Infinity API - Documentação

# 1. Visão Geral

A \*\*Infinity API\*\* conecta o frontend ao banco de dados de uma plataforma onde estudantes postam seus projetos e conhecimentos. Recrutadores podem visualizar esses projetos e entrar em contato com os alunos.  
  
\* \*\*Tecnologias\*\*: FastAPI, SQLAlchemy, Pydantic, SQLite  
\* \*\*Autenticação\*\*: JWT com refresh token  
\* \*\*Hospedagem\*\*: Disponível localmente via GitHub  
\* \*\*Usuários\*\*: Estudantes e Recrutadores

# 2. Autenticação

A autenticação é feita via JWT. Após o login, são retornados dois tokens:  
  
\* \*\*Access Token\*\*: válido por 30 minutos  
\* \*\*Refresh Token\*\*: válido por 7 dias  
  
🔐 Fluxo de Autenticação  
  
1. POST /auth/login → envia email e password → recebe access\_token e refresh\_token  
2. Para acessar rotas privadas, envie o access\_token no cabeçalho:  
Authorization: Bearer <access\_token>  
3. Quando o token expirar, use GET /auth/refresh para obter um novo access\_token.

# 3. Endpoints

### 🔓 Públicas  
  
POST /auth/sign-up: Cadastro de usuário  
Body: { "name": "string", "email": "string", "password": "string", "type": true, "course": "string", "cell": "string", "company": "string" }

Type – True == Estudante

Type – False == Recrutador

POST /auth/login: Login do usuário  
Body: { "email": "string", "password": "string" }  
  
POST /auth/login-form: Login via OAuth2PasswordRequestForm (Form Data: username, password)  
  
GET /auth/refresh: Gera novo access token usando o token atual (Cabeçalho: Authorization)  
  
GET /auth/users: Lista todos os usuários (Query Params: name string opcional)

# Privadas - Estudante

POST /project/register: Cadastra projeto  
Body: { "title": "string", "photo": "string", "description": "string", "body": "string", "tags": [1,2] }  
  
GET /project/list: Lista projetos do estudante  
  
PUT /project/edit?project\_id=<int>: Edita projeto  
Body: { "title": "string", "photo": "string", "description": "string", "body": "string", "tags": [1,2] }  
  
DELETE /project/delete?project\_id=<int>: Deleta projeto

# Privadas - Acesso geral

GET /project/list\_all\_projects: Lista todos os projetos  
  
GET /students/list: Lista todos os estudantes  
Retorno: [{ "name": "string", "course": "string", "phone": "string" }]

# Privadas - Tags

POST /tag/register: Cadastra tag { "name": "string" }  
GET /tag/list: Lista todas as tags [{ "id": 1, "name": "string" }]  
GET /tag/tags/search\_projects?tags\_id=1&tags\_id=2: Lista projetos por tags (Query Params tags\_id array de int)  
PUT /tag/edit/{tag\_id}: Edita nome da tag { "name": "string" }  
DELETE /tag/delete/{tag\_id}: Deleta tag pelo ID

# Privadas - Students CRUD

POST /students/register: Cadastra estudante { "name": "string", "cell": "string", "course": "string" }  
PUT /students/edit\_id?student\_id=<int>: Edita estudante pelo ID  
PUT /students/edit?student\_id=<int>: Edita estudante logado { "name": "string", "cell": "string", "course": "string" }

# 4. Modelos de Dados

Estudante: { "id": 1, "name": "string", "cell": "string", "course": "string" }  
Projeto: { "id": 1, "student\_id": 1, "title": "string", "photo": "string", "description": "string", "body": "string", "tags": [1,2] }  
Tag: { "id": 1, "name": "string" }

# 5. Permissões

| Rota | Estudante | Recrutador |  
|------------------------------|------------|------------|  
| /auth/\* | ✅ | ✅ |  
| /students/\* | ✅ (próprio)| ❌ |  
| /project/\* | ✅ (próprio)| ❌ |  
| /project/list\_all\_projects | ✅ | ✅ |  
| /tag/\* | ✅ | ✅ |

# 6. Exemplos de Uso

📬 Requisição com cURL  
curl -X POST http://localhost:8000/auth/login -H "Content-Type: application/json" -d '{"email": "string", "password": "string"}'  
  
📦 Postman  
- Criar uma coleção com autenticação JWT no cabeçalho  
- Rotas organizadas por tipo de usuário

# 7. Tecnologias Utilizadas

\* FastAPI  
\* SQLAlchemy  
\* Pydantic  
\* SQLite  
\* JWT