WEBGIS NA IDENTIFICAÇÃO DE ÁREAS DE ALTA INCIDÊNCIA DE DENGUE.

Guilherme Fragnani Bez Fontana[1], Ana Claudia Garcia Barbosa[2]

Resumo:
O seguinte trabalho tem como objetivo o desenvolvimento de uma plataforma computacional para análise espacial de dados epidemiológicos da dengue, integrando tecnologias de geoprocessamento e banco de dados geoespaciais. O sistema foi implementado utilizando PostgreSQL com extensão PostGIS para armazenamento e consulta de dados, aliado a bibliotecas de visualização para geração de mapas temáticos interativos e mapas de calor. Os resultados buscam demonstrar uma abordagem na identificação de padrões espaciais da doença, permitindo a associação com fatores ambientais e socioeconômicos. A plataforma mostrou-se capaz de transformar dados brutos em informações geoespaciais estratégicas para vigilância em saúde pública, destacando áreas prioritárias para intervenção. Apesar das limitações em parâmetros fixos de análise, a ferramenta oferece uma solução acessível para monitoramento epidemiológico, com potencial de expansão. Como contribuição, o trabalho evidencia a importância da computação aplicada à saúde pública, proporcionando uma base tecnológica para tomada de decisão baseada em evidências geográficas. O estudo reforça o papel das tecnologias da informação como aliadas no enfrentamento de desafios em saúde coletiva.

Palavras-chave: geoprocessamento; desenvolvimento de software; webgis; mapa de calor; dengue

[1]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), guilherme.fragnani@unesc.net
[2]Curso de Ciência da Computação, Universidade do Extremo Sul Catarinense (Unesc), agb@unesc.net

ABSTRACT: The objective of this work is the development of a computational platform for the spatial analysis of dengue epidemiological data, integrating geoprocessing technologies and geospatial databases. The system was implemented using PostgreSQL with the PostGIS extension for data storage and querying, combined with visualization libraries for generating interactive thematic maps and heat maps. The results aim to demonstrate an approach for identifying spatial patterns of the disease, allowing for associations with environmental and socioeconomic factors. The platform proved capable of transforming raw data into strategic geospatial information for public health surveillance, highlighting priority areas for intervention. Despite limitations due to fixed analysis parameters, the tool offers an accessible solution for epidemiological monitoring, with potential for expansion. As a contribution, this work highlights the importance of computing applied to public health, providing a technological foundation for evidence-based decision-making using geographic data. The study reinforces the role of information technologies as key allies in addressing public health challenges.

Keywords: geoprocessing; software development; web-gis; heat map; dengue

1 INTRODUÇÃO

O presente trabalho aborda a relevância do geoprocessamento na área da Ciência da Computação, especialmente em sua aplicação na saúde pública. O geoprocessamento, tecnologia que permite a coleta e analise de dados demográficos, demonstra relevância na identificação de áreas de risco de doenças como a dengue, auxiliando no planejamento, monitoramento e avaliação de ações em saúde. Seu uso por secretarias de saúde para mapear áreas urbanas e direcionar intervenções está alinhado com as diretrizes do SUS. A eficácia de mapas temáticos na pesquisa e controle de áreas, aplicada desde a descrição de endemias até a análise de vulnerabilidade social, contribui para a reorganização da rede de assistência a saúde (Nardi et al., 2013).

O problema de pesquisa centraliza-se na crescente incidência da dengue, uma das doenças transmitidas por mosquitos mais comuns no mundo, com um impacto global significativo evidenciado pelo aumento de casos e mortes (Procopio et al., 2024). Fatores como urbanização, mudanças climáticas e controle inadequado de vetores contribuem para a disseminação da doença. Neste contexto, a computação permite o processamento

e análise de grandes volumes de dados epidemiológicos e geográficos por meio de técnicas como geoprocessamento, análise espacial e visualização de dados. Essas ferramentas facilitam a identificação de padrões de propagação, a detecção de áreas de risco e a otimização de estratégias de intervenção, tornando-se aliadas essenciais no combate a dengue.

A justificativa para a escolha do tema reside no potencial do desenvolvimento de uma solução de geoprocessamento para identificar áreas de alta incidência de dengue de forma mais eficaz. A utilização de técnicas de geoprocessamento e ferramentas como o PostGIS permite uma análise ágil de dados georreferenciados, facilitando a tomada de decisões para o controle e prevenção da doença.

O objetivo geral deste trabalho é desenvolver uma solução baseada em técnicas de geoprocessamento para facilitar a análise, identificação e monitoramento de focos de dengue, contribuindo para o controle e prevenção da doença.

Os objetivos específicos deste trabalho consistem em organizar informações de saúde relacionadas aos casos de dengue e dados geográficos de forma estruturada e acessível, aplicando técnicas de geoprocessamento para georreferenciar e visualizar a distribuição espacial da doença em uma interface clara, o que facilitará a análise espacial. Além disso, busca-se realizar análises de densidade espacial por meio de mapas de calor, identificando áreas de alta incidência e concentração de casos, com o intuito de auxiliar na detecção de possíveis focos da doença em uma determinada região, contribuindo assim para um monitoramento mais direcionado.

A revisão de literatura sobre estudos similares demonstra o crescente interesse na aplicação do geoprocessamento para a análise da dengue, como evidenciado nos trabalhos de Magalhaes (2012) sobre o caso de Fortaleza, e o estudo na região leste de Teresina de Rocha et al. (2021). O presente trabalho se diferencia pela utilização de mapas de calor gerados dinamicamente e pela implementação de um sistema Web-GIS que oferece uma visualização em tempo real e uma interação mais contínua com os dados.

O desenvolvimento desta aplicação contou com o uso de diversos *softwares*, *frameworks* e bibliotecas para otimizar a implementação e aprimorar a eficiência do sistema. Para a estruturação do backend e frontend, foram utilizados NestJS e React. O banco de dados foi implementado com PostgreSQL, em conjunto com a biblioteca PostGIS, que facilita o

armazenamento e manipulação de dados geoespaciais. Além disso, a geração e visualização dos mapas foram viabilizadas por meio da biblioteca Mapbox, enquanto a API Nominatim foi utilizada para a funcionalidade de geocodificação reversa, permitindo a conversão de endereços em coordenadas geográficas.

A aplicação integra o mapa no frontend com as geometrias armazenadas no banco de dados, possibilitando aos usuários criar e visualizar dados espaciais interativamente. Esse recurso permite mapear e representar pontos de possíveis casos de dengue. Adicionalmente, a implementação de mapas de calor permite uma análise dinâmica das áreas com maior concentração de casos suspeitos, facilitando a identificação de possíveis regiões críticas.

Dada sua aplicação na área da saúde pública, o sistema possibilita a formulação de estratégias para a prevenção da dengue. A ferramenta possibilita o compartilhamento de projetos e mapas entre equipes, permitindo a colaboração e a tomada de decisões baseadas em dados espaciais. Além disso, para integrar informações já existentes, a aplicação possibilita a importação de arquivos Shapefile ou CSV, esses dados podem ser visualizados em formato tabular ou convertidos em pontos geográficos no mapa por meio da geocodificação reversa.

2 MATERIAIS E MÉTODOS

O desenvolvimento deste projeto tem como base tecnológica a criação de uma aplicação de software voltada para o auxílio dos setores públicos no combate a dengue.

A aplicação permite a criação de projetos colaborativos, nos quais múltiplos usuários podem interagir simultaneamente. Dessa forma, equipes podem compartilhar informações, criar, visualizar e analisar dados geoespaciais de forma integrada.

Uma das principais funcionalidades do sistema é a organização dos dados por meio de camadas, que consistem na agregação de grupos de geometrias. Essas camadas podem ser ativadas, desativadas e sobrepostas, permitindo uma visualização estruturada e personalizada das informações.

Além disso, a aplicação possibilita a importação de dados preexistentes por meio de arquivos CSV, que podem ser visualizados em formato tabular ou convertidos em pontos geográficos a partir de endereços. Também suporta a importação de arquivos shapefile, que armazenam atri-

butos geográficos, como localização, formato e características espaciais.

Para a execução desta pesquisa, foi necessário abordar diversos tópicos fundamentais para o desenvolvimento da aplicação. No que diz respeito ao banco de dados, para demonstrar como foi implementado o armazenamento dos dados na aplicação. No frontend, uma interface com tecnologias modernas que facilitam a visualização interativa de mapas, a manipulação de camadas e a análise de dados em tempo real. Já no backend, ele ficou responsável pelo processamento das requisições, integração com serviços externos e manipulação dos dados geoespaciais.

Além disso, a aplicação incorporou algumas funcionalidades, como a importação de shapefiles e CSVs, podendo converter endereços em coordenadas geográficas e a sobreposição de informações de camadas. Por fim, foi implementada a geração de mapas de calor, uma ferramenta para auxiliar na identificação de focos críticos de dengue e na tomada de decisões estratégicas pelos órgãos públicos.

2.1 BANCO DE DADOS

O armazenamento e processamento eficiente de dados geoespaciais são fundamentais para a implementação de um sistemas de monitoramento epidemiológico, como no caso do mapeamento de focos de dengue. Para atender a essa demanda, este trabalho adota o sistema de gerenciamento de banco de dados PostgreSQL (versão 16.3), integrado a extensão PostGIS (versão 3.5). Essa combinação é amplamente utilizada no meio acadêmico e técnico por sua eficiência no tratamento de dados espaciais, como citado no artigo de Vinueza-Martinez et al. (2024), e também utilizado para implementar o projeto no artigo de White et al. (2023).

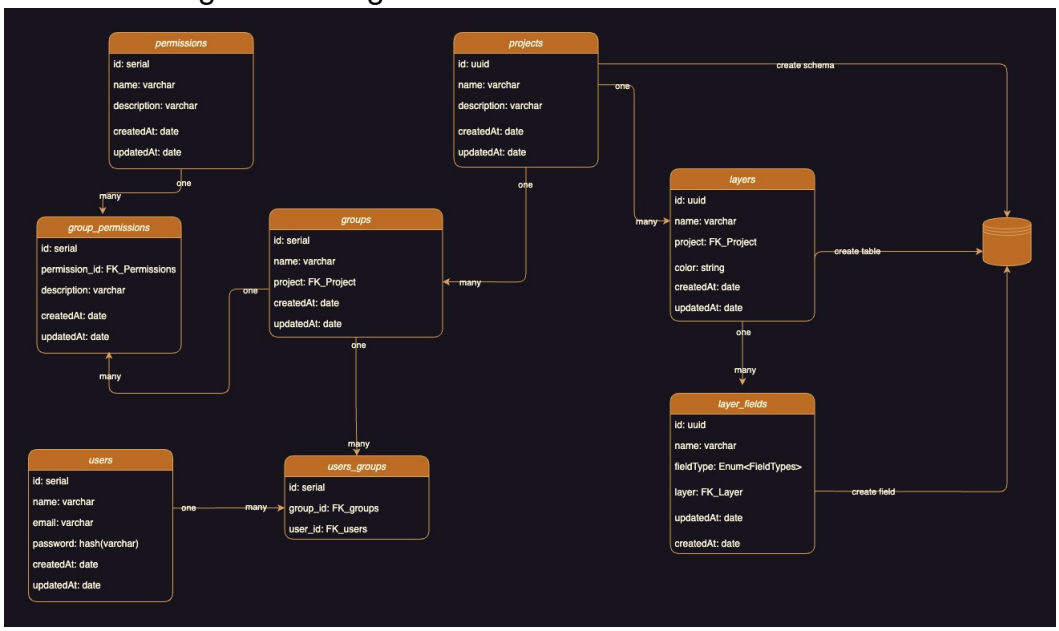
A capacidade de realizar consultas complexas envolvendo coordenadas geográficas, aliada a performance proporcionada pela indexação espacial baseada em R-tree, que é uma estrutura de dados do tipo árvore balanceada, projetada especificamente para indexação espacial e consultas multidimensionais, torna essa solução adequada para lidar com grandes volumes de dados geoespaciais. Além disso, a robustez e a flexibilidade do PostgreSQL garantem a escalabilidade necessária para futuras ampliações do sistema, sem comprometer a confiabilidade das informações.

As propriedades ACID do PostgreSQL, combinadas com as capacidades geoespaciais do PostGIS, o tornam confiável para sistemas de monitoramento da saúde pública. No contexto deste projeto, elas garantem que operações como a correlação geográfica de focos de dengue mante-

nham precisão analítica mesmo em cenários de alta concorrência ou falhas de infraestrutura, assegurando a qualidade dos dados para tomada de decisões em saúde coletiva.

Para garantir segurança e flexibilidade no gerenciamento de usuários, o sistema implementa um modelo de Role-Based Access Control (RBAC), no qual cada usuário, ao ser cadastrado, é automaticamente provisionado como administrador de um projeto padrão, com permissões completas (CRUD – Create, Read, Update, Delete) sobre seus dados.

Figura 1 - Diagrama de classes do banco de dados



Fonte: Elaborado pelo autor.

Como pode-se observar na figura 1, as permissões são armazenadas em uma tabela estática (permissions), configurada como read-only, funcionando como um catálogo fixo de operações autorizáveis. A associação entre usuários e permissões é gerenciada por meio de grupos funcionais, seguindo um esquema many-to-many implementado por tabelas de junção (user-groups e group-permissions).

Para a parte da tabela de projetos (projects), o sistema adota uma estrutura onde cada novo projeto criado gera automaticamente um schema dedicado no banco de dados. Essa abordagem transforma cada projeto em um container lógico independente.

As camadas (layers) criadas, são transformadas em tabelas dentro do schema em que se relacionam com o projeto, por exemplo, uma camada de ”Pontos de Foco de Dengue” se torna uma tabela no schema em

que possui sua relação. A tabela layers possui um campo chamado ”color”, este campo permite o armazenamento de um hexadecimal de alguma cor, para quando a camada for mostrada pelo frontend, as geometrias sejam facilmente identificadas por suas cores, assim visualmente relacionando com sua camada de pertencimento.

Os campos das camadas (layer-fields), são transformados em colunas dentro da camada em que é relacionada, onde campos do tipo de coordenadas geográficas viram colunas GEOMETRY, datas de registro viram colunas TIMESTAMP, e atributos descritivos são armazenados como VARCHAR ou outros tipos adequados.

Essa modelagem permite que operações espaciais sejam executadas com maior eficiência do que se todas fossem armazenadas em uma única tabela, já que cada tabela contém apenas os dados de uma camada específica dentro do contexto de um único projeto, otimizando tanto o desempenho das consultas quanto a organização lógica dos dados.

2.2 FRONTEND

Para o desenvolvimento do frontend, optou-se pelo React (v18.3) devido a sua ampla adoção na comunidade e ecossistema robusto de bibliotecas auxiliares, e para estilização, o Tailwind CSS (v3.4) para maior fluidez no desenvolvimento. O React facilita a construção de interfaces dinâmicas e reativas, essenciais para uma aplicação que demanda atualizações em tempo real, como no monitoramento de focos de dengue. Além disso, sua arquitetura baseada em componentes permite um desenvolvimento modular, simplificando a manutenção e a escalabilidade do projeto.

A fim de acelerar o desenvolvimento e garantir consistência visual, adotou-se ShadCN UI, uma coleção de componentes reutilizáveis construídos sobre Radix UI e Tailwind CSS. Essa biblioteca oferece uma base sólida para a criação de interfaces acessíveis e responsivas, reduzindo a necessidade de estilização manual e mantendo um padrão de design coeso em toda a aplicação.

Para o gerenciamento eficiente de requisições HTTP e cache de dados, foi adotada a biblioteca TanStack Query (v5.64) - anteriormente conhecida como React Query. Seus hooks useQuery e useMutation simplificam a obtenção, atualização e sincronização de dados com o backend, eliminando a necessidade de gerenciamento manual de estados de carregamento e erro. Essa abordagem melhora significativamente a performance da aplicação, evitando recarregamentos desnecessários de dados

já armazenados em cache.

As requisições HTTP são realizadas através da biblioteca Axios (v1.7.9), que oferece interceptores para tratamento centralizado de erros e configurações padrão, como headers de autenticação. Essa combinação com o TanStack Query garante uma camada de comunicação com o backend de fácil manutenção.

O gerenciamento de rotas na aplicação, utilizou-se o React Router DOM (v7.1), que permite a criação de uma SPA (Single Page Application) com navegação dinâmica e lazy loading de componentes, uma técnica de otimização que permite carregar componentes, módulos ou recursos JavaScript apenas quando necessários. Seu sistema de rotas aninhadas e proteção de rotas via Loader Functions, carregam dados assíncronos antes da renderização de uma rota, facilitando a implementação de autenticação e autorização, garantindo que apenas usuários com permissão acessem determinadas views.

Para a visualização de dados geoespaciais, foi escolhida a biblioteca Mapbox GL JS (v3.9), que oferece renderização vetorial de alta performance e suporte a interações avançadas com mapas. O Mapbox se destaca por seu free tier generoso, permitindo até 50.000 renderizações mensais gratuitas para a maioria dos casos de uso em projetos acadêmicos e aplicações de pequeno a médio porte. A cada mês, esse limite é reiniciado, garantindo custo zero para testes e implantações iniciais. Além disso, o Mapbox GL JS fornece plugins para a geração de mapas de calor, importantes para a visualização de densidade de casos de dengue..

A edição e criação de geometrias diretamente no mapa, foi integrado o Mapbox GL Draw (v1.5), um plugin oficial que permite desenhar polígonos, linhas e pontos de forma interativa. Essa funcionalidade é essencial para que agentes públicos possam demarcar áreas de risco, delimitar regiões de intervenção ou ajustar manualmente os dados espaciais coletados. Sua integração simplificada com o Mapbox GL JS e sua compatibilidade com formatos geoespaciais padrão (como GeoJSON) facilitam a importação e o processamento posterior dessas geometrias no backend.

2.3 BACKEND

No backend,o *framework* NestJS (v10.4), juntamente do Nodejs (v20.14) foi selecionado devido a sua arquitetura modular e ao suporte nativo ao TypeScript, que facilita o desenvolvimento de aplicações estruturadas e de fácil manutenção, oferece também flexibilidade para adaptações

futuras. Além disso, o NestJS inclui ferramentas para validação de dados, injeção de dependências e criação de APIs RESTful, aspectos fundamentais para um sistema que demanda precisão e desempenho. A escolha também considera a crescente adoção do *framework* em projetos de médio e grande porte, reforçando sua estabilidade e suporte pela comunidade, e tendo como patrocinadoras grandes nomes como Microsoft, Red Hat e Jetbrains.

Para a manipulação do banco de dados pelo backend, foi escolhido o Drizzle ORM (v0.31.4), a ORM (*Object-Relational Mapping* ou Mapeamento Objeto-Relacional) mapeia as tabelas do banco de dados e transforma em objetos para a aplicação. A escolha do Drizle justifica-se pela sua abordagem moderna, que combina a expressividade do SQL com a segurança de tipos do TypeScript. O Drizzle não abstrai completamente o SQL, aceitando também consultas de forma explícita quando necessário, enquanto ainda oferece uma API para operações comuns. Essa flexibilidade é importante neste sistema por conta das consultas usando o PostGIS, que precisam ser escritas SQL puro.

Para a parte de autenticação, foi adotado o uso de JWT (*JSON Web Token*), um padrão aberto para compartilhar informações entre partes, como um cliente e um servidor, através de um objeto JSON. O sistema utiliza um par de chaves privada e pública para assinar e verificar os tokens, garantindo integridade e autenticidade nas requisições. A chave privada é utilizada exclusivamente pelo backend para assinar os tokens gerados no momento do login, enquanto a chave pública é usada para verificar a assinatura dos tokens nas requisições subsequentes.

Essa estratégia reforça a segurança do sistema, evitando a falsificação de tokens e permitindo a verificação sem expor informações sensíveis. Além disso, foi implementado um controle de permissões para acesso aos *endpoints* da API. Todos os *endpoints*, com exceção do de login, exigem um token JWT válido e passam por um processo de verificação de permissões com base no perfil do usuário, essas informações ficam armazenadas na tabela ”permissions” no banco de dados. Dessa forma, garante-se que cada rota só seja acessada por usuários autorizados, respeitando os níveis de acesso definidos para cada operação do sistema.

A atribuição de permissões específicas a cada rota é realizada por meio de *decorators* personalizados, que definem, de forma declarativa, os requisitos de acesso diretamente nos controladores. Esse mecanismo garante que cada funcionalidade seja acessível apenas por usuários devi-

damente autorizados, conforme o nível de permissão estabelecido.

Os *decorators* de permissão foram implementados utilizando a funcionalidade de metadata do NestJS, permitindo associar, de forma declarativa, as permissões necessárias diretamente nos métodos dos controladores. Internamente, os *decorators* armazenam informações específicas de cada rota, como a lista de permissões exigidas, por meio da API de reflexão (*Reflector*) do NestJS.

Para efetuar a verificação em tempo de execução, foi utilizado um *Guard* customizado, que intercepta as requisições antes da execução dos *handlers*. O *Guard* recupera as permissões definidas nos *decorators* e as compara com as permissões associadas ao usuário autenticado, que são extraídas do token JWT. Caso o usuário não possua a permissão necessária, a requisição é automaticamente bloqueada com uma resposta de acesso negado.

Nos tratamentos de erros, adotou-se o padrão funcional Either, que representa o resultado de uma operação como um valor de sucesso (Right) ou falha (Left). Diferentemente do uso tradicional de exceções, o Either torna os erros parte do fluxo de dados, eliminando o custo de desempenho associado aos tratamentos de erros comuns. Além disso, exige que o tipo de retorno seja sempre declarado e tratado, o que favorece um controle mais preciso do comportamento da aplicação e evita erros silenciosos, o padrão é usado como mostra na figura 2.

Figura 2 - Exemplo de função com retorno Either

Fonte: Elaborado pelo autor.

A estrutura de domínio da aplicação foi organizada em quatro camadas para favorecer a separação de responsabilidades e a escalabilidade do sistema. A camada ”gis” concentra as funcionalidades relacionadas ao contexto de geoprocessamento, como o tratamento de geometrias e operações espaciais. A camada ”auth” é responsável pela autorização e controle de acesso, incluindo validação de permissões e verificação de *tokens*. Já a camada ”database” abstrai a manipulação direta do banco de dados, sendo responsável por operações como a criação, modificação e buscas específicas de dados de tabelas. Por fim, a camada ”user” trata da lógica associada a gestão de usuários, como o gerenciamento de permissões e o registro de novos perfis. Essa organização permite uma maior coesão interna em cada módulo e facilita a manutenção e evolução da aplicação.

O sistema adota o padrão *Repository* como camada de abstração para a interação com o banco de dados. Cada *Repository* é responsável por encapsular a lógica de acesso e manipulação de entidades específicas, como usuários, permissões e dados espaciais, expondo apenas métodos essenciais a aplicação, como criação, atualização, exclusão e consultas. Dessa forma, o restante do sistema permanece desacoplado dos detalhes da persistência, facilitando a manutenção e a possibilidade de substituição futura da tecnologia de armazenamento sem impacto direto nas regras de negócio.

Acima dos repositórios, foram implementados os *Use Cases*, que representam os fluxos de negócio da aplicação. Cada Use Case representa uma operação específica, como adicionar uma nova camada no mapa, criar uma nova tabela geoespacial, ou buscas com filtros em tabelas específicas. Os *Use Cases* fazem chamadas aos repositórios, aplicam regras de validação e garantem que todas as operações sejam consistentes com os requisitos do domínio, também se pode utilizar como um exemplo de *Use Case* o código representado na figura 2.

E acima dos *Use Cases*, estão os *Controllers*, responsáveis por receber as requisições HTTP, interpretar os dados de entrada e delegar a execução para o Use Case apropriado, transformando dados da requisição em objetos apropriados para o domínio e formatando a resposta enviada de volta ao cliente. Eles não contêm regras de negócio, mantendo-se focados apenas na comunicação entre a interface externa, o tratamento do erro que irá retornar ao usuário caso aconteça, podemos observar um exemplo de *controller* na figura 3.

Figura 3 - Exemplo de Controller

```
@Put('/:layerFieldId')  no usages  Guilherme Fragnani *
@Permissions(PERMISSIONS_ENUM.UPDATE_LAYER_FIELDS)
@ApiOperation({summary: 'Update layer field', description: 'Update layer field'})
async updateLayerField(
  @Param('layerFieldId') layerFieldId: IdLayerParamSchema,
  @Body(new ZodValidationPipe(updateLayerFieldParamSchema)) body: UpdateLayerFieldParamSchema
) : Promise<Right<UpdateLayerFieldError, Laye...  {
  const {name} = body

  const updatedLayerField : TUpdateLayerFieldUseCaseRespon...  = await this.updateLayerFieldUseCase
    .execute({name, layerFieldId})

  if (updatedLayerField.isLeft()) {
    switch (updatedLayerField.value.constructor) {
    case UpdateLayerFieldError:
      throw new Error(updatedLayerField.value.message)
    default:
      throw new Error('Unexpected error')
    }
  }

  return updatedLayerField
}
```

Fonte: Elaborado pelo autor.

2.4 IMPORTAÇÃO DE SHAPEFILE E CSV

Para a importação de arquivos CSV para o banco de dados, foi implementado um processo de importação que segue várias etapas para garantir a integração segura e consistente dos dados.

Inicialmente, o arquivo CSV é recebido no backend, e seu con-

teúdo é lido. Utilizando a biblioteca csv-parser, o sistema extrai dinamicamente os cabeçalhos do arquivo, que correspondem aos nomes das colunas a serem criadas na tabela de destino. A partir dessas informações, uma nova camada é criada no banco de dados, juntamente com seus respectivos campos.

Após a definição da estrutura, o arquivo CSV é temporariamente armazenado no sistema de arquivos do servidor e, em seguida, copiado para dentro do contêiner Docker onde o banco de dados PostgreSQL está hospedado. Essa transferência é realizada por meio de comandos auxiliares que interagem diretamente com o Docker.

Com o arquivo disponível no ambiente do banco de dados, a inserção dos dados na tabela é feita utilizando o comando SQL nativo COPY, que permite carregamento em massa de um arquivo para o banco de dados. O comando é construído dinamicamente para refletir a estrutura da nova tabela, considerando delimitadores personalizados e a presença ou não de cabeçalhos, conforme especificado na configuração do CSV. Após a conclusão da operação, o arquivo temporário é removido do contêiner para evitar o acúmulo de arquivos.

Para a importação de arquivos Shapefile (.shp) para o banco de dados, inicia-se com o recebimento de múltiplos arquivos, correspondentes a estrutura típica de um Shapefile (como .shp, .dbf, .shx), dentro de um arquivo .zip, enviados em conjunto pelo cliente.

O sistema identifica automaticamente o arquivo principal (.shp) entre os arquivos recebidos. Em seguida, uma nova camada é criada no banco de dados para representar o novo conjunto de dados geográficos. Essa camada é registrada com atributos como nome e cor, e vinculada a um projeto específico.

Os arquivos são temporariamente salvos no sistema de arquivos do servidor e posteriormente copiados para o contêiner Docker que hospeda o banco de dados. Para a importação dos dados, é utilizado o utilitário shp2pgsql, este comando é instalado na inicialização do contêiner, que converte o Shapefile em comandos SQL compatíveis com o PostGIS. Esses comandos são executados diretamente no banco utilizando a ferramenta psql, populando a nova tabela espacial criada.

Após a importação, o sistema realiza a limpeza dos arquivos temporários do contêiner para manter o ambiente organizado. Posteriormente, é feita a extração automática dos campos da tabela recém-criada para atualizar o modelo de campos no domínio da aplicação, refletindo a

estrutura dos dados importados.

Toda a operação, tanto de importação de Shapefiles ou CSVs é encapsulada em um fluxo com tratamento de erros, utilizando o padrão Either para garantir que falhas sejam capturadas de maneira controlada e retornadas ao sistema de forma previsível, sem que exceções não tratadas interrompam o fluxo da aplicação, e realizadas dentro de uma transação, garantindo com que qualquer erro que aconteça, nenhum dado seja persistido.

2.5 GERAÇÃO DINÂMICA DE GEOMETRIAS A PARTIR DE ENDEREÇOS DE UMA TABELA

Durante o desenvolvimento do sistema, foi implementada uma funcionalidade para geração dinâmica de geometrias a partir de informações de endereço de uma tabela. O objetivo é permitir que usuários usem dados de uma tabela que não possua informações espaciais e, ainda assim, tenham suas geometrias geradas automaticamente no banco de dados.

O processo começa com uma requisição, onde o identificador da camada e os campos de endereço como logradouro, bairro, município, estado, CEP e país são enviados no corpo da requisição.

Inicialmente, o sistema busca todos os registros da tabela associada a camada que ainda não possuem geometria, para cada um desses registros, os campos de endereço são concatenados em uma única *string*, formando um endereço completo.

Com os endereços prontos, o sistema utiliza a API pública do Nominatim para realizar a geocodificação, transformando o endereço em latitude e longitude, obtendo também o identificador único (osm-id) daquele local. Para otimizar o desempenho e reduzir o número de chamadas a API, foi implementado um cache em memória, onde se o endereço já foi consultado anteriormente na mesma execução, o sistema reutiliza o resultado armazenado, evitando consultas repetidas.

Quando uma resposta válida é obtida, o registro correspondente na tabela é atualizado dentro de uma transação de banco de dados. A geometria é criada para representar um ponto e armazenada na coluna geom, enquanto o identificador é gravado na coluna id-geom. Este processo também é feito dentro de uma transação, garantindo que, caso ocorra alguma falha, o banco de dados não fique em um estado inconsistente.

Se algum endereço não puder ser geocodificado ou se a API não retornar resultados, o registro simplesmente é ignorado, permitindo que o

restante da operação continue normalmente sem interrupções.

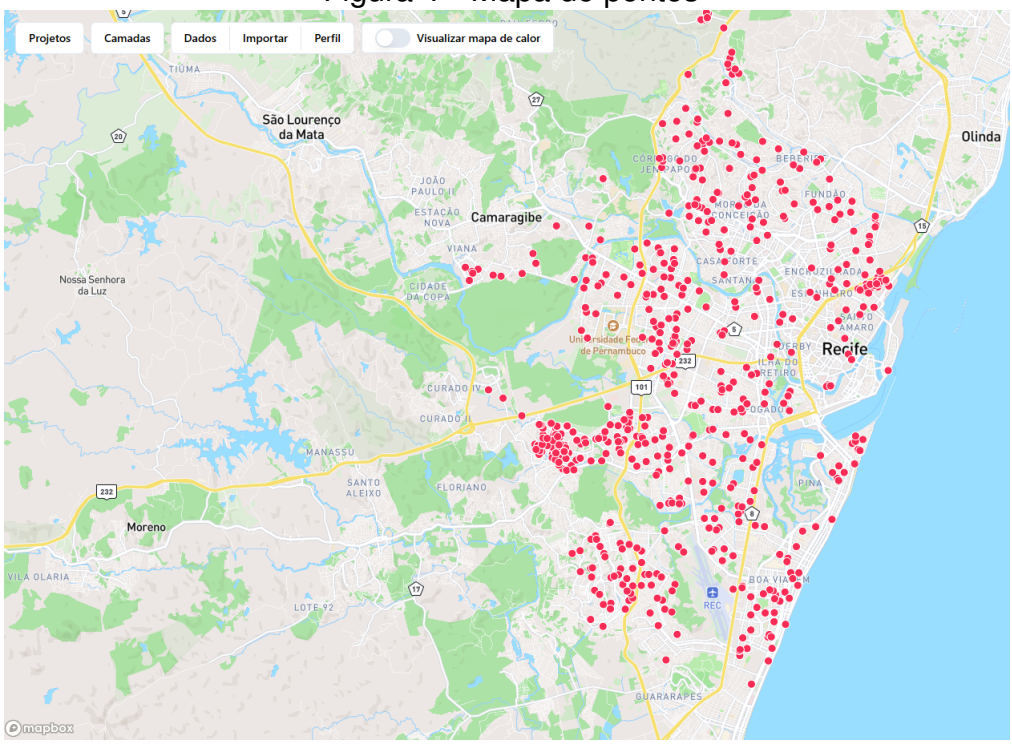
Essa abordagem proporciona uma maneira de popular camadas de dados com informações espaciais, mesmo quando o arquivo de origem não possui coordenadas. Além de aumentar a flexibilidade para os usuários, essa solução contribui para a construção de uma base de dados georreferenciada utilizando apenas informações textuais simples.

2.6 MAPA DE CALOR

Com a implementação deste projeto, foi pensado também em uma forma de identificar áreas de maior risco, que foi onde surgiu a necessidade de oferecer uma visualização mais eficiente e clara desses dados no mapa.

Para isso, foi implementada a geração de mapas de calor utilizando um plugin do Mapbox, configurando a camada com o tipo ‘heatmap’. A visualização foi desenvolvida de forma dinâmica, permitindo ao usuário alternar entre a exibição tradicional de pontos, utilizando o tipo ‘circle’, como pode ser observado na figura 4, e a visualização por densidade, ativando o modo de mapa de calor, como no exemplo da figura 5. A configuração foi realizada de maneira que, ao carregar as geometrias no mapa, o sistema cria uma nova camada com o ID correspondente a camada de dados.

Figura 4 - Mapa de pontos

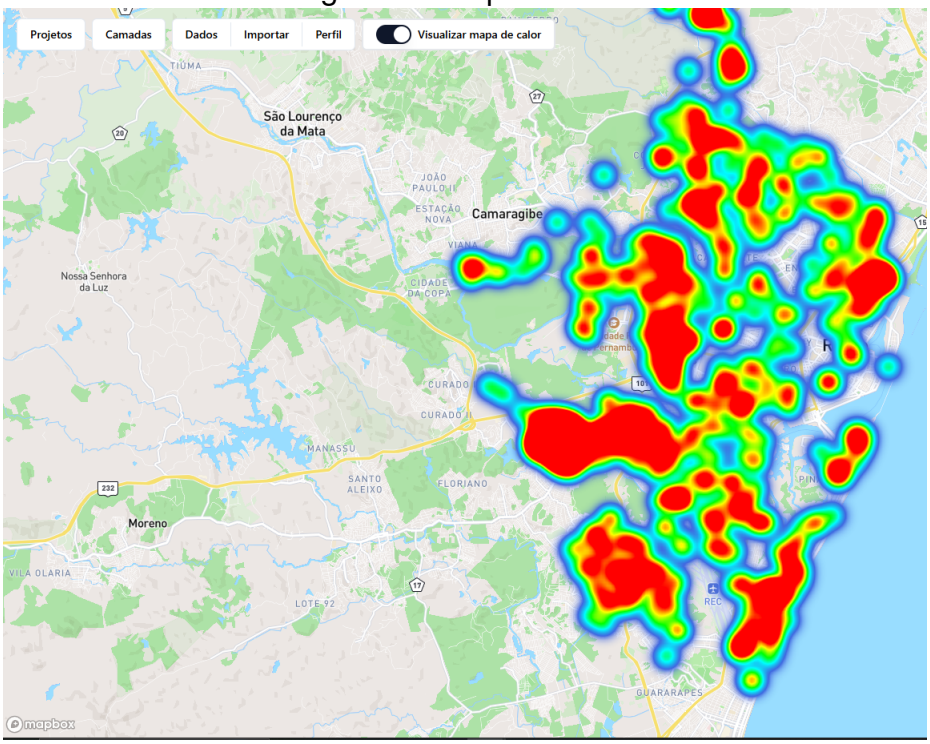


Fonte: Elaborado pelo autor.

A adoção do mapa de calor teve como principal objetivo facilitar a interpretação espacial dos dados. Em vez de exibir todos os pontos individualmente, o que poderia gerar uma poluição visual, o mapa de calor destaca áreas com maior concentração de pontos, evidenciando padrões de agrupamento de forma imediata.

Além disso, a geração automática de geometrias baseada em endereços poderia ocasionalmente resultar em dados inconsistentes, e a visualização por mapa de calor auxilia na identificação de anomalias, como concentrações inesperadas de geometrias em locais errados devido a falhas de geocodificação.

Figura 5 - Mapa de calor



Fonte: Elaborado pelo autor.

Essa abordagem também busca melhorar a experiência do usuário na análise dos dados, especialmente em projetos com um grande volume de informações, oferecendo uma visualização mais limpa. A possibilidade de alternar entre o modo de pontos individuais e o modo de mapa de calor permite que o usuário escolha a melhor forma de visualização conforme a necessidade da análise, tornando a aplicação adaptável a diferentes cenários de uso.

3 DISCUSSÃO E RESULTADOS

O avanço tecnológico na área da computação tem promovido transformações no modo como os dados de saúde pública são coletados, organizados e analisados. A crescente disponibilidade de dados geográficos e epidemiológicos, aliada a evolução de sistemas de informação geográfica e bancos de dados espaciais, permite uma abordagem integrada no combate a doenças como a dengue. O presente trabalho se insere nesse contexto, ao desenvolver uma plataforma de análise espacial voltada a vigilância epidemiológica da dengue, utilizando tecnologias como PostgreSQL com a extensão PostGIS para manipulação de dados geoespaciais e bibliotecas de geoprocessamento para visualização e análise.

O sistema foi concebido com objetivos específicos bem definidos, os quais orientaram a estruturação técnica do projeto. O primeiro objetivo consistiu em organizar informações de saúde relacionadas aos casos de dengue, associando-os a dados geográficos de forma estruturada. A utilização de banco de dados relacionais com suporte a geometrias possibilitou a indexação espacial e a realização de consultas que foram fundamentais para as análises no projeto.

O segundo objetivo foi aplicar técnicas de georreferenciamento e visualização interativa para exibir a distribuição dos casos de dengue sobre um mapa digital. Isso foi possível por meio da criação de uma interface visual, onde os dados são representados em camadas temáticas sobrepostas, essa visualização permite a identificação de áreas afetadas, facilitando a comunicação entre gestores públicos, pesquisadores e profissionais da saúde.

Um dos diferenciais do sistema foi a aplicação de análise de densidade espacial por meio de mapas de calor, ferramenta que reforça a dimensão analítica da plataforma. Esses mapas representaram a intensidade de ocorrência de casos em determinadas regiões, destacando zonas com alta densidade de notificações, visualizadas no sistema como pontos, estas informações tomam destaque para ações de prevenção e controle da dengue. A análise de densidade foi complementada com a possibilidade de sobreposição de outras camadas, como redes de infraestrutura urbana, vegetação, presença de corpos d'água e indicadores socioeconômicos. Essa abordagem permitiu inferências mais amplas sobre as condições que favorecem a proliferação em determinados territórios.

O resultado obtido com o desenvolvimento desta ferramenta des-

taca a importância da computação aplicada a saúde, pois transforma dados brutos, muitas vezes isolados e desorganizados, em conhecimento espacial estruturado. A plataforma busca oferecer uma melhoria para o monitoramento epidemiológico, permitindo a antecipação de surtos, o reconhecimento de padrões territoriais e a priorização de recursos para determinadas regiões.

Entretanto, o sistema ainda pode ser aprimorado em termos de visualização e arquitetura tecnológica. Atualmente, os mapas de calor utilizam parâmetros fixos de densidade e dispersão, o que reduz sua eficácia em regiões com poucos dados, e a inclusão de controles interativos para ajuste desses parâmetros tornaria a análise mais adaptável. Além disso, o uso de ferramentas como Django Rest Framework, GeoDjango e GeoPandas no backend poderia ter otimizado o tratamento de dados espaciais e facilitado futuras integrações com modelos analíticos ou preditivos.

Mesmo com essas limitações, o trabalho mostra como a computação pode contribuir de forma direta e aplicada para a vigilância epidemiológica. Ao tornar os dados espaciais mais compreensíveis e acessíveis, a ferramenta serve como um elo entre o conhecimento técnico da computação e as demandas sociais e territoriais da saúde pública.

No contexto de soluções computacionais aplicadas a saúde e ao geoprocessamento, diversos trabalhos vêm sendo desenvolvidos com o objetivo de integrar dados espaciais e epidemiológicos para apoiar a tomada de decisões em políticas públicas, dentre eles um trabalho que se destaca é o ”An open-source platform for geospatial participatory modeling in the cloud” de White et al. (2023), que propõe a plataforma OpenPlains para modelagem participativa geoespacial na nuvem.

A principal diferença entre os projetos está na abrangência e robustez da arquitetura. Enquanto o presente projeto tem como foco principal a visualização de dados espaciais em mapas interativos e mapas de calor, a plataforma OpenPlains oferece uma solução completa para a criação, execução e compartilhamento de modelos geoespaciais participativos por meio de uma arquitetura modular baseada em GRASS GIS, Actinia e Django Rest Framework.

A plataforma OpenPlains possui vantagens significativas em relação a este projeto projeto, como a capacidade de executar modelos complexos em tempo real, utilizando processamentos assíncronos com suporte a WebSockets e filas de tarefas (Celery), o que permite a execução de simulações espaciais com feedback interativo para os usuários. Além disso, a

integração com bibliotecas como GeoDjango, PostGIS e OpenLayers, bem como o uso de formatos otimizados garantem maior desempenho na manipulação de grandes volumes de dados espaciais.

Apesar disso, o projeto apresentado é mais leve, no sentido técnico de menor complexidade de infraestrutura e dependências, e com curva de aprendizado reduzida, o que pode torná-lo mais acessível financeiramente para iniciativas com recursos limitados, ou em contextos onde a análise e visualização simples de dados espaciais é suficiente para os objetivos.

4 CONCLUSÃO

Este trabalho teve como objetivo o desenvolvimento de uma plataforma web voltada a análise espacial de dados epidemiológicos, com foco específico na vigilância da dengue. A proposta partiu da premissa de que o avanço das tecnologias de informação geográfica, junto da crescente disponibilidade de dados de saúde pública podem oferecer suporte a tomada de decisão em políticas de combate a doenças endêmicas. O sistema foi feito para integrar informações geográficas e epidemiológicas, utilizando tecnologias como PostgreSQL e PostGIS para armazenamento e manipulação de dados espaciais, além de bibliotecas de geoprocessamento para visualização e análise.

O sistema permitiu a organização e espacialização dos dados de casos de dengue, viabilizando consultas geográficas e visualizações interativas sobre mapas digitais. O uso de camadas temáticas sobrepostas buscou proporcionar a interpretação das informações e a identificação de áreas críticas. A implementação de mapas de calor representou um dos principais diferenciais da ferramenta, onde buscam melhorar a visualização para priorização de áreas com maior incidência, auxiliando na otimização de recursos e ações preventivas.

Outro ponto foi a possibilidade de integrar outras camadas informativas ao ambiente de análise, como infraestrutura urbana, cobertura vegetal e indicadores socioeconômicos. Essa integração possibilitou outras formas de abordagem definidas pelo usuário, contribuindo para a identificação de fatores associados a proliferação do vetor da dengue em determinados contextos territoriais.

Além disso, o projeto pode evoluir com a incorporação de funcionalidades avançadas de análise temporal e espacial, utilizando técnicas de séries temporais e modelagem preditiva. Isso permitiria a visualização di-

nâmica da progressão dos casos de dengue ao longo do tempo, bem como a implementação de algoritmos de aprendizado de máquina para prever surtos futuros com base em dados históricos e geoespaciais. Tais recursos fortaleceriam a aplicação como uma ferramenta computacional para a tomada de decisões em saúde pública e ações preventivas.

Em suma, o trabalho desenvolvido demonstra a viabilidade e a relevância de soluções computacionais aplicadas a área da saúde, especialmente no contexto da vigilância epidemiológica. A transformação de dados brutos em conhecimento espacial estruturado e visualmente compreensível, representa uma contribuição para o enfrentamento da dengue e pode ser estendida para outras doenças e contextos territoriais. O projeto reforça a importância da interdisciplinaridade entre computação, saúde pública e geotecnologias, e favorece novas investigações e aprimoramentos no uso de sistemas de informação geográfica aplicados a saúde.

REFERÊNCIAS

MAGALHAES, G. B. **O Uso do Geoprocessamento e da Estatística nos Estudos Ecológicos em Epidemiologia: O Caso da Dengue em 2008 na Região Metropolitana de Fortaleza**. 2012. 1980-1726 p. Disponível em: <http://www.seer.ufu.br/index.php/hygeiaHygeia8>.

NARDI, S. M. T. et al. **Geoprocessamento em saúde pública: fundamentos e aplicações**. Revista do Instituto Adolfo Lutz, 2013, v. 72, n. 3, p. 185–191.

PROCOPIO, A. C. et al. **Integrated One Health strategies in Dengue**. [S.l.]: Elsevier B.V., 2024.

ROCHA, M. do Espírito Santo Abreu da et al. **O uso do geoprocessamento na análise da dengue na região leste de Teresina - PI**. [S.l.]: Even3, 2021.

VINUEZA-MARTINEZ, J. et al. **Geographic Information Systems (GISs) Based on WebGIS Architecture: Bibliometric Analysis of the Current Status and Research Trends**. [S.l.]: Multidisciplinary Digital Publishing Institute (MDPI), 2024.

WHITE, C. T. et al. **An open-source platform for geospatial participatory modeling in the cloud**. Environmental Modelling and Software, 2023, Elsevier Ltd, v. 167.