**DOCUMENTACIÓN COMPLETA DEL PROYECTO**

**Sistema de Gestión Clínica - Salud Vital Ltda.**

**Objetivo General**

Desarrollar un sistema de gestión clínica basado en Django que permita administrar pacientes, médicos, consultas, tratamientos y recetas médicas de manera eficiente.

**Objetivos Específicos**

* Registrar y gestionar la información de pacientes, médicos y especialidades.
* Controlar las consultas médicas y sus tratamientos asociados.
* Gestionar la prescripción de recetas médicas y medicamentos.
* Permitir la visualización de horarios de atención de los médicos.
* Ofrecer una API REST documentada para la interoperabilidad con otros sistemas.
* Facilitar el acceso al personal autorizado mediante una interfaz web intuitiva.

**3. Descripción General del Sistema**

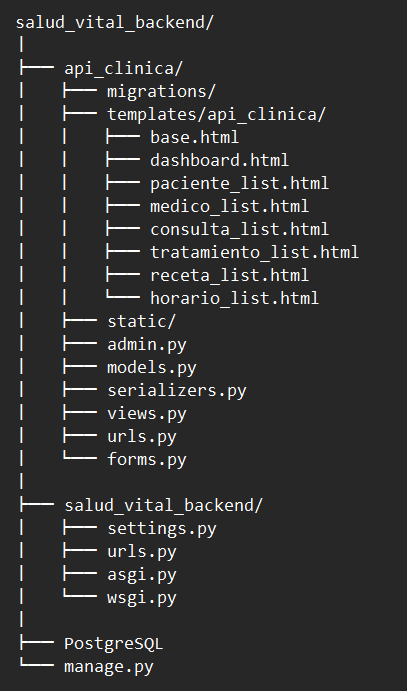
El sistema permite realizar las principales operaciones CRUD (Crear, Leer, Actualizar y Eliminar) sobre los datos clínicos más relevantes.  
Está diseñado bajo una **arquitectura cliente-servidor**, donde Django actúa como servidor backend y las vistas HTML brindan la capa de presentación.

Además, incluye una **API REST** con documentación integrada mediante **Swagger** y **Redoc**, lo que facilita su integración con otros servicios o aplicaciones móviles.

**Tecnologías Utilizadas**

|  |  |
| --- | --- |
| **Tecnología** | **Uso Principal** |
| **Python 3.13** | **Framework backend y ORM** |
| **Django REST Framework** | **Creación de la API REST** |
| **Bootstrap 5.3** | **Diseño y estilo del frontend** |
| **PostgreSQL** | **Sistema de base de datos** |
| **DRF-YASG** | **Generación de documentación Swagger** |
| **HTML + CSS** | **Interfaz visual y plantillas Django** |

**5. Estructura del Proyecto**



**6. Modelos Principales**

**Modelo Médico**

Contiene los datos del personal médico, como nombre, apellido, especialidad, rut y correo.

**Modelo Paciente**

Registra información personal y médica básica del paciente (nombre, edad, sexo, dirección, contacto).

**Modelo Medicamento**

Define los medicamentos disponibles, su tipo (oral, inyectable, tópico) y dosis.

**Modelo Consulta**

Registra las consultas médicas realizadas, enlazando al médico y al paciente.

**Modelo Tratamiento**

Describe los tratamientos indicados a los pacientes tras una consulta médica.

**Modelo Receta**

Permite gestionar las recetas médicas y los medicamentos indicados en cada caso.

**Modelo HorarioAtencion**

Define los horarios de atención de cada médico con campos para día, hora de inicio y hora de término.

**7. API REST**

El sistema expone una API REST desarrollada con Django REST Framework, con endpoints para cada entidad.  
Los principales endpoints son:

|  |  |  |  |
| --- | --- | --- | --- |
| Recurso | Endpoint | Método | Descripción |
| Pacientes | /api/pacientes/ | GET / POST | Listar o crear pacientes |
| Médicos | /api/medicos/ | GET / POST | Listar o crear médicos |
| Especialidades | /api/especialidades/ | GET / POST | Gestión de especialidades |
| Consultas | /api/consultas/ | GET / POST | Registrar y listar consultas |
| Tratamientos | |  | | --- | |  |  |  | | --- | | /api/tratamientos/ | | GET / POST | Gestionar tratamientos médicos |
| Recetas | /api/recetas/ | GET / POST | Crear y listar recetas |
| Horarios | /api/horarios/ | GET / POST | Ver horarios de atención |

Documentación disponible en:

* Swagger UI: /docs/
* Redoc UI: /redoc/

**8. Interfaz Web**

El sistema cuenta con una interfaz HTML construida sobre Bootstrap.  
El Dashboard principal muestra estadísticas resumidas:

* Total de pacientes, médicos, especialidades y consultas.
* Accesos rápidos para crear nuevos registros.
* Enlaces a la documentación y administración del sistema.

Cada módulo (Pacientes, Médicos, Consultas, etc.) tiene:

* Listado general con botones para Editar y Eliminar.
* Formularios intuitivos para agregar nuevos registros.

**9. Vista de Horario de Atención**

Se agregó un módulo de Horario de Atención para mostrar los días y horas disponibles de los médicos.

* Modelo: HorarioAtencion
* Vista: horario\_list
* URL: /horarios/
* Enlace visible en el menú principal bajo el apartado Gestión y en el Dashboard con un botón de acceso rápido.

**10. Seguridad y Buenas Prácticas**

* Se utilizan entornos virtuales para el manejo de dependencias (python -m venv eva2).
* Se aplica CSRF Protection en formularios.
* Validación de datos a través de serializers.
* Uso de filtros (DjangoFilterBackend) para consultas personalizadas.
* Manejo de usuarios y roles mediante el panel de administración Django.

**11. Conclusiones**

El sistema Salud Vital Ltda. representa una solución moderna y escalable para la gestión de clínicas pequeñas y medianas.  
Su desarrollo en Django garantiza seguridad, modularidad y facilidad de mantenimiento.  
La integración de vistas web y una API REST permite expandir el sistema hacia aplicaciones móviles o servicios externos.

Este proyecto demuestra el uso completo del ecosistema Django, desde modelos y vistas hasta plantillas, API y documentación automatizada**.**