Back-End

Todas as informações a respeito do back-end do software Patrimônio

- Swagger
- Banco

Swagger

Descrição Técnica do Back-End

O backend do projeto é estruturado de forma modular, com separação clara de responsabilidades entre rotas, controllers, middlewares e utilitários. A aplicação é desenvolvida utilizando **Node.js** com **Express** e toda a API está documentada de forma interativa utilizando o **Swagger UI**, acessível pelo endereço:

> http://localhost:${PORT}/api/v1/swagger onde **PORT** é definida pela variável de ambiente `API_PORT` ou assume o valor padrão **3006**.

Lembre-se de ter o projeto rodando no docker para funcionar!

Estrutura de Diretórios:

```
/api/
├── src/
│   ├── controller/        # ◫ Controladores responsáveis pela lógica de cada domínio.
│   │   ├── otp/            # Envio de OTPs e recuperação de senha
│   │   ├── patrimonio/
│   │   ├── setor/
│   │   ├── usuario/
│
│   ├── docs/              # ◫ Scripts de geração e de documentação Swagger.
│   │   └── compare-docs.ts
│   │   └── generate-docs.ts
│   │   └── swagger.ts
│   │   └── swaggerSchemas.ts
│
│   ├── middlewares/       # ⚙ Middlewares diversos, incluindo o multer para uploads
```

```
│   │   ├── authFacade/
│   │   ├── basicAuth/
│   │   ├── checkSetor/
│   │   ├── multer/
│   │
│   ├── router/            # 📂 Definição de todas as rotas do sistema organizadas por domínio
│   │   ├── otp/
│   │   ├── patrimonio/
│   │   ├── setor/
│   │   ├── usuario/
│   │   └── router.ts      # Roteador principal que importa e agrupa os módulos acima
│   │
│   ├── utils/             # 📂 Funções auxiliares e utilitários reutilizáveis.
│   │   ├── administrador/
│   │   ├── email/
│   │   ├── otp/              # gerar OTP
│   │   ├── qrcode/
│   │   ├── types/
│   │   ├── validations/      # Validação cpf
│   │   └── expirarSetor.ts   # Validação de expiração do setor
│   │
│   └── app.ts         # 📂 Ponto de entrada da aplicação e inicialização dos serviços
```

Documentação Interativa com Swagger

A API conta com documentação completa utilizando **Swagger**, onde é possível:

- Visualizar **todos os endpoints disponíveis**
- Ver os **parâmetros esperados** por cada rota
- Testar requisições diretamente pela interface
- Visualizar exemplos de **respostas de sucesso e erro**

Exemplo de rotas documentadas:

- `POST /api/v1/patrimonio/cadastrar`
  permite que um usuário com a devida permissão cadastre um novo patrimônio no sistema.

- **`PUT /api/v1/patrimonio/editar`**
  Permite que um usuário com a devida permissão edite um patrimônio.
- **`GET /api/v1/usuario/buscar`**
  Retorna os usuários cadastrados.
- **`POST /api/v1/setor/cadastrar`**
  Permite ao usuário com a devida permissão cadastrar um novo setor no sistema.

Essas e outras rotas estão organizadas por grupos lógicos (Patrimônio, Usuário, Setor) dentro do Swagger UI.

Upload de Arquivos com Multer

A aplicação utiliza o middleware **Multer** para lidar com uploads de arquivos (como csv), salvando-os em diretórios específicos com nomes padronizados conforme o campo da requisição.

Banco

Este sistema utiliza **Microsoft SQL Server** como banco relacional, com mapeamento feito via **TypeORM.**

Principais entidades do sistema:

```
# □□ Estrutura do Banco de Dados (SQL Server + TypeORM)

□□ Tabelas Principais
├── usuarios              # □□ Usuários do sistema
│   ├── id (int, PK)                    # Identificador único
│   ├── setor_id (uuid, FK → setores.id)    # Setor vinculado
│   ├── cpf (varchar[11], único)
│   ├── email (varchar[255], único)
│   ├── password (varchar)
│   ├── nome (varchar)
│   ├── sobrenome (varchar)
│   ├── matricula (varchar[20])
│   ├── permissao (varchar)               # SUPERADMIN | ADMIN | ASSISTENTE
│   ├── otp (varchar)                   # Código OTP associado
│   ├── is_active (bit, default true)       # Usuário ativo/inativo
│   ├── is_superuser (bit, default false)   # Indica se é superusuário
│   ├── date_joined (datetime2)           # Data de criação
│   ├── last_login (datetime2)            # Último login
│
├── setores              # □□ setores aos quais os patrimônios pertencem
│   ├── id (uuid, PK)                   # Identificador único
│   ├── nome (varchar[100])
│   ├── abreviacao (varchar[20])
│   ├── comarca (varchar[100])
│   ├── defensoria (varchar[100])
│   ├── predio (varchar[100])
│   ├── is_supersetor (bit, default false)  # Indica se é um setor principal
│   ├── expiration_date (datetime2, null)   # Data de expiração
```

```
│   ├── validated_year (datetime2, null)     # Último ano de validação
│   ├── is_active (bit, default true)        # Setor ativo/inativo
│
├── patrimonios             # 🏢 patrimônios cadastrados
│   ├── id (uuid, PK)                        # Identificador único
│   ├── setor_id (uuid, FK → setores.id)     # Setor responsável
│   ├── user_created_id (int, FK → usuarios.id)   # Usuário que cadastrou
│   ├── user_edit_id (int, FK → usuarios.id, null)# Último editor
│   ├── user_validate_id (int, FK → usuarios.id, null)# Validador
│   ├── especificacao (varchar[255])
│   ├── gestao (varchar[5], null)
│   ├── tombamento (varchar[16], null)
│   ├── serie (varchar[100], null)
│   ├── subitem (varchar[2], null)
│   ├── valor (bigint, default 0, null)
│   ├── qrcode (varchar[100], null)
│   ├── observacao (varchar[300], null)
│   ├── status (int, default 0)
│   ├── is_new (bit, default true)           # Indica se é um bem recém-criado
│   ├── is_active (bit, default true)        # Indica se o bem está ativo
│   ├── date_create (datetime2)              # Data de criação
│   ├── date_edit (datetime2, null)          # Data de última edição
│   ├── date_validate (datetime2, null)      # Data de validação
│
├── otps                    # 🔐 Códigos temporários de autenticação para nova senha
│   ├── id (uuid, PK)                        # Identificador único
│   ├── user_id (int, FK → usuarios.id)      # Usuário vinculado
│   ├── codigo (varchar[10])                 # Código OTP
│   ├── data_criacao (datetime2, default GETDATE())   # Gerado em
│   ├── data_expiracao (datetime2, default +5min)     # Expira em
│
└── logs                    # 📜 Histórico de ações e auditoria
    ├── id (int, PK)
    ├── user_id (int, FK → usuarios.id)
    ├── action_flag (smallint, unsigned)
    ├── action_time (datetime2)
    ├── object_id (varchar[100])
    ├── object_repr (varchar[255])
    ├── change_message (varchar[255])
```