🏥 SGHSS – Sistema de Gerenciamento Hospitalar de Saúde e Serviços

Capa

Autor: Deleon Santos
Curso: Desenvolvimento Fullstack / Backend
Disciplina: Projeto Integrador – Back-end
Título: Documentação Técnica do Sistema SGHSS
Ano: 2025

Sumário

1. Introdução

O **SGHSS (Sistema de Gerenciamento Hospitalar de Saúde e Serviços)** é uma aplicação **Back-end desenvolvida em Python com Flask**, projetada para gerenciar

consultas médicas, pacientes, médicos e secretarias.
O objetivo principal é **automatizar o fluxo de agendamento e registro de atendimentos**, garantindo segurança, autenticação de usuários, e rastreabilidade através de logs e tokens JWT.

A aplicação segue boas práticas de desenvolvimento seguro e está em conformidade com a **Lei Geral de Proteção de Dados (LGPD)**, garantindo o sigilo e integridade das informações médicas e pessoais.

2. Requisitos Funcionais e Não Funcionais

Requisitos Funcionais

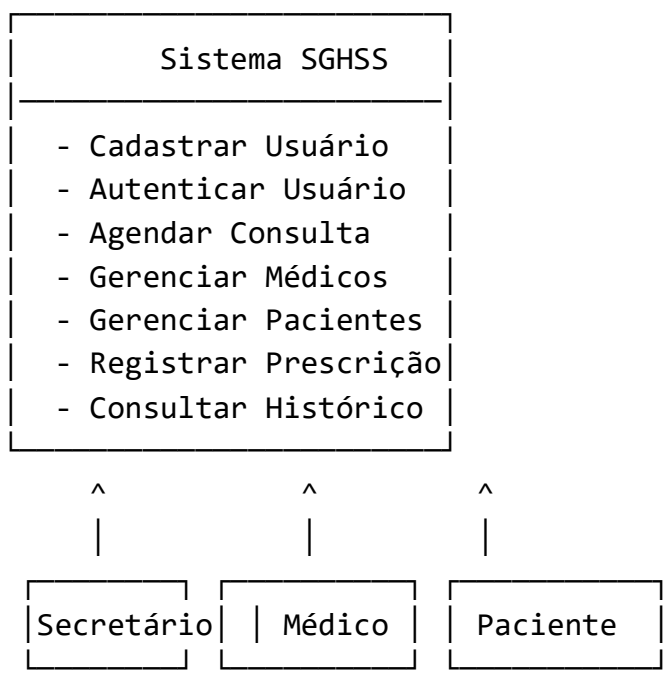
ID	Descrição	Prioridade
RF01	Cadastrar pacientes	Alta
RF02	Cadastrar médicos	Alta
RF03	Cadastrar secretarias	Alta
RF04	Realizar autenticação (login/logout)	Alta
RF05	Agendar consultas	Alta
RF06	Listar e filtrar consultas (por status e data)	Média
RF07	Finalizar consulta e registrar prescrição	Média
RF08	Gerar logs de atividade	Média
RF09	Documentar rotas via Swagger	Alta

Requisitos Não Funcionais

ID	Descrição
RNF01	O sistema deve ser implementado em **Python 3.12** com **Flask**
RNF02	O banco de dados utilizado deve ser **SQLite3**
RNF03	A API deve seguir o padrão **RESTful**
RNF04	O sistema deve utilizar **JWT (JSON Web Token)** para autenticação
RNF05	As senhas devem ser armazenadas de forma **criptografada**
RNF06	O sistema deve registrar logs de operações sensíveis
RNF07	Deve ser possível testar via **Swagger**, **Postman** ou **Insomnia**

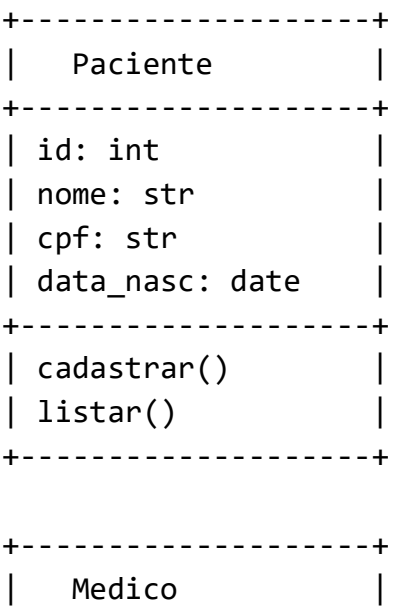
3. Modelagem

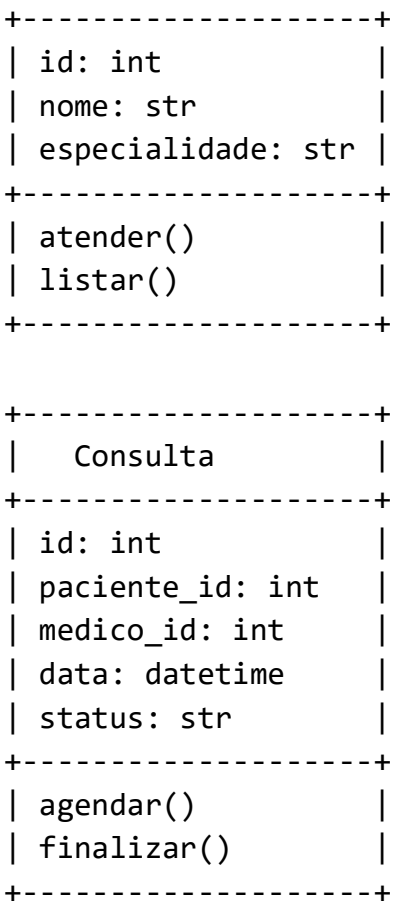
3.1 Diagrama de Caso de Uso



(Representação simplificada do caso de uso principal.)

3.2 Diagrama de Classes





3.3 Estrutura do Banco de Dados

Tabelas Principais:

- `pacientes (id, nome, cpf, data_nasc)`
- `medicos (id, nome, especialidade)`
- `secretarios (id, nome, matricula)`
- `consultas (id, paciente_id, medico_id, data, status)`
- `usuarios (id, nome, email, senha_hash, tipo)`

4. Implementação

4.1 Arquitetura do Sistema

O projeto é modularizado em pacotes:

```
app/
├── models/
│   ├── paciente.py
│   ├── medico.py
│   ├── consulta.py
│   └── secretario.py
├── routes/
│   ├── consultas_medico.py
│   ├── consultas_secretaria.py
│   └── auth.py
├── extensions/
│   ├── __init__.py
│   └── db.py
├── static/swagger.json
├── config.py
└── app.py
```

4.2 Endpoints (Exemplos Principais)

◇ *Autenticação*

POST /api/login
Entrada:

```
{
  "email": "usuario@email.com",
  "senha": "123456"
}
```

Saída esperada:

```
{
  "token": "eyJhbGciOiJIUzI1NiIsIn..."
}
```

◇ *Cadastro de Paciente*

POST /api/pacientes

```
{
  "nome": "João Silva",
  "cpf": "12345678900"
}
```

◇ *Listar Médicos*

GET /api/medicos

```
[
  {"id": 1, "nome": "Dra. Ana Paula", "especialidade":
"Cardiologia"},
  {"id": 2, "nome": "Dr. João Lima", "especialidade": "Pediatria"}
]
```

◇ *Agendar Consulta*

POST /api/consultas

```
{
  "paciente_id": 1,
  "medico_id": 2,
  "data": "2025-11-10T10:00:00"
}
```

4.3 Autenticação e Segurança

- Autenticação via **JWT** (Bearer Token).
- As senhas são armazenadas com **hash bcrypt**.
- Cada rota protegida exige `@jwt_required()`.
- Implementação conforme **LGPD**: dados pessoais visíveis apenas a usuários autorizados.

5. Plano de Testes (Postman/Insomnia)

Caso de Teste	Endpoint	Método	Resultado Esperado
CT01	`/api/login`	POST	Retorna token JWT válido
CT02	`/api/pacientes`	POST	Cria paciente e retorna JSON com ID
CT03	`/api/medicos`	GET	Retorna lista de médicos
CT04	`/api/consultas`	POST	Cria nova consulta
CT05	`/api/consultas/{id}`	PUT	Atualiza status da consulta

(Capturas de tela do Postman podem ser anexadas nesta seção.)

6. Conclusão

O sistema **SGHSS** demonstra o funcionamento completo de um **back-end seguro, escalável e documentado**, com autenticação, CRUD de entidades médicas, e rotas RESTful.
O projeto foi desenvolvido com foco em **boas práticas, modularização e segurança**, pronto para integração com um front-end simples ou sistemas móveis.

7. Referências

- Documentação oficial do Flask (https://flask.palletsprojects.com)
- JWT Authentication (https://jwt.io)
- LGPD – Lei nº 13.709/2018
- PEP8 – Python Enhancement Proposals

8. Anexos (Opcional)

- Capturas de testes no Postman
- Logs de execução do servidor

- Link do repositório:
  🔗 https://github.com/seu_usuario/seu_repositorio