Product Backlog – Aplicação Web de Dados Limnológicos

Épico 1 – Prototipação e Design

História 1 – Prototipação da Interface (Figma)

Como time de desenvolvimento
Quero prototipar a aplicação no Figma
Para validar a interface antes do desenvolvimento

Tarefas:

- Criar wireframes das telas principais (painel, tabela, mapa, gráficos, exportação).
- Definir paleta de cores alinhada ao INPE.
- Validar protótipo com cliente/professor

Critérios de Aceite:

- Protótipo aprovado antes do início do front-end

Prioridade: Alta

História 2 – Identidade Visual e Responsividade

Como usuário
Quero que a interface siga padrões visuais claros e seja responsiva
Para garantir boa usabilidade em diferentes dispositivos

Tarefas:

- Aplicar identidade visual (cores, logos, tipografia).
- Garantir responsividade para desktop, tablet e mobile

Critérios de Aceite:

- Interface clara, navegável e adaptável a diferentes telas

Prioridade: Alta

Épico 2 – Front-End (React + TypeScript)

História 3 – Estrutura Inicial do Front-End

Como desenvolvedor
Quero configurar o projeto em React + TS
Para ter a base pronta

Tarefas:

- Criar projeto React + TypeScript
- Configurar estrutura de pastas (pages, components, services, styles)
- Configurar estilos

Critérios de Aceite:

- Aplicação inicial roda em localhost sem erros

Prioridade: Alta

História 4 – Tela de Visualização em Tabela (RF02)

Como usuário
Quero visualizar os dados em tabelas
Para analisar informações de forma estruturada

Tarefas:

- Criar componente de tabela interativa
- Implementar filtros por instituição, reservatório, período
- Conectar tabela ao back-end
- Tratar dados incompletos ou inválidos para não quebrar a tabela

Critérios de Aceite:

- Dados exibidos corretamente, filtros funcionando
- Sistema lida com valores ausentes ou incorretos sem travar a tabela

Prioridade: Alta

História 5 – Exportação CSV (RF03)

Como usuário
Quero exportar dados em CSV
Para utilizá-los em análises externas

Tarefas:

- Criar botão de exportação
- Integrar botão ao endpoint do back-end

- Garantir que CSV ignore ou indique dados inconsistentes

Critérios de Aceite:

- CSV exportado contém exatamente os dados filtrados
- Dados inconsistentes aparecem marcados ou são reportados

Prioridade: Média

História 6 – Mapa Interativo com Pontos/Polígonos (RF04)

Como usuário
Quero visualizar os pontos ou áreas de coleta no mapa
Para identificar a localização geográfica dos dados

Tarefas:

- Integrar biblioteca de mapas (Leaflet/Mapbox)
- Exibir pontos e polígonos com base em coordenadas geoespaciais do PostgreSQL/PostGIS
- Permitir clique para ver detalhes das medições
- Tratar dados faltantes ou inválidos para não quebrar o mapa

Critérios de Aceite:

- Pontos e polígonos aparecem corretamente
- Clicando, detalhes da coleta são exibidos
- Aplicação não quebra mesmo com dados incompletos

Prioridade: Alta

História 7 – Gráficos de Séries Temporais (RF05)

Como pesquisador
Quero visualizar gráficos de séries temporais
Para acompanhar evolução dos parâmetros ao longo do tempo

Tarefas:

- Criar componente de gráfico (Chart.js/Recharts)
- Conectar ao endpoint de séries temporais
- Ajustar escalas automaticamente
- Tratar dados ausentes ou inválidos

Critérios de Aceite:

- Gráficos exibem dados corretos em escala temporal
- Dados inconsistentes são indicados ou ignorados sem travar o gráfico

Prioridade: Alta

História 8 – Usabilidade e Desempenho (RNF01, RNF02)

Como usuário
Quero que a aplicação seja rápida e intuitiva
Para acessar os dados de forma simples e eficiente

Tarefas:

- Implementar navegação clara
- Otimizar carregamento de tabelas, gráficos e mapas
- Garantir robustez ao lidar com dados inconsistentes

Critérios de Aceite:

- Carregamento em até 3 segundos para datasets médios
- Navegação fácil para usuários sem conhecimento técnico
- Sistema não trava com dados incorretos

Prioridade: Alta

História 9 – Informação sobre os Dados

Como usuário
Quero uma explicação clara sobre a origem e significado dos dados
Para entender melhor o que estou visualizando

Tarefas:

- Criar seção “Sobre os Dados” no front-end
- Adicionar tooltips explicativos em tabelas e gráficos
- Garantir que o conteúdo seja claro e acessível

Critérios de Aceite:

- Usuário consegue entender de onde vêm os dados e o que cada parâmetro representa
- Informações acessíveis tanto em desktop quanto em mobile

Prioridade: Média

Épico 3 – Back-End Web (Node.js + TypeScript)

História 10 – Estrutura Inicial do Back-End

Como desenvolvedor
Quero configurar Node.js + TypeScript
Para disponibilizar dados para o front-end

Tarefas:

- Criar projeto base (Express + TypeScript)
- Configurar estrutura (controllers, services, routes)
- Implementar middleware básico de erros

Critérios de Aceite:

- Servidor sobe com endpoint inicial /health

Prioridade: Alta

História 11 – Endpoints Read de Entidades

Como desenvolvedor
Quero criar endpoints Read
Para gerenciar instituições, reservatórios, parâmetros e medições

Tarefas:

- Rotas para leitura para cada entidade (GET)
- Controllers e validações implementadas
- Testes via Postman
- Garantir que endpoints lidem com dados inconsistentes sem travar

Critérios de Aceite:

- Read completo funcionando
- Dados inconsistentes tratados e reportados no log

Prioridade: Alta

História 12 – Endpoint de Exportação CSV

Como usuário
Quero exportar dados via API
Para baixar e analisar externamente

Tarefas:

- Implementar rota /export/csv
- Permitir filtros (instituição, reservatório, período)
- Lidar com registros incompletos ou inválidos

Critérios de Aceite:

- Download gera CSV correto com filtros aplicados
- Dados inconsistentes marcados ou ignorados

Prioridade: Média

História 13 – Endpoint de Séries Temporais

Como usuário
Quero acessar dados de séries temporais
Para alimentar gráficos no front-end

Tarefas:

- Implementar rota /series-temporais
- Tratar intervalos de tempo e parâmetros
- Lidar com dados incompletos sem quebrar a resposta

Critérios de Aceite:

- Endpoint retorna JSON pronto para gráficos
- Aplicação front-end não trava com dados inválidos

Prioridade: Alta

Épico 4 – Deploy e Infraestrutura

História 14 – Containerização (Docker)

Como desenvolvedor
Quero usar containers independentes
Para rodar front-end, back-end e banco isoladamente

Tarefas:

- Criar Dockerfile para cada serviço
- Configurar docker-compose para integração
- Garantir rede interna segura entre containers

Critérios de Aceite:

- Containers sobem corretamente e comunicam-se
- Ambiente reproduzível em qualquer máquina

Prioridade: Alta

Épico 5 – Servidor de Ingestão de Dados (Separado)

História 15 – Conexão com PostgreSQL e Modelagem de Tabelas (RP01)

Como desenvolvedor
Quero modelar tabelas no PostgreSQL + PostGIS
Para armazenar e organizar dados corretamente

Tarefas:

- Modelar entidades: instituição, reservatório, parâmetros, medições, coordenadas geoespaciais
- Configurar ORM (TypeORM ou Prisma)
- Testar consultas simples

Critérios de Aceite:

- Banco criado e conectado, consultas retornam dados corretos

Prioridade: Alta

História 16 – Upload de Arquivos CSV

Como servidor de ingestão
Quero receber CSVs e validar dados
Para alimentar o banco sem quebrar o sistema

Tarefas:

- Criar endpoint de upload
- Validar arquivos (formatos, campos obrigatórios)
- Registrar inconsistências em logs

Critérios de Aceite:

- Dados válidos inseridos no banco
- Dados inválidos geram logs claros
- Front-end não é afetado

Prioridade: Alta

História 17 – Validação de Dados Importados

Como servidor de ingestão
Quero verificar consistência dos dados
Para garantir robustez do sistema

Tarefas:

- Scripts de validação de duplicidade, campos ausentes ou inválidos
- Relatórios automáticos de inconsistências

Critérios de Aceite:

- Logs claros e acessíveis
- Sistema não trava com dados incorretos
- Relatórios podem ser enviados ao professor para correção

Prioridade: Alta

História 18 – Ingestão Automática Agendada (verificar viabilidade)

Como servidor de ingestão
Quero processar dados automaticamente
Para manter banco atualizado com novos dados do SIMA

Tarefas:

- Pipeline automático de ingestão (ex.: cron job ou Node schedule)
- Tratamento de inconsistências sem interromper o banco
- Logs e relatórios gerados periodicamente

Critérios de Aceite:

- Pipeline funciona sem intervenção manual
- Dados inconsistentes são reportados sem quebrar a aplicação
- Front-end sempre recebe dados válidos

Prioridade: Alta