

Integração DASA - DSS

(parte 1)

DASA UTM

□□ Descrição

Fornecer ao DASA capacidade de acessar entidades UTM através de um serviço intermediário entre o mapa e o ECO-UTM, o que torna o DASA capaz de mostrar as regiões de **intenção operacional (OIRs) e restrições (Constraints)** com **filtros avançados** e **detalhes via popup**.

O serviço deve fornecer **endpoints** que interagem com o **DSS** para obter e processar as informações do ECO-UTM.

□ Estimativas

- Desenvolvimento do backend: 9-11 dias
 - Integração com DSS: 9-11 dias
 - Testes unitários e de integração: 7-9 dias
 - Revisão e ajustes após feedback do frontend: 7-9 dias
 - **Total estimado: ~5 a 6 semanas**
-

□□ Dependências

- **Time de frontend:** para validar a estrutura dos endpoints e a forma como os dados serão consumidos.
 - **Time de backend:** para fornecer suporte ao acesso dos serviços ECO-UTM.
-

□□ Definições técnicas

- Protocolo de comunicação com DSS: **REST/HTTP**
- Formato de dados geoespacial: **GeoJSON**
- Estratégia de atualização de dados: **polling**
- Autenticação: **JWT**

- Desempenho: implementar **cache e paginação** (REDIS) nos endpoints para evitar sobrecarga ao DSS.
-

📋 Tarefas

1. Configurar ambiente

- Criar projeto FastAPI
- Configurar ambiente de desenvolvimento e dependências
- Implementar estrutura básica do projeto
 - `/api/regions/oirs` → Retorna OIRs com filtros aplicáveis
 - `/api/regions/constraints` → Retorna Constraints com filtros aplicáveis
 - `/api/regions/details/{id}` → Retorna detalhes de um OIR ou Constraint específico

2. Modelar dados

- Criar modelos **Pydantic** para OIRs e Constraints
- Adaptar a resposta do DSS para os modelos internos

3. Desenvolver interface com o DSS

- Implementar cliente para comunicação com o DSS
- Desenvolver métodos para busca de OIRs e Constraints
- Tratar erros de comunicação com o DSS

4. Implementar os filtros nos endpoints

- **Horário/Duração**
- **Provedor responsável**
- **Altitude**
- **Tipo de operação (VLOS, EVLOS, BVLOS)**

5. Implementar endpoint de detalhes

- **OIR:** nome do provedor, horário, coordenadas, altitude, contato de emergência
- **Constraint:** nome do responsável, tipo (FRZ, temporária, altitude), duração, altitude

6. Implementar testes unitários e de integração

- Testar filtros e respostas
- Simular falhas no DSS e validar comportamento

7. Documentar

- Documentar endpoints usando **FastAPI + Swagger**
- Compartilhar exemplos de uso para o frontend

☐ Critérios de Aceitação

1. Todos os endpoints devem estar documentados via Swagger
 2. O sistema deve retornar dados geoespaciais em formato compatível exigido pela equipe de frontend
 3. O sistema deve lidar adequadamente com falhas de comunicação com o DSS
 4. Os filtros devem funcionar em combinação (AND lógico)
 5. Os popups devem conter todas as informações especificadas na descrição
 6. Garantir que, caso o DSS esteja temporariamente indisponível, o sistema não quebre e forneça um fallback adequado.
-

☐☐ Nível de prioridade

- Sim
-

Revision #4

Created 18 March 2025 13:54:07 by João Oliveira

Updated 18 March 2025 17:26:25 by João Oliveira