dispositivos inteligentes capazes de detectar o *Aedes aegypty* de forma automatizada que são viabilizados pelos trabalhos citados na Subseção 3.1. Este trabalho tenta estabelecer um processo para considerar os dados desses novos dispositivos e assim, de forma dinâmica, computa-los para gerar informação que será disponibilizada para os usuários que tomarão as decisões referentes à políticas públicas de controle do mosquito.

As referências cidades na Subseção 3.2 reforçam uma das hipóteses que fundamentam este trabalho: o uso de variáveis climáticas pode contribuir para a predição de ocorrências do mosquito ou, pelo menos, apontar regiões propensas à proliferação. Este aspecto será discutido no Capítulo 7 onde o modelo de aprendizado de máquina for apresentado. Como contribuição, este trabalho também tenta propor uma metodologia para a manutenção dos dados de variáveis climáticas que serão utilizadas na computação das predições do número de mosquitos *A. aegypti*.

4 PROJETO E DESENVOLVIMENTO DO SISTEMA

Neste capitulo será apresentado o usuário final da plataforma, além disso será discutido sobre o mapeamento do problema que estamos abordando. Serão abordadas as decisões arquiteturais e as *features* que compõe o sistema. A última seção aborda a modelagem do banco de dados.

Para desenvolver uma ferramenta capaz de auxiliar no combate da proliferação do mosquito *A. aegypti*, o presente trabalho se propõe a definir um processo automatizado de coleta de dados necessários para a elaboração de visualizações que potencialmente ajudarão os agentes competentes. O sistema desenvolvido também tem o o potencial de ser uma plataforma capaz de ser integrada com diferentes modelos de previsão que trarão valores de estimativas de números de mosquitos para as regiões, tornando possível a implementação de mecanismos gráficos para chamar a atenção dos agentes responsáveis para as regiões com maiores chances de aparição do mosquito. O tabalho aborda a concepção do modelo preditivo na Capítulo 7, a criação de novos modelos para serem integrados com a plataforma é discutida na seção 7.7.5.

4.1 Usuário

Ao passo que estamos elaborando uma ferramenta com a finalidade de servir como um auxílio para uma tarefa cotidiana de um indivíduo, precisamos entender um pouco melhor quem seria o nosso usuário. Para isso, conversamos com o secretário adjunto da Secretaria de Estado de Inovação, Ciência e Tecnologia, José Fernando César de Mattos. Nessa reunião, tentamos achar formas de conseguir mais informações para fundar o desenvolvimento do trabalho, assim como buscar colaboração para o mesmo.

O usuário final para a plataforma foi pensado como sendo os agentes públicos responsáveis por fazer o monitoramento de doenças transmissíveis por vetores. No caso de Porto Alegre, o monitoramento dos focos do mosquito *A. aegypti* é feito pela Secretaria Municipal de Saúde em parceria com a empresa mineira Ecovec.

O objetivo é facilitar o acesso às informações de identificações e fornecer previsões para os agentes de monitoramento de saúde elaborarem as medidas de prevenção e controle do mosquito para assim minimizarem os caso de dengue e outras doenças transmitidas pelo mosquito. Esse trabalho será facilitado com a possibilidade de integração da plataforma com armadilhas automatizadas.

4.2 Mapeamento do Problema

Nessa seção é discutido o problema principal que o trabalho aborda. Iremos fazer um mapeamento dos tópicos importante através de uma ferramenta chamada *mind-map*. A seção começa com a explicação sobre a ferramenta e, na sequência, a utilizamos no contexto do projeto.

4.2.1 Mind-Maps

Uma das formas que de se entender o problema principal que um projeto de software irá resolver é a utilização de *mind maps*. O uso de *mind maps* enriquece a fase inicial dos projetos por ser uma ferramenta de visualização e, principalmente, que proporciona trocas e discussões sobre qual é o problema a ser resolvido e instiga o levantamento de ideias sobre como resolvê-lo. O tópico principal a ser discutido é colocado no centro do diagrama. Tópicos que contribuem com o problema ou que podem ser importantes para a resolução do mesmo são adicionados como extensões dos tópicos mais gerais conforme o contexto. Assim, o problema inicial a ser resolvido é a raiz do diagrama que pode ramificar para diferentes tópicos folhas mais específicos.

4.2.2 Elaboração de um Mind-Map para o Problema da Dengue

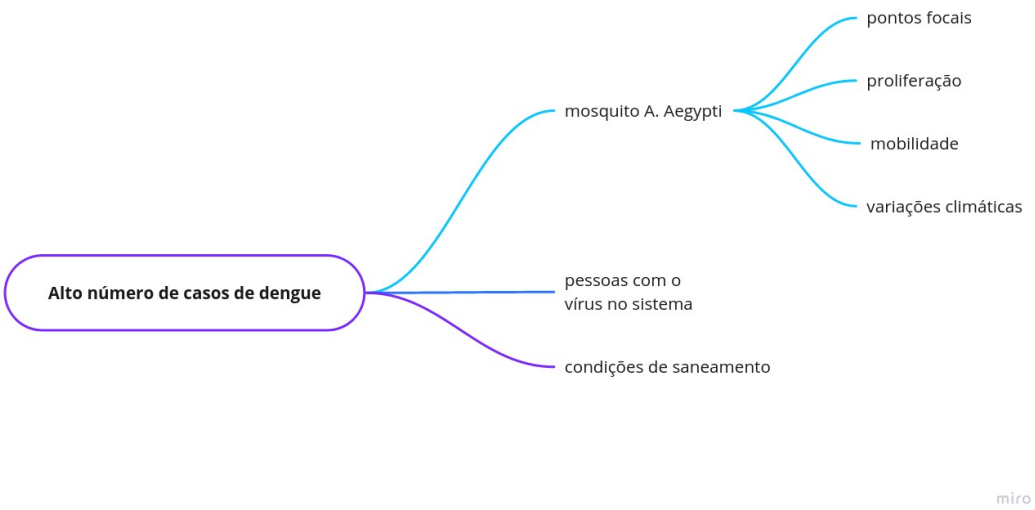
No contexto do trabalho, o problema principal que está sendo abordado é o "alto número de casos de dengue". Mais especificamente na cidade de Porto Alegre, região alvo para o desenvolvimento inicial do trabalho, mas a aplicação poderá ser facilmente utilizada em diferentes regiões. Com isso foi criado um tópico em destaque no diagrama que faz referência a esse problema. De acordo com o que indica o Ministério da Saúde (2009), a principal causa do aumento dos números da dengue é a baixa contenção do mosquito *A. aegypti*, o que facilita a circulação do mosquito e, por consequência, a contaminação de mais pessoas. Portanto, adicionamos uma extensão do problema principal chamada "mosquito *A. aegypti*".

Existem outros quatro tópicos referentes ao mosquito que podem, diretamente, colaborar com o número de casos de dengue. São eles: pontos focais, proliferação, mobilidade e variações climáticas. Pontos focais, pois onde há ninhos do mosquito, haverá

maior número de indivíduos da espécie. Proliferação e mobilidade tem a ver com como o mosquito se reproduz e até onde um indivíduo pode alcançar a partir do seu ninho. Por fim, o uso de variáveis climáticas é motivado por estudos anteriores que indicam que a quantidade do mosquito em uma determinada região pode estar ligada às variações de temperatura, períodos de chuva etc, conforme discutido no capítulo de trabalhos relacionados. Esses tópicos estão ligados com a quantidade de indivíduos do mosquito e são características que podem ser quantificadas.

A Figura 4.1 é o resultado do exercício no contexto do projeto.

Figura 4.1: Mind-Map do Problema



Fonte: O autor

4.2.3 Funcionalidades

Um subconjunto de funcionalidades foi idealizado para alcançar o objetivo do sistema e atacar o problema principal descoberto no processo da seção anterior. As funcionalidades foram levantadas a partir da observação de outros sistemas como o (ECOVEC, 2021) e da tentativa de se adicionar a possibilidade de se ter predições do *A. agypti* em uma plataforma automatizada. Abaixo são listadas das funcionalidades principais que são alcançadas com a soma de todos os módulos que serão apresentados nas seções seguintes.

Requisito 1 *Dispositivos móveis devem ser capazes de informar o sistema quando um mosquito* A. aegypti *for identificado.*

Essa funcionalidade diz respeito à porta de entrada dos dados no sistema. Armadilhas automatizadas, dispositivos móveis, e até mesmo algum outro sistema onde agentes, de forma manual, cadastram as identificações, poderiam ser usados. Com a inserção dos dados de identificações, parte dos dados necessários para as funcionalidades já estariam disponíveis para serem trabalhados.

Requisito 2 *O usuário do sistema deve poder visualizar as identificações em um mapa que representa a área da região de interesse.*

Ao visualizar as identificações em um mapa para a região de interesse, o usuário terá a capacidade de ver um panorama em tempo real, o que de antemão já ajuda para tomadas de decisões.

Requisito 3 *Os bairros serão representados por polígonos no mapa.*

A região de interesse utilizada para o contexto desse trabalho é a cidade de Porto Alegre. As sub-regiões de interesse para as visualizações são os bairros.

Requisito 4 *O usuário deve poder selecionar um período desejado para ver as identificações, gráfico e ranking.*

Como, por vezes, o agente pode desejar visualizar como estava a situação geral das identificações no mapa, a capacidade de informar um período específico para definir as visualizações ajuda na tarefa de comparação de períodos e auxilia na identificação de padrões.

Requisito 5 *O usuário deve poder ver um ranking com os bairros mais afetados ordenados de forma decrescente por número de identificações em um determinado período*

Utilizando o período selecionado pelo usuário, a capacidade de ver o ranking dos bairros ajuda na priorização de ações de combate do mosquito.

Requisito 6 *O usuário deve poder ver um gráfico mostrando a quantidade de identificações do mosquito por dia para um determinado período*

A visualização histórica dos dados em formato de gráfico facilita o processo de avaliação de ações e políticas públicas para minimizar o número de mosquitos *A. aegypti*, da mesma forma que também ajuda a criar um senso de necessidade de tomada de ação quando há aumentos de identificações com relação a dias anteriores.

Requisito 7 *As identificações devem ser renderizadas no mapa como forma de marcadores e terão uma opacidade inversamente proporcional ao número de dias desde o início do período selecionado*

Para trazer uma semântica de tempo para as identificações visíveis no mapa, a utilização do recurso de opacidade se mostrou satisfatório para mostrar as identifi-

cações. Assim as mais antigas são menos aparentes e as mais recentes são mais destacadas no mapa.

Requisito 8 *O usuário poderá habilitar o mapa de calor das identificações*

Quando o usuário habilita o mapa de calor das identificações, há uma certa interpolação das identificações em um raio de trezentos metros. Dessa forma, as identificações podem sugerir um rastro no mapa da região de interesse.

Requisito 9 *Ao clicar em uma identificação, o usuário poderá ver as informações do endereço, assim como a data da identificação*

Ao ser capaz de ver o endereço de onde ocorreu uma identificação, os agentes tem a capacidade de enviar unidades de inspeção para checar a área e eliminar qualquer possível foco nas proximidades.

Requisito 10 *Haverão duas formas de visualização das informações: históricas e a visualização preditiva*

O usuário poderá alternar entre dois modos para ver dois tipos de informações: históricas e preditivas.

Requisito 11 *A visualização histórica mostra as identificações no período selecionado*

A visualização histórica mostra todos os dados referentes às identificações conforme listado nas funcionalidades acima.

Requisito 12 *Na visualização preditiva, o usuário pode selecionar uma data para previsão, o sistema irá apresentar uma estimativa de risco para os bairros que serão coloridos de acordo com o risco*

As estimativas de risco são basicamente diferentes colorações de acordo com o número de mosquitos identificados em cada área.

Requisito 13 *O usuário poderá ver as chunks no mapa*

Chunks são micro-regiões quadradas de lado igual a trezentos metros. As *chunks* são utilizadas para agrupar as identificações e gerar as predições com base nas informações mais bem definidas.

Requisito 14 *O sistema deve ser alimentado todos os dias com as informações do tempo de forma automatizada*

Com a alimentação do sistema de forma automatizada, os cálculos de predições poderão acontecer periodicamente sem a necessidade de interferência manual.

Requisito 15 *O usuário deve poder procurar por bairros e ver as informações agregadas, tanto históricas como de previsão*

Uma tela que possibilita o usuário a selecionar um bairro desejado e visualizar as informação de risco com base nas predições para os próximos dias, e também ver o histórico para os últimos dias, poderá facilitar na checagem do estado de cada bairro.

Requisito 16 *O usuário deve poder ver a lista de chunks de cada bairro em uma tabela que mostra o endereço, predições e histórico de forma isolada*

Com uma visão detalhada das *chunks* de um determinado bairro, o usuário poderá selecionar as áreas dos bairros que mais exigem atenção e deliberar que alguma ação seja tomada na região com uma maior assertividade. Ao clicar em uma determinada *chunk* que exige atenção, usuário será redirecionado para o mapa que focará no elemento selecionado. Uma coluna importante para a tabela é o endereço do centroide da *chunk*.

Requisito 17 *O usuário deve poder cadastra armadilhas de forma manual*

Armadilhas que já são utilizadas para o controle do mosquito precisam ser suportadas para que haja uma continuidade no trabalho que é feito atualmente. Com isso, o usuário poderá adicionar uma armadilha dando o nome e as coordenadas geográficas para a mesma.

Requisito 18 *o usuário poderá inserir identificações de forma manual para as armadilhas*

Como o processo de coleta e contagem dos mosquitos capturados pelas armadilhas que hoje em dia são utilizadas ainda é manual, o usuário deve ser capaz de informar a quantidade de mosquitos e a data da coleta para que o nosso sistema processe o número de identificações para a região e distribua a quantidade entre os dias desde a última coleta.

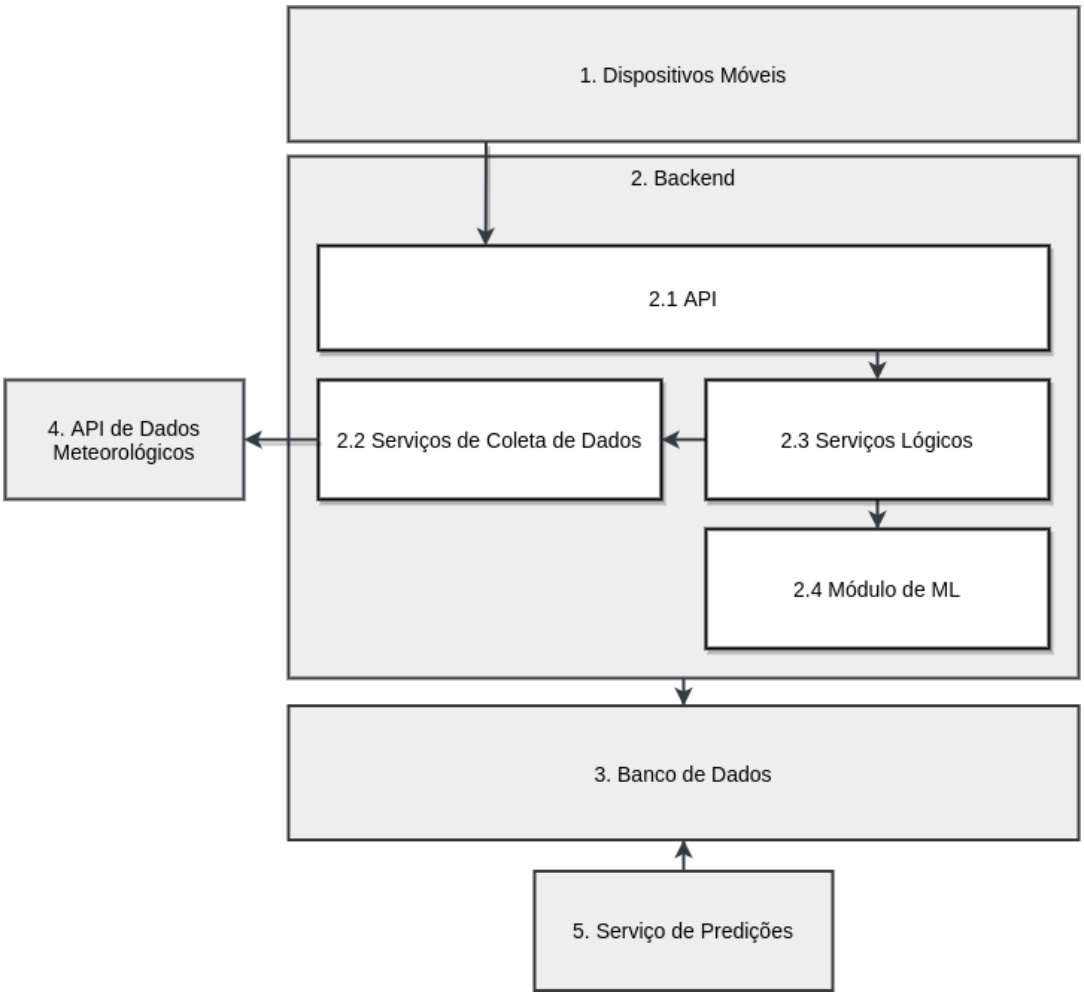
Esse conjunto de funcionalidades foi utilizado para nortear o desenvolvimento do trabalho. Nas próximas seções, o processo de construção do sistema para cobrir essas funcionalidade será melhor descrito.

4.3 Arquitetura do sistema

Nessa seção será apresentado o modelo arquitetural do sistema. Definimos o conjunto de módulos fundamentais para a cobertura dos requisitos e a integração entre eles.

Para o contexto do trabalho, foi pensada uma arquitetura representada na Figura

Figura 4.2: Arquitetura do sistema e dependências entre módulos



Fonte: O autor

4.2. PostgreSQL foi utilizado como SGBD (Sistema de gerenciamento de banco de dados), ele é um serviço a parte da aplicação no qual o *backend* depende. Todos os serviços para a lógica da aplicação estão no *backend* e são expostos por APIs para os dispositivos móveis e para a plataforma. Os dispositivos móveis compreendem os dispositivos que vão interagir com o sistema, alimentando-o com os dados sobre as identificações. Existe também um componente de software que compreende plataforma administrativa para o acesso às funcionalidades. Há um serviço que integra com uma API que fornece dados meteorológicos que serão utilizados para as estimativas de risco de infestações.

Outro módulo importante para o sistema é um micro-serviço Python feito para o cálculo e armazenar as predições no banco de dados. Ele expõe uma API *"/predict"* que recebe uma data e o número de dias para o futuro que gostaríamos de fazer a predição. O resultado da chamada é a inserção das predições de todas as *chunks* no banco de dados para um determinado dia.

4.4 Banco de Dados

O banco de dados utilizado para a elaboração do trabalho foi o PostgreSQL. Nessa seção é abordada a concepção das tabelas e de suas relações. Também é apresentado o *framework* utilizado para fazer o mapeamento das entidades do sistema para a estrutura do banco de dados.

4.4.1 *Framework* de Acesso ao Banco - Java

O banco de dados da aplicação foi contruido baseado em um *framework* em Java que cria as entidades no banco de dados relacional de forma automática a partir das classes Modelo que definimos no código. Tais classes modelo possuem seu atributos e métodos que são identificados pelo JPA (*Java Persistence API*) e utilizados no seu funcionamento. O Spring JPA incorpora a aplicação rotinas que, ao inicializarmos a aplicação, as tabelas são criadas e as alterações nas estruturas também são aplicadas no banco.

Existem interfaces utilizadas para o acesso das informações das tabelas e também para executarmos operações sobre a base de dados. Podemos ver um exemplo desse tipo de interface na Listagem 1. Nesse exemplo de interface de um repositório, são apresentados duas formas de executarmos as os *statements*: através de métodos seguindo um padrão de escrita nas assinaturas (como em *findAllByLocationId* ou *findAllByTimeBetween*), ou por meio da escrita dos *statements* em SQL (como em *findAllByLocationIdAndTimeBetween*). O presente trabalho utilizou a primeira forma para as operações mais simples. Para aquilo que envolvia alguma operação mais complexa, foi optado por fazer definir o SQL de forma manual no código.

4.4.2 Estrutura da Base de Dados

O banco de dados utilizado pela sistema, tabelas que guardam dados de quatro tipos de categoria:

Dados Geográficos: característicos por estarem armazenando informações sobre pontos geográficos, polígonos e localidades.

Tabelas Históricas: armazenam informações sobre as variáveis utilizadas para o controle do mosquito.

Listing 1: Exemplo API de um Repositório

```
1  public interface IdentificationRepository extends
    ↪  CrudRepository<Identification, String>{
2
3      String GET_ALL_BY_LOCATION_AND_DATE_RANGE =
4      """
5          select distinct
6              ident.*
7          from
8              locations l inner join
9              chunks_intersects ci on (l.code =
    ↪  ci.intersection) inner join
10             chunks c on (c.id = ci.chunks_id) inner
    ↪  join
11             identifications ident on (ident.grid_line
    ↪  = c.grid_line and ident.grid_column =
    ↪  c.grid_column)
12         where
13             l.code = :locationCode and
14             ident.time between :startDate and
    ↪  :endDate
15     """;
16     //...
17     Collection<Identification>
        ↪  findAllByLocationId(String locationId);
18     Collection<Identification> findAllByTimeBetween(
19         Date dateStart,
20         Date dateEnd
21     );
22     @Query(value =
        ↪  GET_ALL_BY_LOCATION_AND_DATE_RANGE,
        ↪  nativeQuery = true)
23     Collection<Identification>
        ↪  findAllByLocationIdAndTimeBetween(
24             @Param("locationCode") Integer
                ↪  locationCode,
25             @Param("startDate") Date startDate,
26             @Param("endDate") Date endDate
27     );
28     //...
29 }
```

Tabelas Transacionais: utilizadas para armazenar as informações sobre as entidades do sistema.

Tabelas Relacionais: guardam as informações sobre o relacionamento entre entidades.

A seguir, são descritas as tabelas do sistema. Falaremos o motivo delas existirem e também discutiremos os seus relacionamentos. A Figura 4.3 apresenta o diagrama ER (*Entity Relationship*) para o estado final do banco de dados.

chunks: guarda as informações das microrregiões utilizadas para discretizar a área da localidade que desejamos monitorar. Nela armazenamos qual é a sua posição no *grid* de microrregiões que cobre a área monitorada (*grid_column* e *grid_line*), *centroid_id* é chave estrangeira para a tabela de *geo_points* a fim de podermos saber qual é o ponto que constitui o contro da microrregião. Já *bottom_left_id*, *bottom_right_id*, *top_left_id*, *top_right_id* também são chave estrangeira para a tabela *geo_points*, porém representam as quatro arestas da microrregião.

Essa entidade é central para o sistema, pois é sobre ela que que aconteceram os cálculos de predições, e cada uma delas possui identificações relacionadas. Uma *chunk* pode possuir N identificações, predições e informações sobre o clima. A representação de quais são as localidades que uma *chunk* pode intersectar são armazenadas na tabela *chunks_intersects*.

locations: os bairros são armazenados nessa tabela. Uma localidade possui um código (*code*) utilizado para a lógica de cálculo das intersecções. O nome da localidade é armazenado na coluna *neighborhood*. Assim como nas *chunks*, as localidades também possuem, tal registro é armazenado na coluna *centroid_id* que faz referência para a tabela *geo_points*. Uma localidade pode ter N polígonos para representar a sua área, com isso há um relacionamento para a tabela de *locations_polygon*.

identifications: guarda os registros das identificações que foram inseridas pelos dispositivos. Podemos considerar os registros de identificações como dados históricos. Funcionam como uma espécie de *log* que podem ser agregados de acordo com a consulta que está sendo executada. Assim como as *chunks*, a tabela de identificações também possui duas colunas para o controle de sua posição no *grid* que cobre a região de interesse (*grid_column e grid_line*). As coordenadas latitude (*lat*) e longitude (*lng*) correspondem ao local onde a identificação foi feita. O serviço de inserção de uma identificação calcula para qual localidade e qual microrregião (*chunk*) a identificação está localizada (*location_id* e *chunk_id*). Por fim, para po-

dermos ter uma noção de quando aconteceu a identificação, foi adicionada uma coluna chamada *time* e tornar possível a lógica de selecionar apenas os registros importantes para cada funcionalidade. Uma identificação terá, necessariamente, microrregião associada. A coluna *chunk_id*, já citada, é chave estrangeira para o id do registro da tabela *chunks*.

prediction: armazena as informações das predições computadas pelo sistema. Não há a necessidade de se calcular uma coordenada específica para cada predição, ao invés disso, armazenamos o número de mosquitos que o modelo previu para determinada microrregião (*chunk_id* - chave estrangeira para a tabela *chunks*) e um determinado dia *date*. Essa tabela difere das demais no sentido de que sua chave primária é composta por duas colunas (*date* e *chunk_id*) e não por uma coluna específica tal como utilizamos o id como identificador único.

weather: é a tabela que armazena os dados coletados dos serviços meteorológicos. Nela vamos encontrar os valores de umidade, pressão, temperatura e chuva. Os valores das variáveis climáticas são coletados de forma diária, portanto a coluna *date* é responsável por guardar o dia em que cada registro foi coletado. Como os dados são armazenados de forma individual para cada microrregião, há uma referência para a tabela *chunks* por parte da chave estrangeira *chunk_id*.

geo_points: as coordenadas utilizadas em diferentes entidades do nosso sistema são armazenadas nessa tabela. Os campos que compõem os pontos geográfico são latitude (*lat*) e longitude (*lng*).

trap: para persistir as informações das armadilhas que podem ser integradas no sistema, foi criada essa tabela. Cada armadilha possui um nome e a localização (*location_id*) representada por um ponto geográfico na tabela *geo_points*. Também armazenamos a data de quando a armadilha foi utilizada na alimentação de identificações de forma manual pela última vez para ser usado nos cálculos de divisão das identificações entre os dias (*last_time_feed*).

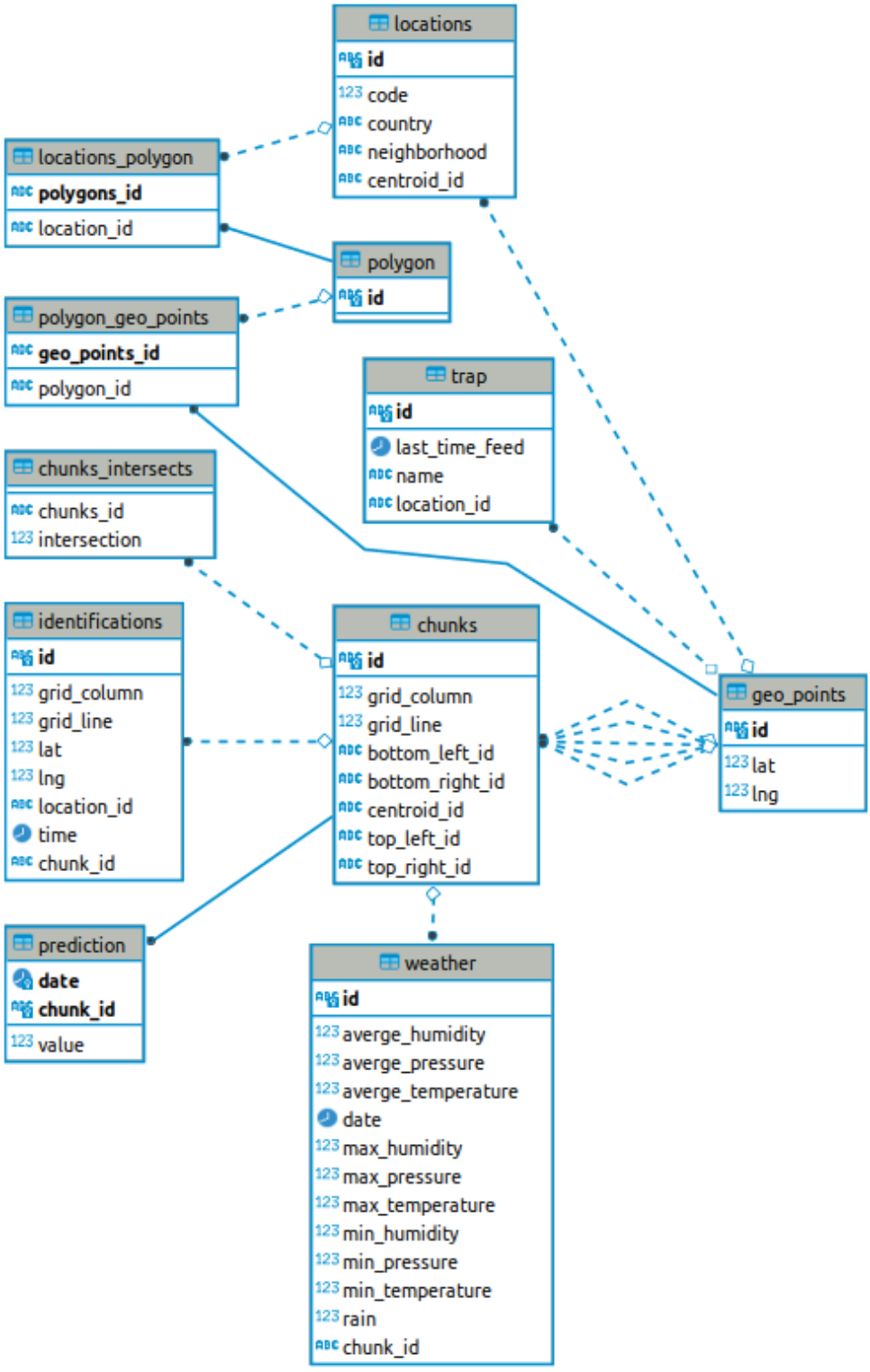
polygon: polígonos são combinações de pontos geográficos (*geo_points*). Eles possuem apenas o id que será referenciado na tabela de relacionamento *polygon_geo_points*.

polygon_geo_points: cada polígono possui N *geo_points*. Para fazer a referência entre esses múltiplos pontos geográficos e um polígono, existe essa tabela.

location_polygon: uma localização (registro da tabela *locations*) pode possuir N polígonos. A função dessa tabela é armazenar essa relação.

chunks_intersects: tabela que relaciona as microrregiões (*chunks*) com as localidades. O sistema calcula na inserção das microrregiões as suas intersecções com cada uma das localidades (bairros). Com isso é possível consolidar os dados de cada micro região em um nível maior (de localidade). Cada *chunk* pode fazer intersecção zero ou mais bairros. Cada bairro terá múltiplas *chunks* para compor a sua área.

Figura 4.3: Diagrama ER do Banco de Dados

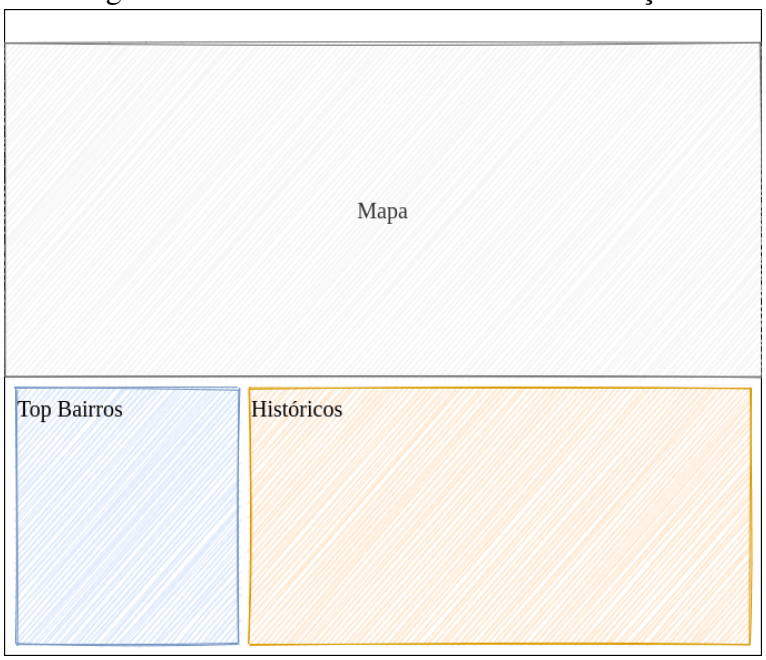


Fonte: O autor

5 INTERFACE COM O USUÁRIO

Para contemplar as funcionalidades idealizadas, no início do processo de desenvolvimento foi feito um simples *mock* que tenta propor os componentes da UI. O *mapa*, *ranqueamento dos bairros* e *históricos* foram os componentes considerados no início do processo. Alguns componentes foram agregados durante o desenvolvimento e serão mostrados ao longo do trabalho. A Figura 5.1 mostra como foi o esboço inicial da tela principal da aplicação para o usuário.

Figura 5.1: Mock do dashboard de visualização



Fonte: O autor

A percepção inicial da plataforma se deu por meio do modelo da Figura 5.1. Ela traz a ideia principal para satisfazer boa parte das funcionalidades iniciais pensadas em 4.2.3. Por meio da visão do mapa, ranqueamento e gráfico histórico, o usuário poderia de antemão ter as visões necessárias para tomar as decisões baseadas nas identificações existentes.

5.1 Definição dos Componentes

A presente seção tem como objetivo apresentar os componenetes utilizados para a elaboração da interface com o usuário.

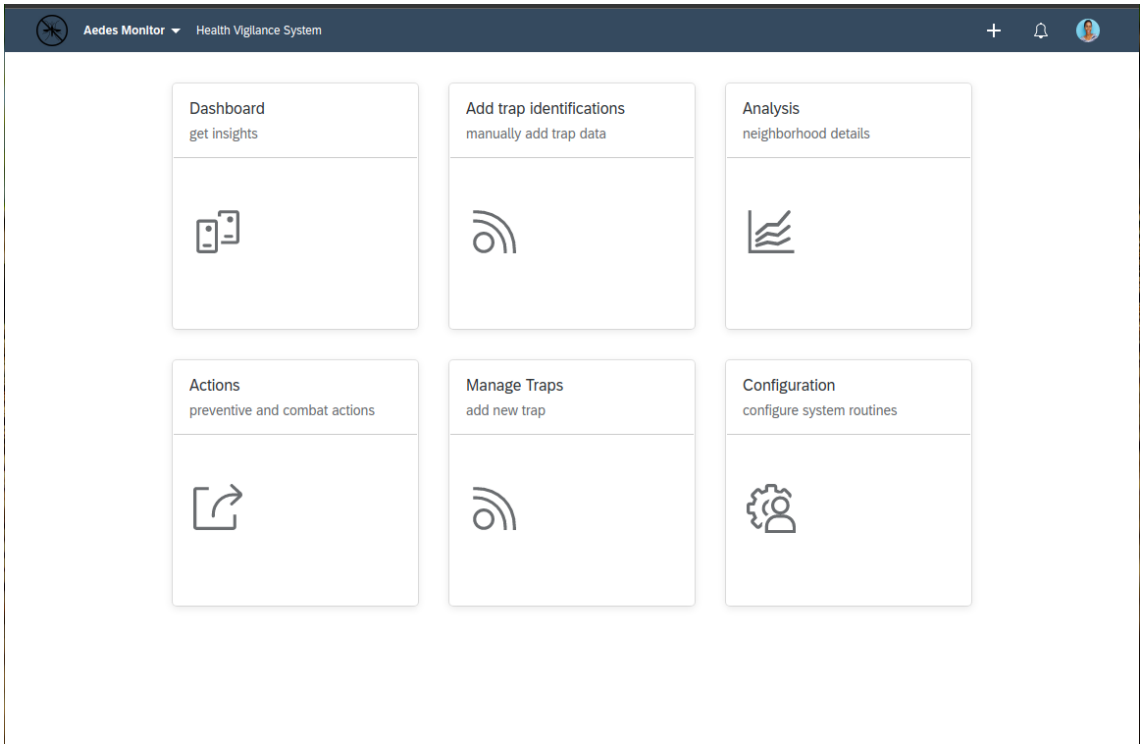
5.1.1 SAP Fiori *Guidelines*

Para o desenvolvimento da interface com o usuário, optou-se por seguir um padrão já existente e validado no mercado. Existem diretivas (*guidelines*) que definem como utilizar componentes já estilizados. Alguns exemplos de *guidelines* são o Google Material (GOOGLE, 2021), Grommet (GROOMET, 2021) e o SAP Fiori Guideline (SAP, 2021a). O trabalho foi desenvolvido utilizando uma biblioteca (SAP, 2021b) que fornece componentes que seguem o estilo proposto pelo SAP Fiori para o framework ReactJs.

5.1.2 Menu principal

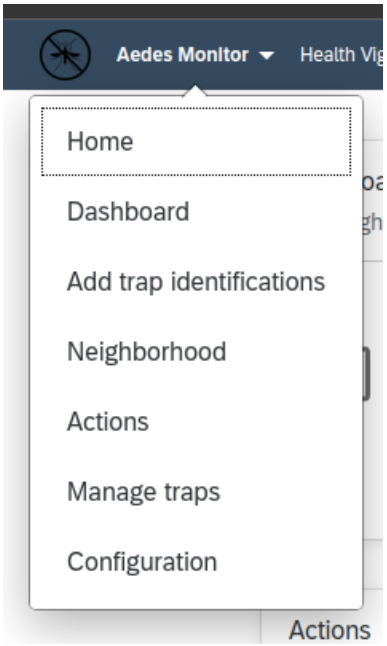
Ao acessar a plataforma, o usuário terá a visão das principais rotinas que ele poderá executar no dia a dia de uso da aplicação. Tais funcionalidades foram divididas em diferentes telas que podem ser acessadas pelos *Tiles* como representados na Figura 5.2. Além da tela de menu principal, o componente utilizado para a barra de navegação superior também possui um menu que permite o acesso às funcionalidades a partir de qualquer tela da aplicação como mostra a Figura 5.3.

Figura 5.2: Menu principal



Fonte: O autor

Figura 5.3: Menu da *Shell Bar*

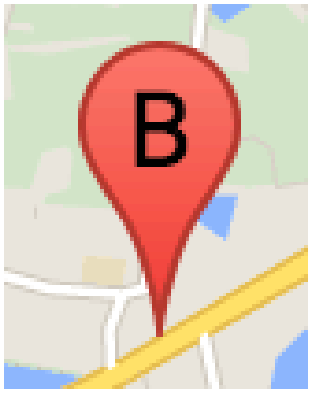


Fonte: O autor

5.1.3 Mapa

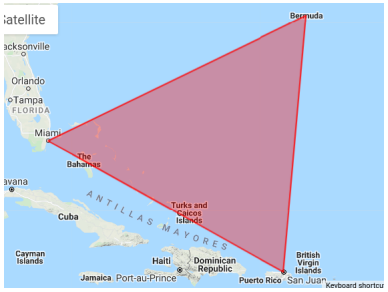
Para a renderização do mapa utilizou-se a API do Google Maps. Ela fornece meios para manipular objetos, polígonos, dentre outras funcionalidades utilizadas. A escolha foi dada pela vasta documentação e pelo fato de o Google Maps ser uma ferramenta popular para o consumo de mapas na comunidade. A flexibilidade proporcionada pelo Google Maps por meio de sua API, permitiu construirmos o conceito de visualização desejado considerando os objetos disponíveis. Os objetos são *Marker*, *Polygon* e *Heatmap*.

Figura 5.4: Objeto do tipo *Marker*



Fonte: Documentação Google Maps

Figura 5.5: Objeto do tipo *Polygon*



Fonte: Documentação Google Maps

Figura 5.6: Objeto do tipo *Heatmap*

Fonte: Documentação Google Maps

O objeto do tipo Marker mostrado na Figura 5.4 foi utilizado para representar as identificações no mapa. Sua API permite dizer onde ele deve ser renderizado no contêiner do mapa pelo parâmetros latitude e longitude. Além disso, é possível informar um percentual de opacidade desejado para o objeto, com isso podemos satisfazer o requisito funcional citado em 4.2.3 (noção temporal para a representação das identificações).

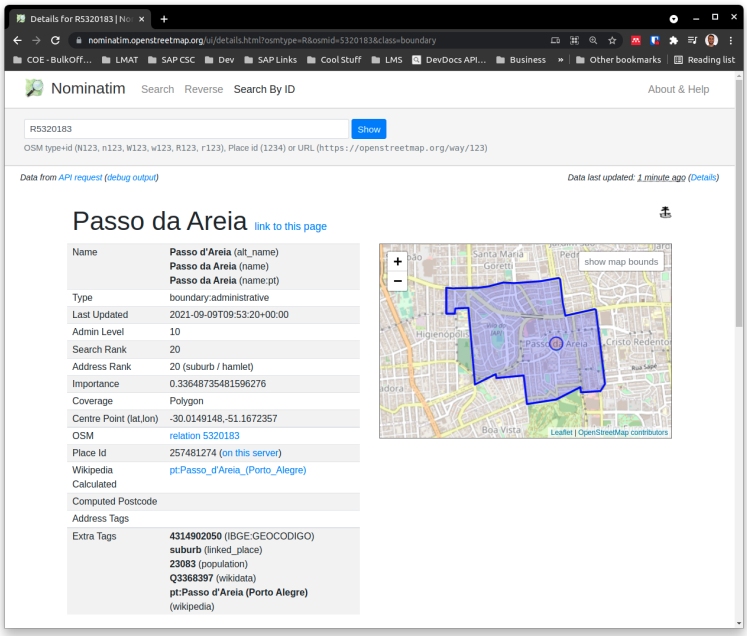
Para fazer as delimitações das *chunks* e dos bairros, foi preciso utilizar o objeto do tipo polígono mostrado na Figura 5.5. Para delimitar as áreas dos polígonos, é preciso passar como parâmetro uma sequência de pontos geográficos de latitude e longitude. A ordem desses pontos importa pois as linhas são traçadas entre os pontos conforme a sequência. Pare representar as *chunks* foi necessário apenas informar os quatro pontos que compõem as arestas do quadrado da micro-região. Já para podermos representar os bairros, foi necessário fazer um processo de coleta de dados que será explicado a seguir.

Existem repositórios que disponibilizam a malha de polígonos que representam determinadas divisões administrativas para uma região de interesse. No caso deste trabalho, a região de interesse é a cidade de Porto Alegre. Para a inserção da malha, foi optado por utilizar arquivos do tipo *geojson*. Esse tipo de arquivo possui informações importantes sobre as regiões desejadas. Um de seus atributos compõe as áreas representadas por

polígonos. Esses polígonos estão exatamente da forma e ordem de pontos geográficos que precisamos para alimentar o objeto do tipo polígono do Google Maps. Depois de levantar as possibilidades de fonte de dados para os polígonos, foi optado por fazer um trabalho manual de extração dos bairros da cidade em uma plataforma que fornece o *gejson* a partir do nome da localidade. O nome da plataforma é Nominatim (NOMINATIM, 2021).

Na Figura 5.7, podemos observar um exemplo de busca pelo polígono que compõe o bairro de Passo da Areia. Ao fazer o *download* do objeto teremos algo como é mostrado na Listagem 2.

Figura 5.7: Polígono Passo da Areia



Fonte: O autor

Os objetos extraídos para cada um dos bairros foram combinados em um arquivo *json* que foi utilizado como base para a inserção dos dados no banco por meio de um *script* Python. O *script* foi escrito em Python 3 e é responsável por iterar sobe todos os detalhes dos bairros e inserir as coordenadas presentes no atributo *geojson*.

O objeto *Heatmap* representado na Figura 5.6 foi utilizado para duas funcionalidades chave da aplicação: mostrar os rastros das identificações do mosquito e também proporcionar uma visão do risco baseado nas previsões, nas visões histórica e preditiva, respectivamente como será apresentado no próximo capítulo.

Listing 2: Exemplo de um objeto de uma localidade

```
{
    "details": [
        {
            "boundingbox": [
                "-30.0218697",
                "-30.0074038",
                "-51.1818262",
                "-51.1608098"
            ],

            "display_name": "Passo da Areia [...]",
            "geojson": {
                "type": "Polygon",
                "coordinates": [
                    [
                        [
                            -51.1818262,
                            -30.0085291
                        ],
                        [
                            -51.1818253,
                            -30.0085634
                        ],
                        [
                            -51.1818243,
                            -30.0085977
                        ],
                        ...
                    ]
                    ...
                ]
            }
            ...
        }
    ]
    ...
}
```

5.2 Implementação da UI

A seguir serão apresentadas as telas desenvolvidas utilizando os componentes citados anteriormente.

5.2.1 Dashboard

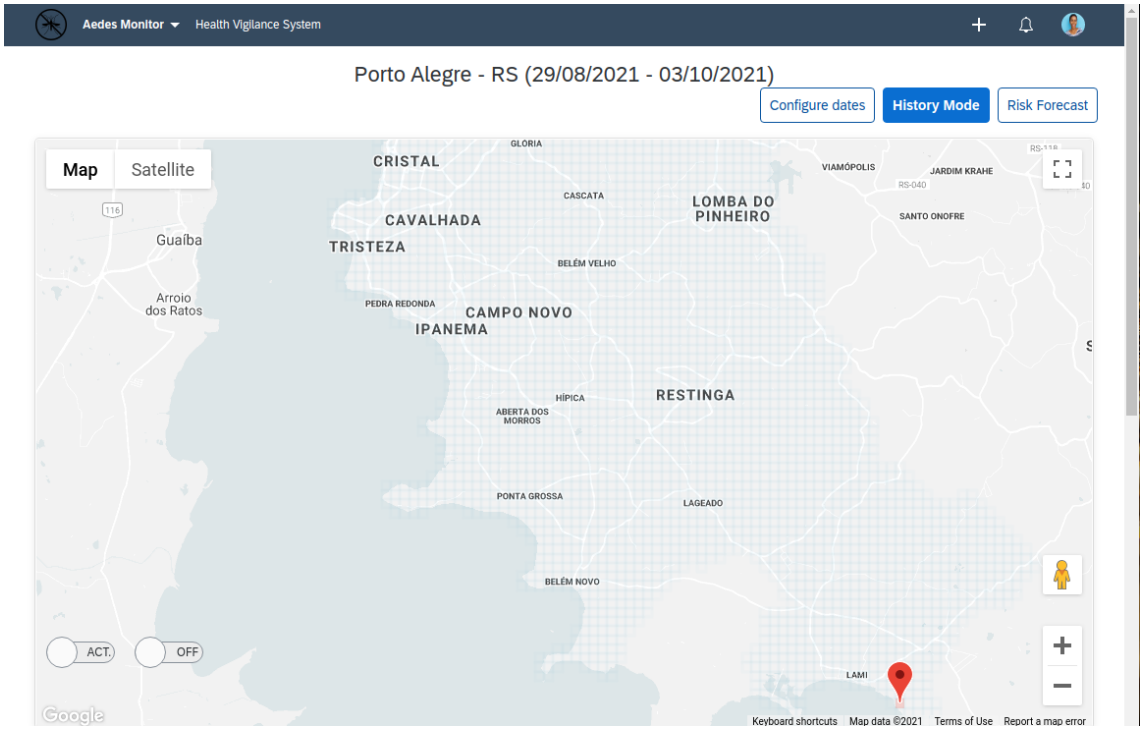
Ao entrar no *dashboard* da aplicação, o usuário deve poder ver os principais aspectos para o monitoramento das identificações e predições de risco. Nesse sentido, as funcionalidades principais citadas na Seção 4.2.3 serão agrupadas nessa página. Utilizaremos o componente de mapa com os objetos representando as entidades importantes para o monitoramento (identificações e *chunks*).

A tela do *dashboard* foi feita com base no *mock* da Figura 5.1. Além do mapa capaz de mostrar as identificações e predições. O usuário poderá acompanhar o ranqueamento dos bairros mais afetados e ver o histórico das identificações em um gráfico de linhas ou selecionar uma visualização de gráfico de barras.

As Figuras 5.8 e 5.9 mostram a visão do usuário ao navegar para a página de visualizações do *dashboard*. Essa visão inicial abrange as configurações padrões que são customizáveis. Por padrão os bairros e *heatmaps* não são mostrados assim como nenhum período está selecionado. A visão inicial engloba os dados históricos, o menu seletor (Figura 5.16) que fica na canto superior direito - logo acima do mapa - indica que o modo histórico está habilitado. Podemos observar na Figura 5.8 que as *chunks* estão cobrindo a área da cidade de Porto Alegre. Na Figura 5.12 temos em destaque uma *chunk* que possui identificações. Essas identificações fizeram com que a área da *chunk* ficasse em destaque na cor vermelha. Também podemos notar na Figura 5.12 que o botão que faz os *heatmaps* aparecerem estava habilitado fazendo com que um círculo aparecesse ao redor da identificação. Ao ativar a visualização dos bairros, a malha de polígonos que representa essas divisões administrativas é apresentada no mapa como mostra a Figura 5.11.

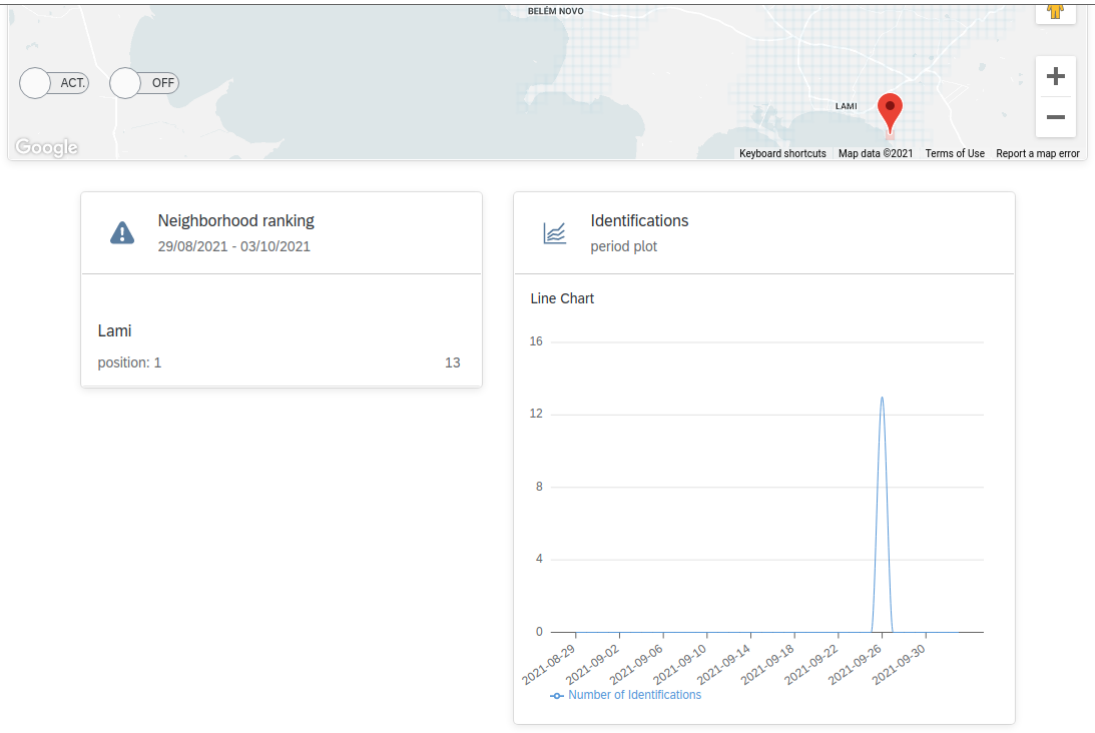
A Figura 5.13 mostra o que acontece se o usuário clicar em "*Configure Dates*", um *dialog* é aberto com um seletor de datas. Ao clicar na caixa de seleção de datas, o usuário poderá escolher no calendário o início e o fim do período desejado para a exibição dos dados (5.14). Ao selecionar o período desejado, o mapa irá mostrar apenas

Figura 5.8: Página de *dashboard*



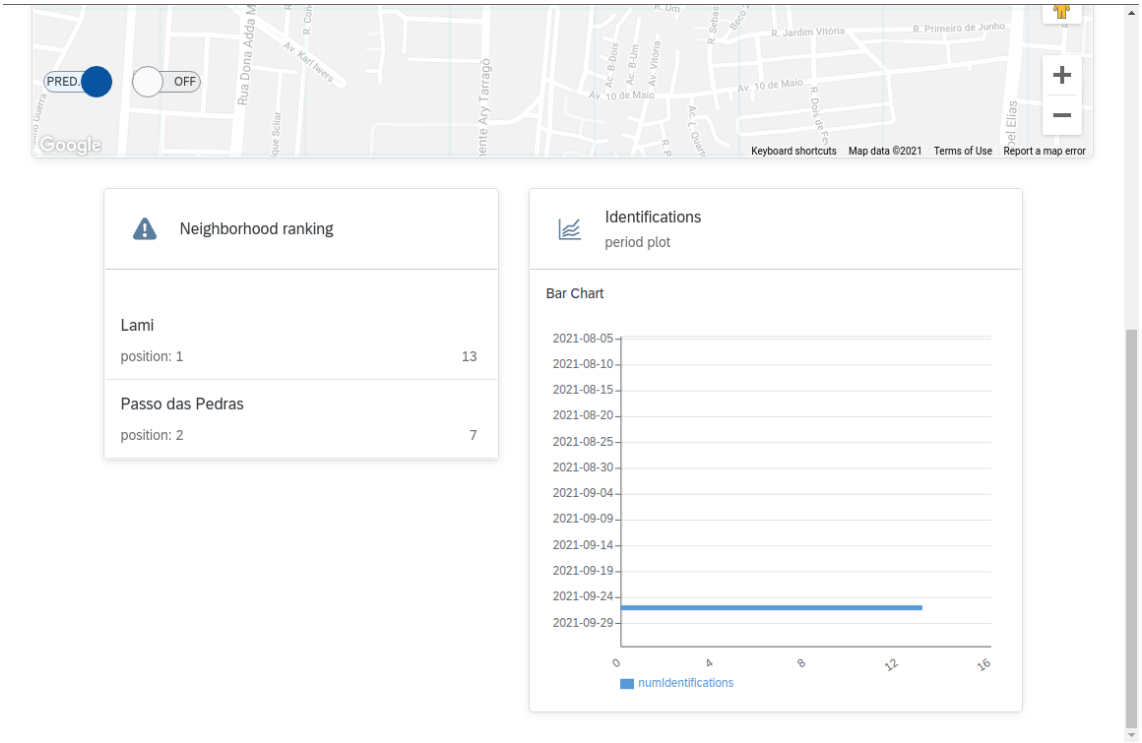
Fonte: O autor

Figura 5.9: Página de *dashboard* continuação



Fonte: O autor

Figura 5.10: *Dashboard* - histórico gráfico de barras



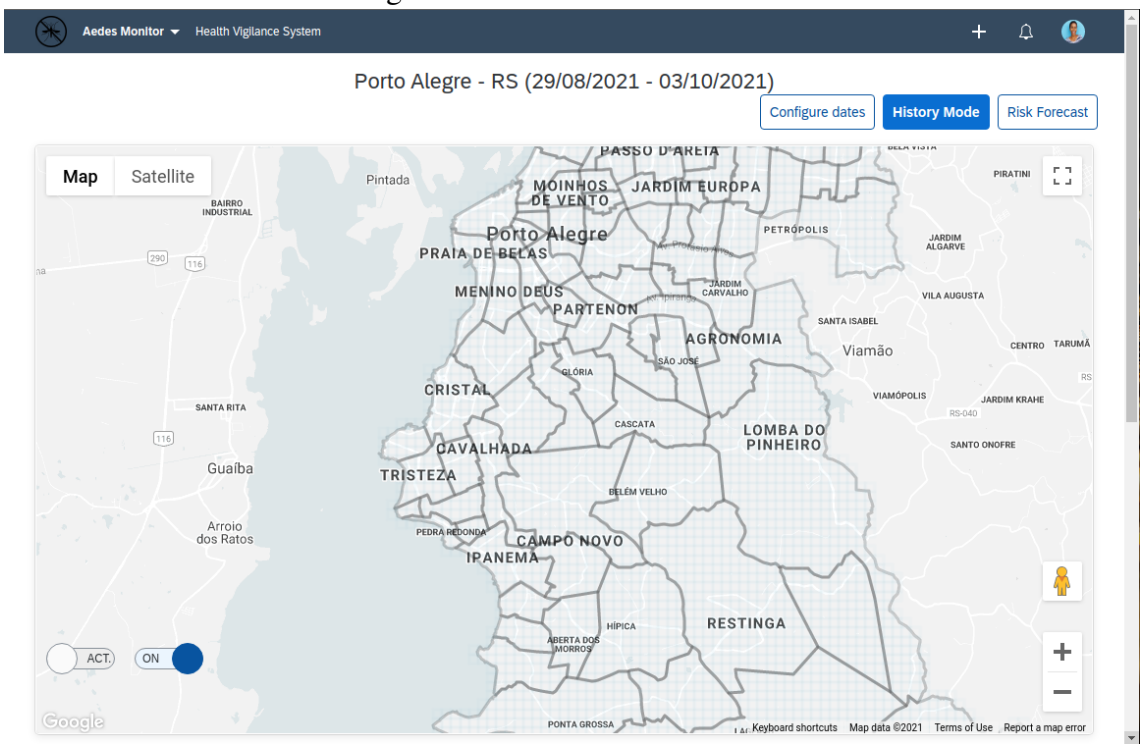
Fonte: O autor

as identificações dentro do período informado. O ranqueamento dos bairros com maior número de identificações (Figura 5.9) e os gráficos do histórico (Figura 5.10) também são influenciados pelo período.

Regiões onde não houveram identificações não são destacadas no mapa. Porém as regiões onde o sistema foi informado de pelo menos uma identificação será destacada no mapa em duas cores: amarela e vermelha. Os parâmetros para definir a cor acontece de forma programática, porém é possível de no futuro a aplicação incorporar páginas de configuração onde o usuário define os limites para cada classificação para que as cores sejam aplicadas com base em parâmetros customizáveis. O esquema de cores para dados históricos pode ser observado na imagem 5.15.

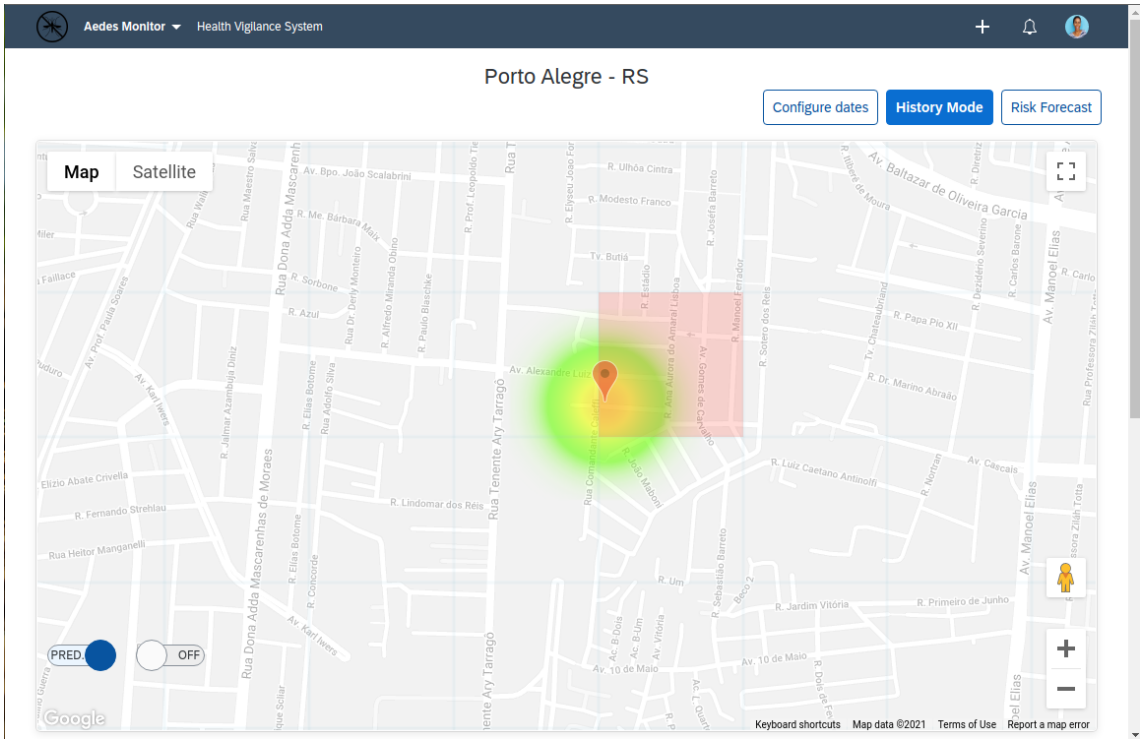
Os objetos marcadores e *heatmaps* foram implementados para aparecerem de forma dinâmica conforme o zoom que o usuário está aplicando no mapa. Com isso, é possível dar foco para a agregação do grupo quando o usuário está com uma visão mais geral do mapa. Quando o usuário se aproxima de uma área de interesse a ponto de enxergar com mais detalhes as micro-regiões, os marcadores se tornarão visíveis (Figura 5.12), tornando assim possível uma melhor noção da região da ocorrência da captura do mosquito.

Figura 5.11: *Dashboard* - bairros



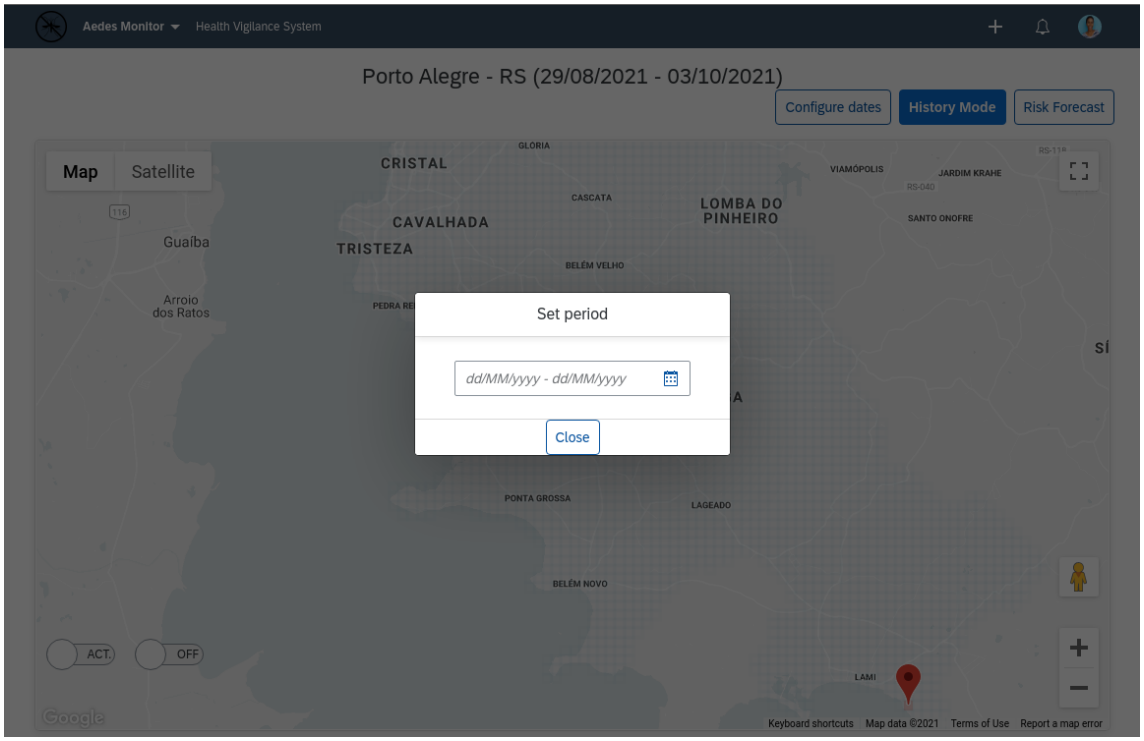
Fonte: O autor

Figura 5.12: *Dashboard* - *heatmap* histórico



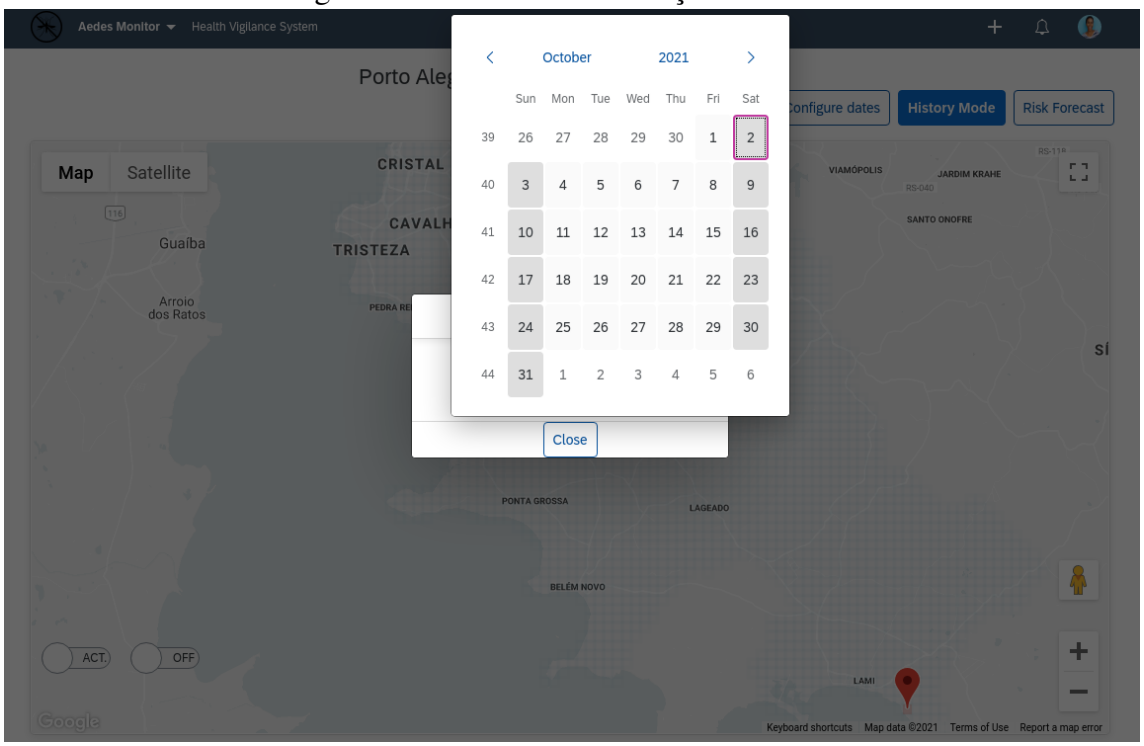
Fonte: O autor

Figura 5.13: *Dashboard* - seletor de período



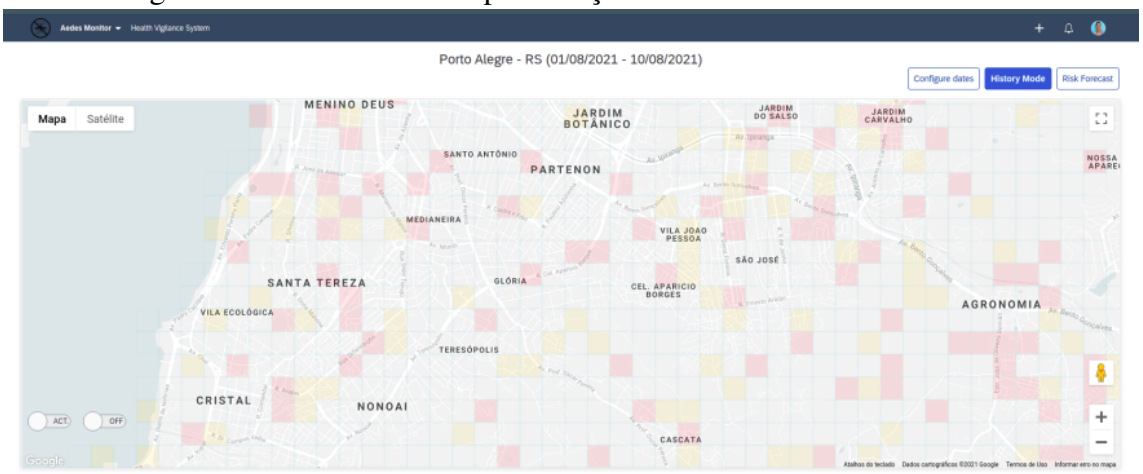
Fonte: O autor

Figura 5.14: *Dashboard* - seleção das daatas



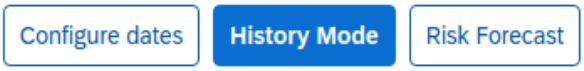
Fonte: O autor

Figura 5.15: *Dashboard* - representação histórica com dados mocados



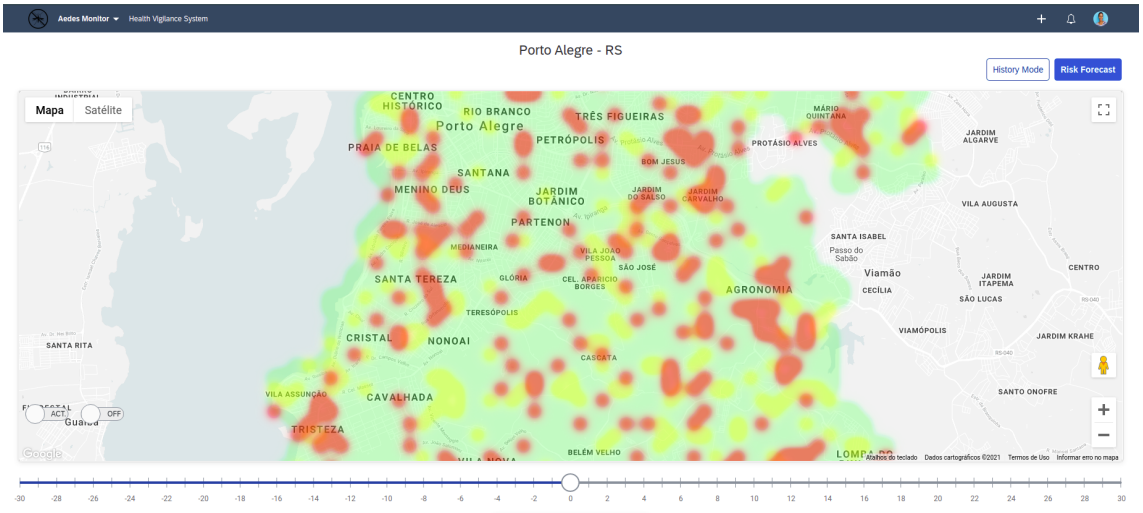
Fonte: O autor

Figura 5.16: *Dashboard* - seletores de visualização



Fonte: O autor

Figura 5.17: *Dashboard* - predições de risco



Fonte: O autor

Ao clicar em *"Risk Forecast"* o usuário terá acesso à visualização de predição de risco como mostra a Figura 5.17. Essa tela é constituída pelo mesmo componente de mapa da visão histórica, porém é composta por *heatmaps* de diferentes cores. Os cálculos do risco serão feitos pelo *backend* e armazenados em uma tabela. Essa tabela é utilizada para retornar para a UI os valores previstos para cada *chunk*. Com isso o

usuário poderá selecionar o dia para o qual ele deseja ver o risco. O cálculo pode inferir três diferentes níveis de criticidade: normal, atenção e risco (verde, amarelo e vermelho, respectivamente). Abaixo do mapa há um barra de seleção. Ela representa uma escala que varia de -30 a 30 que é utilizada pelo usuário para selecionar quantos dias para o futuro ele deseja ver o estado de predições, caso o valor selecionado seja positivo. Caso o valor selecionado seja negativo, ele poderá acessar os valores que foram preditos para o passado, o que pode ser utilizado para avaliar a eficiência do modelo. Quando o seletor está posicionado em 0, o mapa estará mostrando as predições que foram feitas para o dia atual. Cada *chunk* possui a referência para o seu centro (centroide), esse valor é utilizado como referência para a renderização da cor do risco para a área.

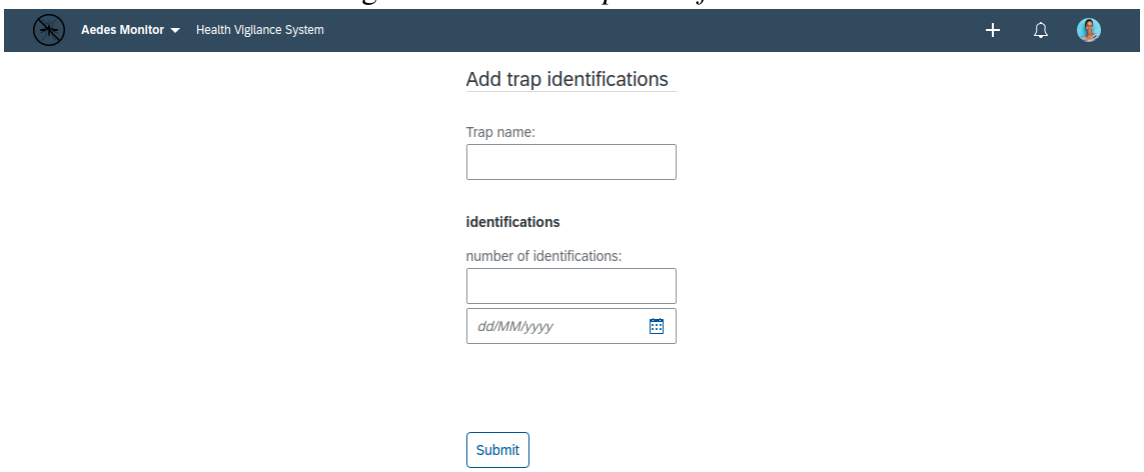
Nessa subseção foram mostrados os recursos que a interface com o usuário oferece para o fim de proporcionar uma boa experiência de monitoramento. A possibilidade de ter esses dados em tempo real implica em uma possível melhora de efetividade na identificação de pontos que necessitam atenção. Podemos dizer que a tela do *dashboard* é de caráter analítico, assim como as visualizações dispostas na tela de *"Analysis"* descrita na Seção 5.2.3.

5.2.2 *Add trap identifications*

O sistema proposto pelo trabalho apresenta uma nova forma de ter os dados de identificações do mosquito de forma automatizada. Porém, também é preciso olhar para o processo já existente quanto à coleta, identificação e registro dos dados de identificação.

Atualmente, as quantidades de mosquitos identificados são registradas de forma manual em uma plataforma chamada DATASUS (SAÚDE, 2021). Esse processo é feito por uma tela de cadastro. O ideal seria ter uma integração com esse sistema para que o ponto de entrada dessa informação para o mundo digital exigisse apenas uma ação humana. No entanto, no contexto desse trabalho, não foram desferidos esforços para realizar essa integração e, na busca de achar uma forma de inserir dados sobre as coletas feitas nas armadilha, foi implementada uma tela para tal ação. Na Figura 5.18 podemos ver os campos que essa tela nos permite inserir.

Figura 5.18: *Add Trap identifications*



Fonte: O autor

5.2.3 *Analysis*

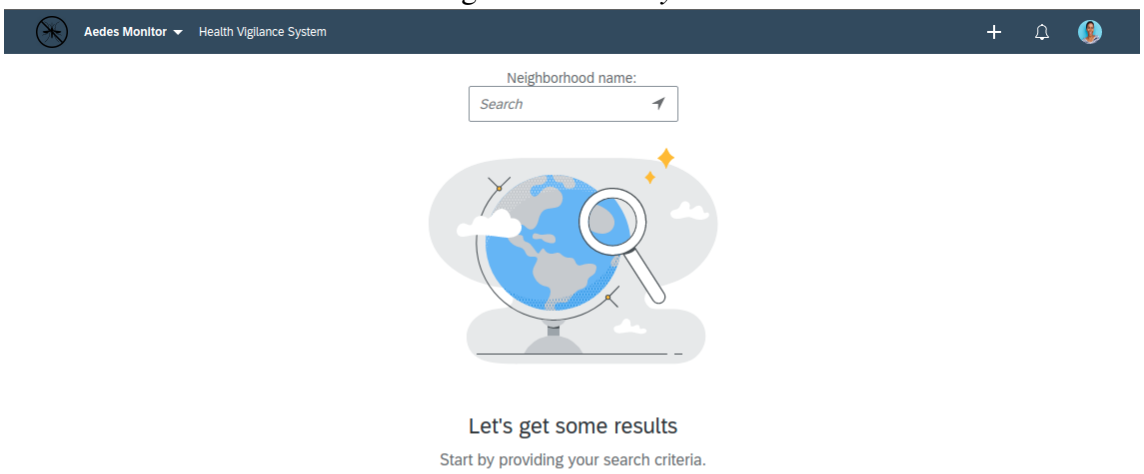
O *dashbord* introduzido na Seção 5.2.1 tem a capacidade de mostrar uma visão mais geral da situação da cidade como um todo. Para elevar o nível de detalhamento, foi feita uma página que permite o usuário voltar a atenção para as divisões administrativas (os bairros). Chamamos essa tela de *Analysis*, conforme mostra a Figura 5.19.

O usuário poderá procurar por seus bairros de interesse como na Figura 5.20. Ao selecionar um dos bairros, um componente com algumas informações é mostrado. Essas informações foram divididas em dois componentes: *overview* e *chunks* , eles serão detalhados na subseções que seguem (5.2.4 e 5.2.5).

5.2.4 *Overview*

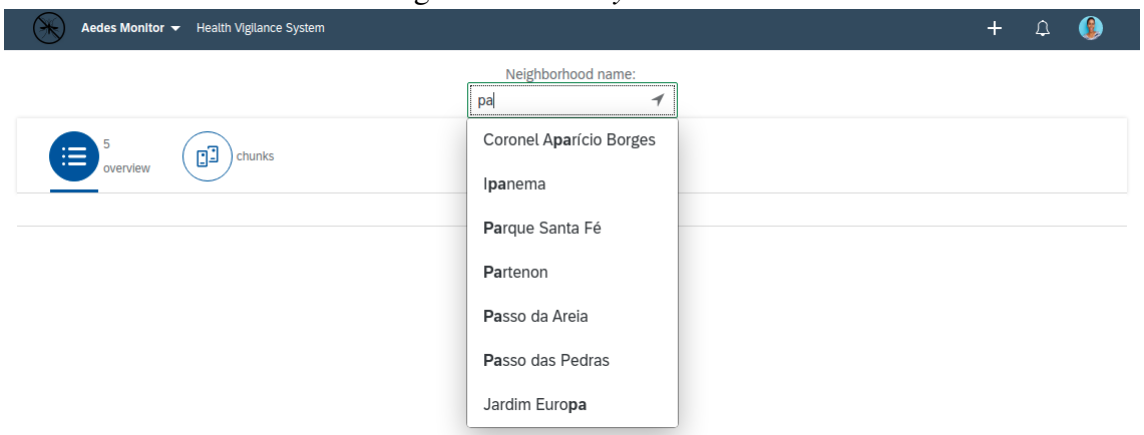
O componente *overview* foi pensado no cenário onde o usuário tem o interesse de ter rápido acesso à informações históricas e dados de predições. Essas informações são baseadas em um número já predefinido de dias e não pode ser alterada de forma dinâmica

Figura 5.19: *Analysis*



Fonte: O autor

Figura 5.20: *Analysis Search*



Fonte: O autor

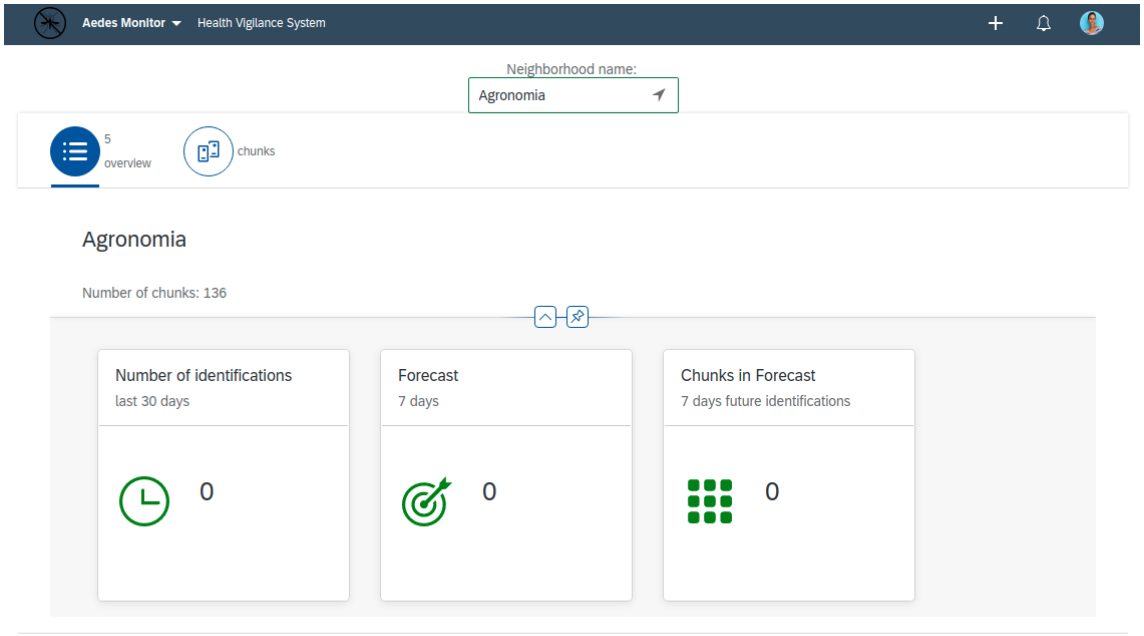
na aplicação. Os dados são consolidados e disponibilizados em *tiles*. Os *tiles* definidos são:

Number of identifications : número de identificações que foram registradas para as *microrregiões* que compõem o bairro de interesse.

Forecast : número de predições que foram feitas para os próximos 7 dias de acordo com o bairro de interesse.

Chunks in Forecast : O número de *chunks* do bairro de interesse que são afetadas pelas previsões para os próximos 7 dias.

Figura 5.21: *Analysis - overview*



Fonte: O autor

As informações de previsões visíveis nos *tiles* são calculadas com modelos preditivos feitos para o número desejado de dias no futuro. Na forma como o sistema foi projetado, os valores de predições serão computadas por processos automatizados a serem executados diariamente e armazenados em tabelas para servir as funcionalidades. Tais processos serão descritos no Capítulo 6 em mais detalhes.

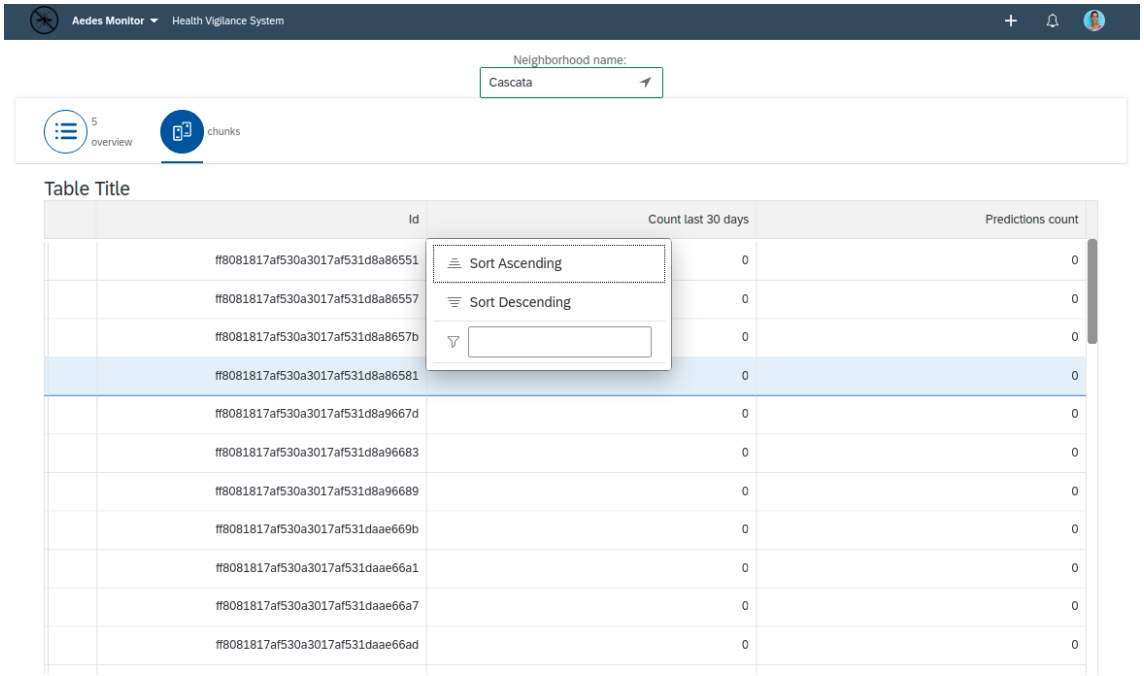
5.2.5 *Chunks*

A subseção *Chunks* da página *Analysis* apresenta informações sobre as microrregiões que compõem a localidade observada. O objeto é disponibilizar uma tabela que destrincha os valores observados nos *tiles* da seção *Overview* (5.2.4). Há a possibilidade de adicionar mais informações nessa tabela para guiar o usuário à alguma ação com base nos números de cada microrregião da tabela. Na versão atual do trabalho, são listados: identificação interna da microrregião, número de identificações nos últimos dias e o número de registros que são previstos para os próximos dias.

Uma funcionalidade implementada a fim de tornar possível uma ação direcionada à uma microrregião com um risco maior de haver identificações nos dias futuros permite que o usuário clique na linha da microrregião de interesse e seja direcionado ao mapa do *dashboard* com um zoom na região selecionada. Essa funcionalidade somada com o ordenamento dinâmico permitido pelo componente utilizado para a tabela, faz com que os usuário tenham um controle sobre quais são as regiões que merecem mais atenção.

A Figura 5.22 mostra a página de *Analysis* quando o componente de *Chunks* é selecionado. A tela deixa de renderizar o componente de *Overview* e mostra a tabela com as *chunks*. Essas *chunks* são apenas as microrregiões que tiveram alguma intersecção com as áreas do polígono que compõe a região do bairro de interesse.

Figura 5.22: *Analysis - chunks*

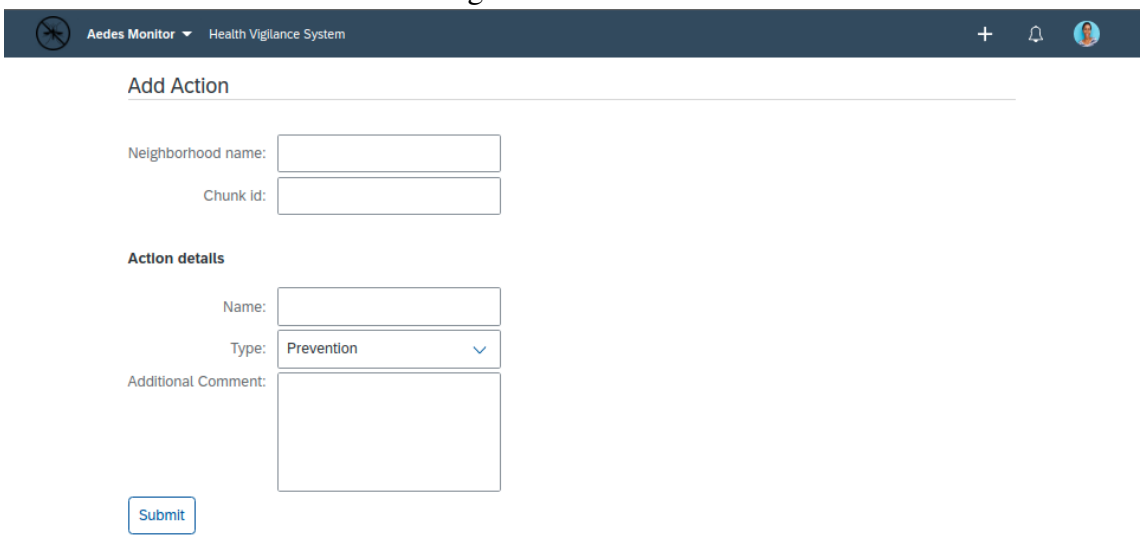


Fonte: O autor

5.2.6 *Actions*

Com a finalidade de projetar um sistema que possa ser melhorado e utilizado para rotinas que compõem o dia a dia dos agentes responsáveis pelo controle do mosquito *A. aegypti*, foi considerada uma funcionalidade que extrapola o conceito de monitoramento e colabora com a parte do combate do vetor. Tornar possível o registro de ações de controle e prevenção dos focos do mosquito pode ajudar no processo de construção e manutenção de conhecimento histórico de medidas para o combate do mosquito. Além de poder ajudar no surgimento de modelos capazes de utilizar essas informações para sugerir ações para determinadas localidades com base nos dados históricos de identificações, medidas registradas, dados climáticos etc. A página de cadastro de ações de controle está representada na Figura 5.23.

Figura 5.23: *Actions*



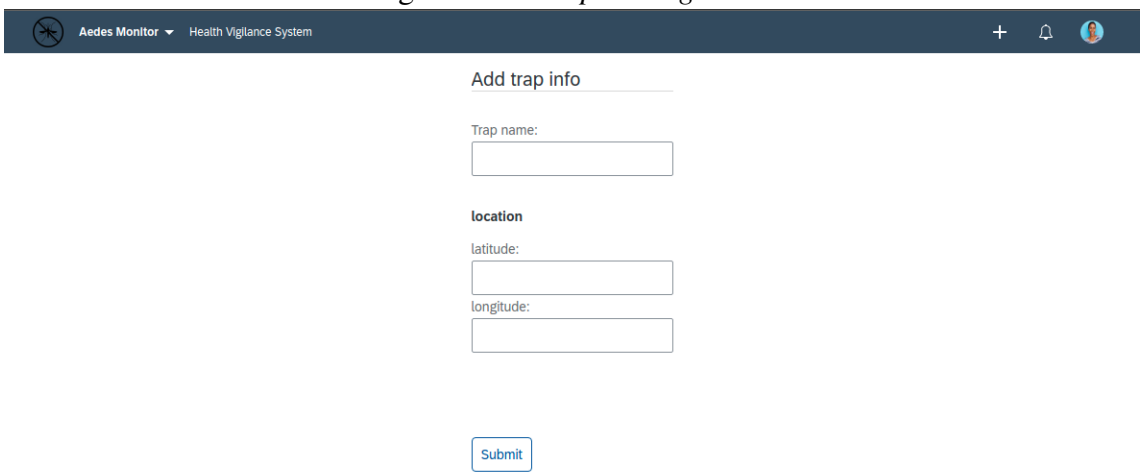
Fonte: O autor

5.2.7 *Manage Traps*

As armadilhas já existente no sistema atual de monitoramento do mosquito *Aedes aegypti* na cidade de Porto Alegre podem continuar em operação e agregar registros aos dados utilizados para o controle. Para que seja possível fazer o registro dessas armadilhas

no nosso sistema e que as identificações adicionadas manualmente (Figura 5.2.2) possam ser vinculadas à localidade física das armadilhas, é preciso fazer um cadastro manual. Esse processo de cadastro das armadilhas poderá feito pela página *Manage Traps* (Figura 5.24).

Figura 5.24: *Trap Management*



Fonte: O autor

5.2.8 *Configuration*

A Figura 5.2 mostra um *tile* de configuração. Ele deve levar a uma página onde o usuário deve poder configurar os processos automatizados utilizados no processo de coleta e processamento dos dados que são mantidos no sistema. Tais configurações não foram implementadas, as possíveis configurações que poderiam ser incorporadas são as seguintes:

Agendamento da coleta de dados climáticos: para configurar o *backend* e definir quando o processo que interage com a a API de climática deve executar a sua rotina.

Agendamento do processo de predições: com a finalidade de configurar o *backend* para executar o processo de cálculo de predições com base nos novos dados do dia. Esse processo deve ser executado após o citado anteriormente para que ele leve em consideração os dados climáticos do dia.

6 DEFINIÇÃO DOS SERVIÇOS

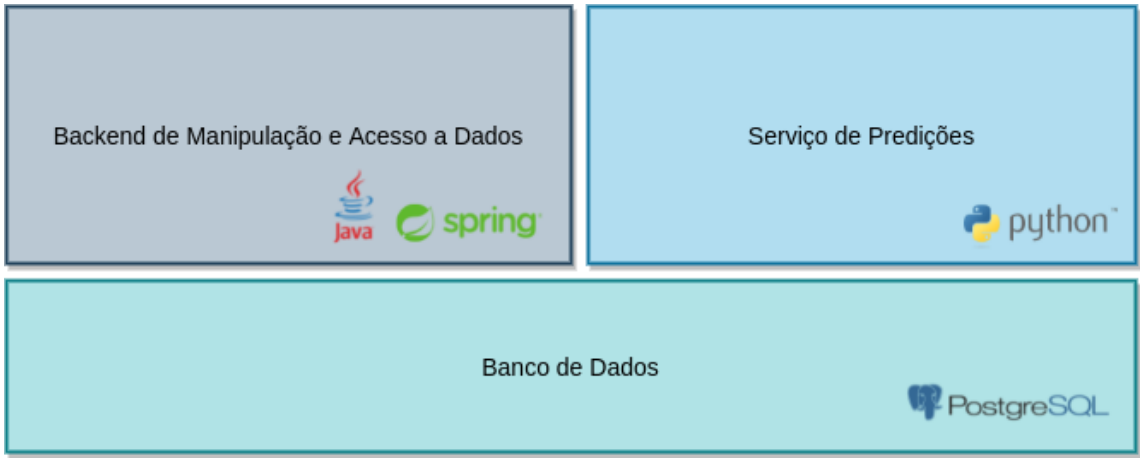
O conjunto de serviços que torna possível o comprimento dos requisitos é entregue por alguns componentes de software que serão descritos no presente capítulo. As tecnologias utilizadas para implementar o sistema com base na arquitetura pode ser encontrada na Figura 6.1. A Figura 4.2 apresenta as relações de dependência entre os módulos do sistema.

As regras de negócio foram implementadas no *backend*, com isso o *frontend* é responsável apenas por consultar as informações no *backend* desenvolvido na aplicação. Os componentes da UI foram separados em telas e cada tela faz a utilização serviços de consulta de API que, por sua vez, fazem as consultas dos dados com base nos parâmetros.

Para a implementação dos serviços necessários para servir os dispositivos e realizar as coletas de dados, foi feita uma aplicação em Java utilizando Spring Boot como *framework* (WALLS, 2015). A utilização do Spring Boot, permite que a aplicação seja auto contida tendo o servidor embutido (Tomcat). Além disso, o Spring permite a utilização de princípios que facilitam o desenvolvimento tal como convenção sobre configuração (GUTIERREZ, 2016) e inversão de controle (PISA; PISA, 2009).

O serviço responsável por realizar as predições e armazená-las no banco de dados foi criado na linguagem Python por conta da existência de bibliotecas que facilitam o tratamento de dados como Pandas (MCKINNEY, 2012), por exemplo. O serviço foi exposto em HTTP com a utilização da biblioteca Flask (GRINBERG, 2018).

Figura 6.1: Tecnologias

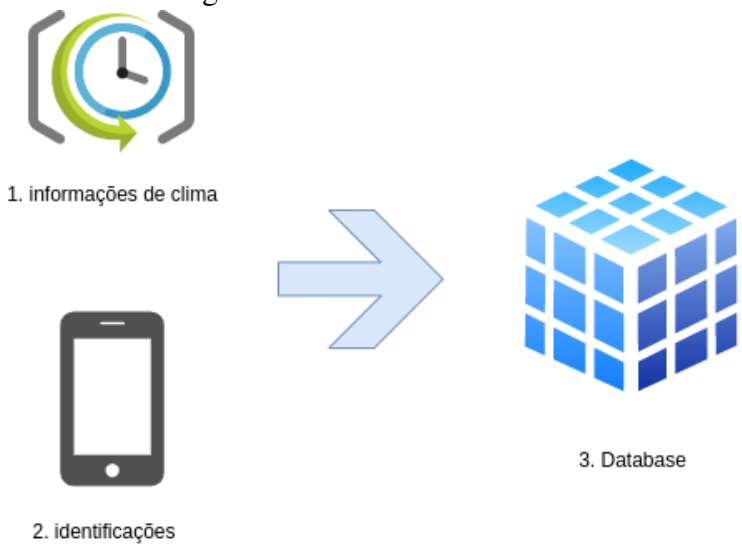


Fonte: O autor

6.1 Serviços de Coleta de Dados

Os serviços de coleta de dados são responsáveis por inserir no contexto do sistema as informações necessárias para o seu funcionamento. A solução proposta depende de diferentes fontes de dados e cada tipo de dado deve ser propriamente tratado para que faça sentido na aplicação. Os serviços de coleta de dados podem ser categorizados de acordo com a informação que estão tratando: identificações e dados meteorológicos.

Figura 6.2: *Processo de Coleta*



Fonte: O autor

Uma ilustração abstrata de como funciona o processo de coleta de dados pode ser encontrada na Figura 6.2. Nela, as fontes de dados estão à esquerda. Onde 1 representa as informações de clima que são coletadas de forma automatizada por um *background job* e as identificações que são inseridas por meio dos dispositivos móveis.

6.1.1 Coleta das Identificações

As identificações podem ser coletadas de duas formas. A primeira já foi citada na Seção 5.2.2, ela categoriza a inserção manual das identificações das armadilhas do controle que já é feito na cidade de Porto Alegre. A segunda forma é por meio de uma API que expõe um recurso à web para dispositivos móveis e armadilhas autônomas inserirem um registro da localidade juntamente com o *timestamp* do momento da identificação.

O processo de inserção de identificações de forma manual, feito através da tela *Add Trap Identifications* (5.2.2), considera o momento em que foi feita a última inserção

e divide o número de identificações informado pelos dias decorridos. Tal lógica foi necessária pois os agentes que inspecionam as armadilhas executam a rotina de verificação e contagem em uma base semanal. Ao dividirmos a quantidade de mosquitos pelos dias, teremos uma aproximação do número de capturas de cada dia, porém há uma perda de fidelidade na caracterização temporal desses dados.

6.1.2 Coleta dos Dados Meteorológicos

Com o intuito de fornecer mais recurso para previsões de contagens do mosquito *Aedes aegypti*, este trabalho se propõe a utilizar dados meteorológicos juntamente com as identificações históricas. Esse tipo de abordagem já foi feita em outros trabalhos como citado nos trabalhos relacionados. Na Seção 7.3 também discutimos sobre a correlação entre as variáveis meteorológicas e número de mosquitos.

Para inserção desse tipo de dado, é necessário ter acesso à informações de estações climáticas que fazem, de forma automatizada, leituras de temperatura, umidade, pressão, quantidade de chuva etc. Existem serviços de terceiros que combinam informações de diferentes estações climáticas à modelos matemáticos e fornecem uma forma fácil de acesso a esse tipo de informação com um bom nível de precisão. Esse tipo de serviço foi utilizado para a coleta desses dados. Utilizamos a API de clima *Weather API* (OPENWEATHER, 2021). O plano de desenvolvimento para estudantes foi utilizado para poder fazer as chamadas da API sem custo.

Uma das vantagens da utilização da *Weather API* da OpenWeather é que ela proporciona uma boa granularidade para a consulta dos dados. Pode-se passar para uma consulta as coordenadas latitude e longitude. Ela utiliza os valores das estações próximas à localização informada e calcula os valores que melhor representam o ponto consultado. Tal granularidade nos permite fazer consulta para todas as microrregiões de interesse (*chunks*) de forma independente. Um exemplo de resposta para uma consulta da API utilizada pode ser encontrada na Listagem 3.

O sistema possui uma rotina automatizada que é executada periodicamente, a sugestão de execução é na mudança de todos dias (zero hora) . Nosso *backend* solicita à API os dados climáticos do dia anterior, e ela retorna os valores de hora em hora. Essa chamada é feita para cada uma das microrregiões que estão mapeadas na aplicação. Ao receber os dados para cada hora, existe uma classe que calcula as médias para o dia e armazena os valores em uma tabela no banco de dados que será utilizada posteriormente

como base para os modelos preditivos.

6.2 Serviços de Processamento de Dados

Com base nos dados coletados tal como descrito na Seção 6.1.2, precisamos calcular as predições para podermos mostrar na granularidade de cada microrregião. O sistema alcança isso da mesma forma como na coleta dos dados. A partir de um agendamento feito para toda noite, é executada uma rotina que considera as séries temporais de todas as microrregiões, levando em conta os valores para o novo dia. Para atingir o objetivo de proporcionar predições ao usuário, um micro-serviço especializado foi elaborado em Python. Esse serviço foi construído pensando em ser extensivo para poder incorporar diferentes versões do modelo preditivo.

O serviço de predições desenvolvido nese trabalho recebe como parâmetro a data base para ser considerada nos cálculos. A data base é utilizada com a finalidade de delimitar a janela de dados que será utilizada para o cálculo. No início da elaboração do trabalho, tal serviço foi projetado para receber como parâmetro os valores das séries temporais para cada microrregião. Dessa forma, estaríamos criando um tráfego de informações elevado (a aplicação Java consultaria os valores do banco de dados e as enviaria para o serviço em Python). Para não adicionarmos o custo extra de fazer a chamada do método *predict* passando todas as informações necessárias pelo método HTTP, o serviço de predições passou a ter acesso direto à base de dados. Com esse acesso direto, é possível, de forma simplificada, fazer as consulta aos valores das tabelas por meio de *queries* SQL que retornam tabelas a serem carregadas como *Dataframes* da biblioteca Pandas, o que facilita o tratamento na etapa que antecede o método de predição.

6.3 REST APIs

Como foi dito, a UI e os dispositivos móveis precisam ser capazes de realizar consultas na base de dados. Essas consultas são servidas através de *REST APIs* (RODRÍGUEZ et al., 2016) que podem ser consumidas por diferentes clientes. Qualquer dispositivo capaz de acessar a internet e operar com o protocolo HTTP será capaz de utilizar as APIs aqui propostas. Nesse sentido, a aplicação ganha possibilidade de extensão e integração com diferentes tipos de equipamentos (como por exemplo, possíveis armadilhas

Listing 3: exemplo *Weather API*

```
{
    "lat": 33.44,
    "lon": -94.04,
    "timezone": "America/Chicago",
    "timezone_offset": -21600,
    ...
    "hourly": [
        {
          "dt": 1618315200,
          "temp": 282.58,
          "feels_like": 280.4,
          "pressure": 1019,
          "humidity": 68,
          "dew_point": 276.98,
          "uvi": 1.4,
          "clouds": 19,
          "visibility": 306,
          "wind_speed": 4.12,
          "wind_deg": 296,
          "wind_gust": 7.33,
          "weather": [
            {
              "id": 801,
              "main": "Clouds",
              "description": "few clouds",
              "icon": "02d"
            }
          ],
          "pop": 0
          ...
        },
        ...
    ]
    ...
}
```

automatizadas que podem vir a ser implementadas).

Para tornar possível os serviços que o sistema se propõe, foram criadas APIs para abstraírem a execução de rotinas e também para retornar dados que são utilizados pela aplicação ou que podem ser utilizados por outras ferramentas desenvolvidas no futuro. Tais APIs foram implementadas utilizando o protocolo *Hypertext Transfer Protocol* (HTTP) da camada de aplicação Open System Interconnection (OSI) (KUMAR; DALAL; DIXIT, 2014), assim, é preciso informar o método HTTP (*GET, POST, PUT* ou *DELETE* - existem outros mas esses são os principais que nossa aplicação utiliza) na requisição. No Apêndice A são listadas as APIs e suas funcionalidades.

Cabe ressaltar que a implementação de um sistema pode se basear nas funcionalidades e interfaces propostas por este trabalho, porém os detalhes de tecnologia podem variar de acordo com o projeto. Pode-se, por exemplo, optar pela utilização de outros protocolos para a comunicação entre os serviços. Porém, o que é aqui apresentado foi o utilizado para a construção da prova de conceito durante a elaboração do trabalho.

7 MODELO PREDITIVO DE IDENTIFICAÇÕES DO *A. AEGYPTI*

O sistema que este trabalho propõe pode ter um modelo preditivo acoplado. Assim, visualizações referentes a previsões podem ser apresentadas para o usuário, tornando possível a tomada de decisão com base em dados históricos. No intuito de mostrar essa capacidade, no presente capítulo é apresentado o processo de desenvolvido um modelo que foi integrado à plataforma como prova de conceito. O foco desse trabalho não é desenvolver um modelo preditivo, mas sim mostrar a possibilidade dessa integração com a plataforma. Tal integração pode ser alcançada com a adição de um serviço Python como citado na Seção 6.2.

O modelo criado nesse trabalho é capaz de inferir a quantidade de mosquitos que serão identificados em cada uma das regiões para *N* dias no futuro. Assim, com base nas informações históricas de identificações e clima, será retornada uma estimativa de número de mosquitos que será traduzido para uma valoração de grau de risco a ser apresentada ao usuário final da plataforma e colorir a região com verde, amarelo e vermelho conforme apresentado na Figura 5.17.

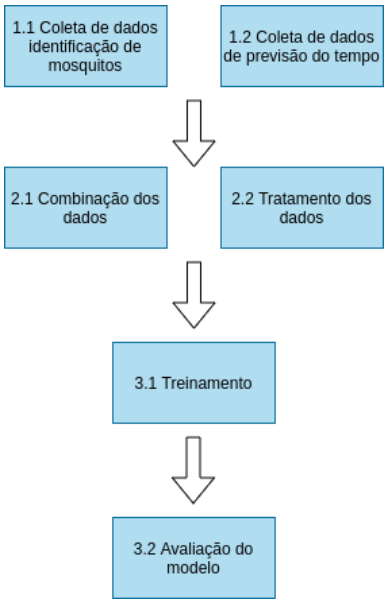
Para validar a hipótese de que é possível inferir a quantidade de mosquitos para uma determinada região com base em dados históricos de identificações temperaturas, umidade, pressão e chuva, apresentamos uma análise feita para extrair as correlações entre as séries temporais. Na sequência descrevemos os dados utilizados para o treinamento do modelo e os detalhes sobre o treinamento e validação do modelo.

7.1 Processo de Elaboração do Modelo

Os dados foram coletados, tratados e utilizados na construção do modelo e na avaliação. A Figura 7.1 mostra a metodologia seguida para a sua construção.

A unidade de análise para o nosso modelo é um segmento de área da região de interesse. Esse segmento é representado por um quadrado de lado igual a trezentos metros. A medida do lado está diretamente relacionada à mobilidade média de um mosquito da espécie *A. aegypti* em um território urbano (NASCIMENTO, 2017). O segmento unitário será chamado de *chunk*. Uma *chunk* pode também ser interpretada como um quadrante em um *grid* sobreposto ao mapa da cidade. A Figura 7.2 exemplifica uma região que foi segmentada em *chunks*.

Figura 7.1: Diagrama para o processo realizado para a elaboração do modelo



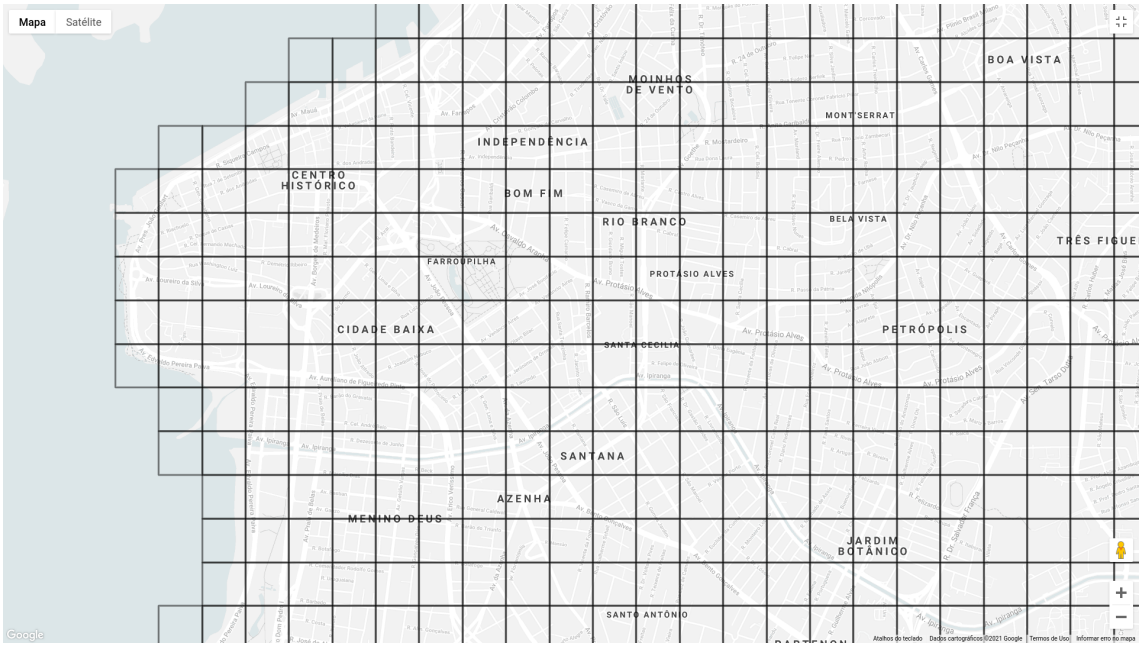
Fonte: O autor

7.2 Descrição dos Dados Utilizados para o Modelo

No intuito de criar o modelo preditivo com base na cidade alvo deste trabalho, tentamos ter acesso aos dados de identificações do mosquito que são capturados pela malha de armadilhas dispostas em Porto Alegre. Os dados não foram disponibilizados em tempo hábil para serem utilizados neste trabalho porém as condições pareceram favoráveis para, em um futuro próximo, eles serem cedidos para estudos. A seguinte alternativa foi adotada: utilizar os dados disponíveis para alguma outra região do mundo que tenha algumas características semelhantes ao clima da cidade alvo. Após certa pequisa, foram encontrados dados de identificações do mosquito para a cidade de Brownsville, no Texas - USA para o ano de 2017.

A Figura 7.3 mostra os históricos de identificações do mosquito *A. aegypti* (*y_aeg*), temperatura máxima(*t_max*), temperatura mínima (*t_min*), temperatura média (*t_average*) e precipitação (*precipitation*). Uma das formas de quantificarmos a relação entre diferentes séries temporais é através do cálculo da correlação entre elas.

Figura 7.2: Exemplo de grid sobrepondo uma região de interesse



Fonte: O autor

7.3 Análise de Correlação Entre as Variáveis do *Dataset*

Foram levadas em consideração as seguintes variáveis para a análise: quantidade de mosquitos identificados, precipitação, temperatura máxima, temperatura mínima, temperatura média e pressão. Os dados foram transformados em séries temporais utilizando a data como índice.

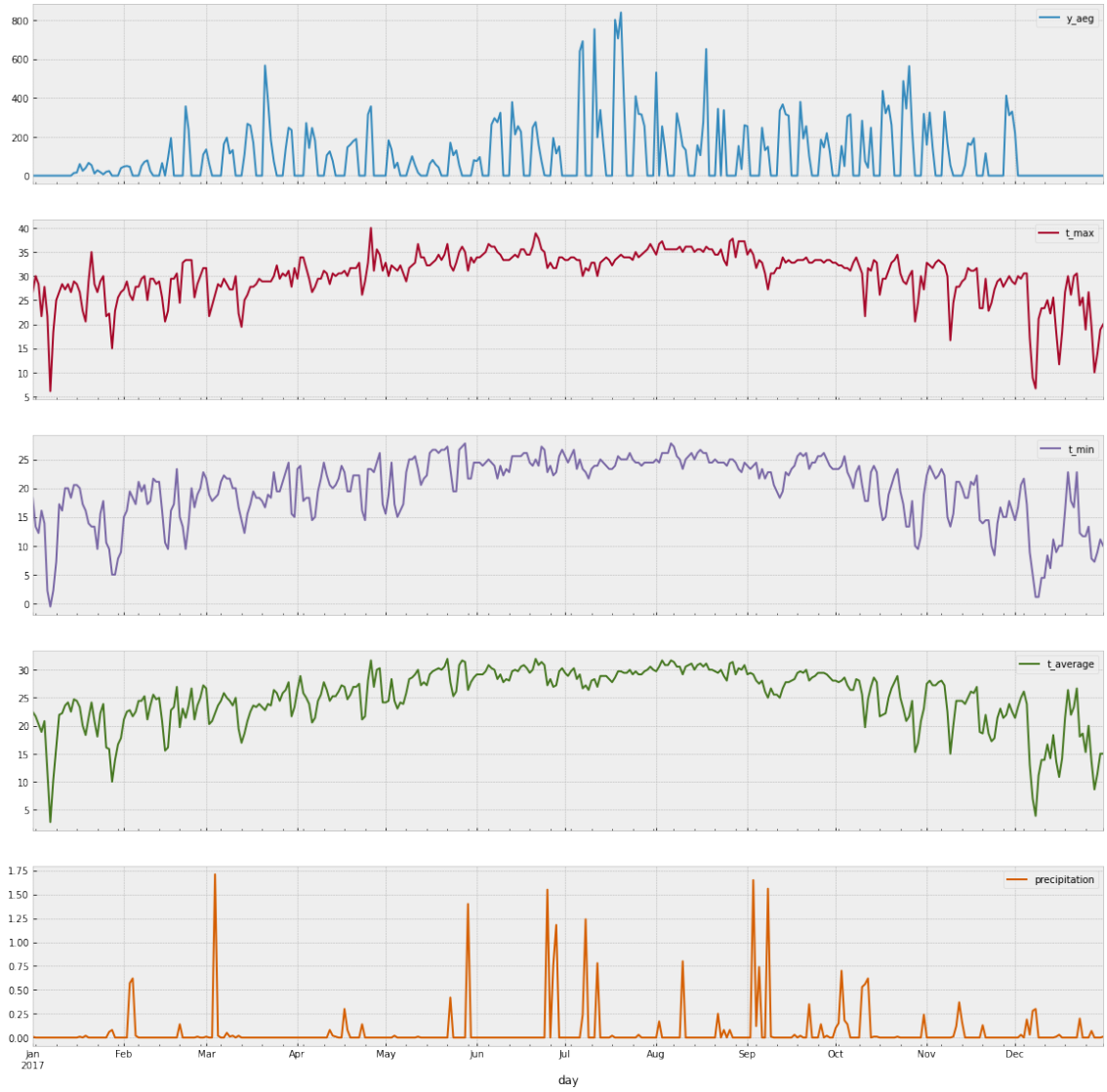
Os dados que temos compreendem valores para uma cidade inteira. O nosso objetivo é conseguir um modelo que consiga fazer predições na granularidade das *chunks*. Para chegarmos em algo próximo a isso, para cada dia, foi feita a média dos valores de identificações entre todas as armadilhas.

$$a_i = \frac{\sum_{j=0}^{n} B_{ij}}{n} \tag{7.1}$$

onde a_i é o valor da média para o dia i, B é a matriz que contém os valores de identificações onde as linhas representam os dias, as colunas representam as armadilhas e n é o número de armadilhas.

Com a finalidade de preencher os dias que não possuem registros de variáveis, foi aplicada uma interpolação linear nos valores das séries temporais. Um dos últimos tratamentos feitos nas séries de variáveis foi aplicar a média móvel para diferentes tamanhos de janela: sete, quatorze e trinta dias. A Figura 7.4 mostra os gráficos dos resultados.

Figura 7.3: Histórico dos dados para o ano de 2017 - Brownsville - TX

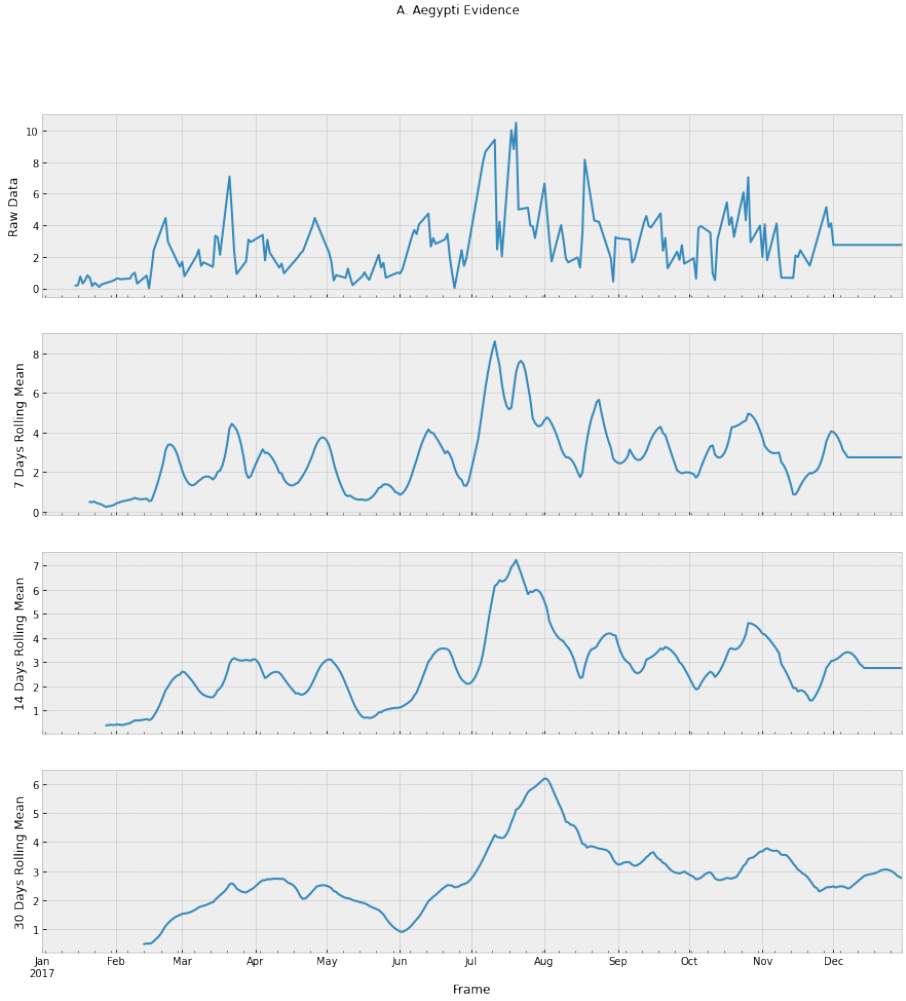


Fonte: O autor

Para o seguimento da análise, foi utilizado o resultado das séries com a média móvel que aplica uma janela deslizante de 30 dias. Uma função foi implementada para aplicar diferentes deslocamentos de dias para as séries que estamos comparando com a série da variável alvo (número de identificações) utilizando a correlação de Pearson (BENESTY et al., 2009). Calculando a média entre os melhores dias para cada uma das variáveis, tivemos que o melhor número geral para a aplicação do deslocamento para a esquerda, com relação à variável alvo, seria algo em torno de 36 dias.

Agora, vamos detalhar um pouco mais sobre essa análise feita. A correlação de Pearson (Equação 7.2) foi aplicada para cada uma das variáveis e seus deslocamentos temporais comparadas com a variável *target* de número de mosquitos. Aplicando um deslocamento negativo nos registros da variável avaliada, temos a noção de o quanto os

Figura 7.4: Médias Móveis Identificações dos Mosquitos



Fonte: O autor

dados do passado influenciam nas identificações do presente. A correlação de Pearson é uma ferramenta comumente utilizada para medir a relação linear entre duas variáveis normalmente distribuídas. Se o valor da correlação for 1, então ambas as séries são perfeitamente correlacionadas, se o valor for zero, então não há relação entre as variáveis, se o valor for -1, então ambas as variáveis possuem uma relação perfeitamente inversa. A Tabela 7.1 mostra os melhores valores do coeficiente de Pearson para a base de dados e os dias de deslocamento correspondentes.

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}} \tag{7.2}$$

Valores resultados das correlações são mais altos para as variáveis de temperatura e pressão. Ao analisar a variável de precipitação, podemos perceber que, para os dados utilizados, sua correlação com o número de identificações é baixo. O valor diferente de 1 encontrado para y_aeg se deve à aplicação da média móvel na série utilizada como *target*

Tabela 7.1: Melhores coeficientes para as variáveis

Variável	Melhor Deslocamento (dias)	Coeficiente de Pearson
y_aeg	0	0.43
Precipitação	60	0.32
Temperatura (máx.)	39	0.64
Temperatura (min.)	45	0.66
Temperatura (méd.)	39	0.65
Pressão	35	0.67

Fonte: O autor

para a função de correlação.

Nas Figuras 7.5 e 7.6 é possível notar que a correlações de temperatura são altas durante todo o período mas que sofrem um leve aumento próximo à 30 dias. Os valores de pressão se mantém praticamente constantes em um coeficiente de Pearson relativamente alto (próximo de 0.5). A correlação da precipitação com a quantidade de mosquitos sofre um grande aumento também aos 30 dias, porém o valor continua sendo baixo (próximo de 0.2).

Considerando as variações dos valores do coeficiente de Pearson para as variáveis estudadas, pode-se sugerir o uso das mesmas para tentar chegar a uma aproximação do valor de identificações para um dia no futuro. Seria uma possibilidade utilizar diferentes deslocamentos para cada uma das variáveis de acordo com os melhores valores encontrados, porém, para a simplificação da busca por um modelo capaz de resolver esse problema, foi utilizado algo próximo ao valor médio dos melhores dias, dados históricos de até 30 dias no passado.

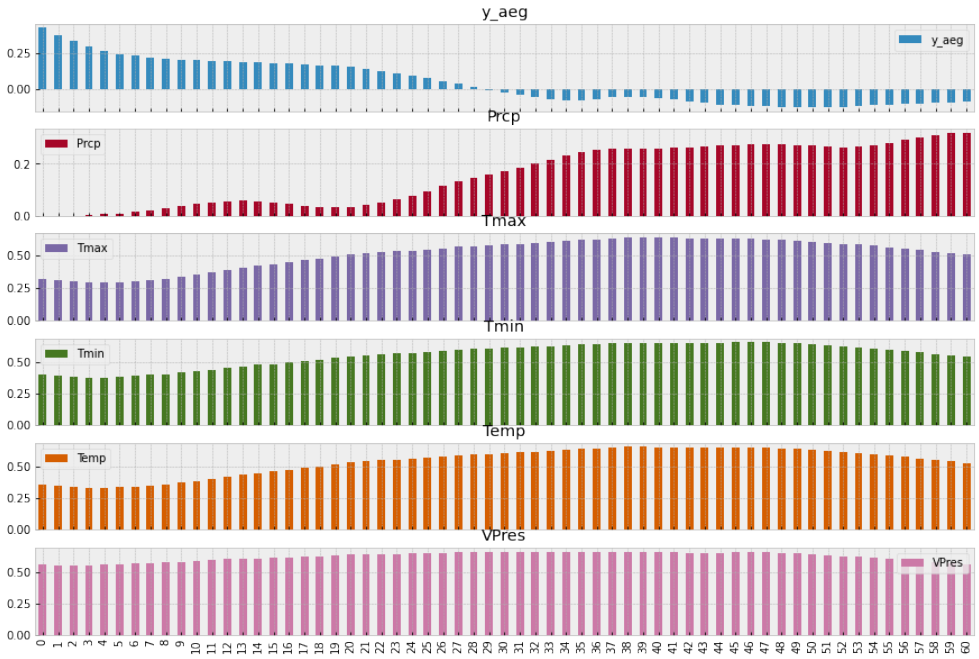
Também foi feita uma breve análise sobre de similaridade do percentual de variação entre as séries estudadas. Para isso foram plotados em diferentes gráficos os percentuais de mudança dos melhores deslocamento das variáveis no eixo (y) e no eixo (x) o percentual de mudança da variável alvo. Ao identificarmos *clusteres*, podemos dizer que as variáveis estão sofrendo variações de forma similar. Isso acontece praticamente para todas as variáveis do estudo. Além disso, pode-se calcular a distância entre as duas séries como a soma do módulo da diferença entre os valores das séries das variáveis do estudo a série da variável alvo, como na Equação 7.3. Temos como resultado valores entre 12,70 e 13,69 para as variáveis climáticas. Para a variável de identificações, como o melhor deslocamento é zero, a soma das distâncias entre cada ponto também será zero pois ambas as

séries são idênticas e variam com o mesmo percentual em todos os pontos.

$$d_{x,y} = \sum_{i=0}^{n} \|x_i - y_i\| \tag{7.3}$$

Nas seguintes seções, será apresentado um modelo que faz, como prova de conceito, a utilização dessas variáveis para tentar inferir valores de identificações no futuro.

Figura 7.5: Correlação das Variáveis com Identificações



Fonte: O autor

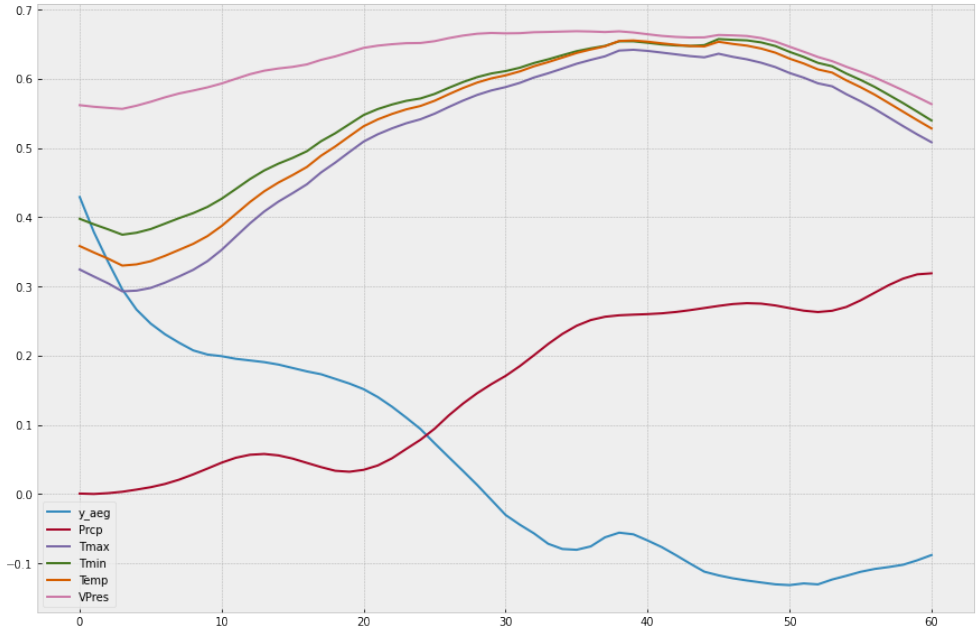
7.4 Estrutura dos Dados

Para o treinamento do modelo, foram utilizados duas fontes: uma tabela com o número de identificações do mosquito e outra com as dados sobre o clima para o ano de 2017. Nas seguintes subseções são descritos os campos utilizados como variáveis para o modelo.

7.4.1 Dados de Identificações

A Tabela 7.2 mostra os campos utilizados para o treinamento do modelo. Os dados foram coletados de uma plataforma que disponibiliza dados públicos ferentes a

Figura 7.6: Correlação das Variáveis com Identificações Linhas



Fonte: O autor

estudos. O arquivo das identificações foi encontrado em formato csv, importado e tratado em Python com a utilização da biblioteca Pandas. A base de dados utilizada é fruto do trabalho desenvolvido por Myer et al. (2020).

Tabela 7.2: Dados identificações utilizados no modelo

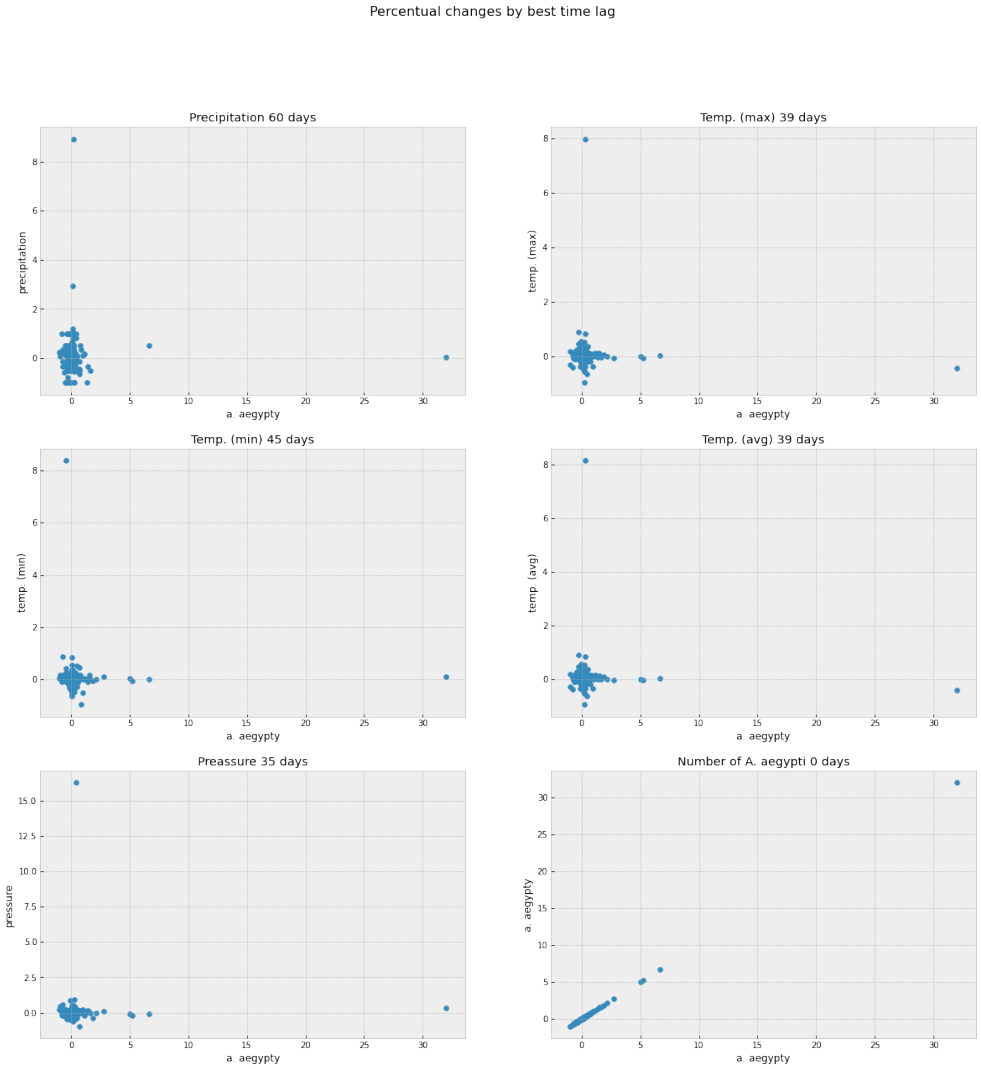
Nome do campo	**Tipo**	**Descrição**
Date	*data*	dia da identificação
y_aeg	N	número de mosquitos do tipo *A. aegypti* identificados
Lat	$\mathbb{R}$	latitude em que a identificação aconteceu
Long	$\mathbb{R}$	longitude em que a identificação aconteceu

Fonte: O Autor

7.4.2 Dados de Clima

O objetivo é ter as médias, máximas e mínimas diárias. Esses dados foram extraídos de uma plataforma que oferece dados meteorológicos. Os dados foram convertidos de graus Fahrenheit para Celsius. A Tabela 7.3 mostra os dados sobre clima utilizados (OCEANIC; ADMINISTRATION, 2021).

Figura 7.7: Percentual de Mudanças



Fonte: O autor

7.5 Tratamento dos dados

Para o modelo poder avaliar as condições diárias de um determinado número de dias no passado, é preciso ter as condições diárias de clima e número histórico de identificações para cada *chunk*. Para isso, foi feito um método que mapeia as coordenadas de cada identificação para uma determinada posição no *grid*. Com isso, foram adicionadas duas novas colunas nos dados: *gridLine* e *gridColumn*. A Tabela 7.4.

Como o nosso sistema é capaz de coletar os dados para cada *chunk* registrada, o modelo deve ser capaz de considerar diferentes valores para as condições climáticas também. Entretanto, os dados obtidos para a cidade que estamos utilizando como base para o treinamento são os mesmos para toda a extensão do seu território. Este fato se dá pois a API utilizada para coletar os dados históricos só permite fazer consultas com datas

Tabela 7.3: Dados do clima utilizados no modelo

Nome do campo	Tipo	Descrição
day	*data*	dia da identificação
t_max	$\mathbb{R}$	temperatura máxima registrada no dia
t_min	$\mathbb{R}$	temperatura mínima registrada no dia
t_average	$\mathbb{R}$	temperatura média do dia
precipitation	$\mathbb{R}$	volume de precipitação

Fonte: O Autor

Tabela 7.4: Dados agregados por dia e *chunk*

Nome do campo	Tipo	Descrição
Date	*data*	dia da identificação
gridLine	$\mathbb{N}$	linha do **chunk**
gridColumn	$\mathbb{N}$	coluna do **chunk**
y_aeg	$\mathbb{N}$	número de identificações de mosquitos *A. aegypti* agregado por chunk e dia

Fonte: O Autor

na granularidade de coordenadas no o alcance máximo de um ano no passado (no plano disponibilizado para estudantes). Para o treinamento, consideramos a cidade inteira como uma grande *chunk*, com a finalidade de treinar com os dados globais e depois podermos escalar o modelo para que ele trabalhe para as microrregiões de forma independente.

Na sequência, para facilitar o treinamento e definir uma instância com todos os atributos necessário para o processo, foi feita uma operação de mesclagem entre as tabelas 7.4 e 7.3 utilizando a data como chave primária. Assim, é possível dizer quais eram os valores da variáveis climáticas para cada leitura.

Até essa etapa do processamento, para cada *chunk* era possível recuperar os dados climáticos e o agregado de identificações para datas que haviam identificações. Dias sem identificações não tinham os valores sobre os parâmetros climáticos. Uma tarefa importante do processamento foi adicionar, para cada dia do ano de referência e *chunk* que não possuíam valores identificações, os dados climáticos referentes ao dia com o valor da coluna *"y_ae"* setado em zero. Com isso, garantirmos que temos valores para todos os dias do período da base de treinamento.

7.6 Referências para o Treinamento

Para a implementação do modelo, foram utilizadas bibliotecas que auxiliaram o processo de desenvolvimento. A seguir, essas bibliotecas são descritas.

Tensorflow (ABADI et al., 2015) é uma plataforma que oferece uma solução fim a fim que disponibiliza algoritmos de aprendizado de máquina cuja computação pode ser executada em uma abrangente variedade de sistemas com miníma ou nula mudança no código. Além de disponibilizar a interface para a execução de algoritmos de aprendizado de máquina, o Tensorflow também permite a o *deploy* de modelos em ambientes produtivos para servir clientes que irão consumi-los.

Keras (CHOLLET et al., 2015) é uma interface de alto nível para o Tensorflow. Ela permite executar o processo de prototipação e execução de modelos com maior velocidade. O seu foco é em *deep learning*.

7.7 Definição do Problema de Treinamento

O problema que o trabalho aborda envolve séries temporais e pode ser categorizado como um problema de regressão, ou seja, nosso objetivo será tentar achar uma função que mapeia dados históricos dos N últimos dias para um valor que tenta se aproximar do valor futuro de uma determinada variável, no caso, número de identificações. O problema também poderia ser modelado como um problema de classificação, tendo como saída três tipos de categorias: risco baixo, médio e alto. Porém caso nossa escolha fosse modelar como classificação, teríamos de fazer um processo de mapeamento dos valores de identificações históricos para as três categorias previamente ao treinamento, perdendo, assim, informação pela discretização aplicada.

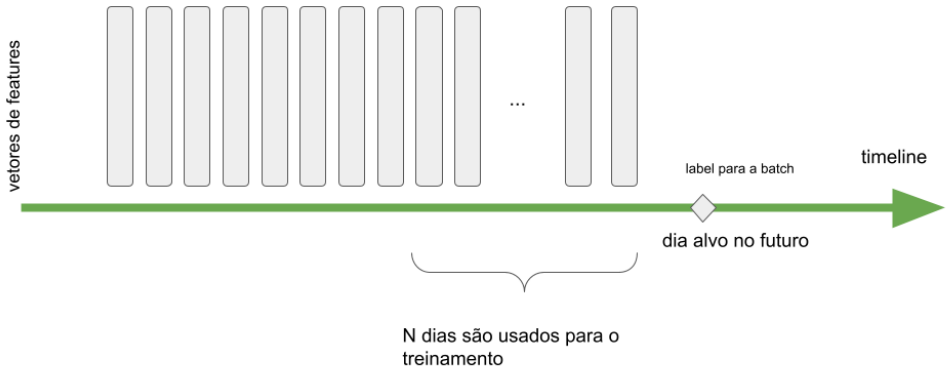
As Figuras 7.8 e 7.9 mostram duas interações do processo de treinamento. Cada *batch* (nesse contexto de treinamento, a *batch* está sendo considerada como o seguimento de dias com os valores das *features*) vai considerar os N dias do passado para calcular a predição que será comparada com o valor da label da variável *target* do futuro. Cada interação soma um dia na data base que é utilizada para considerar os últimos N dias e dia *target* (a *label* é o rótulo que corresponde ao valor real). A característica serial do treinamento (deslisamento do período dos dias considerados) foi adotada para podermos

Figura 7.8: Treinamento (primeira interação)

Fonte: o Autor

tirar vantagem das camadas LSTM que incorporamos na rede. Assim, mesmo que os parâmetros dos dias de um passado distante não estejam sendo passados para a *bath*, estarão tendo alguma influência na computação por conta da memória que esse tipo de camada implementa.

Figura 7.9: Treinamento (interação N)



Fonte: o Autor

7.7.1 Arquitetura da Rede

A Figura 7.10 representa o modelo criado para tentar prever o número de identificações. Sua camada de entrada espera um *input* de três dimensões onde as duas últimas

representam número de dias históricos e número de *features*, respectivamente. A primeira dimensão espera nulo mas é necessária para que o treinamento possa ser feito utilizando o processo de *batching*. Na sequência, o modele possui uma camada LSTM que recebe as conexões da camada de entrada e retorna sequências que são encaminhadas para uma camada do tipo *dense* que direciona a sua saída para outra camada LSTM. A segunda camada LSTM se conecta com uma camada de *dropout* que ajuda a prevenir o fenômeno de *overfitting*. Na sequência o modelo possui duas camadas do tipo *dense* onde a última tem como saída um único valor que corresponde ao número de identificações da previsão.

A concepção da arquitetura se baseou em teste, porém o processo de alteração da rede não foi documentado por não se o foco do trabalho, esse processo de documentação, comparação de diferentes arquiteturas é uma oportunidade de trabalhos futuros para esse tema. Isso poderá ser melhor desempenhado, ao se conseguir acesso a mais dados sobre as identificações de mosquito nas regiões de interesse.

7.7.2 Pré-Processamento

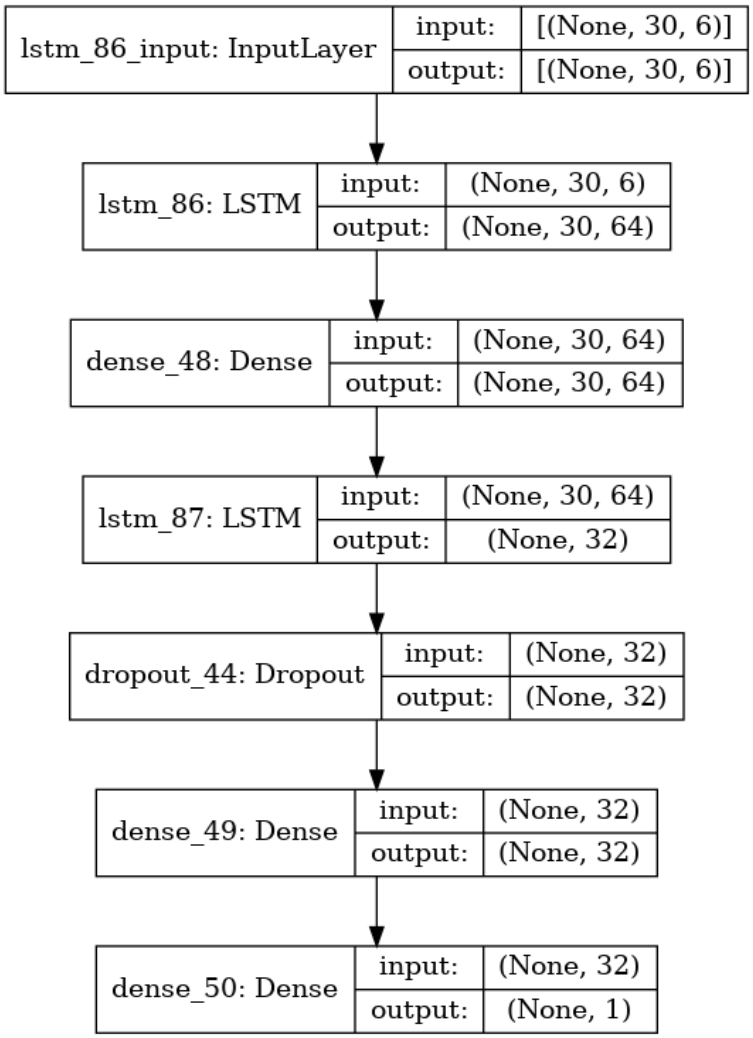
Para podermos utilizar os dados no modelo, foram aplicadas algumas boas práticas como a remoção dos valores faltantes e normalização dos valores a partir de uma transformação de escala. A função utilizada para escala é salva no repositório para ser incorporada no serviço de predição.

Com os valores propriamente normalizados, foram criadas as instâncias. Uma instância é composta por 30 valores das 6 variáveis utilizadas. Cada instância é uma janela na série temporal. Para treinamento foi usado 90% dos valores do banco de dados. Para validação foram utilizados os outros 10%, esse processo de separação foi feito na etapa de pré-processamento.

7.7.3 Fit

Para treinar o modelo, foram utilizadas as instâncias resultantes do pré-processamento. O modelo foi compilado utilizando o algoritmo de otimização socástica Adam (KINGMA; BA, 2014), com a função de perda *Mean Squared Error*. O objetivo do modelo é minimizar o erro quadrado. A função *fit* foi utilizada para efetuar o treinamento. O número de *epochs* utilizado foi 500. A Figura 7.11 mostra a evolução da métrica *Mean Squared*

Figura 7.10: Arquitetura do Modelo



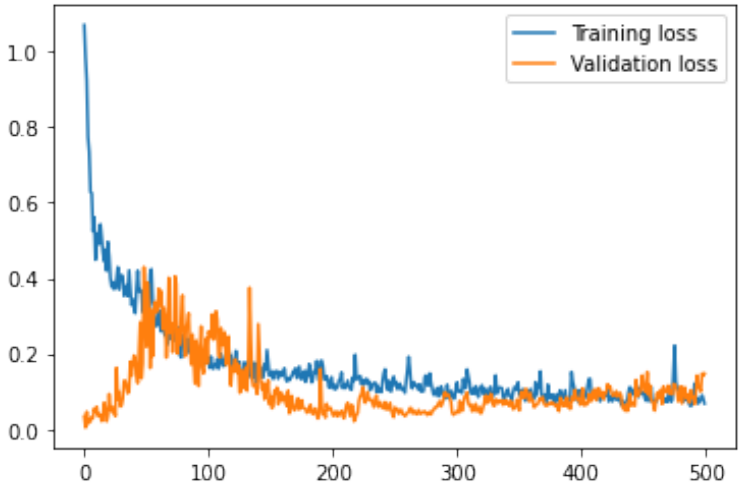
Fonte: o Autor

Error a cada *epoch* do treinamento.

Ao analisarmos a Figura 7.11, podemos perceber que para os dados do treinamento, o modelo conseguiu minimizar o erro. Porém, para os dados de validação, o modelo acabou piorando nas primeiras iterações e depois conseguiu ser minimizado para algo próximo aos valores de erro alcançados para os dados de treinamento.

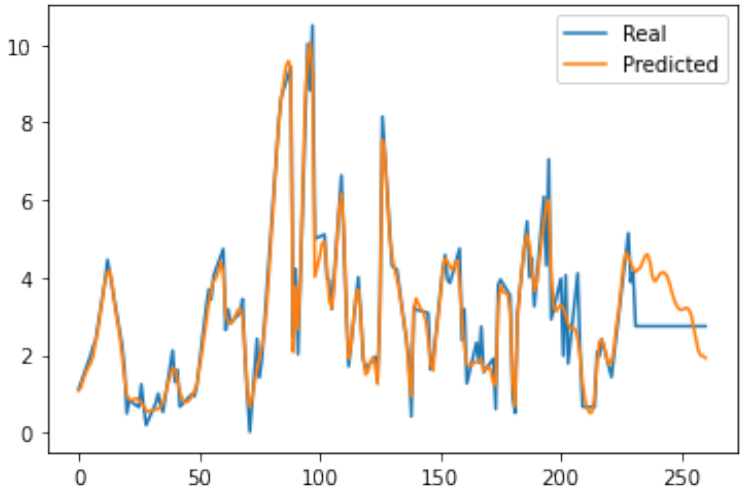
A imagem 7.12 mostra os valores de previsão (em laranja) comparados com os valores reais das identificações do mosquito (azul). O modelo conseguiu se moldar para o problema de forma satisfatória (considerando os dados do treinamento). Porém, mais validações seriam necessárias para considerar o modelo útil pois podemos estar observando o fenômeno de *overfitting*. Uma vez que o modelo for exposto a mais dados, poderemos ter uma melhor noção sobre a sua efetividade. Isso pode ser considerado em trabalhos futuros que tiverem acesso a mais dados históricos.

Figura 7.11: *Loss* do treinamento



Fonte: o Autor

Figura 7.12: Predições versus real



Fonte: o Autor

7.7.4 *Deploy* do Modelo

Para que o modelo possa ser disponibilizado para o consumo por diferentes clientes, duas coisas do processo de treinamento precisam ser salvas: a arquitetura da rede com os seus pesos e o objeto utilizado para escalar (normalizar) os dados de treinamento. A arquitetura da rede com os pesos será carregada no serviço de predições para que as predições possam ser feitas.

Como o modelo foi treinado com os dados escalados por uma função de normalização, essa função precisa ser de conhecimento do servidor de predições também, para que os novos dados utilizados nas predições também possam ser normalizados antes da aplicação da função *predict*. O resultado da função *predict* sobre os dados normalizados

precisa sofrer a função inversa da normalização para que os dados estejam no mesmo domínio de valores dos dados utilizados para a execução da predição.

7.7.5 Sobre Novos Modelos

O modelo criado como prova de conceito considerou 7 dias no futuro para as predições como mencionado nas seções anteriores. Porém novos modelos podem ser construídos com a finalidade de fazer predições para um valor diferente de dias no futuro, isso seria necessário para poder cobrir a funcionalidade proposta na tela de predições com o *slider* de até 30 dias. Em um contexto real de uso do sistema, poderia-se, por exemplo ter um modelo que enxerga 30 dias para o futuro outro 14 e um terceiro com uma visão de 7 dias. Assim a cada dia, esses três modelos armazenariam os valores de suas predições em suas respectivas tabelas. Na hora de mostrar os dados na interface do usuário, o sistema poderia sempre optar por mostrar as predições do modelo com a visão mais mais próxima do dia corrente.

A estratégia de manter os valores das predições precisaria ser modelada a nível de banco de dados para comportar essa funcionalidade. Porém a utilização de diferentes tabelas (ou então de uma coluna para identificar o modelo na tabela de predições) traria uma flexibilidade de possibilitar auditorias e avaliações de desempenho dos diferentes tipos de modelos. No contexto do trabalho, se diferentes modelos forem utilizados para prever valores para diferentes números de dias no futuro, os valores preditos por modelos com uma visão mais distante seriam sobrescritos pelos valores dos modelos com uma visão de curto alcance.

8 CONCLUSÃO

Nesse trabalho foi projetado e implementado um sistema capaz de proporcionar visualizações sobre detecções do mosquito *A. aegypti*. Foram definidas interfaces com o usuário que possuem a finalidade de facilitar o trabalho dos agentes responsáveis pela tomada de decisão de onde disferir ações de controle e de prevenção para evitar a proliferação do mosquito. A aplicação web foi desenvolvida em JavaScript e ReactJs com componenetes SAP Fiori. Também foi utilizada uma API do Google Maps que possibilitou a interação com o recurso de mapa para a criação das visualizações.

Foi definido um processo de coleta e tratamento de dados automatizado para variáveis climáticas. Também foram definidas APIs para a comunicação com o serviço de *backend* feito em Java com o *framework Spring*. Como parte do processo, os dispositivos móveis (celulares ou armadilhas autônomas) informam o sistema a cada detecção positiva, fechando assim o ciclo de alimentação de dados. Foram definidos processos capazes de processar os registros armazenados e gerar as predições que serão mostradas nas interfaces. O conjunto de tabelas definido no banco de dados torna possível a extração dos dados de forma fácil para serem disponibilizados para outros trabalhos e, eventualmente, para avaliações da qualidade das predições feitas.

Para fins de prova de conceito, foram feitas análises nas variáveis utilizadas no estudo, comparando as correlações entre elas e a variável alvo. Tal avaliação sugeriu que os dados podem ter correlação e, assim, podem contribuir para a tarefa de inferir valores de detecções futuras. Foi definida uma rede neural para trabalhar com séries temporais e executar a tarefa de regressão com a finalidade de se aproximar do valor de identificações previstas. A rede foi utilizadas para treinar um modelo que utilizou como dados de entrada valores de identificações de uma região onde haviam identificações por um período de um ano. O modelo treinado foi avaliado utilizando a métrica de erro quadrático (*mean squared error*) e pode-se notar que houve uma adaptação do modelo para os dados que foram utilizados no treinamento.

8.1 Limitações

O objetivo inicial do trabalho era a utilização de dados de históricos identificações dos mosquitos para a cidade de Porto Alegre no treinamento do modelo com técnicas de aprendizado de máquina. Essa abordagem foi possível pois não obtivemos acesso aos

dados durante o desenvolvimento do projeto. Nesse sentido, o modelo que discutimos no trabalho foi idealizado com os dados de outra cidade que, apesar de ter uma certa semelhança em termos de clima, há diferentes características com relação à cidade de Porto Alegre. A criação do modelo envolve envolve a abstração das *chunks* (microrregiões) por todo o território da cidade utilizada no treinamento, algo que pode levar à uma perda de informação por se utilizar a média dos valores de todas as *chunks* para o treinamento. Tal tipo de abstração necessita mais validações e testes com mais dados para podermos dizer se ela tem uma boa generalização.

Além do que foi citado acima, o modelo utilizado para computar as previsões de quantidades dos mosquitos *A. aegypti* não foi validado a ponto de se ter uma confiança em sua precisão devido a carência de dados de detecções dos mosquitos. Esse problema deve ser superado com o acesso aos dados históricos relacionados aos esforços atuais de monitoramento. Porém, as características dos dados disponíveis de forma histórica diferem do que o sistema elaborado no trabalho propõe, essa diferença está relacionada ao represamento semanal dos dados. Ao colocar o sistema em produção e ele passar a receber todos os dias as identificações oriundas de dispositivos capazes de detectar o mosquito de forma autônoma, pode-se considerar um novo treinamento que levará em conta essa nova frequência de alimentação, o que pode acarretar em uma maior precisão por parte do modelo.

O trabalho tenta promover, como citado no parágrafo anterior, a utilização de dados dinâmicos e fazer a computação das previsões de forma diária. Esse processo seria possível através da implementação da malha de armadilhas com dispositivos autônomos na região de interesse para o estudo, no caso, Porto Alegre. Esses dispositivos, no momento da escrita do trabalho, estão sendo desenvolvidos por outros grupos de pesquisa, suas viabilidades estão sendo consideradas e podem vir, no futuro, a se tornar a forma padrão de monitoramento do vetor. Até isso acontecer, deve-se considerar a validação da solução em áreas menores, controle e testes. Deve haver, portanto, uma colaboração entre o desenvolvimento de um sistema de monitoramento do padrão de disseminação do *A. aegypti* (como o que se propõe esse trabalho) com os grupos de pesquisa que estão desenvolvendo os dispositivos autônomos de detecção.

8.2 Trabalhos Futuros

O sistema proposto pode ser melhorado no sentido de ser valido com as pessoas que fazem o monitoramento de vetores. A plataforma pode ser generalizada para poder funcionar com qualquer região do mundo, para isso precisaria ser feita uma melhor integração com sistemas de capazes de fornecer as divisões administrativas desejadas (bairros, por exemplo). O desenvolvimento da plataforma pode ser melhorado no sentido de implementar padrões de consultas que melhoram a experiência do usuário e reduzem os tempos de carregamento para das páginas de visualização.

Melhorias podem ser feitas no modelo de previsão e também sugere-se o desenvolvimento de uma interface que recebe o modelo como parâmetro tornando possível a substituição do agente responsável pela predição conforme a necessidade. Ainda sobre o modelo, a forma como os modelos armazenam os dados pode ser melhorada tal como sugerido na Seção 7.7.5. Também fica como sugestão, o uso de variáveis socio econômicas atreladas às regiões para o treinamento do modelo.

Sugere-se que sejam desenvolvidos dispositivos IoT capazes de executar o algoritmo de identificação e de fazer chamadas HTTP para informar o sistema sobre as identificações. Com uma maior cobertura da cidade e da instantaneidade das identificações, o sistema pode ser melhor validado e melhorado, abrindo a possibilidade para o surgimento de modelos preditivos mais eficazes que podem ser integrados com o sistema.

REFERÊNCIAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Available from Internet: <https://www.tensorflow.org/>.

ALPAYDIN, E. **Introduction to machine learning**. [S.l.]: MIT press, 2020.

ARTHUR, B. J. et al. Mosquito (aedes aegypti) flight tones: Frequency, harmonicity, spherical spreading, and phase relationships. **The Journal of the Acoustical Society of America**, Acoustical Society of America, v. 135, n. 2, p. 933–941, 2014.

BEE, T. K.; LYE, K. H.; YEAN, T. S. Modeling dengue fever subject to temperature change. In: IEEE. **2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery**. [S.l.], 2009. v. 5, p. 61–65.

BENESTY, J. et al. Pearson correlation coefficient. In: **Noise reduction in speech processing**. [S.l.]: Springer, 2009. p. 1–4.

BHATT, S. et al. The global distribution and burden of dengue. **Nature**, Nature Publishing Group, v. 496, n. 7446, p. 504–507, 2013.

BRADY, O. J. et al. Refining the global spatial limits of dengue virus transmission by evidence-based consensus. Public Library of Science San Francisco, USA, 2012.

CHOLLET, F. et al. **Keras**. 2015. <https://keras.io>.

CONTROL, C. for D.; PREVENTION. **Life Cycle of Aedes aegypti and Ae. albopictus Mosquitoes**. 2020. <https://www.cdc.gov/mosquitoes/about/life-cycles/aedes.html>.

ECOVEC. **Plataforma Onde Está o Aedes**. 2021. <http://ondeestaoaedes.com.br/>.

FERNANDES, M. S.; CORDEIRO, W.; RECAMONDE-MENDOZA, M. Detecting aedes aegypti mosquitoes through audio classification with convolutional neural networks. **Computers in Biology and Medicine**, Elsevier, v. 129, p. 104152, 2021.

GOOGLE. **Material Design**. 2021. <https://material.io/design/guidelines-overview>.

GRINBERG, M. **Flask web development: developing web applications with python**. [S.l.]: "O'Reilly Media, Inc.", 2018.

GROOMET. **v2**. 2021. <https://v2.grommet.io/>.

GUTIERREZ, F. **Pro Spring Boot**. [S.l.]: Springer, 2016.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KUMAR, S.; DALAL, S.; DIXIT, V. The osi model: Overview on the seven layers of computer networks. **International Journal of Computer Science and Information Technology Research**, v. 2, n. 3, p. 461–466, 2014.

LI, W.; CHEN, Y. Risk factor identification and spatiotemporal diffusion path during the dengue outbreak. In: IEEE. **2016 4th International Workshop on Earth Observation and Remote Sensing Applications (EORSA)**. [S.l.], 2016. p. 348–352.

LUZA, A. L. et al. Mapeamento dinâmico da probabilidade de infestação por vetores urbanos de arbovírus nos municípios do rio grande do sul, 2016-2017. **Epidemiologia e Serviços de Saúde**, SciELO Public Health, v. 30, p. e2020154, 2021.

MCKINNEY, W. **Python for data analysis: Data wrangling with Pandas, NumPy, and IPython**. [S.l.]: "O'Reilly Media, Inc.", 2012.

MORES, G. B. et al. Site occupancy by aedes aegypti in a subtropical city is most sensitive to control during autumn and winter months. **The American journal of tropical medicine and hygiene**, The American Society of Tropical Medicine and Hygiene, v. 103, n. 1, p. 445, 2020.

MYER, M. H. et al. Mapping aedes aegypti (diptera: Culicidae) and aedes albopictus vector mosquito distribution in brownsville, tx. **Journal of medical entomology**, Oxford University Press US, v. 57, n. 1, p. 231–240, 2020.

NASCIMENTO, F. A. M. d. **Padronização e implementação do uso de armadilhas de oviposição nas ações de monitoramento do mosquito Aedes aegypti (Diptera: Culicidae) no município de Natal, RN**. Dissertation (Master) — Brasil, 2017.

NOMINATIM. **Nominatim**. 2021. <https://nominatim.openstreetmap.org/ui/search.html>.

NUNES, L. R. Aplicativo móvel para mapeamento colaborativo de focos de aedes aegypti utilizando técnicas de machine learning. 2020.

OCEANIC, N.; ADMINISTRATION, A. **National Weather Service**. 2021. <www.weather.gov>.

ONG, S.-Q. et al. Implementation of a deep learning model for automated classification of aedes aegypti (linnaeus) and aedes albopictus (skuse) in real time. **Scientific Reports**, Nature Publishing Group, v. 11, n. 1, p. 1–12, 2021.

ONLINE, S.; NET, S. Monitoramento dos casos de arboviroses urbanas causados por vírus transmitidos por aedes (dengue, chikungunya e zika), semanas epidemiológicas 1 a 53, 2020. v. 52, n. Tabela 1, p. 1–31, 2021.

OPENWEATHER. **Weather API**. 2021. <https://openweathermap.org/api>.

PISA, F. di; PISA, F. di. Inversion of control. **Beginning Java™ and Flex: Migrating Java, Spring, Hibernate, and Maven Developers to Adobe Flex**, Springer, p. 89–129, 2009.

RODRÍGUEZ, C. et al. Rest apis: a large-scale analysis of compliance with principles and best practices. In: SPRINGER. **International conference on web engineering**. [S.l.], 2016. p. 21–39.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. **Learning internal representations by error propagation**. [S.l.], 1985.

SAP. **Fiori Guidelines**. 2021. <https://experience.sap.com/fiori-design/>.

SAP. **Fiori ReactJs library**. 2021. <https://sap.github.io/ui5-webcomponents-react/>.

SAÚDE, M. da. **Diretrizes Nacionais para a Prevenção e Controle de Epidemias de Dengue**. 2009. <https://bvsms.saude.gov.br/bvs/publicacoes/diretrizes_nacionais_prevencao_controle_dengue.pdf>.

SAÚDE, M. da. **DATASUS**. 2021. <https://datasus.saude.gov.br/>.

WALLS, C. **Spring Boot in action**. [S.l.]: Simon and Schuster, 2015.

WILSON, A. L. et al. The importance of vector control for the control and elimination of vector-borne diseases. **PLoS Neglected Tropical Diseases**, Public Library of Science, v. 14, n. 1, p. 1–31, 1 2020. ISSN 19352735. Available from Internet: <https://doi.org/10.1371/journal.pntd.0007831.t002>.

APÊNDICE A — DEFINIÇÕES DAS APIS

GET - /identifications: retorna todas as identificações do sistema.

POST - /identifications: adiciona a identificação informada no *payload* da *request*.

PATCH - /identifications: edita uma identificação conforme os valores informados no *payload* da *request*.

POST - /identifications/batch: adiciona múltiplas identificações ao mesmo tempo.

GET - /identifications/<identificationId>: retorna a identificação cujo o id seja igual ao valor informado em <*identificationId*>.

GET - /location/<location>: retorna as identificações uma localidade (bairro) que tenha o mesmo código informado em <*location*>.

GET - /getLocationIdentificationsBetweenDates: retorna todas as identificações de uma determinada localidade e período. Os parâmetros *locationId, startDate* e *endDate* são esperados.

GET - /getLocationIdentificationsBetweenDates: retorna todas as identificações de uma determinada localidade e período. Os parâmetros *locationId, startDate* e *endDate* são esperados para a delimitação do período.

GET - /getAllIdentificationsBetweenDates: retorna todas as identificações entre as duas datas *startDate* e *endDate*.

GET - /getNumberOfIdentificationPerLocationBetween: retorna a soma de identificações de cada bairro dentro do período determinado pelas datas *startDate* e *endDate*.

GET - /getIdentificationDayCounts: tem como retorno o número de identificações geral de cada dia dentro do período definido pelas datas *startDate* e *endDate*.

GET - /geo/chunks: retorna todas a microrregiões do sistema.

POST - /geo/chunks: adiciona a microrregião passada no corpo da requisição.

POST - /geo/chunks/batch: adiciona a lista de objetos do tipo microrregião passada no corpo da requisição.

GET - /geo/locations: retorna todas as localizações (bairros) do sistema.

POST - /geo/locations/batch: adiciona a lista de objetos do tipo location (bairros) passada no corpo da requisição.

GET - /geo/getNumberOfChunksPerLocationId: retorna o número de microrregiões que intersectam o bairro que tem o mesmo código da *locationId* informada como parâmetro.

GET - /geo/getLocationChunks: retorna todas as microrregiões que intersectam o bairro que tem o mesmo código da *locationId* informada como parâmetro.

GET - /geo/getNumberOfChunksWithIdentificationsForecastByLocationIdBetweenPeriod: retorna a contagem de microrregiões que possuem predições para o período especificado pelos parâmetros *startDate* e *endDate* para a região que possui o código especificado pelo parâmetro *locationId*.

GET - /geo/getLocationChunksStatistics: retorna informações estatísticas para uma determinada localidade especificada pelo parâmetro *locationId* e utilizando como data base para os cálculos o valor informado no parâmetro e *baseDate*.

GET - /trap: retorna todas as armadilhas cadastradas.

POST - /trap: adiciona uma armadilha com os dados no objeto que acompanha a requisição.

POST - /trap/identification: adiciona as identificações de forma que a contagem informado na propriedade *numberOfIdentifications* é adicionada na localidade da armadilha que tiver o mesmo nome do valor informado em *trapName*.

GET - /predict: retorna as predições de todas as microrregiões.

POST - /predict: executa as rotinas de predição utilizando como base a data passada no parâmetro *prediction_base_date*.