UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

FRANCISCO ANTONIO VASCONCELOS

APLICAÇÃO WEB DE UM SISTEMA DE INFORMAÇÃO GEOGRÁFICA (SIG) COM OPENLAYERS: UM ESTUDO DE CASO E APRIMORAMENTO NA PLATAFORMA SOBRAL EM MAPAS

SOBRAL

2025

FRANCISCO ANTONIO VASCONCELOS

APLICAÇÃO WEB DE UM SISTEMA DE INFORMAÇÃO GEOGRÁFICA (SIG) COM OPENLAYERS: UM ESTUDO DE CASO E APRIMORAMENTO NA PLATAFORMA SOBRAL EM MAPAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do CAMPUS SOBRAL da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Iális Cavalcante de Paula Júnior

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V45a Vasconcelos, Francisco Antonio.
APLICAÇÃO WEB DE UM SISTEMA DE INFORMAÇÃO GEOGRÁFICA (SIG) COM OPENLAYERS : UM ESTUDO DE CASO E APRIMORAMENTO NA PLATAFORMA SOBRAL EM MAPAS / Francisco Antonio Vasconcelos. – 2026.
51 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2026.
Orientação: Prof. Dr. Iális Cavalcante de Paula Júnior.

1. OpenLayers. 2. SIG para web. 3. GeoServer. 4. Laravel. 5. PostGIS. I. Título.

CDD 621.39

FRANCISCO ANTONIO VASCONCELOS

APLICAÇÃO WEB DE UM SISTEMA DE INFORMAÇÃO GEOGRÁFICA (SIG) COM OPENLAYERS: UM ESTUDO DE CASO E APRIMORAMENTO NA PLATAFORMA SOBRAL EM MAPAS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do CAMPUS SOBRAL da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Aprovada em: 28 de Janeiro de 2026

BANCA EXAMINADORA

Prof. Dr. Iális Cavalcante de Paula Júnior (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Me. Fernando Rodrigues De Almeida Junior
Universidade Federal do Ceará(UFC)

Prof. Madson Pereira Mesquita
Universidade Federal do Ceará(UFC)

À minha família, pela confiança depositada em mim e pelo constante investimento no meu crescimento. A todos que estiveram presentes, oferecendo apoio, incentivo e palavras de encorajamento ao longo desta jornada.

AGRADECIMENTOS

Ao Prof. Dr. Iális Cavalcante de Paula Júnior por me orientar em meu trabalho de conclusão de curso.

Aos meus amigos e colegas, pelo apoio e incentivo durante os estudos realizados na graduação.

A minha família, que nos momentos de minha ausência dedicados ao estudo superior, sempre fez entender que o futuro é feito a partir da constante dedicação no presente.

Ao Doutorando em Engenharia Elétrica, Ednardo Moreira Rodrigues, e seu assistente, Alan Batista de Oliveira, aluno de graduação em Engenharia Elétrica, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Ao aluno Thiago Nascimento do curso de ciência da computação da Universidade Estadual do Ceará que elaborou o *template* do qual este trabalho foi adaptado para a Universidade Federal do Ceará.

Agradeço a todos os professores por me proporcionar o conhecimento não apenas racional, mas a manifestação do caráter e afetividade da educação no processo de formação profissional, por tanto que se dedicaram a mim, não somente por terem me ensinado, mas por terem me feito aprender.

"O sonho é que leva a gente para frente. Se a gente for seguir a razão, fica aquietado, acomodado."

(Ariano Suassuna)

RESUMO

A disponibilização de dados geoespaciais em portais públicos exige soluções que conciliem desempenho, confiabilidade e baixo acoplamento entre camadas. No contexto da Prefeitura Municipal de Sobral (PMS), este trabalho teve como objetivo avaliar e aprimorar a aplicação da biblioteca *OpenLayers* na plataforma "Sobral em Mapas (2026)", um Sistema de Informação Geográfica (SIG) para *web* (WebGIS) cujo *front-end* é construído integralmente com essa biblioteca e interage com o *back-end* em Laravel (2026) integrado ao *GeoServer (2026)* para publicação de serviços *Open Geospatial Consortium* (OGC). A arquitetura refinada contempla o carregamento de camadas temáticas, a implementação de ferramentas de desenho e edição vetorial e o desenvolvimento de um módulo de exportação de mapas. Por meio de estudo aplicado, verificaram-se navegação fluida, consultas espaciais responsivas e consistência cartográfica entre cliente e servidor, evidenciando a eficiência do uso da *OpenLayers* para o consumo e operação de dados geoespaciais municipais. Os resultados reforçam a viabilidade de uma pilha *open source* para SIG na *web* no setor público, destacando benefícios como flexibilidade, redução de custos e padronização das integrações. Como desdobramentos, propõe-se a impressão em escala cartográfica, a ampliação de funcionalidades analíticas e a otimização da usabilidade em dispositivos móveis.

Palavras-chave: *OpenLayers*. SIG para *web*. *GeoServer*. Laravel. OGC. PostGIS. Setor público.

ABSTRACT

Publishing geospatial data on public portals demands solutions that balance performance, reliability, and low coupling across system layers. Focusing on the Municipality of Sobral, this study evaluates and enhances the implementation of the OpenLayers library within "Sobral em Mapas," a municipal WebGIS whose front end is implemented with this library and interacts with a Laravel back end integrated with GeoServer for OGC services and PostGIS data. The refined architecture comprises thematic layers, vector drawing and editing tools, and a custom map export module with server-side request intermediation. In an applied evaluation, the system delivers smooth navigation, responsive spatial queries, and cartographic consistency across client and server, evidencing the efficiency of OpenLayers for municipal geospatial data consumption and interaction. The results support the feasibility of an open-source stack for government WebGIS, highlighting flexibility, cost reduction, and standardized integrations. Future work includes scale-aware cartographic printing, expanded spatial analytics, and usability optimization for mobile devices.

Keywords: OpenLayers. WebGIS. GeoServer. Laravel. OGC. PostGIS. Public sector.

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
DPI	*Dots Per Inch*
ECW	*Enhanced Compression Wavelet*
GIS	*Geographic Information System*
HTML	*HyperText Markup Language*
HTTP	*Hypertext Transfer Protocol*
JPG	*Joint Photographic Experts Group*
KML	*Keyhole Markup Language*
KMZ	*Keyhole Markup Language Zipped*
MDT	Modelo Digital de Terreno
OGC	*Open Geospatial Consortium*
PDF	*Portable Document Format*
PMS	Prefeitura Municipal de Sobral
PNG	*Portable Network Graphic*
SEUMA	Secretaria do Urbanismo e Meio Ambiente
SHP	*Shapefile*
SIG	Sistema de Informação Geográfica
WFS	*Web Feature Service*
WMS	*Web Map Service*
WMTS	*Web Map Tile Service*

SUMÁRIO

1 INTRODUÇÃO

Nas últimas décadas, a utilização de mapas interativos tem ganhado destaque pela sua capacidade de auxiliar na navegação e compreensão de espaços que não são plenamente atendidos por sistemas de mapas tradicionais, como ambientes internos ou áreas específicas de campi universitários. Estudos recentes demonstram que, por meio de ferramentas de código aberto, é possível implementar soluções de SIG, acrônimo inglês de *Geographic Information System* (GIS), modernas e eficientes, sem a necessidade de softwares proprietários. Nesse contexto, os sistemas *web* GIS assumem papel fundamental, pois proporcionam acessibilidade, escalabilidade e integração de dados espaciais em plataformas *web* (BATISTA; ANTUNES, 2021).

Durante a pandemia, os sistemas de SIG baseados na *web* (*WebGIS*) tiveram papel fundamental ao apoiar a educação a distância, o monitoramento da saúde pública, como o acompanhamento de casos e vacinação, a análise espacial de fenômenos sociais e ambientais, demonstrando sua relevância como ferramenta de acesso rápido à informação e suporte à tomada de decisão em situações críticas - áreas como planejamento urbano, saúde pública e conservação ambiental (VINUEZA-MARTINEZ *et al.*, 2024).

A utilização de geotecnologias em ambientes urbanos tem se mostrado uma ferramenta imprescindível para o planejamento e gestão eficaz das cidades, e Sobral, no Ceará, se destaca ao adotar o *WebGIS* como parte de sua estratégia de gestão pública. Até a implementação da plataforma "Sobral em Mapas (2026)", o município enfrentava limitações significativas no uso de geotecnologias para gestão urbana e ambiental. As informações geográficas estavam dispersas em formatos estáticos, como arquivos *Keyhole Markup Language* (KML), *Keyhole Markup Language Zipped* (KMZ), *Shapefile* (SHP), Modelo Digital de Terreno (MDT) e *Enhanced Compression Wavelet* (ECW), disponíveis para *download* no portal da Secretaria do Urbanismo e Meio Ambiente (SEUMA). Essa abordagem dificultava a análise integrada e a participação cidadã, uma vez que não havia uma interface interativa que permitisse a visualização dinâmica e a consulta em tempo real dos dados territoriais (Secretaria de Urbanismo e Meio Ambiente de Sobral, 2025).

1.1 Justificativa

A adoção da biblioteca *OpenLayers* na plataforma "Sobral em Mapas (2026)"visa superar limitações técnicas ao oferecer uma interface *web* interativa para visualização de mapas e dados geoespaciais. Trata-se de uma biblioteca JavaScript de código aberto amplamente utilizada para construção de aplicações de mapeamento na *web*, permitindo a integração de diversos formatos de dados espaciais, como WMS, *Web Feature Service* (WFS) e dados vetoriais. Sua implementação oferece suporte a funcionalidades interativas, como *zoom*, navegação e marcação de pontos, facilitando a análise espacial e o planejamento urbano. Também conta com uma função nativa que permite capturar rapidamente a visualização atual para documentação, compartilhamento ou análise posterior. Isso facilita a coleta de dados visuais sem a necessidade de ferramentas externas, agilizando o processo de tomada de decisão. Além disso, por ser uma ferramenta de código aberto, o *OpenLayers* reduz custos com licenças e oferece maior flexibilidade na personalização e manutenção da plataforma *(OpenLayers Team, 2025a)*.

Para além dos benefícios técnicos, a biblioteca também apresenta relevância no campo educacional. Sua adoção em projetos acadêmicos e trabalhos de conclusão demonstra a efetividade da ferramenta como meio de aprendizado prático, favorecendo o desenvolvimento de competências em geotecnologias, programação *web* e análise espacial aplicada a diferentes contextos sociais. Dessa forma, o *OpenLayers* consolida-se como recurso estratégico, tanto para a formação de profissionais quanto para a implementação de soluções em ambientes reais (Universidade Federal do Pará, 2024).

1.2 Objetivo Geral

Desenvolver nova funcionalidade de exportação de mapas e analisar a aplicação da biblioteca *OpenLayers* na plataforma "Sobral em Mapas (2026)", visando disponibilizar uma interface *web* interativa e acessível para visualização e análise de dados georreferenciados, contribuindo para o planejamento urbano, a gestão territorial e o uso educacional da informação espacial.

1.3 Objetivos Específicos

Diante da crescente utilização de ferramentas digitais de geolocalização, torna-se essencial explorar suas funcionalidades e implementar melhorias. Nesse contexto, a aplicação

desenvolvida trata-se de uma plataforma *web* em SIG para a Prefeitura de Sobral, intitulada "Sobral em Mapas (2026)", em parceria com a SEUMA. Assim, foram definidos os seguintes objetivos específicos:

- Analisar recursos interativos de visualização de mapas utilizando a biblioteca *OpenLayers*, incluindo navegação, *zoom*, marcação de pontos e camadas de dados, com o objetivo de verificar se esses recursos atendem às necessidades dos usuários.
- Implementar e disponibilizar funcionalidades de exportação e captura da visualização atual, de modo a facilitar o compartilhamento e a documentação de análises espaciais na plataforma "Sobral em Mapas (2026)".
- Avaliar a usabilidade e a acessibilidade da plataforma por potenciais usuários como ferramenta de apoio ao planejamento urbano e à gestão pública no município de Sobral. O endereço eletrônico do portal será disponibilizado e, posteriormente, será aplicado um questionário aos participantes, cujos resultados serão analisados para validar o atendimento aos objetivos propostos.

2 FUNDAMENTAÇÃO TEÓRICA

O *OpenLayers* destaca-se como uma das principais bibliotecas para o desenvolvimento de aplicações *WebGIS*, oferecendo ampla flexibilidade e suporte a diversos formatos de dados geoespaciais. Sua evolução contínua e adesão a padrões abertos consolidam-no como uma escolha estratégica para a construção de aplicações de mapeamento *web* robustas e escaláveis. Quando bem explorada, a biblioteca pode servir como base para soluções de grande porte, incluindo plataformas municipais de monitoramento e análise espacial. Nesse cenário, posiciona-se como um concorrente de peso em relação a outras bibliotecas amplamente utilizadas, sobretudo em contextos que demandam maior controle e personalização (FOERSTER *et al.*, 2011).

2.1 Fundamentos dos SIG na *Web*

Os SIG são ferramentas essenciais para a captura, gestão, análise, visualização e interpretação de dados geoespaciais. Eles permitem relacionar fenômenos ao espaço geográfico e apoiar decisões em áreas diversas, como planejamento urbano, monitoramento ambiental, gestão de recursos naturais e saúde pública. Com a crescente disponibilidade de sensores remotos, dispositivos móveis e o compartilhamento de dados, o volume e a diversidade de informações espaciais têm expandido rapidamente.

A evolução dos *WebGIS* foi motivada pela necessidade de tornar tais funcionalidades mais acessíveis, escaláveis e utilizáveis via navegador, eliminando a dependência de *softwares desktop* e promovendo uso remoto por diferentes perfis de usuários. *WebGIS* pode ser entendido como aplicações SIG em que parte ou toda a lógica (visualização, consulta, análise espacial, manipulação de dados) é exposta por meio de interfaces *web*. Pesquisas recentes ressaltam que essas plataformas permitem funcionalidades SIG via *web* com foco em democratização e acesso remoto a mapas e dados de representação territorial (KOMBE; OUTROS, 2022; SHABANI; OUTROS, 2024).

Nos últimos anos, a computação em nuvem tem desempenhado papel central nessa evolução, oferecendo elasticidade no processamento e escalabilidade no armazenamento de grandes volumes de dados. Essa combinação entre *WebGIS* e nuvem cria oportunidades para arquiteturas resilientes, com camadas de dados distribuídas, serviços cartográficos em conformidade com padrões abertos e APIs que permitem integração em múltiplas aplicações. Estudos aplicados mostram, por exemplo, o uso dessa abordagem para apoio à tomada de decisão em

defesa nacional, bem como no desenvolvimento de ferramentas de modelagem ecossistêmica orientadas à comunicação de resultados complexos para gestores e *stakeholders* ambientais (ALMEIDA; OUTROS, 2025).

Essa convergência entre *WebGIS*, computação em nuvem e padrões abertos como os do OGC consolida as bases para plataformas modernas de monitoramento, análise e gestão territorial, estabelecendo um caminho sólido para soluções robustas em contextos científicos, públicos e privados.

2.2 Tecnologias para Mapas Interativos: Uma Análise Comparativa

No desenvolvimento de aplicações *WebGIS*, diferentes bibliotecas e APIs podem ser empregadas para a visualização, manipulação e análise de dados territoriais na *web*. Cada tecnologia apresenta características específicas em termos de desempenho, flexibilidade, curva de aprendizado e aderência a padrões geoespaciais, sendo a escolha fortemente influenciada pelos requisitos do projeto.

Google Maps API é uma das soluções mais difundidas no mercado, reconhecida pela ampla cobertura cartográfica e pela facilidade de integração em aplicações *web* e móveis. Contudo, seu modelo de licenciamento pago e as restrições na customização de camadas limitam a flexibilidade em cenários mais avançados (Wikipedia contributors, 2025a; Google, 2025).

Leaflet caracteriza-se por ser uma biblioteca JavaScript *open-source* notavelmente leve e extensível através de um vasto ecossistema de *plugins*. Sua principal vantagem é a baixa curva de aprendizado, sendo ideal para aplicações que demandam interatividade básica. Contudo, apresenta limitações quando comparado a bibliotecas mais robustas, carecendo de suporte nativo a funcionalidades geoespaciais avançadas, como a manipulação complexa de projeções (Wikipédia, 2023).

Mapbox GL JS diferencia-se pela renderização de alto desempenho, otimizada para o uso de *vector tiles*, tornando-se uma escolha eficiente para a visualização de grandes volumes de dados. No entanto, sua arquitetura pode apresentar menor flexibilidade para integração com serviços que não seguem seus padrões específicos, exigir maior esforço de configuração e apresentar menor aderência a padrões OGC quando comparada a alternativas como o *OpenLayers* (DOE *et al.*, 2024).

OpenLayers é uma biblioteca JavaScript de código aberto, de alto desempenho e rica em funcionalidades, projetada para a criação de aplicações de mapeamento interativo. Sua

principal característica é a vasta gama de recursos nativos, que a posiciona como uma solução completa para o desenvolvimento de SIG complexos e de nível profissional. Uma de suas maiores vantagens é a flexibilidade e o suporte robusto a padrões estabelecidos pelo OGC. A biblioteca oferece compatibilidade nativa com os principais serviços de dados territoriais, como WMS, WFS e *Web Map Tile Service* (WMTS) *(OpenLayers Team, 2025b)*.

2.3 *OpenLayers*

2.3.1 Histórico e Evolução

A biblioteca *OpenLayers*, lançada inicialmente em 2006 pela empresa MetaCarta, consolidou-se como uma das mais poderosas e flexíveis soluções de código aberto para o desenvolvimento de *WebGIS*. Desde sua concepção, o projeto tem passado por uma evolução constante, visando facilitar a criação de aplicações de mapeamento ricas e complexas. Sua arquitetura permite que desenvolvedores integrem diversos tipos de dados espaciais, permite também um alto nível de personalização e controle sobre os mapas e as interações, tornando-se a escolha ideal para projetos mais complexos e específicos (Wikipedia contributors, 2025b).

2.3.2 Arquitetura e Funcionamento Interno

O *OpenLayers* é composto por uma série de módulos que lidam com diferentes aspectos de um mapa interativo. Entre os principais componentes da arquitetura do *OpenLayers*, estão:

Map: O objeto principal responsável por gerenciar a visualização do mapa e suas interações.

Layer: Define os tipos de camadas a serem exibidas no mapa, como camadas *raster* ou vetoriais.

Source: Fontes de dados que alimentam as camadas do mapa.

Interaction: Define como o usuário pode interagir com o mapa, por exemplo, movendo-se pelo mapa, fazendo *zoom*, desenhando ou editando formas.

Control: Fornece controles adicionais, como botões de *zoom*, impressão, controle de camadas, entre outros *(OpenLayers Team, 2025a)*.

2.3.3 Funcionalidades Principais

A arquitetura do *OpenLayers* é modular e se baseia em um conjunto de componentes principais que, combinados, oferecem um controle granular sobre a visualização e a manipulação de dados geográficos. Suas funcionalidades essenciais podem ser agrupadas da seguinte forma:

Gerenciamento de Camadas: O conceito central da biblioteca é a sobreposição de camadas. O *OpenLayers* suporta diversos tipos, cada um com uma finalidade específica. As principais são a *TileLayer*, utilizada para exibir mapas base em formato de mosaico, e a *VectorLayer*, que renderiza dados vetoriais a partir de fontes como *GeoJSON*, KML ou serviços WFS. Há também a *ImageLayer* para imagens estáticas georreferenciadas.

Suporte a Fontes de Dados e Formatos: Cada camada é alimentada por uma fonte *(Source)*, que define a origem dos dados. O *OpenLayers* se destaca por sua vasta compatibilidade com padrões OGC e formatos de mercado, essa flexibilidade permite a integração de dados de múltiplas origens em uma única aplicação.

Manipulação de Projeções Cartográficas: Uma funcionalidade robusta e crucial do *OpenLayers* é sua capacidade de gerenciar diferentes sistemas de referência de coordenadas. A biblioteca pode reprojetar dados dinamicamente, permitindo, por exemplo, exibir dados em coordenadas geográficas sobre um mapa base na projeção *Web Mercator*. Para sistemas mais complexos, é possível estender essa capacidade com a integração da biblioteca Proj4js.

Controles e Interações: A interatividade do mapa é gerenciada por meio de controles e interações. Os controles são os elementos visíveis da interface, como botões de *zoom*, escala gráfica e visão geral. As interações capturam ações do usuário, como o *pan* (arrastar o mapa), *zoom* e interações de desenho (*draw*) ou seleção (*select*), permitindo a manipulação e consulta de feições vetoriais *(OpenLayers Team, 2025a)*.

Para ilustrar a estrutura básica de inicialização de um mapa, o código-fonte 1 a seguir apresenta e renderiza um mapa base do *OpenStreetMap*:

Código-fonte 1 – Exemplo de inicialização de um mapa com *OpenLayers*

```
1 import Map from 'ol/Map';
2 import View from 'ol/View';
3 import TileLayer from 'ol/layer/Tile';
4 import OSM from 'ol/source/OSM';
5
```

```
new Map({
  target: 'map',
  layers: [
    new TileLayer({
      source: new OSM(),
    }),
  ],
  view: new View({
    center: [0, 0],
    zoom: 2,
  }),
});
```

Fonte: Elaborado pelo autor (2025), com base na documentação do *OpenLayers*.

2.3.4 Casos de Uso e Aplicações Conhecidas

Devido à sua robustez, flexibilidade e aderência a padrões abertos, a biblioteca *OpenLayers* é amplamente adotada em uma diversidade de projetos que demandam a manipulação complexa de dados geoespaciais. Suas aplicações abrangem desde o setor público até soluções comerciais e científicas de grande escala.

Um dos principais campos de aplicação é no desenvolvimento de plataformas *WebGIS* para a gestão pública. Exemplos notáveis são as portais Fortaleza em Mapas e Sobral em Mapas, que utilizam *OpenLayers* para fornecer à gestão e aos cidadãos uma ferramenta de visualização e análise de dados sobre a infraestrutura urbana, zoneamento, equipamentos públicos e informações sociodemográficas (Instituto de Planejamento de Fortaleza (IPPLAN), 2025).

No setor científico e ambiental, a biblioteca é frequentemente empregada em sistemas que exigem a visualização de grandes volumes de dados dinâmicos. Isso inclui sistemas de monitoramento de desmatamento, análise de bacias hidrográficas, modelagem de dispersão de poluentes ou a exibição de dados de sensores em tempo real. A capacidade do *OpenLayers* de lidar com diferentes projeções e renderizar camadas vetoriais complexas sobre bases de imagens de satélite o torna ideal para essas finalidades (NETO *et al.*, 2025).

Além disso, *OpenLayers* serve como a base tecnológica para inúmeros visualizadores

de dados georreferenciados em empresas de logística, agricultura de precisão e geomarketing, onde a análise da dimensão espacial dos dados é um diferencial competitivo.

2.4 Comparativo com Outras Bibliotecas

A escolha da biblioteca de mapas adequada depende dos objetivos do projeto, nível de personalização desejado e complexidade das funcionalidades de representação territorial envolvidas. Diversas bibliotecas JavaScript se destacam no desenvolvimento de aplicações cartográficas interativas, entre elas o *OpenLayers*, o *Leaflet*, o *Mapbox GL JS* e a *Google Maps API*. A Tabela 1 apresenta um comparativo entre essas principais bibliotecas, destacando suas características, vantagens e limitações.

Tabela 1 – Comparativo entre bibliotecas de mapeamento *Web*.

Biblioteca	**Principais Características**	**Vantagens**	**Limitações/desvantagens**
OpenLayers	Suporte a diversos formatos (WMS, WFS, *GeoJSON* entre outros), alto grau de personalização, aderência a padrões OGC.	Flexibilidade, controle completo, suporte a *WebGL*, múltiplas projeções.	Requer conhecimento técnico maior para configuração.
Leaflet	Foco em simplicidade, desempenho e leveza. Arquitetura extensível através de um vasto ecossistema de *plugins*.	Extrema facilidade de uso, baixa curva de aprendizado, ideal para projetos que necessitam de mapas interativos de forma rápida.	Suporte nativo limitado a funcionalidades avançadas como projeções complexas e renderização *WebGL*.
Mapbox GL JS	Motor de renderização de alto desempenho focado em *vector tiles* e estilização do lado do cliente via *WebGL*.	Excelente performance com grandes volumes de dados, visualizações ricas e fluidas, e alta qualidade visual dos mapas.	Licenciamento pago e limitações na personalização profunda.
Google Maps API	Plataforma comercial integrada a um vasto ecossistema de serviços do Google.	Riqueza de dados base, alta confiabilidade e funcionalidades prontas para uso, como cálculo de rotas e geocodificação.	Licenciamento restritivo e custos elevados em escala. Baixa flexibilidade para customização profunda e uso de fontes de dados externas.

Fonte: Elaborado pelo autor (2025).
Nota: Elaborado com base na documentação oficial de cada biblioteca.

A partir da Tabela 1, observa-se que o *OpenLayers*, embora exija maior conhecimento técnico inicial, é a opção que melhor equilibra flexibilidade, aderência a padrões OGC e independência de provedores comerciais, o que corrobora sua adoção neste trabalho.

2.5 Requisitos para Uso Prático

Para a integração do *OpenLayers* em uma aplicação *WebGIS*, é necessário considerar alguns requisitos técnicos essenciais que garantem o correto funcionamento da aplicação e o desempenho adequado no processamento de dados geográficos, como *font-end*, *back-end*, banco de dados geoespacial e servidor *web*.

O *front-end* requer um ambiente JavaScript moderno para o carregamento e renderização dos mapas. O *OpenLayers* é compatível com frameworks populares, como React, Laravel (2026), Angular e Vue, além de poder ser utilizado diretamente em *HyperText Markup Language* (HTML) com JavaScript puro (CUI *et al.*, 2022).

A disponibilização de dados cartográficos no *back-end* pode ser realizada por meio de servidores especializados, como *GeoServer (2026)* ou *MapServer*, ambos com suporte aos padrões WMS e WFS definidos pelo OGC (CUI *et al.*, 2022).

O uso de sistemas de gerenciamento de banco de dados com suporte espacial é altamente recomendável. O PostGIS, extensão espacial do PostgreSQL (2024), permite o armazenamento, consulta e manipulação de grandes volumes de dados georreferenciados de forma eficiente (CUI *et al.*, 2022).

É necessário um servidor para hospedar a aplicação, como Apache ou Nginx. Além disso, pode-se empregar soluções de *cache*, como o *GeoWebCache*, para otimizar o carregamento de camadas WMS e melhorar a escalabilidade da aplicação (CUI *et al.*, 2022).

2.6 Trabalhos Relacionados

A literatura recente no domínio dos *WebGIS* e da publicação de mapas na *Web* evidencia avanços significativos na adoção de tecnologias *open source* e na interoperabilidade entre servidores de mapas e bibliotecas de visualização geoespacial.

No contexto da publicação de dados geoespaciais na *Web*, Destro (2021) desenvolveu um *Protótipo de Sistema de Informação Geográfica Online Open Source*, com o objetivo de demonstrar a viabilidade de um SIG acessível via navegador, construído integralmente com ferramentas de código aberto. O autor propôs uma arquitetura modular composta por um servidor de mapas e um cliente Web interativo, ressaltando a importância da adoção de padrões do OGC, como WMS e WFS, para garantir interoperabilidade entre serviços. O trabalho destacou ainda o uso de bibliotecas de *front-end*, como o *OpenLayers*, capazes de consumir dados diretamente de

servidores como o *GeoServer (2026)*, reforçando o potencial das soluções livres em ambientes institucionais.

De forma complementar, Bastos e Camboim (2022) analisaram métodos de disponibilização e publicação de mapas topográficos na *Web*, comparando diferentes abordagens para veiculação cartográfica de dados vetoriais e *raster*. O estudo avaliou o desempenho, a eficiência de renderização e a escalabilidade de serviços de mapas utilizando o *GeoServer (2026)* e clientes baseados em JavaScript, como o *OpenLayers*. Os autores destacaram a importância da escolha adequada de protocolos - WMS, WMTS e WFS - conforme o tipo de dado e o nível de interatividade desejado, além de discutirem desafios como padronização de simbologia e desempenho em redes com baixa largura de banda.

A partir desses estudos, observa-se uma convergência entre as práticas em desenvolvimento de sistemas *Web* geoespaciais e a crescente adoção de ferramentas abertas para a publicação e análise de dados geográficos.

3 METODOLOGIA

Ao longo desta seção, descrevem-se e detalham-se as etapas de desenvolvimento e integração do *OpenLayers* na aplicação **"Sobral em Mapas (2026)"**, construída utilizando o *framework* Laravel (2026). A metodologia adotada neste trabalho caracteriza-se como pesquisa aplicada, tendo o estudo de caso como abordagem metodológica. O capítulo apresenta os procedimentos adotados para a implementação das funcionalidades do sistema, a configuração dos componentes da arquitetura *WebGIS* e as ferramentas utilizadas no processo de desenvolvimento.

3.1 Visão Geral da Aplicação

A aplicação "Sobral em Mapas (2026)"constitui-se como uma plataforma *WebGIS* proposta para o município de Sobral, Ceará, cujo objetivo central é disponibilizar uma interface interativa para a visualização e análise de dados geoespaciais por parte de cidadãos e gestores públicos. A arquitetura da solução emprega a biblioteca *OpenLayers* no *front-end*, responsável pela renderização dos mapas dinâmicos, e o *framework* Laravel (2026) no *back-end* para a gestão e o fornecimento dos dados. Dentre suas principais funcionalidades, a aplicação oferece a exibição de mapas base interativos com a sobreposição de camadas temáticas de relevância municipal, como rotas e dados de Zoneamento e Áreas de Risco. Para a manipulação de dados, a plataforma incorpora ferramentas de desenho e edição de geometrias, além de mecanismos de consulta espacial que permitem ao usuário obter informações detalhadas de feições diretamente no mapa. Por fim, o sistema também contempla a exportação dos mapas e dados geográficos em múltiplos formatos, garantindo a interoperabilidade com outras ferramentas de análise.

3.2 Ferramentas Utilizadas

As ferramentas e tecnologias empregadas no desenvolvimento da aplicação "Sobral em Mapas (2026)"foram selecionadas com o propósito de garantir uma plataforma robusta, interativa e escalável, capaz de atender às demandas de visualização e análise de dados geográficos do município de Sobral.

O *OpenLayers* foi a biblioteca JavaScript escolhida para a construção e manipulação de mapas interativos. Por ser uma solução de código aberto amplamente consolidada, oferece suporte nativo a diversos formatos e serviços geoespaciais, além de permitir ampla personalização de camadas, projeções e interações no ambiente *WebGIS (OpenLayers Team, 2025b)*.

O Laravel (2026), *framework* PHP de código aberto, foi adotado para o desenvolvimento do *back-end* da aplicação. Ele é responsável pelo fornecimento de APIs RESTful que disponibilizam os dados cartográficos ao cliente, além de gerenciar a autenticação de usuários, o controle de acesso e a integração com o banco de dados *(Laravel Team, 2024)*.

Para o gerenciamento e publicação de dados espaciais, utilizou-se o *GeoServer (2026)*, um servidor de mapas de código aberto compatível com os principais padrões OGC. Essa ferramenta possibilita a distribuição eficiente dos dados geográficos, permitindo que o *OpenLayers* consuma e exiba essas informações de forma dinâmica *(GeoServer Project, 2024)*.

O PostgreSQL (2024), em conjunto com sua extensão espacial PostGIS, foi adotado como banco de dados principal da aplicação. Essa combinação possibilita o armazenamento estruturado e eficiente de dados georreferenciados, além de oferecer suporte a consultas espaciais complexas, como exploração de proximidade e rede, utilizando funções e índices espaciais que permitem calcular distâncias, medir áreas e analisar relações de mapeamento, o que é essencial para análises geográficas e operações com sobreposição de camadas *(PostGIS Development Group, 2024)*.

Por fim, o Docker foi utilizado para a conteinerização da aplicação, garantindo um ambiente de desenvolvimento e implantação homogêneo. O uso dessa ferramenta facilita a replicação da infraestrutura, simplifica a configuração dos serviços e assegura maior portabilidade entre ambientes de teste, desenvolvimento e produção *(Docker Inc., 2024)*.

3.3 Arquitetura da Solução

A arquitetura da solução, conforme ilustrada na Figura 1, foi projetada para ser modular, com separação clara entre o *front-end* e o *back-end*, utilizando a abordagem *client-server*.O fluxo de interação entre as camadas ocorre conforme descrito a seguir.

1. **Cliente (Usuário):** O usuário interage com a aplicação através de um navegador *web*, que executa a camada de apresentação do sistema.
2. **Camada de Apresentação (*front-end*):** Construída com o *OpenLayers*, responsável pela criação e manipulação do mapa interativo, e com o *Blade*, utilizado para renderizar as páginas HTML no Laravel (2026). Esta camada é encarregada da renderização dos mapas e de toda a interface do usuário. Ela realiza dois tipos principais de requisições:
    - **Requisições de Aplicação:** Chamadas à API RESTful do La-

ravel (2026) para obter dados não espaciais, listas de camadas disponíveis, autenticação de usuários e outras lógicas de negócio;

- **Requisições Geoespaciais:** Chamadas diretas aos serviços OGC publicados pelo *GeoServer (2026)* para buscar os fragmentos de imagem do mapa e os dados vetoriais das camadas.

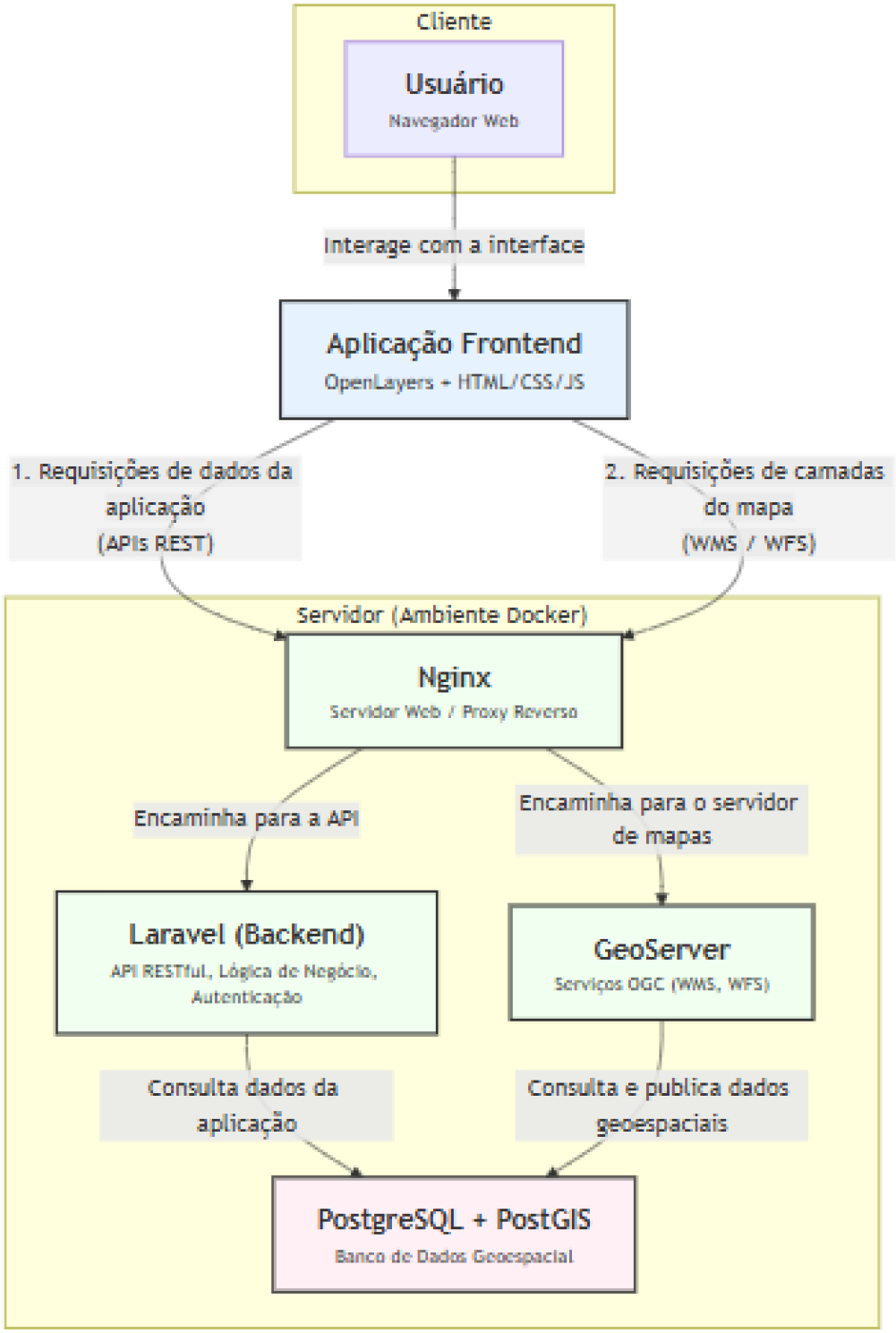
3. **Camada de Servidor:** Todos os componentes do servidor são orquestrados e executados em contêineres Docker isolados, conforme descrito a seguir:
    - **Nginx:** Atua como um *gateway* (porta de entrada) e *proxy* reverso, recebendo todas as requisições do cliente e as direcionando para o serviço apropriado — requisições de API são encaminhadas ao Laravel (2026), enquanto requisições de mapas são enviadas ao *GeoServer (2026)*;
    - **Laravel (2026) (*back-end*):** Serve como o núcleo da aplicação, controlando as regras de negócio, o acesso aos dados e a comunicação com o banco de dados para operações não geoespaciais ou de gerenciamento;
    - ***GeoServer (2026)*:** Atua como servidor de mapas especializado, conectando-se ao banco de dados para interpretar os dados geográficos e publicá-los em formatos padronizados que o *OpenLayers* pode consumir.
4. **Camada de Persistência:** O PostgreSQL (2024), com a extensão PostGIS, funciona como o repositório central de dados, armazenando tanto informações tabulares quanto dados vetoriais, suas geometrias e sistemas de coordenadas.

Essa arquitetura desacoplada permite que cada componente seja mantido e escalado de forma independente, constituindo uma solução robusta e moderna para uma plataforma *WebGIS*.

3.4 Funcionalidades Desenvolvidas com *OpenLayers*

As funcionalidades desenvolvidas com a biblioteca *OpenLayers* são apresentadas a seguir, detalhando o conceito teórico, o processo de implementação e o respectivo trecho de código que demonstra seu funcionamento na aplicação.

Figura 1 – Diagrama da arquitetura da aplicação Sobral em Mapas (2026)



Fonte: Elaborado pelo autor(2025)

3.4.1 Exibição de Mapa Base

O primeiro passo para a construção do mapa foi a exibição de uma camada base utilizando o *OpenStreetMap*, que fornece um fundo cartográfico gratuito sobre o qual outras camadas geográficas podem ser sobrepostas.

O Código-Fonte 2 apresenta o trecho responsável pela inicialização do mapa principal da aplicação "Sobral em Mapas (2026)". Nesse trecho, é criada uma instância do objeto `ol.Map`, responsável por renderizar o mapa no elemento `HTML`, identificado como `map`. A camada base (`TileLayer`) utiliza o serviço `OpenStreetMap (OSM)` como fonte de dados, com a propriedade `preload` configurada para otimizar o carregamento prévio de blocos de imagem (*tiles*) nas áreas adjacentes ao campo de visão, proporcionando uma navegação mais fluida. A visão (`View`) define o centro do mapa em coordenadas geográficas de Sobral (longitude -40.35, latitude -3.69), convertidas para a projeção `EPSG:3857` por meio da função `ol.proj.fromLonLat`. Além disso, são definidos limites de ampliação (`minZoom` e `maxZoom`) e a projeção padrão do mapa. O trecho também personaliza as interações de *zoom* do usuário, ajustando a sensibilidade e a animação da rolagem do mouse, resultando em uma experiência de navegação mais responsiva e ergonômica.

Código-fonte 2 – Inicialização do mapa base com *OpenLayers* na aplicação Sobral em Mapas (2026)

```
1
2  export function initializeMap() {
3      var map = new ol.Map({
4          target: "map",
5          layers: [
6              new ol.layer.Tile({
7                  source: new ol.source.OSM(),
8                  preload: 10, // Preload de 4 n veis de
                       tiles ao redor
9              }),
10         ],
11         view: new ol.View({
12             center: ol.proj.fromLonLat([-40.35, -3.69]),
13             zoom: 15,
```

```
14              maxZoom: 17,
15              minZoom: 10,
16              projection: ol.proj.get("EPSG:3857"),
17          }),
18          interactions: ol.interaction.defaults({
19
20              mouseWheelZoom: new ol.interaction.
                    MouseWheelZoom({
21                  duration: 100, // Dura  o da anima  o
                        de zoom, em milissegundos
22                  zoomDelta: 10, // Controla o quanto o zoom
                        aumenta ou diminui a cada rota  o do
                        scroll
23              }),
24          }),
25      });
26
27      window.map = map;
28      loadSobralBoundary();
29  }
```

Fonte: Adaptado de (ASRSQ, 2025). Repositório Sobral em Mapas.

3.4.2 Adição de Marcadores e Camadas Temáticas

Além do mapa base, foram adicionadas camadas temáticas que representam diferentes informações geoespaciais, como zoneamento urbano, áreas de risco e projeções de planejamento territorial. Essas camadas são fornecidas pelo servidor *GeoServer (2026)* por meio do serviço WMS, permitindo que dados geográficos sejam exibidos de forma dinâmica e interativa no navegador.

O Código-Fonte 3 apresenta o trecho de código responsável pela configuração e renderização dessas camadas. Nessa implementação, o objeto `ol.layer.Tile` cria uma camada de mosaico, enquanto `ol.source.TileWMS` define a origem dos dados, especificando parâmetros

como o nome da camada (`LAYERS`), formato da imagem (`FORMAT`), transparência e versão do protocolo WMS (`VERSION`). O carregamento das imagens é realizado por uma função personalizada (`tileLoadFunction`), que executa requisições assíncronas via `XMLHttpRequest` e converte as respostas em imagens renderizáveis no mapa. Esse método também inclui tratamento de erros e controle de cache, garantindo maior estabilidade e desempenho no carregamento dos mosaicos.

Código-fonte 3 – Carregamento das camadas temáticas a partir do *GeoServer (2026)*

```
const geoServerLayer = new ol.layer.Tile({
    source: new ol.source.TileWMS({
        url: "api/proxy-wms",
        params: {
            LAYERS: layerName,
            TILED: true,
            FORMAT: "image/png",
            TRANSPARENT: true,
            VERSION: "1.3.0",
            SRS: crs,
        },
        serverType: "geoserver",
        crossOrigin: "anonymous",
        tileLoadFunction: function (imageTile, src) {
            const xhr = new XMLHttpRequest();
            xhr.open("GET", src, true);
            xhr.responseType = "blob";

            xhr.onload = async function () {
                if (xhr.status === 200) {
                    const reader = new FileReader();
                    reader.readAsDataURL(xhr.response);
                    reader.onload = function () {
                        imageTile.getImage().src = reader.
                            result;
```

```
                    };
                } else {
                    await showError(`Erro ao carregar tile
                        ${src}. Status: ${xhr.status}`,
                        geoServerLayer.get("name"));
                }
            };
            xhr.onerror = async function () {
                await showError(`Erro de rede ao carregar
                    tile ${src}.`, geoServerLayer.get("name"
                    ));
            };
            xhr.send();
        },
        cacheSize: 2048,
        preload: 4,
    }),
    name: layerName,
    zIndex: order || 1,
    maxResolution: maxScale || undefined,
});
```

Fonte: Adaptado de (ASRSQ, 2025). Repositório Sobral em Mapas.

3.4.3 Ferramentas de Desenho e Edição

Além da exibição de camadas temáticas provenientes do *GeoServer (2026)*, a aplicação também oferece ferramentas que permitem aos usuários desenhar e editar elementos geoespaciais diretamente sobre o mapa. Essa funcionalidade é primordial para permitir a marcação, a medição e o delineamento de áreas de interesse, como zonas urbanas ou pontos de referência.

O primeiro passo para habilitar o desenho foi a criação de uma camada vetorial utilizando `ol.layer.Vector`, responsável por armazenar as feições (pontos, linhas e polígonos)

criadas pelo usuário. Essa camada é configurada com um estilo dinâmico, definindo propriedades visuais como a cor da borda, o preenchimento semitransparente e o formato dos símbolos de ponto, conforme apresentado no Código-Fonte 4.

Código-fonte 4 – Camada vetorial para desenho e edição

```
const vectorLayer = new ol.layer.Vector({
    source: source,
    style: function (feature) {
        return new ol.style.Style({
            stroke: new ol.style.Stroke({
                color: selectedLineColor,
                width: 2,
            }),
            fill: new ol.style.Fill({
                color: hexToRGBA(selectedLineColor, 0.4),
                    // Cor preenchida semi-transparente
            }),
            image: new ol.style.Circle({
                radius: 5,
                fill: new ol.style.Fill({
                    color: selectedLineColor, // Cor da
                        bolinha
                }),
                stroke: new ol.style.Stroke({
                    color: "#000000",
                    width: 1,
                }),
            }),
        });
    },
});
window.map.addLayer(vectorLayer);
```

Fonte: Adaptado de (ASRSQ, 2025). Repositório Sobral em Mapas.

A camada criada serve como o repositório principal das geometrias desenhadas, permitindo que o usuário interaja com o mapa de forma intuitiva. A partir dessa base, são adicionadas as interações de desenho e modificação do *OpenLayers*, que permitem criar, mover e ajustar polígonos e marcadores. Essas interações utilizam o mesmo `source` da camada vetorial, garantindo que todas as edições sejam refletidas em tempo real sobre o mapa.

3.4.4 Exportação de Mapas (Implementação Adicional)

A funcionalidade de exportação de mapas foi desenvolvida como uma ampliação personalizada do sistema, com o objetivo de permitir que o usuário selecione uma área específica no mapa e a exporte em diferentes formatos de imagem (*Portable Network Graphic* (PNG), *Joint Photographic Experts Group* (JPG)) ou como documento *Portable Document Format* (PDF) para impressão ou outro fim. Essa implementação amplia as capacidades nativas do *OpenLayers*, aproveitando sua API baseada em elementos `canvas`, responsável pela renderização das camadas do mapa no navegador.

O processo tem início com a captura das coordenadas correspondentes à área de seleção definida pelo usuário na interface. Essa área é delimitada por meio do posicionamento e dimensionamento da janela de captura, conforme ilustrado no Código-Fonte 5. Esses valores são utilizados pela função principal de exportação da área, disponível no Apêndice A, responsável por compor a imagem final da região selecionada.

Código-fonte 5 – Seleção da área de impressão

```
1 export function getSelectionAreaPixels() {
2     const selectionArea = document.getElementById('
        selection-area');
3     const mapElement = document.getElementById('map');
4
5     if (!selectionArea || !mapElement) return null;
6
7     const selectionRect = selectionArea.
        getBoundingClientRect();
8     const mapRect = mapElement.getBoundingClientRect();
```

```
9
10      return {
11          left: selectionRect.left - mapRect.left,
12          top: selectionRect.top - mapRect.top,
13          width: selectionRect.width,
14          height: selectionRect.height
15      };
16  }
```

Fonte: Adaptado de (ASRSQ, 2025). Repositório Sobral em Mapas.

Antes da exportação, o método `map.renderSync()` é chamado para garantir a renderização imediata de todas as camadas visíveis. Isso é importante porque o *OpenLayers* trabalha com renderização assíncrona — as camadas podem ser atualizadas em momentos diferentes, e o uso desse método garante que todas as *tiles* e vetores estejam totalmente desenhados antes da captura da imagem. Em seguida, o código percorre todos os `canvas` gerados internamente pelo *OpenLayers*, identificados pelo seletor `.ol-layer canvas`. Cada camada de mapa é desenhada em seu próprio `canvas`, o que permite um controle mais preciso sobre transparência e transformações aplicadas. O código recria essa estrutura em um `canvas` temporário, aplicando as mesmas transformações para alinhar corretamente todas as camadas e depois recortar apenas a área correspondente à seleção do usuário.

Dessa forma, a aplicação utiliza os recursos gráficos já processados pelo *OpenLayers*, sem depender de renderizações externas, assegurando que as camadas vetoriais e *raster* sejam combinadas fielmente à visualização do usuário no momento da exportação. O uso da biblioteca jsPDF viabiliza a conversão direta para PDF, enquanto o método `toDataURL()` do `canvas` transforma o conteúdo em uma *string base64* de imagem, que é uma representação de dados binários convertidos em uma sequência de texto, permitindo a exportação pelo navegador.

3.5 Integração com Laravel (2026)

A integração entre a camada de apresentação, implementada com *OpenLayers*, e o servidor desenvolvido com Laravel (2026), constitui um dos pilares da arquitetura adotada para a plataforma. Nesse contexto, o Laravel (2026) assume papel de provedor de uma API RESTful, responsável por expor recursos geoespaciais e tabulares para consumo pelo *front-end*, além de

gerenciar autenticação, autorização, controle de acesso e lógica de negócio. A comunicação entre *front-end* e *back-end* ocorre por meio de chamadas *Hypertext Transfer Protocol* (HTTP) a *endpoints* definidos no Laravel (2026), os quais retornam dados estruturados ou referências de camadas territoriais que são posteriormente exibidas pelo *OpenLayers*.

3.5.1 Integração com o GeoServer (2026) via API

O Laravel (2026) atua como intermediário entre *OpenLayers* e *GeoServer (2026)*, permitindo que o *front-end* solicite e receba camadas geoespaciais no formato adequado. O controlador `GeoServerProxyController`, disponível no Apêndice A, que permite integração com o Laravel (2026), é responsável por encaminhar as requisições do cliente ao *GeoServer (2026)* e devolver as respostas processadas.

O método `proxyWms` recebe os parâmetros da requisição WMS e retorna as imagens das camadas no formato PNG, enquanto o método `getLegendGraphic` obtém as legendas das camadas publicadas. Essa estrutura garante uma comunicação segura e controlada, mantendo o *GeoServer (2026)* protegido e otimizando o carregamento das camadas no mapa.

3.5.2 Armazenamento e Recuperação de Dados Geoespaciais

Com dados geográficos do sistema armazenados no banco de dados PostGIS integrado ao *GeoServer (2026)*, o *back-end* acessa esses dados por meio de um serviço intermediário denominado `GeoServerService`, ilustrado no Código-Fonte 6, que tem a função de abstrair a comunicação com o *GeoServer (2026)*, encaminhando as requisições para o cliente responsável. Além disso, executa as chamadas HTTP ao servidor geoespacial, atuando como uma camada de integração e controlando a autenticação e o fluxo de dados entre o *GeoServer (2026)* e a aplicação *web*.

Código-fonte 6 – Serviço para recuperação de camadas WMS

```
1 class GeoServerService
2 {
3     protected $geoServerClient;
4
5     public function __construct(GeoServerClient
        $geoServerClient)
```

```
6       {
7           $this->geoServerClient = $geoServerClient;
8       }
9
10      public function getWmsLayer(array $params)
11      {
12          return $this->geoServerClient->getWms($params);
13      }
14
15      public function getLegendGraphic(string $layer)
16      {
17          return $this->geoServerClient->getLegendGraphic(
                $layer);
18      }
19  }
```

Fonte: Adaptado de (ASRSQ, 2025). Repositório Sobral em Mapas.

Por meio desse serviço, o sistema requisita as camadas geográficas renderizadas pelo *GeoServer (2026)* e recupera dinamicamente as legendas correspondentes às camadas publicadas.

4 RESULTADOS

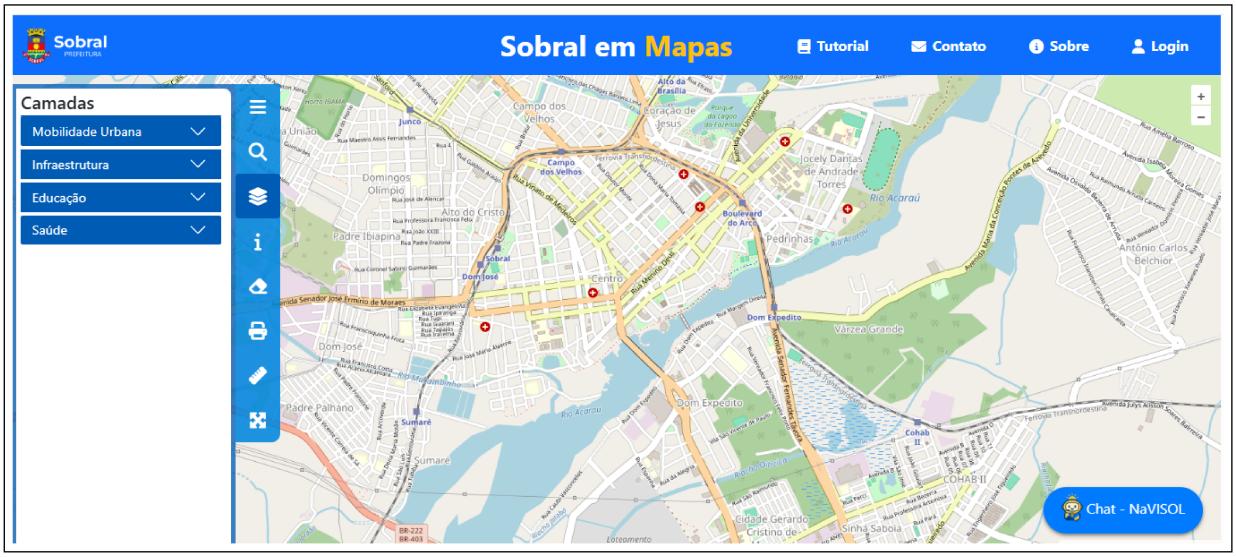
Este capítulo apresenta a análise crítica dos resultados obtidos com o desenvolvimento e aprimoramento da aplicação "Sobral em Mapas (2026)". A avaliação abrange aspectos como desempenho, usabilidade e eficácia das funcionalidades implementadas, buscando demonstrar se os objetivos estabelecidos no início do trabalho foram alcançados e se a solução proposta se mostra adequada ao contexto de um sistema *WebGIS* municipal.

Para validar a aplicação, foi disponibilizado o endereço eletrônico da plataforma e um formulário de avaliação, direcionados à comunidade acadêmica e a potenciais usuários. As respostas coletadas foram analisadas com o intuito de mensurar a percepção dos usuários sobre a utilidade, acessibilidade e funcionalidade do sistema. O questionário utilizado encontra-se disponível no Apêndice B.

4.1 Demonstração de Resultados no *Front-end*

Como resultado de implementação da biblioteca *OpenLayers* no *front-end* da aplicação, a Figura 2 apresenta a página inicial do sistema. Nessa interface, o mapa é carregado automaticamente, estando centralizado na área urbana de Sobral. Além disso, o menu lateral oferece uma navegação fluida entre as diferentes funcionalidades disponíveis, facilitando o acesso aos recursos do sistema de forma intuitiva e organizada.

Figura 2 – Interface da página inicial

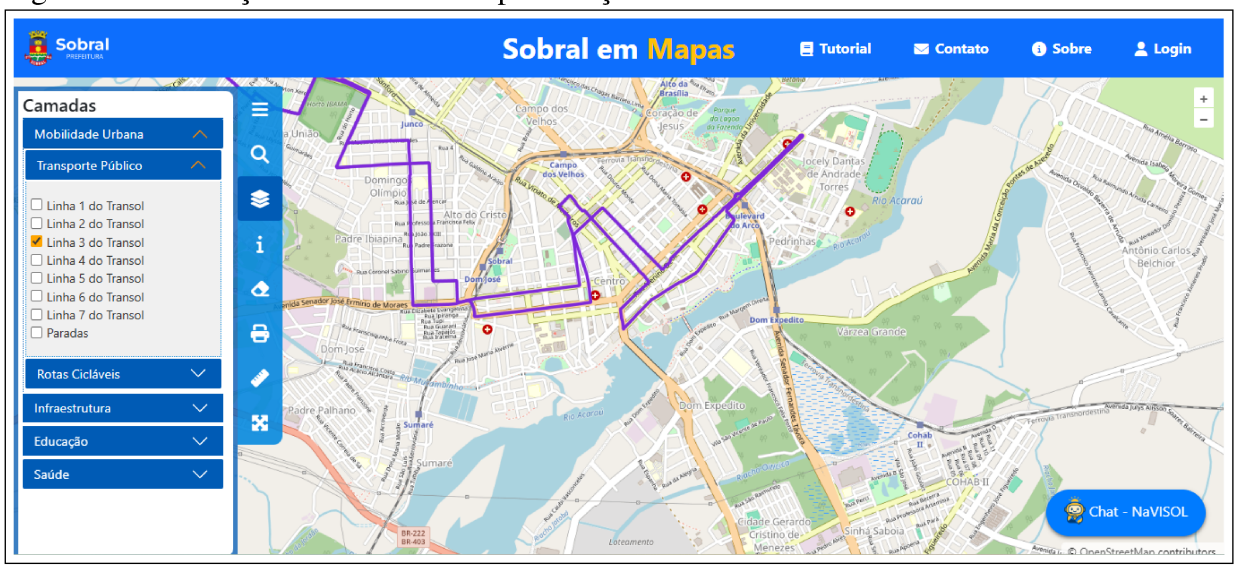


Fonte: Elaborado pelo autor (2025).

A partir do menu lateral, ilustrado na Figura 3, o usuário pode selecionar e adicionar

diferentes camadas temáticas ao mapa, como rotas do transporte público Transol, ciclovias e a localização dos postos de saúde, entre outras opções. O carregamento das camadas apresentou comportamento estável e responsivo, com renderização adequada e boa legibilidade, como também demonstrado na Figura 3. Esses resultados validam a integração eficiente entre os componentes do sistema em diferentes níveis de *zoom*.

Figura 3 – Utilização do menu lateral para adição de camadas temáticas



Fonte: Elaborado pelo autor (2025).

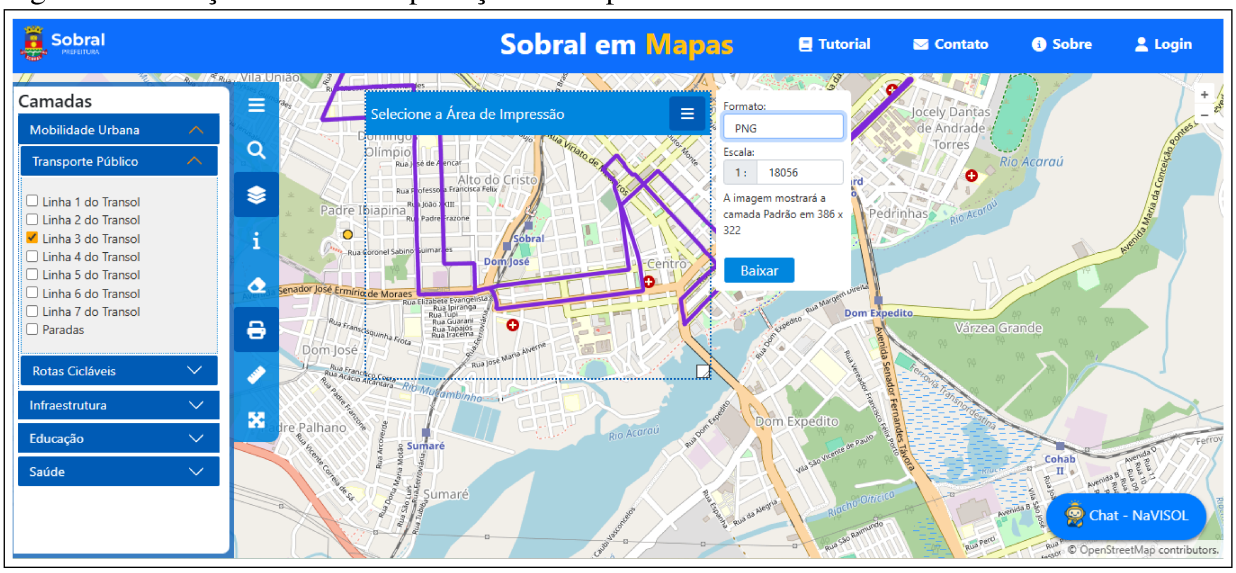
É possível também medir linhas ou áreas no mapa, conforme mostra a Figura 4. A implementação da camada vetorial e dos controles de interação do *OpenLayers* permite ao usuário criar polígonos, linhas e pontos de forma intuitiva, definir a cor dos traçados e limpar todos os desenhos aplicados. Essa funcionalidade foi implementada a partir das interações de desenho (*Draw Interaction*) da biblioteca, reforçando sua adequação para aplicações operacionais no contexto de SIG.

Além das ferramentas de medição, o sistema também permite exportar o mapa em diferentes formatos, como PNG, JPG e PDF. Essa funcionalidade constitui um aprimoramento que possibilita ao usuário definir a área desejada para impressão, ajustar a escala e realizar o *download* do recorte selecionado. A Figura 5 apresenta a interface de seleção e exportação, destacando o processo de delimitação da área e escolha do formato de saída.

Figura 4 – Ferramenta de medição e desenho no mapa

Fonte: Elaborado pelo autor (2025).

Figura 5 – Seleção de área e exportação do mapa



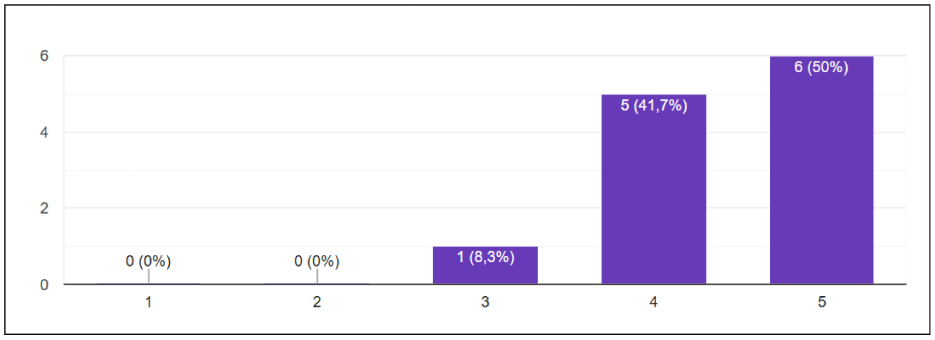
Fonte: Elaborado pelo autor (2025).

4.2 Validação de Usabilidade

Com o objetivo de mensurar e validar a usabilidade da plataforma, foram avaliados os resultados obtidos a partir do formulário aplicado aos potenciais usuários após os testes. A maioria dos entrevistados consideraram a navegação fluida e fácil, corroborando eficiência e robustez da arquitetura utilizada. O desempenho geral da aplicação é satisfatório com carregamento das camadas temáticas dentro de um intervalo considerado adequado, conforme ilustrado nas Figuras 6 e 7.

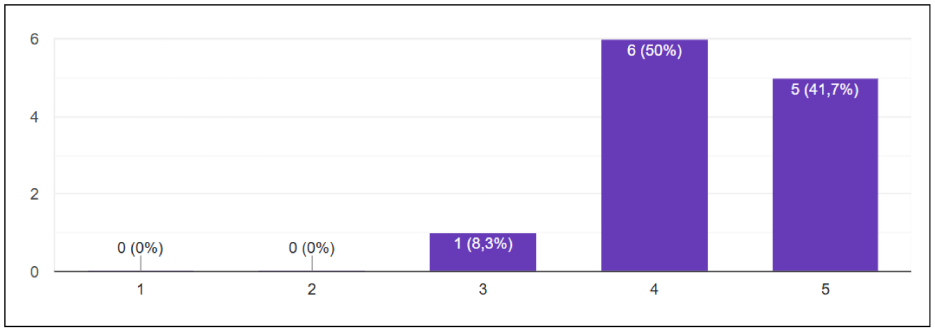
De forma geral, a análise integrada dos resultados evidencia que a plataforma atingiu

Figura 6 – Facilidade de navegação no mapa



Fonte: Elaborado pelo autor (2025).

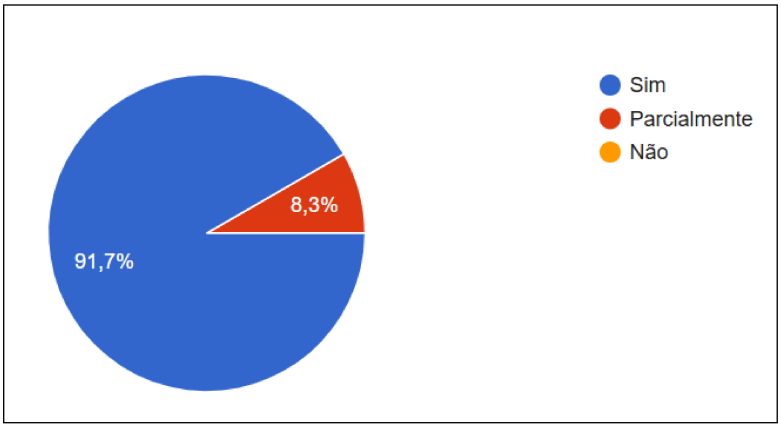
Figura 7 – Carregamento do mapa



Fonte: Elaborado pelo autor (2025).

de maneira consistente os objetivos propostos. A combinação do Laravel (2026), *GeoServer (2026)* e *OpenLayers* mostrou-se uma arquitetura poderosa para um *WebGIS* agregando valor tecnológico à plataforma municipal, atendendo as necessidades de interatividade como marcações de pontos, sobreposições de camadas, entre outros, conforme exibe a Figura 8.

Figura 8 – Atendimento de necessidades interativas



Fonte: Elaborado pelo autor (2025).

O sistema atual não apenas mantém as funcionalidades essenciais de um SIG, como também incorpora melhorias significativas, principalmente na impressão em escala e em dispositivos móveis.

5 CONCLUSÕES E TRABALHOS FUTUROS

O desenvolvimento deste trabalho permitiu evidenciar que o uso da biblioteca *OpenLayers* atendeu de forma satisfatória aos objetivos funcionais e de desempenho estabelecidos em aplicações *WebGIS*. A solução demonstrou ser capaz de realizar a exibição interativa de mapas, o gerenciamento de camadas temáticas e a integração com serviços geoespaciais de maneira eficiente e responsiva. A integração entre o Laravel (2026) no *back-end*, o *GeoServer (2026)* na publicação dos dados geoespaciais e o *OpenLayers* no *front-end* proporcionou um fluxo consistente, seguro e modular de dados geográficos, garantindo desempenho e manutenção facilitada da solução desenvolvida.

Os resultados obtidos a partir da avaliação com 12 usuários indicaram que a aplicação alcançou plenamente seu objetivo principal, especialmente no que se refere à usabilidade e à efetividade dos recursos interativos. As funcionalidades de navegação, controle de zoom, sobreposição de camadas temáticas e realização de consultas espaciais atendem de forma satisfatória às demandas de análise e exploração territorial. Além disso, a implementação dos mecanismos de exportação e captura da visualização do mapa ampliou as possibilidades de uso da plataforma, permitindo o compartilhamento e a documentação de informações territoriais de maneira prática e eficiente, tanto em contextos administrativos quanto técnicos. A análise de usabilidade, por sua vez, evidenciou interface intuitiva e nível adequado de acessibilidade, reforçando a viabilidade da aplicação como ferramenta de apoio ao planejamento urbano e à gestão pública municipal no município de Sobral.

Como continuidade deste projeto, propõem-se aprimoramentos voltados à ampliação das funcionalidades e do desempenho da aplicação. Entre eles, destacam-se a implementação de um módulo de impressão em escala real, com suporte a formatos padrão (A4–A0) e configuração de *Dots Per Inch* (DPI) variável. Recomenda-se ainda, para desenvolvimentos futuros, a adição de recursos analíticos espaciais, como cálculo de áreas de influência e a interseção entre camadas, bem como o aprimoramento da aplicação para dispositivos móveis, melhorando acessibilidade e desempenho.

Durante o desenvolvimento do trabalho, alguns desafios impactaram o andamento das atividades, especialmente no que se refere ao processo de homologação da aplicação. O acesso limitado à equipe responsável pela validação do sistema, agravado pelo período eleitoral e pela subsequente mudança de gestão municipal, dificultou a realização de testes contínuos com os usuários finais. Como forma de mitigação desses obstáculos, adotou-se uma abordagem

iterativa de desenvolvimento, com ajustes progressivos baseados em testes internos, avaliações com potenciais usuários e contribuições da comunidade acadêmica, realizadas sempre que possível. Essa estratégia permitiu a continuidade do projeto e contribuiu para a consolidação das funcionalidades implementadas, garantindo que a aplicação atendesse, dentro das limitações impostas, aos requisitos propostos.

REFERÊNCIAS

ALMEIDA, F.; OUTROS. A cloud-based webgis tool for communicating integrated ecosystem modeling results. **Journal of Environmental Management**, v. 360, p. 120560, 2025. Acesso em: 20 set. 2025. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0301479725003482>.

ASRSQ. **Sobral em Mapas: Aplicação WebGIS baseada em OpenLayers e Laravel**. 2025. Acesso em: 4 out. 2025. Disponível em: <https://github.com/ASRSQ/sobralmapas>.

BASTOS, T. L.; CAMBOIM, S. P. Comparação e análise de métodos de disponibilização e publicação de mapas topográficos na web. **Revista Brasileira de Cartografia**, v. 74, n. 3, p. 540–560, 2022. Acesso em: 8 nov. 2025. Disponível em: <https://seer.ufu.br/index.php/revistabrasileiracartografia/article/view/75024>.

BATISTA, L.; ANTUNES, R. d. S. **Mapa interno interativo do campus UTFPR-Ct**. 2021. Trabalho de Conclusão de Curso — Bacharelado em Sistemas de Informação, Universidade Tecnológica Federal do Paraná. Acesso em: 22 ago. 2025. Disponível em: <http://repositorio.utfpr.edu.br/jspui/handle/1/26526>.

CUI, L.; LI, W.; WANG, X.; NIU, X. Design and implementation of open-source webgis system for orchard land management. **Advances in Computer, Signals and Systems**, v. 6, n. 3, p. 26–34, 2022. Acesso em: 9 out. 2025. Disponível em: <https://www.clausiuspress.com/article/4213.html>.

DESTRO, D. N. **Protótipo de sistema de informação geográfica online open source**. Criciúma: [s.n.], 2021. Trabalho de Conclusão de Curso — Engenharia de Computação, Universidade do Extremo Sul Catarinense. Acesso em: 8 nov. 2025. Disponível em: <http://repositorio.unesc.net/handle/1/11949>.

Docker Inc. **Docker Documentation**. [S.l.], 2024. Acesso em: 4 out. 2025. Disponível em: <https://docs.docker.com>.

DOE, J.; SMITH, J.; OUTROS. Vector data rendering performance analysis of open-source web mapping libraries. **ISPRS International Journal of Geo-Information**, v. 14, n. 9, p. 336, 2024. Acesso em: 10 out. 2025. Disponível em: <https://www.mdpi.com/2220-9964/14/9/336>.

FOERSTER, T.; SCHAEFFER, B.; BRAUNER, J.; JIRKA, S. Open geospatial consortium (ogc) standards and their role in webgis interoperability. **Computers & Geosciences**, v. 37, n. 7, p. 727–735, 2011.

GeoServer. **GeoServer: Open Source Server for Sharing Geospatial Data**. 2026. <https://geoserver.org>. Acesso em: 29 jan. 2026.

GeoServer Project. **GeoServer User Manual**. [S.l.], 2024. Acesso em: 4 out. 2025. Disponível em: <https://geoserver.org>.

Google. **Google Maps Platform Documentation**. 2025. Acesso em: 2 out. 2025. Disponível em: <https://developers.google.com/maps>.

Instituto de Planejamento de Fortaleza (IPPLAN). **Fortaleza em Mapas**. 2025. Acesso em: 3 out. 2025. Disponível em: <https://www.ipplan.fortaleza.ce.gov.br/o-que-fazemos/dados-e-evidencias/fortaleza-em-mapas>.

KOMBE, E.; OUTROS. Comprehensive review of gis and webgis. **ResearchGate Preprint**, 2022. Acesso em: 15 set. 2025. Disponível em: <https://www.researchgate.net/publication/363316662_Comprehensive_review_of_GIS_and_web_GIS>.

Laravel. **Laravel: The PHP Framework for Web Artisans**. 2026. <https://laravel.com>. Acesso em: 29 jan. 2026.

Laravel Team. **Laravel Documentation**. [S.l.], 2024. Acesso em: 4 out. 2025. Disponível em: <https://laravel.com/docs>.

NETO, A.; NERES, M.; CARVALHO, M. Uma arquitetura de data lake de dados ambientais e socioeconômicos para fomentar pesquisa e inovação. In: **Anais do XVI Workshop de Computação Aplicada à Gestão do Meio Ambiente e Recursos Naturais**. SBC, 2025. p. 296–305. Acesso em: 5 out. 2025. Disponível em: <https://sol.sbc.org.br/index.php/wcama/article/view/36107>.

OpenLayers Team. **OpenLayers 3: A Web Mapping Framework**. 2025. Acesso em: 30 ago. 2025. Disponível em: <https://openlayers.org>.

OpenLayers Team. **OpenLayers — Welcome**. 2025. Acesso em: 2 out. 2025. Disponível em: <https://openlayers.org/>.

PostGIS Development Group. **PostGIS 3.4 Manual**. [S.l.], 2024. Acesso em: 4 out. 2025. Disponível em: <https://postgis.net/documentation/>.

PostgreSQL. **PostgreSQL**. 2024. Acesso em: 03 fev. 2026. Disponível em: <https://www.postgresql.org>.

Secretaria de Urbanismo e Meio Ambiente de Sobral. **Dados Geográficos de Sobral**. 2025. Acesso em: 30 ago. 2025. Disponível em: <https://seuma.sobral.ce.gov.br/iinformativos/dados-geograficos>.

SHABANI, S.; OUTROS. Web gis and cloud computing for defense decision making. In: **International Conference on Geoinformatics**. [s.n.], 2024. Acesso em: 15 set. 2025. Disponível em: <https://www.researchgate.net/publication/395751162_WEB_GIS_AND_CLOUD_COMPUTING_FOR_DEFENSE_DECISION_MAKING>.

Sobral em Mapas. **Sobral em Mapas**. 2026. <https://spacesolutions.alphi.media/sobralmapas/public>. Acesso em: 29 jan. 2026.

Universidade Federal do Pará. **CicloGis: um sistema web para a comunidade do ciclismo de Belém**. Belém: [s.n.], 2024. Trabalho de Conclusão de Curso — Engenharia da Computação, UFPA.

VINUEZA-MARTINEZ, J.; CORREA-PERALTA, M.; RAMIREZ-ANORMALIZA, R.; ARIAS, O. F.; PAREDES, D. V. Geographic information systems (giss) based on webgis architecture: Bibliometric analysis of the current status and research trends. **Sustainability**, v. 16, n. 15, p. 6439, 2024. Acesso em: 22 ago. 2025.

Wikipedia contributors. **Google Maps — Wikipédia, The Free Encyclopedia**. 2025. Acesso em: 2 out. 2025. Disponível em: <https://en.wikipedia.org/wiki/Google_Maps>.

Wikipedia contributors. **OpenLayers**. 2025. Acesso em: 3 out. 2025. Disponível em: <https://en.wikipedia.org/wiki/OpenLayers>.

Wikipédia. **Leaflet — Wikipédia, a enciclopédia livre**. 2023. Acesso em: 3 out. 2025. Disponível em: <https://pt.wikipedia.org/wiki/Leaflet>.

APÊNDICE A – CÓDIGOS-FONTES UTILIZADOS NA IMPLEMENTAÇÃO DA BIBLIOTECA OPENLAYERS NA APLICAÇÃO SOBRAL EM MAPAS

Código-fonte 7 – Exportação da área selecionada

```
export function exportVisibleMapArea(map) {
  const selection = getSelectionAreaPixels();
  if (!selection) {
    console.error('Área de seleção não encontrada.');
    return;
  }

  const { left, top, width, height } = selection;
  const format = document.getElementById('format').value;
  // Espera o mapa renderizar tudo
  map.renderSync(); // Garante renderização imediata

  setTimeout(()=> {
    const mapCanvas = document.createElement('canvas');
    mapCanvas.width = width;
    mapCanvas.height = height;
    const mapContext = mapCanvas.getContext('2d');
    // Itera sobre todos os canvas das camadas do OL
    document.querySelectorAll('.ol-layer canvas').forEach
       ((canvas) => {
      if (canvas.width > 0 && canvas.height > 0) {
        const opacity = canvas.parentNode.style.opacity;
        mapContext.globalAlpha = opacity === '' ? 1 :
           Number(opacity);

        const transform = canvas.style.transform;
        if (transform && transform.startsWith('matrix'))
           {
```

```
26            const matrix = transform
27            .match(/^matrix\(([^)]+)\)$/)[1]
28            .split(',')
29            .map(Number);
30            mapContext.setTransform(...matrix);
31          }
32          // Recorta apenas a parte da sele  o
33          mapContext.drawImage(
34          canvas,
35          left, top, width, height,
36          0, 0, width, height
37          );
38        }
39      });
40
41      // Gera o conte do final
42      const mime = format === 'jpg' ? 'image/jpeg' : 'image
           /png';
43      const imgData = mapCanvas.toDataURL(mime);
44
45      if (format === 'pdf') {
46        const { jsPDF } = window.jspdf;
47        const pdf = new jsPDF({
48          orientation: width > height ? 'landscape' : '
              portrait',
49          unit: 'px',
50          format: [width, height]
51        });
52        pdf.addImage(imgData, 'PNG', 0, 0, width, height);
53        pdf.save('mapa_selecionado.pdf');
54      } else {
55        const link = document.createElement('a');
```

```
56          link.download = `mapa_selecionado.${format}`;
57          link.href = imgData;
58          link.click();
59        }
60      }, 500);
61    }
```

Fonte: Adaptado de (ASRSQ, 2025). Repositório Sobral em Mapas.

Código-fonte 8 – Integração do Laravel (2026) com o *GeoServer (2026)*

```
1   class GeoServerProxyController extends Controller
2   {
3     protected $geoServerService;
4
5     public function __construct(GeoServerService
          $geoServerService)
6     {
7       $this->geoServerService = $geoServerService;
8     }
9
10    public function proxyWms(Request $request)
11    {
12      // Pega todos os par metros da requisi  o
13      $params = $request->all();
14
15      // Faz a requisi  o ao GeoServer passando os
            par metros
16      $response = $this->geoServerService->getWmsLayer(
            $params);
17
18      // Verifica se a requisi  o foi bem-sucedida
19      if ($response->successful()) {
```

```
20          // Verifica se o corpo da resposta n o est  vazio
                ou corrompido
21          if (!empty($response->body())) {
22            return response($response->body(), 200)
23            ->header('Content-Type', 'image/png');  //
                 Retorna a imagem PNG
24          } else {
25            // Retorna uma mensagem de erro caso o corpo
                 esteja vazio (indica problema na camada)
26            return response()->json(['message' => 'Problema
                 ao carregar a camada. Verifique no GeoServer.'
                 ], 500);
27          }
28        } else {
29          // Retorna uma mensagem de erro caso a requisi  o
                falhe
30          return response()->json(['message' => 'Erro ao
               carregar a camada WMS do GeoServer.'], 500);
31        }
32      }
33
34      public function  getLegendGraphic(Request $request)
35      {
36        $layer = $request->input('layer');
37
38        $response = $this->geoServerService->getLegendGraphic
             ($layer);
39        if ($response->successful()) {
40          return response($response->body())
41          ->header('Content-Type', 'image/png')
42          ->header('Cache-Control', 'public, max-age=3600');
               // Cache por 1 hora;
```

```
43          }
44
45          return response()->json(['error' => 'Failed to
                retrieve legend graphic'], 500);
46      }
47  }
```

Fonte: Adaptado de (ASRSQ, 2025). Repositório Sobral em Mapas.

APÊNDICE B – QUESTIONÁRIO UTILIZADO PARA AVALIAÇÃO DA APLICAÇÃO

Questão 1. Qual é o seu nível de familiaridade com mapas digitais e sistemas SIG(Sistema de Informação Geográfica)?

(a) Nenhuma familiaridade

(b) Básica

(c) Intermediária

(d) Avançada

Questão 2. Com que frequência você utiliza a aplicação com mapas digitais como Google Maps ou Sobral em Mapas?

(a) Primeira vez

(b) Ocasionalmente

(c) Frequentemente

(d) Diariamente

Questão 3. Como você avalia a facilidade de navegação do mapa?

(a) Muito difícil

(b) Difícil

(c) Indiferente

(d) Fácil

(e) Muito fácil

Questão 4. As ferramentas de zoom, movimentação e seleção de camadas atenderam suas expectativas?

(a) Sim, plenamente

(b) Parcialmente

(c) Não atenderam

Questão 5. As ferramentas de zoom, movimentação e seleção de camadas atenderam suas expectativas?

(a) Sim, plenamente

(b) Parcialmente

(c) Não atenderam

Questão 6. Você encontrou dificuldades ao visualizar ou carregar camadas?

(a) Nunca

(b) Alguma vez

(c) Frequentemente

(d) Sempre

Questão 7. Avalie a velocidade de carregamento do mapa:

(a) Muito lenta

(b) Lenta

(c) Indiferente

(d) Rápida

(e) Muito rápida

Questão 8. O mapa atendeu suas necessidades de interatividade (ex.: clique em pontos, legendas, sobreposição de camadas)?

(a) Sim

(b) Parcialmente

(c) Não

Questão 9. O design e a clareza dos elementos do mapa (ícones, cores, legendas) foram satisfatórios?

(a) Muito ruim

(b) Ruim

(c) Indiferente

(d) Bom

(e) Excelente

Questão 10. Em sua opinião, quais funcionalidades poderiam ser adicionadas ou melhoradas?

Questão 11. Você recomendaria a aplicação Sobral em Mapas para outras pessoas?

(a) Sim

(b) Talvez

(c) Não