📌 SGHSS – Sistema de Gerenciamento Hospitalar

API RESTful completa para gestão hospitalar, desenvolvida em **Python + Flask**, com autenticação via **JWT**, documentação interativa com **Swagger UI**, e integração robusta com banco de dados usando **SQLAlchemy**.

📖 Índice

📘 Introdução

O **SGHSS** (Sistema de Gerenciamento Hospitalar de Saúde e Serviços) é uma aplicação backend projetada para facilitar o gerenciamento de consultas médicas, usuários (pacientes, médicos e secretários) e processos administrativos essenciais dentro de uma unidade de saúde.

A API foi construída com foco em ser simples, escalável e segura. A utilização de autenticação JWT garante que todas as transações sejam protegidas, seguindo rigorosamente as boas práticas de desenvolvimento RESTful.

🚀 Funcionalidades

O sistema segrega as funcionalidades de acordo com o nível de acesso do usuário, garantindo o princípio da menor privilégio.

Médico

Funcionalidade	Descrição
Consultar Agenda	Visualizar todas as consultas agendadas.
Finalizar Consulta	Registrar o término do atendimento.
Registrar Diagnóstico	Anotar o diagnóstico do paciente.
Prescrever	Emitir prescrição de medicamentos.
Solicitar Exames	Solicitar exames complementares.
Cancelar Consulta	Cancelar um agendamento.

👤 Paciente

Funcionalidade	Descrição
Listar Médicos	Ver médicos credenciados disponíveis.
Ver Consultas	Visualizar consultas agendadas.
Agendar	Marcar nova consulta.
Cancelar Consulta	Cancelar um agendamento.

🏢 Secretaria

Funcionalidade	Descrição
Cadastrar Pacientes	Registrar novos pacientes.
Cadastrar Médicos	Registrar novos médicos.
Listar Usuários	Listar usuários gerais (pacientes, médicos, secretários).
Buscar Consulta	Buscar consulta por ID.
Bloquear Pacientes	Restringir acesso de pacientes.

Excluir Médicos	Remover registros de médicos.

🔐 Autenticação

O sistema implementa:

- **Login com JWT** (com tempo de expiração definido em 60 minutos).
- **Atualização de senha** para o usuário atualmente autenticado.

🛠️ Tecnologias Utilizadas

Tecnologia	Descrição
🐍 **Python 3.12**	Linguagem principal de desenvolvimento.
⚗ **Flask**	Microframework web para a API REST.
🧪 **Flask-JWT-Extended**	Implementação de autenticação baseada em JSON Web Tokens.
🗄️ **Flask-SQLAlchemy**	ORM (Object-Relational Mapper) para manipulação de banco de dados.
🔄 **Flask-CORS**	Controle de acesso HTTP (Cross-Origin Resource Sharing).
📘 **Swagger UI**	Geração e visualização da documentação interativa da API.
🔐 **Werkzeug Security**	Utilizado para *hashing* seguro de senhas.

📁 Estrutura do Projeto

A arquitetura do projeto segue um padrão modular com foco em separação de responsabilidades (Models, Routes e Extensions):

```
SGHSS/
├── app/
│   ├── models/          # Definições de modelos de dados (SQLAlchemy)
│   ├── routes/          # Definições de rotas da API
│   ├── extensions.py    # Inicialização de extensões (DB, JWT, CORS)
│   └── __init__.py      # Configuração da aplicação Flask
│
├── requirements.txt     # Dependências do projeto
├── run.py               # Ponto de entrada da aplicação
```

```
├── swagger.json          # Arquivo de definição do Swagger
└── README.md
```

📌 Endpoints da API

Abaixo, uma visão geral dos endpoints principais. A documentação Swagger completa já foi definida e está acessível através da rota /api/docs.

🔐 Autenticação

Método	Endpoint	Descrição
POST	api/login	Gerar token JWT.
PUT	api/editar_senha	Atualizar senha do usuário autenticado.

👨‍⚕️ Grupo Médico

Método	Endpoint	Descrição
GET	api/consulta/consulta_agenda_medica	Agenda do médico.
POST	api/consulta/{id}/atendimento	Finalizar consulta e registrar diagnóstico.
POST	api/consulta/{id}/prescreve_tratamento	Prescrever medicamentos.
POST	api/consulta/{id}/solicita_exame	Solicitar exames.
PUT	api/consulta/{id}/ cancelamento_consulta	Cancelar consulta.

👤 Grupo Paciente

Método	Endpoint	Descrição
GET	api/consulta/lista_medico_credenciado	Listar médicos credenciados.
GET	api/consulta/agendamento_paciente	Ver consultas agendadas do paciente.
GET	api/consulta/agenda_medica/	Ver agenda de um médico

	{medico_id}	específico.
POST	api/consulta/novo_agendamento	Agendar nova consulta.
PUT	api/consulta/{id}/ paciente_cancelamento	Cancelar consulta pelo paciente.

🏢 Grupo Secretaria

Método	Endpoint	Descrição
POST	api/cadastra/novo_medico	Cadastrar novo médico.
DELETE	api/deleta/medico/{id}	Excluir médico.
POST	api/cadastra/novo_paciente	Cadastrar novo paciente.
POST	api/bloqueia/paciente/{id}	Bloquear paciente.
GET	api/consulta/cadastro_geral_usuarios	Listar todos os usuários.
GET	/consulta/{id}	Buscar consulta por ID.
GET	/consulta/consultas_geral_marcadas	Listar todas as consultas agendadas.

📜 Trechos Importantes do Código

🔐 Autenticação JWT

A criação do token é fundamental e inclui o nível de acesso do usuário:

```
token = create_access_token(
    identity={
        "id": usuario.id,
        "nivel_acesso": usuario.nivel_acesso
    },
    expires_delta=timedelta(minutes=60)
)
```

🛡️ Proteção de Rotas

O decorador @jwt_required() protege a rota, e a lógica interna valida o nível de acesso (medico):

```
@jwt_required()
def rota_protegida():
    usuario = get_jwt_identity()
```

```
    if usuario['nivel_acesso'] != 'medico':
        return jsonify({"erro": "Acesso negado"}), 403
    # ... lógica da rota para médicos
```

🩺 Finalizar Atendimento

Exemplo de transação de banco de dados via SQLAlchemy para atualizar o status da consulta:

```
consulta.diagnostico = data["diagnostico"]
consulta.status = "realizada"
db.session.commit()
```

⚙️ Como Clonar e Executar o Projeto

Siga os passos abaixo para configurar e rodar a API localmente no seu ambiente.

📥 1. Clonar o repositório

git clone https://github.com/deleonsantos/SGHSS.git

🧱 2. Criar ambiente virtual

python -m venv venv

▶ 3. Ativar ambiente

Sistema Operacional	Comando
Windows	venv\Scripts\activate
Linux/macOS	source venv/bin/activate

📦 4. Instalar dependências

pip install -r requirements.txt

▶ 5. Executar a aplicação

python run.py

A API estará rodando em:

http://127.0.0.1:5000

Acesse a documentação interativa **Swagger UI** em:

http://127.0.0.1:5000/api/docs

📦 Requirements.txt

As dependências necessárias para o projeto estão listadas abaixo:

```
Flask==3.0.3
Flask-Cors==6.0.1
Flask-JWT-Extended==4.6.0
Flask-SQLAlchemy==3.1.1
flask-swagger-ui==4.11.1
Werkzeug==3.0.3
SQLAlchemy==2.0.23
greenlet==3.0.1
python-dotenv==1.0.1
```

🧪 Testando a API no Postman

Para testar as rotas protegidas, você deve primeiro obter um token JWT fazendo login.

🔑 Autenticação

Para todas as rotas protegidas, adicione o seguinte cabeçalho (Header) no Postman após obter seu token no /login:

Chave	Valor
Authorization	Bearer SEU_TOKEN_AQUI

📩 Exemplo POST /login

Body (JSON):

```
{
  "email": "deleon@santos",
  "senha": "4556949"
}
```

📝 Conclusão

O SGHSS é um sistema robusto e confiável para o gerenciamento hospitalar. Sua arquitetura modular e o uso de tecnologias modernas, como Flask e JWT, garantem um sistema escalável e seguro.

Com a documentação Swagger e os padrões REST bem definidos, o sistema está pronto para uso e fácil integração com aplicações front-end.