1. Requisitos Funcionais

1.1. Módulo: Localização

- O sistema deve permitir consultar a localização de exames, setores, blocos, vagas de emprego ou notícias por meio de parâmetros na URL.
- A resposta deve incluir campos como existe, exame, setor, bloco, coordenada e referencias.
- O sistema deve retornar erro ou resposta inválida se nenhuma query for fornecida.
- Administradores devem poder criar, editar ou excluir setores via endpoints POST, PATCH e DELETE em /setores.

1.2. Módulo: Eventos

- O sistema deve listar todos os eventos (futuros, em andamento e encerrados) via GET /eventos.
- Deve permitir filtrar eventos por:
  - status (futuros, em_andamento, encerrado);
  - dias (eventos nos próximos N dias).
- Deve permitir consultar detalhes de um evento específico via GET /eventos/:id.
- Administradores devem poder gerenciar eventos (criar, atualizar, excluir) via POST, PATCH e DELETE em /eventos.

1.3. Módulo: Notícias

- O sistema deve retornar notícias dos últimos 30 dias por padrão via GET /noticias.
- Deve permitir filtrar notícias por:
  - recentes (últimos N dias);
  - bloco, setor, exame (notícias relacionadas a esses contextos).
- Deve permitir consultar uma notícia específica via GET /noticias/:id.

- Administradores devem poder publicar, editar ou excluir notícias via POST, PATCH e DELETE em /noticias.

1.4. Módulo: Sobre a Fundação

- O sistema deve retornar informações institucionais (missão, visão, valores, compliance) via GET /sobre.
- Administradores devem poder atualizar essas informações via PATCH /sobre.

1.5. Módulo: Vagas de Emprego

- O sistema deve listar vagas ativas por padrão via GET /vagas.
- Deve permitir filtrar vagas por:
  - encerradas (exibir vagas finalizadas);
  - cargo;
  - contrato (ex: CLT);
  - data_publicacao (ordenar por data).
- Deve permitir consultar detalhes de uma vaga específica via GET /vagas/:id.
- Administradores devem poder gerenciar vagas via POST, PATCH e DELETE em /vagas.

🔹 2. Requisitos Não Funcionais

2.1. Usabilidade

- A API deve ser intuitiva e acessível até para usuários leigos.
- As respostas devem seguir estruturas padronizadas e consistentes entre endpoints.

2.4. Manutenibilidade

- A estrutura da API deve facilitar futuras atualizações ou inclusão de novos módulos.

2.5. Integração

- Os dados devem seguir formatos padronizados (JSON) para facilitar integração

com sites e apps.

- Dados devem sempre estar atualizados com as informações corretas na Fundação

---

- 3. Requisitos de Segurança e Controle de Acesso

- Endpoints com métodos POST, PATCH e DELETE devem ser restritos a usuários administradores autenticados.
- Endpoints com método GET devem ser públicos, sem necessidade de autenticação.
- A API deve validar parâmetros de entrada para evitar erros (ex: query vazia em /localizacao deve retornar erro).
- Imagens utilizadas como referência (ex: em localizações, notícias ou setores) não devem conter pessoas identificáveis; caso haja pessoas nas fotos, seus rostos devem ser previamente borrados ou anonimizados antes da publicação.

- 4. Requisitos de Dados

- Todos os campos de data devem seguir o formato ISO 8601 (ex: "2024-08-10T09:00:00").
- Ids devem ser unicos.
- Campos como referencias, requisitos, imagens e tags devem ser arrays de strings.