Perfeito, Érico 👏

Aqui está a **versão final revisada do seu README.md**, pronta para commit no repositório **Gestão Fácil** — com correções de encoding, sumário clicável, bloco "Como Rodar o Projeto", exemplos de middleware e rodapé profissional com autor/licença.

```
# 🧱 Sistema de Gestão de Ativos e Serviços

Este projeto oferece uma estrutura completa para o **gerenciamento de ativos**, **serviços técnicos** e **fluxo operacional**, com **controle de SLA** e autenticação segura via **JWT**.

---

## 📑 Sumário

- [🚀 Tecnologias Utilizadas](#-tecnologias-utilizadas)
- [📂 Estrutura do Projeto](#-estrutura-do-projeto)
- [🧰 Como Rodar o Projeto](#-como-rodar-o-projeto)
- [🔑 Entidades Principais](#-entidades-principais)
- [🔗 Entidades e Relacionamentos](#-entidades-e-relacionamentos)
- [🧭 Diagrama Conceitual Resumido](#-diagrama-conceitual-resumido)
- [🔐 Testes de Autenticação](#-testes-de-autenticação)
- [🧩 Middleware de Autenticação](#-middleware-de-autenticação)
- [🧪 Testes Recomendados](#-testes-recomendados)
- [🧱 Estruturas JSON para Testes](#-estruturas-json-para-testes)
- [🔄 Fluxo do Ciclo de Vida de um Serviço](#-fluxo-do-ciclo-de-vida-de-um-serviço)
- [📄 Documentação Automática (Swagger)](#-documentação-automática-swagger)
- [🛠️ Manutenção Administrativa](#-manutenção-administrativa)
- [🔒 Segurança e Keep-Alive](#-segurança-e-keep-alive)
- [👤 Autor](#-autor)
- [🪪 Licença](#-licença)

---

## 🚀 Tecnologias Utilizadas

* **Node.js** — Plataforma principal do backend
* **Express** — Framework para rotas e middlewares REST
* **Sequelize** — ORM para banco relacional
* **JWT (JSON Web Token)** — Autenticação segura
* **Arquitetura em Camadas** — Separação clara entre *models*, *controllers* e *routes*
* **Swagger** — Documentação interativa da API
* **Keep-Alive** — Mantém API ativa no Render em modo gratuito

---

## 📂 Estrutura do Projeto
```

```
GestaoFacil/
├── src/
│   ├── app.js # Ponto de entrada principal
│   ├── config/ # Configurações do Sequelize e .env
│   ├── models/ # Entidades Sequelize
│   ├── controllers/ # Lógica de negócio
│   ├── routes/ # Rotas Express REST
│   ├── middlewares/ # Autenticação e validação
│
├── api/ # Versão alternativa da API
│   ├── public/ # Interface web de teste
│   ├── cert/ # Certificados SSL
│
├── migrations/ # Migrations do banco
├── .env # Variáveis de ambiente
├── package.json # Dependências e scripts
```

```
---

## 🧰 Como Rodar o Projeto

1. **Instalar dependências**
   ```bash
   npm install
```

2. **Configurar o banco e variáveis**

```
cp .env.example .env
# Edite credenciais e JWT_SECRET
```

3. **Executar migrações e seeds**

```
npx sequelize-cli db:migrate
npm run seed
```

4. **Iniciar o servidor**

```
npm run dev
```

5. **Testar endpoint**
```

```
GET http://localhost:3000/teste
```

🔑 Entidades Principais

👤 Cliente

- `id`, `nome`, `cnpj`, `contatos`
- Relacionamentos: possui **Ativos** e **Serviços**

🕵️ Usuário

- `id`, `nome`, `cargo`, `email`, `telefone`
- Pode ser **Solicitante** ou **Responsável** por serviços

🏢 Local

- `id`, `nome`
- Contém vários **Ativos**

⚙️ Ativo

- `id`, `nome`, `numeroSerie`, `status`, `detalhes`
- Pertence a um **Cliente** e está em um **Local**

🧩 Tipo de Serviço

- `id`, `nome`, `descricao`
- Classifica serviços e define **SLA**

🛠️ Serviço

- `id`, `descricao`, `status`, `dataAgendada`, `dataConclusao`
- Relacionado a **Cliente**, **Ativo**, **Tipo de Serviço**, e **Usuários**

🔗 Entidades e Relacionamentos

Modelo	Campos principais	Relacionamentos
Cliente	id, nome, cnpj, contatos	hasMany(Ativo), hasMany(Servico)
Ativo	id, nome, numeroSerie, status	belongsTo(Cliente), belongsTo(Local), hasMany(Servico)
Serviço	id, descricao, status, datas	belongsTo(Cliente), Ativo, TipoServico, Usuário
Local	id, nome	hasMany(Ativo)
Usuário	id, nome, cargo, email	relacionado a Serviços como solicitante/responsável

---

Diagrama Conceitual Resumido

```
Cliente 1---* Ativo *---1 Local
Cliente 1---* Servico *---1 Ativo
Usuario 1---* Servico (solicitante/responsavel)
Servico *---1 TipoServico
```

Testes de Autenticação

Registro de Usuário

```
POST /auth/register
{
  "nome": "Erico",
  "email": "erico@teste.com",
  "cargo": "admin",
  "telefone": "85999999999",
  "password": "123456",
  "confirmPassword": "123456"
}
```

Login

```
POST /auth/login
{
  "email": "erico@teste.com",
  "password": "123456"
}
```

Retorno:

```
{ "token": "<JWT>" }
```

Middleware de Autenticação

Exemplo: `src/middlewares/auth.js`

```
const auth = require("../middlewares/auth");

// Rota protegida
router.get("/v1/clientes", auth(), clienteController.listar);
```

```
// Rota restrita a administradores
router.post("/v1/admin/relatorios", auth(["admin"]), relatorioController.gerar);
```

Testes Recomendados

- Registro com senhas diferentes
- E-mail duplicado
- Login com senha incorreta
- Acesso sem token
- Token expirado

Fluxo do Ciclo de Vida de um Serviço

```
Aberto → Em andamento → Concluído → Encerrado
```

Documentação Automática (Swagger)

```
npm install swagger-ui-express swagger-jsdoc
```

app.js

```
const swaggerUi = require('swagger-ui-express');
const swaggerJsdoc = require('swagger-jsdoc');

const swaggerOptions = {
  swaggerDefinition: {
    openapi: '3.0.0',
    info: { title: 'Gestão Fácil API', version: '1.0.0' }
  },
  apis: ['./src/routes/*.js']
};

const swaggerDocs = swaggerJsdoc(swaggerOptions);
app.use('/docs', swaggerUi.serve, swaggerUi.setup(swaggerDocs));
```

Disponível em `/docs`

Manutenção Administrativa

Fix Client Services

POST `/v1/servicos/admin/fix-client-services`

```
{
  "clienteId": 12,
  "numeroSerie": "C52-HIK-2025",
  "nome": "Câmera Pátio Central"
}
```

Query param opcional:
`?dryRun=true` → simula sem aplicar alterações.

Segurança e Keep-Alive

Para evitar exposição de dados sensíveis e manter o serviço ativo:

- `KEEP_ALIVE_URL` derivada do próprio serviço
- `/teste` serve apenas conteúdo estático
- `.env` nunca deve ser versionado

Variáveis principais:

```
APP_MODE=production
KEEP_ALIVE_ENABLED=true
KEEP_ALIVE_URL=https://gestaofacil.onrender.com/teste
KEEP_ALIVE_INTERVAL_MS=300000
PUBLIC_API_BASE_URL=https://gestaofacil.onrender.com
```

☑ Checklist:

- `.env` fora do versionamento
- `KEEP_ALIVE_ENABLED` ajustado no deploy
- Logs não expõem tokens ou headers

Autor

Érico de Freitas Neto
Desenvolvedor Full-Stack | Sistemas de Videomonitoramento e Gestão de Ativos
GitHub: EricofreitasNeto
erico@teste.com

Licença

Distribuído sob a licença **MIT**.
Sinta-se livre para usar e adaptar conforme necessário, mantendo os créditos originais.