Documentação do Projeto - Indústrias Wayne

Este documento descreve a estrutura e o funcionamento do projeto final "Sistema de Gerenciamento de Segurança - Indústrias Wayne".  
O sistema foi desenvolvido utilizando Python (Flask) no backend e HTML, CSS e JavaScript no frontend.

# 1. Arquitetura do Sistema

O projeto segue uma arquitetura cliente-servidor. O backend, construído com o framework Flask, fornece as rotas e APIs necessárias para comunicação com o frontend.  
O frontend é composto por páginas HTML, estilizadas com CSS e dinamizadas com JavaScript (Fetch API), que consomem os dados providos pelo backend.

# 2. Tecnologias Utilizadas

- Backend: Python 3 + Flask  
- Banco de Dados: SQLite  
- Frontend: HTML5, CSS3, JavaScript (ES6)  
- Bibliotecas: Flask-SQLAlchemy, Werkzeug, Flask-Login

# 3. Estrutura do Projeto

/app.py ...................... Código principal do servidor Flask  
/seed.py ..................... Criação do banco e dados iniciais  
/templates ................... Páginas HTML renderizadas pelo Flask  
/static/css .................. Estilos visuais do sistema  
/static/js ................... Lógica de interação frontend (Fetch API)  
/wayne.db .................... Banco de dados SQLite gerado automaticamente

# 4. Integração Backend e Frontend

A comunicação entre o frontend e o backend é realizada por meio de requisições HTTP usando a Fetch API do JavaScript.  
Os endpoints do Flask retornam e recebem dados em formato JSON, garantindo uma integração leve e eficiente.  
  
Por exemplo:  
- O frontend envia uma requisição fetch('/api/resources') para listar recursos.  
- O backend responde com um JSON contendo a lista de recursos do banco de dados.  
- As funções JavaScript atualizam dinamicamente o conteúdo da página com esses dados.  
  
Essa integração é bidirecional: o frontend envia dados via POST ou PUT para criar e atualizar registros, e o backend processa, valida e grava no banco de dados.

# 5. Autenticação e Sessão

O sistema utiliza autenticação por sessão via Flask. Quando o usuário faz login, o backend cria uma sessão persistente armazenando o ID do usuário.  
Com isso, as rotas da API que requerem autenticação verificam o identificador do usuário ativo na sessão antes de autorizar o acesso.

# 6. Banco de Dados

O banco de dados utiliza SQLite e contém duas tabelas principais:  
- User: armazena informações de login, senha criptografada e papel (funcionário, gerente, admin);  
- Resource: armazena dados dos recursos de segurança (nome, tipo, status, local e observações).

# 7. Fluxo de Funcionamento

1. O usuário acessa a página de login.  
2. Após autenticação, é redirecionado para o dashboard principal.  
3. O JavaScript do frontend requisita estatísticas e lista de recursos ao backend.  
4. O backend busca as informações no banco e retorna via JSON.  
5. O frontend exibe as informações na tela e permite criar, editar ou excluir registros, conforme o nível de permissão.

# 8. Considerações Finais

O sistema integra de forma eficiente o backend em Python e o frontend web, demonstrando domínio em desenvolvimento full stack.  
Pode ser facilmente expandido para uso em produção com melhorias como autenticação JWT, uso de banco de dados PostgreSQL e deploy em nuvem.