Roadmap - Projeto Antártica WebMapa

Sistema Interativo de Visualização de Dados Geoespaciais da Baía do Almirantado

> **Versão:** 1.0
> **Última atualização:** Outubro 2025
> **Objetivo:** Desenvolver plataforma web completa para visualização, análise e disseminação de dados científicos georreferenciados da Antártica

---

Estado Atual do Projeto

Implementado

- [x] Aplicação Next.js 15 com TypeScript
- [x] Mapa interativo usando Leaflet 1.9.4
- [x] Carregamento de 2.085 fotografias georreferenciadas via CSV
- [x] Sistema de clustering de marcadores (Marker Cluster)
- [x] Popups informativos com preview de imagens do Google Drive
- [x] Interface com painéis glass-morphism para: Layers, Search, Export, Info
- [x] Marcadores customizados com ícones de câmera
- [x] Múltiplas camadas base (OSM, Satellite, Terrain, CartoDB)
- [x] Controles de zoom e navegação
- [x] Design responsivo moderno

Dados Disponíveis

- **2.085 fotografias** georreferenciadas
- **Metadados:** PATH, NÚMERO, URL, ANO, TIPO, REC_TIME, DATA, LATITUDE, LONGITUDE, ELEVATION, OBSERVADOR
- **Formatos:** Detalhe, Paisagem
- **Período:** 2023
- **Observadores:** CARINA PETSCH e equipe
- **Área:** Baía do Almirantado, Ilha Rei George

Visão Geral das Fases

```
FASE 1 (MVP+)          → FASE 2 (Backend)      → FASE 3 (Features)    → FASE 4 (Análise)     → FASE 5 (Ciência Aberta)
   Funcionalidades        Banco de Dados         Camadas Avançadas      Análise Espacial       Publicação & API
   Essenciais             Geoespacial            & Exportação           & Visualizações        & Colaboração
```

🚀 FASE 1: Funcionalidades Essenciais & UI/UX

Objetivo: Completar MVP com todas as funcionalidades básicas operacionais

1.1 Sistema de Busca Completo

Prioridade: 🟠 ALTA

Entregáveis:

- [ ] **Busca por texto livre**
  - Buscar em: Observador, Tipo, Data, Número da foto
  - Autocompletar com sugestões
  - Highlight de resultados no mapa
- [ ] **Filtros avançados**
  - Filtro por data (range picker)
  - Filtro por observador (dropdown multi-select)
  - Filtro por tipo de registro (Detalhe, Paisagem)
  - Filtro por ano
  - Filtro por elevação (slider com range)
- [ ] **Resultados de busca**
  - Lista lateral com thumbnails
  - Paginação de resultados
  - Contador de resultados encontrados
  - Botão "Limpar filtros"
  - Zoom automático para resultado selecionado

Tecnologias:

- `react-select` para multi-select
- `react-datepicker` para seleção de datas
- `fuse.js` para busca fuzzy

1.2 Sistema de Exportação de Dados

Prioridade: 🟠 ALTA

Entregáveis:

- [ ] **Exportação CSV**
  - Exportar dados filtrados
  - Incluir todos os metadados
  - Encoding UTF-8 com BOM para Excel

- ☐ **Exportação XLSX**
  - Múltiplas planilhas
  - Formatação condicional
  - Biblioteca: `xlsx` (SheetJS)
- ☐ **Exportação PDF**
  - Tabela de dados selecionados
- ☐ **Exportação GeoJSON**
  - Para uso em outros softwares GIS
  - Incluir propriedades completas
  - Validação de geometria

Funcionalidades:

- Exportar apenas dados visíveis/filtrados
- Exportar todos os dados
- Preview antes de exportar

Tecnologias:

- `xlsx` (SheetJS)
- `jspdf` + `jspdf-autotable`
- `file-saver`
- `leaflet-draw` para seleção de área

1.3 Base Cartográfica Antártica

Prioridade: MÉDIA-ALTA

- ☐ **Camadas temáticas**
  - Elevação/Topografia
  - Geleiras
  - Estações de pesquisa

Fontes de Dados que podem ser utilizados:

- Quantarctica
- LIMA - USGS
- REMA - PGC
- Antarctic Digital Database

Tecnologias:

- TileServer GL
- GDAL para processamento de rasters

- QGIS para preparação de dados
- MBTiles para armazenamento

1.4 Melhorias de UI/UX

Prioridade: 🟡 MÉDIA

Entregáveis:

- ☐ **Navegação aprimorada**
  - Minimap no canto (overview)
  - Escala gráfica
  - Coordenadas do cursor
- ☐ **Tooltips informativos**
- ☐ **Performance**
  - Loading states melhores
- ☐ **Acessibilidade**
  - ARIA labels
  - Navegação por teclado
  - Alto contraste opcional
- ☐ **Responsividade mobile**
  - Menu hamburguer
  - Touch gestures otimizados
  - Layout adaptativo

Tecnologias:

- `intro.js` para tour guiado
- `leaflet-minimap`
- `leaflet-scale`
- `leaflet-coordinates`

💾 FASE 2: Backend & Banco de Dados Geoespacial

Objetivo: Migrar para arquitetura escalável com banco de dados

2.1 Infraestrutura de Backend

Prioridade: 🔴 ALTA

Entregáveis:

- [ ] **API RESTful**
  - Node.js + Express ou NestJS
  - Autenticação JWT (opcional para admin)
  - Rate limiting
  - CORS configurado
  - Documentação OpenAPI/Swagger
- [ ] **Endpoints principais**

```
GET    /api/photos              - Listar fotos (paginado + filtros)
GET    /api/photos/:id          - Detalhes de uma foto
GET    /api/photos/search       - Busca avançada
GET    /api/photos/bbox         - Fotos em bounding box
POST   /api/photos              - Upload nova foto (admin)
PUT    /api/photos/:id          - Atualizar foto (admin)
DELETE /api/photos/:id          - Deletar foto (admin)
GET    /api/stats               - Estatísticas gerais
GET    /api/export              - Exportar dados
```

- [ ] **Upload de arquivos**
  - Multer para uploads
  - Validação de imagens
  - Processamento de EXIF
  - Thumbnails automáticos (Sharp)
  - Armazenamento em cloud (S3/Cloud Storage)

Tecnologias:

- Node.js + Express ou NestJS
- TypeScript
- Multer + Sharp
- AWS S3 / Google Cloud Storage
- Swagger UI

2.2 Banco de Dados PostGIS (3-4 semanas)

Prioridade: 🟠 ALTA

Entregáveis:

- [ ] **Setup PostgreSQL + PostGIS**
  - Docker Compose para desenvolvimento
  - Configurações otimizadas para dados geoespaciais
  - Backups automáticos
  - Replicação (produção)

- [ ] **Modelo de dados**

```sql
-- Tabela principal de fotografias
CREATE TABLE photos (
  id SERIAL PRIMARY KEY,
  number VARCHAR(50) UNIQUE NOT NULL,
  file_path TEXT,
  google_drive_url TEXT,
  year INTEGER,
  type VARCHAR(50),
  rec_time TIMESTAMP WITH TIME ZONE,
  date DATE,
  location GEOGRAPHY(POINT, 4326), -- PostGIS
  elevation NUMERIC(10, 6),
  observer_id INTEGER REFERENCES observers(id),
  metadata JSONB, -- Metadados flexíveis
  thumbnail_url TEXT,
  created_at TIMESTAMP DEFAULT NOW(),
  updated_at TIMESTAMP DEFAULT NOW()
);

-- Índices espaciais
CREATE INDEX idx_photos_location ON photos USING GIST(location);
CREATE INDEX idx_photos_date ON photos(date);
CREATE INDEX idx_photos_observer ON photos(observer_id);
CREATE INDEX idx_photos_metadata ON photos USING GIN(metadata);

-- Tabela de observadores
CREATE TABLE observers (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255),
  institution VARCHAR(255),
  created_at TIMESTAMP DEFAULT NOW()
);

-- Tabela de camadas/layers
CREATE TABLE layers (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  description TEXT,
  type VARCHAR(50), -- raster, vector
  url TEXT,
  metadata JSONB,
  active BOOLEAN DEFAULT true
);

-- Tabela de áreas de interesse
CREATE TABLE areas_of_interest (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255),
  description TEXT,
  geometry GEOGRAPHY(POLYGON, 4326),
```

```
  created_at TIMESTAMP DEFAULT NOW()
);
```

- [ ] **Migração de dados**
  - Script de importação do CSV para PostgreSQL
  - Validação de coordenadas
  - Limpeza de dados duplicados
  - Log de erros de importação
- [ ] **Queries geoespaciais**

```
-- Buscar fotos em raio
SELECT * FROM photos
WHERE ST_DWithin(location, ST_MakePoint(-58.4, -62.1)::geography, 1000);

-- Buscar fotos em polígono
SELECT * FROM photos
WHERE ST_Within(location::geometry, ST_GeomFromGeoJSON('...'));

-- Heatmap de densidade
SELECT ST_AsGeoJSON(ST_SnapToGrid(location::geometry, 0.01)) as cell,
       COUNT(*) as count
FROM photos
GROUP BY cell;
```

Tecnologias:

- PostgreSQL 15+
- PostGIS 3.4+
- pg-promise (Node.js client)
- Docker + Docker Compose
- Adminer ou pgAdmin

🎨 FASE 3: Features Avançadas & Visualizações

Objetivo: Adicionar funcionalidades avançadas de análise visual

3.1 Sistema de Camadas Múltiplas

Prioridade: 🟠 MÉDIA-ALTA

Entregáveis:

- [ ] **Gerenciador de camadas**
  - Adicionar/remover camadas

- ☐ **Camadas vetoriais**
- ☐ **Camadas raster**
- ☐ **Sobreposições (overlays)**
  - Heatmap de densidade de fotos
  - Tracks GPS
- ☐ **Timeline temporal**
  - Slider temporal para filtrar por data
  - Animação de mudanças temporais

Tecnologias:

- `leaflet-timeline`
- `leaflet-side-by-side`
- `turf.js` para análise espacial
- `d3.js` para visualizações

3.2 Galeria & Visualização de Fotos

Prioridade: 🟡 MÉDIA

Entregáveis:

- ☐ **Galeria de fotos**
  - Grid view com lazy loading
  - Lightbox para visualização ampliada
  - Navegação entre fotos (prev/next)
  - Zoom e pan na imagem
  - Download de alta resolução
- ☐ **Metadata viewer**
  - EXIF completo
  - Localização no mapa
  - Fotos relacionadas (nearby)
  - Timeline de fotos do observador
- ☐ **Integração com Google Drive**
  - Login OAuth (opcional)
  - Upload direto para Drive
  - Sincronização automática

Tecnologias:

- `react-photo-view` ou `yet-another-react-lightbox`

- `react-image-gallery`
- `exifr` para leitura de EXIF
- Google Drive API

3.3 Dashboard & Estatísticas

Prioridade: 🟡 MÉDIA

Entregáveis:

- ☐ **Dashboard administrativo**
  - Total de fotos
  - Fotos por observador
  - Fotos por tipo
  - Fotos por ano
  - Cobertura espacial
- ☐ **Gráficos interativos**
- ☐ **Relatórios automáticos**
  - Relatório mensal de coletas
  - Relatório anual
  - Estatísticas por expedição

Tecnologias:

- `recharts` ou `chart.js`
- `react-grid-layout` para dashboard customizável
- `date-fns` para manipulação de datas

---

4 Integração com QGIS/ArcGIS

Prioridade: 🟡 BAIXA-MÉDIA

Entregáveis:

- ☐ **Plugin QGIS**
  - Baixar dados diretamente do QGIS
  - Upload de dados para o sistema
  - Sincronização de camadas

Tecnologias:

- GeoServer ou MapServer
- PyQGIS para plugin

🗄️ Stack Tecnológica Recomendada

Frontend

- **Framework:** Next.js 15 (App Router)
- **Linguagem:** TypeScript
- **Mapas:** Leaflet / CesiumJS (3D)
- **UI:** Tailwind CSS + shadcn/ui
- **State:** React Context API / Zustand
- **Forms:** React Hook Form + Zod
- **Data fetching:** React Query

Backend

- **Runtime:** Node.js 20+ / Bun
- **Framework:** NestJS (recomendado) ou Express
- **Linguagem:** TypeScript
- **API:** REST + GraphQL (opcional)
- **Validação:** Zod / class-validator

Banco de Dados

- **Principal:** PostgreSQL 15+ com PostGIS 3.4+
- **Cache:** Redis
- **Search:** PostgreSQL Full-Text Search ou Meilisearch
- **ORM:** Prisma (recomendado) ou TypeORM

📖 Recursos & Referências

Documentação Técnica

- Leaflet Documentation
- PostGIS Documentation
- Next.js Documentation
- Turf.js Guide

Última atualização: Outubro 2025
Versão: 1.0
Contato: [Adicionar informações de contato do projeto]

Este roadmap é um documento vivo e será atualizado conforme o projeto evolui.