Documentação da Arquitetura de Software

NatureLens

Visão Geral

Este documento descreve a arquitetura de software do sistema **NatureLens** utilizando o **Modelo C4**. A arquitetura é apresentada em quatro níveis: **Contexto**, **Container**, **Componente** e **Código**.

---

1. Diagrama de Contexto

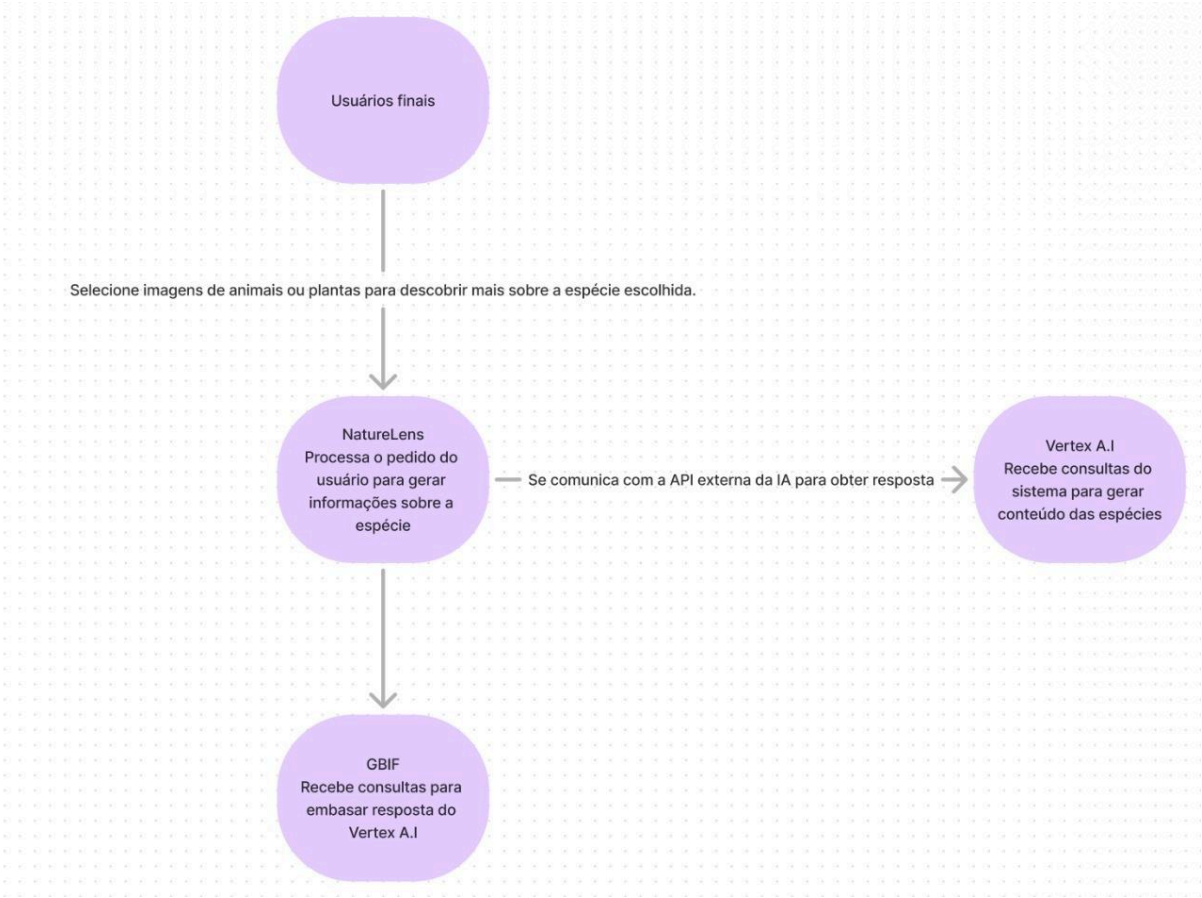
1.1. Descrição do Diagrama de Contexto

O diagrama de contexto oferece uma visão geral de alto nível do sistema **NatureLens** e suas interações com os atores externos.

- **Sistema:** NatureLens

- **Atores Externos:**
  - **Usuários Finais:** Entusiastas da natureza, estudantes, educadores, turistas.
  - **APIs Externas:**
    - **Google OAuth:** Autenticação de usuários.
    - **Vertex AI (Google Vision Pro):** Identificação de espécies em imagens.
    - **GBIF API:** Informações detalhadas sobre espécies.

1.2. Diagrama de Contexto



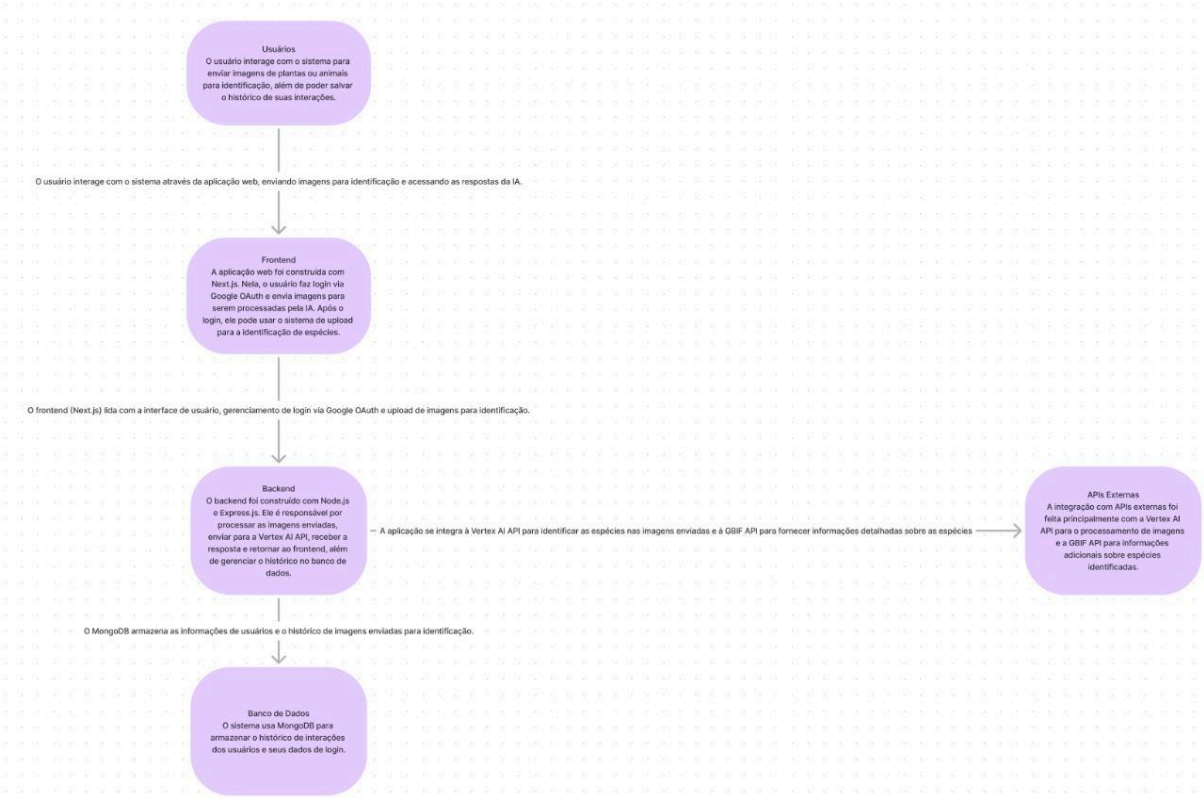
2. Diagrama de Container

2.1. Descrição do Diagrama de Container

O diagrama de container descreve os principais containers de software que compõem o sistema e como eles interagem entre si.

- **Containers Incluídos:**
  - **Frontend (Aplicação Web)**
  - **Backend (API Server)**
  - **Banco de Dados (MongoDB)**
  - **APIs Externas**

2.2. Diagrama de Container



2.3. Descrição dos Containers

- **Frontend (Aplicação Web em Next.js)**
  - **Tecnologias:** Next.js, React, Tailwind CSS
  - **Responsabilidade:** Interface do usuário, permitindo login, upload de imagens e exibição de resultados.
  - **Interações:** Envia requisições ao Backend e redireciona para o Google OAuth.

- **Backend (API Server em Node.js com Express.js)**
  - **Tecnologias:** Node.js, Express.js, Multer, Mongoose

    - **Responsabilidade:** Processamento de imagens, integração com APIs externas, gerenciamento de lógica de negócio.
    - **Interações:** Recebe requisições do Frontend e comunica-se com o MongoDB e APIs externas.

- **Banco de Dados (MongoDB)**
    - **Tecnologias:** MongoDB, Mongoose
    - **Responsabilidade:** Armazenamento de dados de usuários, históricos e metadados.
    - **Interações:** Consultado e atualizado pelo Backend.

- **APIs Externas:**
    - **Google OAuth:** Autenticação de usuários.
    - **Vertex AI (Google Vision Pro):** Processamento de imagens e identificação de espécies.
    - **GBIF API:** Fornecimento de informações detalhadas sobre espécies.

---

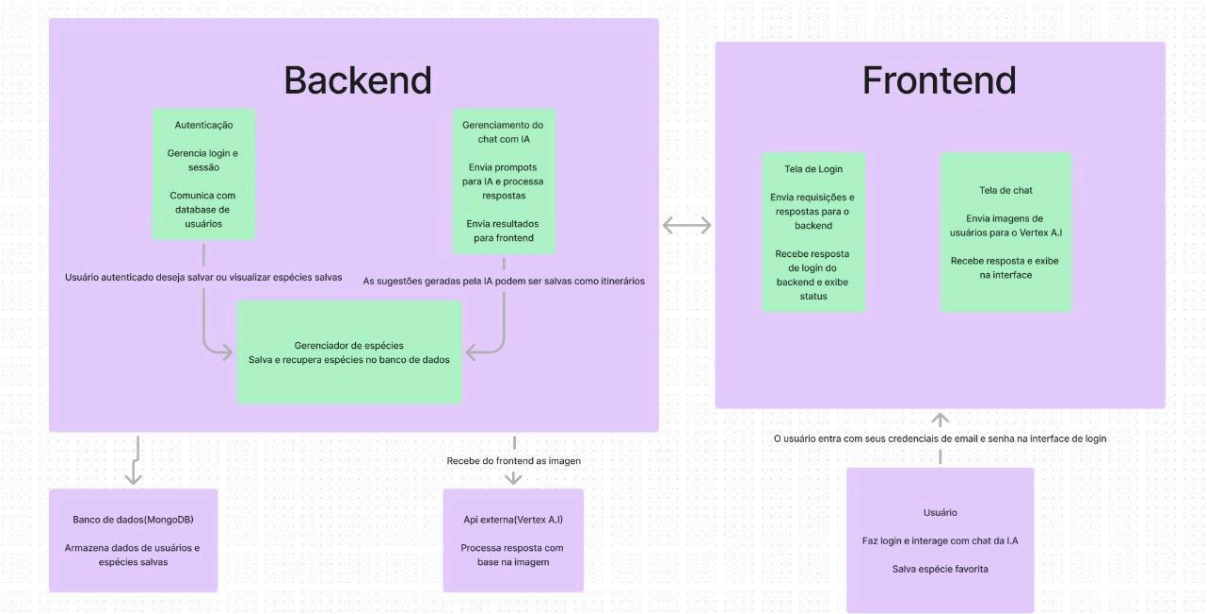
3. Diagrama de Componente

3.1. Descrição do Diagrama de Componente

O diagrama de componente detalha a arquitetura interna do Backend.

- **Container Focado:** Backend (API Server)

3.2. Diagrama de Componentes



3.3. Descrição dos Componentes

- **Authentication Controller**
    - **Responsabilidade:** Gerenciar processos de login e logout.
    - **Interações:** Comunica-se com o Google OAuth e utiliza Middlewares para validar tokens.

- **Image Upload Controller**
    - **Responsabilidade:** Gerenciar o upload de imagens.
    - **Tecnologias:** Multer para upload.
    - **Interações:** Recebe imagens do Frontend e chama o Image Processing Service.

- **Image Processing Service**
  - **Responsabilidade:** Enviar imagens para o Vertex AI e processar resultados.
  - **Interações:** Comunica-se com a API do Vertex AI e repassa dados para o Species Information Service.

- **Species Information Service**
  - **Responsabilidade:** Obter informações adicionais sobre a espécie identificada.
  - **Interações:** Consulta a GBIF API e retorna dados ao Controller.

- **User Model**
  - **Responsabilidade:** Estrutura de dados para usuários.
  - **Interações:** Realiza operações CRUD com o MongoDB.

- **History Model**
  - **Responsabilidade:** Registro de histórico de uploads e resultados.
  - **Interações:** Armazena dados no MongoDB.

- **Middlewares**
  - **CORS Middleware:** Configura políticas de CORS.
  - **Authentication Middleware:** Verifica a autenticidade dos tokens.
  - **Error Handling Middleware:** Captura e trata erros.

---

4. Decisões Arquiteturais

4.1. Decisões Importantes

- **Decisão:** Uso de Next.js para o Frontend.
    - **Descrição:** Escolha do Next.js para facilitar a construção de interfaces e roteamento.
    - **Justificativa:** Framework popular com excelente suporte e documentação.
    - **Impacto:** Desenvolvimento mais rápido e eficiente do frontend.

- **Decisão:** Implementação do Backend com Node.js e Express.js.
    - **Descrição:** Uso de tecnologias JavaScript no servidor.
    - **Justificativa:** Consistência de linguagem entre frontend e backend; grande comunidade de suporte.
    - **Impacto:** Facilita a integração e manutenção do código.

- **Decisão:** Armazenar dados no MongoDB.
    - **Descrição:** Escolha de um banco de dados NoSQL flexível.
    - **Justificativa:** Facilidade de escalabilidade e modelagem de dados não estruturados.
    - **Impacto:** Agilidade no desenvolvimento e capacidade de lidar com grandes volumes de dados.

- **Decisão:** Utilizar APIs externas para processamento e dados.
    - **Descrição:** Integração com o Vertex AI e GBIF API.
    - **Justificativa:** Aproveitar soluções já existentes e confiáveis.
    - **Impacto:** Redução de tempo de desenvolvimento e aumento da precisão dos resultados.

---

5. Considerações Finais

5.1. Padrões e Práticas

- **Padrões:**
  - **Arquitetura em Camadas:** Separação clara entre frontend, backend e serviços externos.
  - **RESTful APIs:** Uso de princípios REST para comunicação entre frontend e backend.
  - **MVC (Model-View-Controller):** Aplicado no backend para organização do código.
- **Práticas:**
  - Controle de Versão com Git e GitHub.
  - Gerenciamento de Configurações Sensíveis com Dotenv.
  - Implementação de Testes Automatizados com Cypress.
  - Segurança na Autenticação com Google OAuth.

5.2. Próximos Passos

- **Melhoria na Gestão de Erros:** Implementação de logs detalhados e monitoramento.
- **Otimização de Performance:** Implementação de cache onde for necessário.
- **Escalabilidade:** Preparação da infraestrutura para suportar mais usuários.
- **Testes Automatizados no Backend:** Implementação contínua de testes.