Documentação Técnica da Solução API Enercheck (.NET 8)

1. Propósito da API Enercheck

A **API Enercheck** é um serviço **RESTful** de gestão de dados (*Data Management*). Desenvolvida em **.NET 8**, ela é especificamente focada na gestão de **usuários e projetos** no setor de energia.

Funcionalidades Chave:

- **CRUD Padrão:** Gerenciamento completo (Cadastro, Consulta, Atualização e Exclusão) de planos e dados do cliente.
- **Análise Inteligente:** Possui um *endpoint* crucial para a submissão e processamento de plantas elétricas.
- **Mecanismo de IA:** Este *endpoint* utiliza um **mecanismo de Inteligência Artificial** (Gemini AI Service) para realizar a análise e validação técnica das plantas.
- **Segurança:** O controle de acesso e a segurança são garantidos pelo **ASP.NET Identity** e **JWT (JSON Web Tokens)**.

---

2. Especificação da API (Endpoints Principais)

1.1. Usuários

Método	Rota	Descrição	Parâmetros	Resposta (Modelo)	Status HTTP
GET	/api/Usuarios	Lista todos os usuários.	-	UsuarioDetalhesDto[]	200
GET	/api/Usuarios/me	Retorna os dados do usuário logado.	-	UsuarioDetalhesDto	200, 401
GET	/api/Usuarios/{id}	Busca usuário por ID.	id: string	Usuario	200, 404

Método	Rota	Descrição	Parâmetros	Resposta (Modelo)	Status HTTP
POST	/api/Usuarios/Cliente	Cadastro de novo usuário cliente.	RegisterDto	UserResponseDto	**201**, 400
POST	/api/Usuarios/Admin	Cadastro de novo usuário admin.	RegisterDto	UserResponseDto	**201**, 400
PUT	/api/Usuarios/{id}	Atualiza dados do usuário.	id: string, Usuario	-	204, 400
PUT	/api/Usuarios/add/plano	Vincula um plano ao usuário logado.	planoId: int	Dados do usuário/plano	200, 404
DELETE	/api/Usuarios/{id}	Remove usuário.	id: string	-	204, 404

1.2. Projetos

Método	Rota	Descrição	Parâmetros	Resposta (Modelo)	Status HTTP
GET	/api/Projetos	Lista todos os projetos.	-	Projeto[]	200
GET	/api/Projetos/{id}	Busca projeto por ID.	id: Guid	Projeto	200, 404
POST	/api/Projetos	Cria novo projeto.	ProjetoCreateDto	ProjetoResponseDto	**201**, 400
PUT	/api/Projetos/{id}	Atualiza projeto.	id: Guid, Projeto	-	204, 400
DELETE	/api/Projetos/{id}	Remove projeto.	id: Guid	-	204, 404
POST	/api/Projetos/projeto/{id}/analisar	**Analisa projeto via Gemini AI**.	id: Guid, IFormFile	Projeto	200, 500

1.3. Planos

Método	Rota	Descrição	Parâmetros	Resposta (Modelo)	Status HTTP
GET	/api/Planos	Lista todos os planos.	-	Plano[]	200
GET	/api/Planos/{id}	Busca plano por ID.	id: int	Plano	200, 404
POST	/api/Planos	Cria novo plano.	Plano	Plano	**201**, 400
PUT	/api/Planos/{id}	Atualiza plano.	id: int, Plano	-	204, 400
DELETE	/api/Planos/{id}	Remove plano.	id: int	-	204, 404

---

2. Modelos de Dados (Principais Entidades)

Estes são os principais modelos de dados utilizados na API:

- **Usuario**: Id, Email, NomeCompleto, NumeroCrea, Empresa, PlanoId, Plano, Projetos, UserReq.
- **Projeto**: ProjetoId, Nome, Descricao, UsuarioId, Usuario, dataInicio, Progresso, Status, Analise.
- **Plano**: PlanoId, Nome, Preco, QuantidadeReq, Ativo, QuantidadeUsers.

Exemplo de Cadastro de Cliente

Requisição: POST /api/Usuarios/Cliente

JSON

```
Content-Type: application/json
{
  "email": "usuario@exemplo.com",
  "senha": "SenhaSegura123!",
  "nomeCompleto": "João da Silva",
  "numeroCrea": "123456",
  "empresa": "Empresa X"
}
```

Resposta (Status 201 - Created):

JSON

```
{
  "id": "string",
  "email": "usuario@exemplo.com",
  "nomeCompleto": "João da Silva",
  "numeroCrea": "123456",
  "empresa": "Empresa X",
  "roles": ["Cliente"]
}
```

3.Fluxo de Autenticação/Autorização (JWT)

A segurança é gerenciada pelo **ASP.NET Identity** e **JWT**.

Passo	Ação	Detalhes
1. Registro	Cadastro de Usuário	POST /api/Usuarios/Cliente ou POST /api/Usuarios/Admin.
2. Login	Autenticação	Usuário faz login (Endpoint de login recomendado, mas não explícito).
3. Token	Recebimento do JWT	Após o login, um **Token JWT** é emitido.
4. Requisições	Envio do Token	O token deve ser enviado no cabeçalho Authorization: Bearer {token} para todos os *endpoints* protegidos.
5. Validação	Verificação	Um *Middleware* valida o token e verifica as *roles* (Admin, Cliente).

---

4. Visão Geral da Arquitetura

Diagrama de Camadas

A arquitetura segue um modelo de camadas com foco em separação de responsabilidades:

Camada	Função Principal	Tecnologias Chave
Controllers	Recebem requisições HTTP, validam dados e chamam serviços.	ASP.NET Core Web API
Services	Contêm a lógica de negócio e integram com APIs externas (ex: GeminiService).	GeminiService
Data/DbContext	Gerencia o acesso ao banco de dados relacional (ORM).	Entity Framework Core
Identity	Gerencia autenticação e autorização de usuários.	ASP.NET Identity
Models/DTOs	Representam entidades e objetos de transferência de dados.	-

---

5. Configuração de Ambiente Local

Pré-Requisitos

- **.NET 8** SDK
- **SQL Server** (local ou remoto)
- **Visual Studio 2022** (Recomendado)
- **Pacotes NuGet Relevantes:** Microsoft.EntityFrameworkCore.SqlServer, Microsoft.AspNetCore.Identity.EntityFrameworkCore, Swashbuckle.AspNetCore.

Passos para Executar Localmente

1. Clone o repositório.
2. Configure a string de conexão no appsettings.json:

```
"ConnectionStrings": {
  "DefaultConnection": "Server=localhost;Database=APIEnercheck;User
  Id=usuario;Password=senha;"
},
```

3. Execute as Migrations do banco de dados:

```
dotnet ef database update
```

4. Compile e execute o projeto:

```
dotnet build
dotnet run
```

5. Acesse o **Swagger UI** para testar os *endpoints*:

Normalmente em https://localhost:{porta}/swagger