Documentación Técnica Completa

Table of Contents

# 1 📋 DOCUMENTACIÓN TÉCNICA COMPLETA

## 1.1 Sistema de Gestión Clínica - Facultad de Ciencias Médicas

**Versión:** 1.0  
**Fecha:** 2025  
**Institución:** Universidad de San Carlos de Guatemala - Facultad de Ciencias Médicas

## 1.2 📖 ÍNDICE

1. [Información General](#X70313355dedc00a303ecc5246a3d8c07be7daed)
2. [Arquitectura del Sistema](#X9e4bf3adcbcb771d9880adccdad2b01afa01a08)
3. [Tecnologías Utilizadas](#X5a566f2977679294d35637d377dad47fb22f615)
4. [Estructura del Proyecto](#X553efa3b3461d540c5c3aef0a9c8f502f4449d7)
5. [Modelos de Datos](#Xc921130fc35dbb6cdfb8095a5f589159e0cef16)
6. [Sistema de Autenticación](#X2fd9e90a413910d5d34a63f8dc16a6fea0dc4fc)
7. [Módulos del Sistema](#Xacd8b6fb9de72b23ae95d72cc2d2d0b5ad9b7e1)
8. [Frontend y Interfaces](#X36b24dce97a7289f1901cdc07202b6d547c31c8)
9. [Configuración y Despliegue](#Xa087115fa84c3e2b2cd2cf61a595b7cf9ec705f)
10. [Guía de Instalación](#Xe9bfaeff80736f447869f7ea5f6b6b59f22393e)
11. [API Endpoints](#Xe3302eafd2719f9eca45c33ed65b84334dead8d)
12. [Seguridad](#Xc013efb945d256cec73859965aa7c39bf482a1e)
13. [Mantenimiento](#X9d68f77d75287ed2d819aa3c40fcb2e856fc7dd)
14. [Consideraciones Finales](#Xc431ba1c5a2e118e27f4671958d9c07cf95c37b)

## 1.3 1. INFORMACIÓN GENERAL

### 1.3.1 1.1 Descripción del Proyecto

El Sistema de Gestión Clínica es una aplicación web desarrollada para la Facultad de Ciencias Médicas de la Universidad de San Carlos de Guatemala. Su objetivo principal es digitalizar y optimizar los procesos de atención médica, gestión de pacientes y generación de reportes en las clínicas universitarias.

### 1.3.2 1.2 Objetivos del Sistema

* **Digitalizarse:** Migrar de registros físicos a digitales
* **Eficiencia:** Reducir tiempos de atención y registro
* **Trazabilidad:** Mantener historial completo de pacientes
* **Reportes:** Generar estadísticas y reportes médicos
* **Seguridad:** Proteger información médica sensible
* **Escalabilidad:** Soportar múltiples clínicas y usuarios

### 1.3.3 1.3 Alcance del Sistema

* Gestión completa de pacientes
* Registro de consultas médicas
* Signos vitales y antecedentes médicos
* Sistema de roles y permisos
* Generación de reportes en PDF
* Asignación de clínicas
* Historial médico completo

### 1.3.4 1.4 Usuarios del Sistema

* **Administradores:** Control total del sistema
* **Médicos Supervisores:** Gestión de clínicas y médicos
* **Médicos:** Atención de pacientes y consultas
* **Personal de Recepción:** Registro de pacientes

## 1.4 2. ARQUITECTURA DEL SISTEMA

### 1.4.1 2.1 Patrón Arquitectónico

El sistema implementa una **arquitectura MVC (Model-View-Controller)** con Flask:

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ VIEW │ │ CONTROLLER │ │ MODEL │  
│ (Templates) │◄──►│ (Routes) │◄──►│ (Database) │  
│ │ │ │ │ │  
│ - HTML/Jinja2 │ │ - Flask Routes │ │ - SQLAlchemy │  
│ - CSS/Bootstrap │ │ - Business Logic│ │ - Models │  
│ - JavaScript │ │ - Validations │ │ - Relationships │  
└─────────────────┘ └─────────────────┘ └─────────────────┘

### 1.4.2 2.2 Arquitectura por Capas

┌─────────────────────────────────────────────────────────────┐  
│ CAPA DE PRESENTACIÓN │  
│ HTML Templates + CSS + JavaScript + AJAX │  
├─────────────────────────────────────────────────────────────┤  
│ CAPA DE APLICACIÓN │  
│ Flask Routes + Forms + Business Logic │  
├─────────────────────────────────────────────────────────────┤  
│ CAPA DE DATOS │  
│ SQLAlchemy ORM + Models + Database │  
├─────────────────────────────────────────────────────────────┤  
│ CAPA DE PERSISTENCIA │  
│ SQLite Database File │  
└─────────────────────────────────────────────────────────────┘

## 1.5 3. TECNOLOGÍAS UTILIZADAS

### 1.5.1 3.1 Backend Framework

* **Flask 2.3.3:** Framework principal de Python
* **Flask-SQLAlchemy 3.1.1:** ORM para base de datos
* **Flask-Login 0.6.2:** Gestión de sesiones de usuario
* **Flask-WTF 1.2.1:** Formularios web y protección CSRF
* **Flask-Mail 0.9.1:** Envío de correos electrónicos
* **Flask-Migrate 4.0.5:** Migraciones de base de datos

### 1.5.2 3.2 Base de Datos

* **SQLite:** Base de datos principal (desarrollo)
* **MySQL-connector-python 8.1.0:** Conector MySQL (producción)
* **PyMySQL 1.1.0:** Driver alternativo MySQL

### 1.5.3 3.3 Frontend Technologies

* **HTML5:** Estructura de páginas
* **CSS3 + Bootstrap 5:** Estilos y responsive design
* **JavaScript ES6:** Interactividad del cliente
* **AJAX:** Comunicación asíncrona
* **Jinja2:** Motor de templates

### 1.5.4 3.4 Seguridad y Validación

* **WTForms 3.0.1:** Validación de formularios
* **Werkzeug 2.3.7:** Utilidades de seguridad
* **Passlib 1.7.4:** Hashing de contraseñas
* **Cryptography 41.0.7:** Funciones criptográficas
* **itsdangerous 2.1.2:** Tokens seguros

### 1.5.5 3.5 Reportes y Documentos

* **ReportLab 4.0.4:** Generación de PDFs
* **Email-validator 2.0.0:** Validación de emails

## 1.6 4. ESTRUCTURA DEL PROYECTO

### 1.6.1 4.1 Estructura de Directorios

clinica\_/  
├── 📁 app/ # Aplicación principal  
│ ├── 📄 \_\_init\_\_.py # Factory pattern de Flask  
│ ├── 📄 models.py # Modelos de datos  
│ ├── 📄 email.py # Funciones de email  
│ │  
│ ├── 📁 auth/ # Módulo de autenticación  
│ │ ├── 📄 \_\_init\_\_.py # Blueprint auth  
│ │ ├── 📄 routes.py # Rutas de autenticación  
│ │ └── 📄 forms.py # Formularios de auth  
│ │  
│ ├── 📁 main/ # Módulo principal  
│ │ ├── 📄 \_\_init\_\_.py # Blueprint main  
│ │ ├── 📄 routes.py # Rutas principales  
│ │ ├── 📄 forms.py # Formularios main  
│ │ ├── 📄 pdf\_reports.py # Generación PDFs  
│ │ └── 📄 plot\_generator.py # Gráficos estadísticos  
│ │  
│ ├── 📁 pacientes/ # Módulo de pacientes  
│ │ ├── 📄 \_\_init\_\_.py # Blueprint pacientes  
│ │ ├── 📄 routes.py # Rutas de pacientes  
│ │ └── 📄 forms.py # Formularios pacientes  
│ │  
│ ├── 📁 reportes/ # Módulo de reportes  
│ │ ├── 📄 \_\_init\_\_.py # Blueprint reportes  
│ │ └── 📄 routes.py # Rutas de reportes  
│ │  
│ ├── 📁 static/ # Archivos estáticos  
│ │ ├── 📁 js/ # JavaScript  
│ │ │ ├── 📄 consultas.js # Lógica de consultas  
│ │ │ └── 📄 reportes.js # Lógica de reportes  
│ │ └── 📁 img/ # Imágenes  
│ │ ├── 📄 logo.png # Logo principal  
│ │ └── 📄 logotipo-cunori.png  
│ │  
│ └── 📁 templates/ # Plantillas HTML  
│ ├── 📁 auth/ # Templates autenticación  
│ │ ├── 📄 login.html  
│ │ ├── 📄 register.html  
│ │ ├── 📄 reset\_password.html  
│ │ └── 📄 gestionar\_roles.html  
│ │  
│ ├── 📁 main/ # Templates principales  
│ │ ├── 📄 base.html # Template base  
│ │ ├── 📄 index.html # Dashboard  
│ │ ├── 📄 consultas.html # Consultas médicas  
│ │ ├── 📄 recepcion.html # Recepción  
│ │ └── 📄 reportes.html # Reportes  
│ │  
│ ├── 📁 pacientes/ # Templates pacientes  
│ │ ├── 📄 lista.html  
│ │ ├── 📄 formulario.html  
│ │ └── 📄 signos\_vitales.html  
│ │  
│ └── 📁 reportes/ # Templates reportes  
│ ├── 📄 index.html  
│ ├── 📄 estadisticas.html  
│ └── 📄 diario.html  
│  
├── 📁 migrations/ # Migraciones base datos  
├── 📄 config.py # Configuración aplicación  
├── 📄 run.py # Punto de entrada  
├── 📄 requirements.txt # Dependencias Python  
└── 📄 README.md # Documentación básica

## 1.7 5. MODELOS DE DATOS

### 1.7.1 5.1 Diagrama de Base de Datos

┌─────────────────┐ ┌─────────────────┐ ┌─────────────────┐  
│ Usuario │ │ Consulta │ │ Paciente │  
├─────────────────┤ ├─────────────────┤ ├─────────────────┤  
│ id (PK) │ │ id (PK) │ │ id (PK) │  
│ nombre\_completo │ │ paciente\_id (FK)│────│ nombre\_completo │  
│ usuario │ │ medico\_id (FK) │ │ edad │  
│ email │────│ clinica\_id (FK) │ │ sexo │  
│ password\_hash │ │ fecha\_consulta │ │ direccion │  
│ fecha\_registro │ │ tipo\_consulta │ │ telefono │  
│ activo │ │ estado │ │ dni │  
│ rol │ │ motivo\_consulta │ │ fecha\_nacimiento│  
│ clinica\_actual │ │ revision\_sist...│ │ religion │  
└─────────────────┘ │ gestas │ │ escolaridad │  
 │ partos │ │ ocupacion │  
┌─────────────────┐ │ abortos │ │ procedencia │  
│ Clinica │ │ hijos\_vivos │ │ numero\_expedie..│  
├─────────────────┤ │ hijos\_muertos │ └─────────────────┘  
│ id (PK) │────│ fecha\_ultima... │  
│ nombre │ │ antecedentes │ ┌─────────────────┐  
└─────────────────┘ │ presion\_arterial│ │ SignosVitales │  
 │ frecuencia\_resp │ ├─────────────────┤  
 │ temperatura │ │ id (PK) │  
 │ peso │ │ presion\_arterial│  
 │ talla │ │ frecuencia\_card │  
 │ frecuencia\_card │ │ frecuencia\_resp │  
 │ saturacion\_ox..│ │ temperatura │  
 │ imc │ │ saturacion │  
 │ diagnostico │ │ glucosa │  
 │ laboratorio │ │ fecha\_registro │  
 │ tratamiento │ │ consulta\_id (FK)│  
 │ indicaciones │────└─────────────────┘  
 └─────────────────┘

### 1.7.2 5.2 Características Especiales de Modelos

#### 1.7.2.1 Modelo Consulta - Antecedentes Gineco-Obstétricos Condicionales:

* **Control por género:** Los campos gineco-obstétricos (gestas, partos, abortos, etc.) solo se muestran y procesan para pacientes femeninos
* **Validación frontend:** JavaScript controla la visibilidad según el género del paciente
* **Validación backend:** El guardado es condicional en las rutas de Flask

#### 1.7.2.2 Sistema de Roles:

* **admin:** Acceso completo al sistema
* **medico\_supervisor:** Gestión de clínicas y reportes
* **medico:** Atención de pacientes

## 1.8 6. FUNCIONALIDADES PRINCIPALES

### 1.8.1 6.1 Módulo de Consultas Médicas

#### 1.8.1.1 Características Avanzadas:

* **Sistema de pestañas:** Organización clara de información
* **Autoguardado:** Persistencia automática cada 30 segundos
* **Búsqueda en tiempo real:** Pacientes con AJAX y debounce
* **Historial médico:** Panel lateral con consultas previas
* **Antecedentes condicionales:** Gineco-obstétricos solo para mujeres

#### 1.8.1.2 Flujo de Trabajo:

1. Buscar Paciente → 2. Seleccionar → 3. Cargar Datos → 4. Consulta  
 ↓ ↓ ↓ ↓  
 AJAX Search Load History Auto-fill Conditional Fields

### 1.8.2 6.2 Gestión de Pacientes

#### 1.8.2.1 Funcionalidades:

* **Registro completo:** Datos demográficos y socioeconómicos
* **Búsqueda avanzada:** Por nombre, DPI, edad, etc.
* **Signos vitales:** Registro y seguimiento
* **Historial médico:** Visualización cronológica

### 1.8.3 6.3 Sistema de Reportes

#### 1.8.3.1 Tipos de Reportes:

* **Diarios:** Consultas por fecha
* **Por clínica:** Estadísticas específicas
* **Pacientes:** Listas y métricas
* **PDFs:** Generación automática con ReportLab

## 1.9 7. GUÍA DE INSTALACIÓN COMPLETA

### 1.9.1 7.1 Instalación en Desarrollo

# 1. Prerrequisitos  
# Python 3.8+, Git, Node.js (opcional)  
  
# 2. Clonar repositorio  
git clone https://github.com/tu-usuario/sistema-clinico.git  
cd sistema-clinico  
  
# 3. Entorno virtual  
python -m venv venv  
  
# Windows:  
venv\Scripts\activate  
# Linux/macOS:  
source venv/bin/activate  
  
# 4. Dependencias  
pip install -r requirements.txt  
  
# 5. Variables de entorno  
cp .env.example .env  
# Editar .env con configuraciones  
  
# 6. Base de datos  
python run.py init-db  
  
# 7. Usuario admin (opcional)  
python scripts/create\_admin.py  
  
# 8. Ejecutar  
python run.py  
# Acceder a http://localhost:5000

### 1.9.2 7.2 Instalación en Producción

#### 1.9.2.1 Preparación del Servidor (Ubuntu/CentOS):

# Ubuntu/Debian  
sudo apt update  
sudo apt install python3 python3-pip python3-venv nginx mysql-server git  
  
# CentOS/RHEL  
sudo yum update  
sudo yum install python3 python3-pip nginx mysql-server git  
  
# Configurar MySQL  
sudo mysql\_secure\_installation

#### 1.9.2.2 Base de Datos:

-- MySQL Setup  
CREATE DATABASE clinica\_db CHARACTER SET utf8mb4 COLLATE utf8mb4\_unicode\_ci;  
CREATE USER 'clinica\_user'@'localhost' IDENTIFIED BY 'password\_seguro';  
GRANT ALL PRIVILEGES ON clinica\_db.\* TO 'clinica\_user'@'localhost';  
FLUSH PRIVILEGES;

#### 1.9.2.3 Configuración de Aplicación:

# 1. Usuario del sistema  
sudo useradd -m -s /bin/bash clinica  
sudo su - clinica  
  
# 2. Despliegue  
git clone https://github.com/tu-usuario/sistema-clinico.git  
cd sistema-clinico  
python3 -m venv venv  
source venv/bin/activate  
pip install -r requirements.txt  
pip install gunicorn  
  
# 3. Variables de entorno (.env)  
DATABASE\_URL=mysql://clinica\_user:password\_seguro@localhost/clinica\_db  
SECRET\_KEY=clave-super-secreta-para-produccion  
MAIL\_USERNAME=tu-email@gmail.com  
MAIL\_PASSWORD=tu-app-password  
  
# 4. Base de datos  
python run.py init-db

#### 1.9.2.4 Systemd Service:

# /etc/systemd/system/clinica.service  
[Unit]  
Description=Sistema Clinico Gunicorn  
After=network.target mysql.service  
  
[Service]  
User=clinica  
Group=clinica  
WorkingDirectory=/home/clinica/sistema-clinico  
Environment="PATH=/home/clinica/sistema-clinico/venv/bin"  
ExecStart=/home/clinica/sistema-clinico/venv/bin/gunicorn -c gunicorn\_config.py run:app  
Restart=always  
RestartSec=3  
  
[Install]  
WantedBy=multi-user.target

# Activar servicio  
sudo systemctl enable clinica.service  
sudo systemctl start clinica.service  
sudo systemctl status clinica.service

#### 1.9.2.5 Nginx Configuration:

server {  
 listen 80;  
 server\_name tu-dominio.com;  
 return 301 https://$server\_name$request\_uri;  
}  
  
server {  
 listen 443 ssl http2;  
 server\_name tu-dominio.com;  
   
 ssl\_certificate /etc/letsencrypt/live/tu-dominio.com/fullchain.pem;  
 ssl\_certificate\_key /etc/letsencrypt/live/tu-dominio.com/privkey.pem;  
   
 location / {  
 proxy\_pass http://127.0.0.1:5000;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 }  
   
 location /static/ {  
 alias /home/clinica/sistema-clinico/app/static/;  
 expires 1y;  
 add\_header Cache-Control "public, immutable";  
 }  
}

#### 1.9.2.6 SSL con Let's Encrypt:

# Instalar Certbot  
sudo apt install certbot python3-certbot-nginx  
  
# Obtener certificado  
sudo certbot --nginx -d tu-dominio.com  
  
# Renovación automática  
sudo crontab -e  
# Agregar: 0 12 \* \* \* /usr/bin/certbot renew --quiet

### 1.9.3 7.3 Configuración de Dominio y DNS

Para que la aplicación sea accesible a través del nombre de dominio clinicascunorishororagua.com en lugar de la dirección IP 154.38.160.20, es necesario configurar el Sistema de Nombres de Dominio (DNS).

#### 1.9.3.1 Paso 1: Apuntar el Dominio a la IP del Servidor (Registro A)

1. **Acceda al panel de control de su proveedor de dominio** (donde compró clinicascunorishororagua.com).
2. Navegue a la sección de **"Gestión de DNS"**, **"Editor de Zonas DNS"** o similar.
3. Cree un nuevo registro o edite el existente para el dominio principal (@).
   * **Tipo de Registro**: A
   * **Nombre/Host**: @ (esto representa el dominio raíz, es decir, clinicascunorishororagua.com)
   * **Valor/Apunta a**: 154.38.160.20
   * **TTL (Time To Live)**: Dejar el valor por defecto (normalmente 1 hora o 3600 segundos).
4. (Opcional) Cree un registro para el subdominio www:
   * **Tipo de Registro**: A o CNAME
   * **Nombre/Host**: www
   * **Valor/Apunta a**: 154.38.160.20 (si es tipo A) o clinicascunorishororagua.com (si es tipo CNAME).
5. **Guarde los cambios**. La propagación de DNS puede tardar desde unos minutos hasta 48 horas.

#### 1.9.3.2 Paso 2: Configurar el Servidor Web (Nginx)

Una vez que el DNS está configurado, el servidor web debe ser instruido para que responda a las solicitudes que llegan para ese dominio.

Modifique el archivo de configuración de Nginx (/etc/nginx/sites-available/clinica o similar):

server {  
 listen 80;  
 server\_name clinicascunorishororagua.com www.clinicascunorishororagua.com;  
  
 # Redirigir todo el tráfico HTTP a HTTPS  
 location / {  
 return 301 https://$host$request\_uri;  
 }  
}  
  
server {  
 listen 443 ssl http2;  
 # Aquí va el mismo nombre de dominio  
 server\_name clinicascunorishororagua.com www.clinicascunorishororagua.com;  
  
 # Configuración de Certificados SSL (Let's Encrypt)  
 ssl\_certificate /etc/letsencrypt/live/clinicascunorishororagua.com/fullchain.pem;  
 ssl\_certificate\_key /etc/letsencrypt/live/clinicascunorishororagua.com/privkey.pem;  
  
 location / {  
 proxy\_pass http://127.0.0.1:5000;  
 proxy\_set\_header Host $host;  
 proxy\_set\_header X-Real-IP $remote\_addr;  
 proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;  
 proxy\_set\_header X-Forwarded-Proto $scheme;  
 }  
  
 location /static {  
 alias /home/clinica/sistema-clinico/app/static;  
 expires 1y;  
 add\_header Cache-Control "public, immutable";  
 }  
}

Después de guardar los cambios, verifique la sintaxis y reinicie Nginx:

sudo nginx -t  
sudo systemctl restart nginx

Con estos pasos, cualquier usuario que escriba http://clinicascunorishororagua.com en su navegador será redirigido a la versión segura (https) y Nginx enviará la solicitud a la aplicación Flask que se ejecuta localmente.

## 1.10 8. API ENDPOINTS DETALLADOS

### 1.10.1 8.1 Endpoints de Autenticación

| Método | Endpoint | Descripción | Parámetros | Respuesta |
| --- | --- | --- | --- | --- |
| POST | /auth/login | Iniciar sesión | usuario, password, remember\_me | JSON con success/redirect |
| POST | /auth/register | Registrar usuario | nombre\_completo, usuario, email, password | JSON con success/error |
| GET | /auth/logout | Cerrar sesión | - | Redirect a login |
| POST | /auth/reset\_password\_request | Solicitar reset | email | JSON con success |
| POST | /auth/reset\_password/<token> | Reset con token | password, password2 | JSON con success |

### 1.10.2 8.2 Endpoints de Pacientes

| Método | Endpoint | Descripción | Rol Requerido | Parámetros |
| --- | --- | --- | --- | --- |
| GET | /pacientes/ | Listar pacientes | medico, admin | page, search |
| POST | /pacientes/nuevo | Crear paciente | medico, admin | Datos del paciente |
| GET | /pacientes/<id> | Ver paciente | medico, admin | - |
| PUT | /pacientes/<id>/editar | Editar paciente | medico, admin | Datos modificados |
| GET | /api/search\_patients | Buscar AJAX | medico, admin | q (query) |

#### 1.10.2.1 Ejemplo Respuesta Búsqueda:

{  
 "patients": [  
 {  
 "id": 1,  
 "nombre\_completo": "María García López",  
 "edad": 35,  
 "genero": "Femenino",  
 "dni": "1234567890123",  
 "telefono": "12345678",  
 "historial\_consultas": [  
 {  
 "id": 15,  
 "fecha\_consulta": "2025-01-15",  
 "diagnostico": "Hipertensión arterial",  
 "medico\_nombre": "Dr. Juan Pérez"  
 }  
 ]  
 }  
 ],  
 "total": 1  
}

### 1.10.3 8.3 Endpoints de Consultas

| Método | Endpoint | Descripción | Parámetros Especiales |
| --- | --- | --- | --- |
| POST | /consulta/<id>/motivo | Guardar motivo | Campos condicionales por género |
| POST | /consulta/<id>/diagnostico | Guardar diagnóstico | diagnostico, laboratorio |
| GET | /get\_patient\_history/<id> | Historial AJAX | Incluye signos vitales |

#### 1.10.3.1 Ejemplo Guardar Motivo (Paciente Femenino):

{  
 "motivo\_consulta": "Control prenatal",  
 "historia\_enfermedad": "Embarazo de 20 semanas...",  
 "revision\_sistemas": "Cardiovascular: Normal",  
 "gestas": "2",  
 "partos": "1",   
 "abortos": "0",  
 "hijos\_vivos": "1",  
 "hijos\_muertos": "0",  
 "fecha\_ultima\_regla": "2024-09-01",  
 "antecedentes": "Diabetes gestacional previa",  
 "presion\_arterial\_examen": "120/80",  
 "peso": "65",  
 "talla": "165"  
}

## 1.11 9. SEGURIDAD AVANZADA

### 1.11.1 9.1 Medidas de Seguridad Implementadas

#### 1.11.1.1 Autenticación Robusta:

# Hash seguro de contraseñas  
def set\_password(self, password):  
 self.password\_hash = generate\_password\_hash(password)  
  
# Verificación con protección contra timing attacks  
def check\_password(self, password):  
 return check\_password\_hash(self.password\_hash, password)  
  
# Tokens seguros para reset  
def get\_reset\_password\_token(self, expires\_in=1800):  
 s = URLSafeTimedSerializer(current\_app.config['SECRET\_KEY'])  
 return s.dumps({'reset\_password': self.id}, salt='password-reset-salt')

#### 1.11.1.2 Protección CSRF:

# Automática en todos los formularios  
csrf = CSRFProtect()  
csrf.init\_app(app)  
  
# En templates  
<input type="hidden" name="csrf\_token" value="{{ csrf\_token() }}"/>

#### 1.11.1.3 Control de Acceso por Roles:

def role\_required(\*roles):  
 def decorator(f):  
 @wraps(f)  
 def decorated\_function(\*args, \*\*kwargs):  
 if not current\_user.is\_authenticated:  
 return redirect(url\_for('auth.login'))  
 if current\_user.rol not in roles:  
 flash('Sin permisos para esta acción.', 'error')  
 return redirect(url\_for('main.index'))  
 return f(\*args, \*\*kwargs)  
 return decorated\_function  
 return decorator

### 1.11.2 9.2 Headers de Seguridad:

@app.after\_request  
def set\_security\_headers(response):  
 response.headers['X-Content-Type-Options'] = 'nosniff'  
 response.headers['X-Frame-Options'] = 'DENY'  
 response.headers['X-XSS-Protection'] = '1; mode=block'  
 response.headers['Strict-Transport-Security'] = 'max-age=31536000; includeSubDomains'  
 return response

### 1.11.3 9.3 Auditoría y Logging:

import logging  
  
# Configuración de logging  
logging.basicConfig(  
 filename='app.log',  
 level=logging.INFO,  
 format='%(asctime)s %(levelname)s %(name)s %(message)s'  
)  
  
# Logs de acceso a datos sensibles  
@bp.route('/pacientes/<int:id>')  
def ver\_paciente(id):  
 current\_app.logger.info(f'Usuario {current\_user.usuario} accedió a paciente {id}')  
 # ... función

## 1.12 10. MANTENIMIENTO Y MONITOREO

### 1.12.1 10.1 Scripts de Mantenimiento

#### 1.12.1.1 Backup Automático:

#!/bin/bash  
# backup\_db.sh  
  
DATE=$(date +%Y%m%d\_%H%M%S)  
BACKUP\_DIR="/backups/clinica"  
DB\_NAME="clinica\_db"  
  
mkdir -p $BACKUP\_DIR  
  
# MySQL Backup  
mysqldump -u usuario -p$PASSWORD $DB\_NAME > $BACKUP\_DIR/clinica\_$DATE.sql  
gzip $BACKUP\_DIR/clinica\_$DATE.sql  
  
# Limpieza de backups antiguos  
find $BACKUP\_DIR -name "\*.sql.gz" -mtime +30 -delete  
  
echo "Backup completado: clinica\_$DATE.sql.gz"

#### 1.12.1.2 Mantenimiento de Base de Datos:

-- Ejecutar semanalmente  
OPTIMIZE TABLE paciente;  
OPTIMIZE TABLE consulta;  
OPTIMIZE TABLE signos\_vitales;  
OPTIMIZE TABLE usuario;  
  
-- Verificar índices  
SHOW INDEX FROM consulta;

#### 1.12.1.3 Script de Limpieza:

# cleanup.py  
from datetime import datetime, timedelta  
from app.models import \*  
  
def cleanup\_old\_sessions():  
 """Eliminar sesiones expiradas"""  
 old\_date = datetime.utcnow() - timedelta(days=30)  
 # Lógica de limpieza  
  
def optimize\_database():  
 """Optimizar tablas"""  
 db.engine.execute("OPTIMIZE TABLE consulta")  
 db.engine.execute("OPTIMIZE TABLE paciente")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 cleanup\_old\_sessions()  
 optimize\_database()

### 1.12.2 10.2 Monitoreo del Sistema

#### 1.12.2.1 Health Check Endpoint:

@bp.route('/health')  
def health\_check():  
 try:  
 # Verificar base de datos  
 db.session.execute('SELECT 1')  
   
 return jsonify({  
 'status': 'healthy',  
 'timestamp': datetime.utcnow().isoformat(),  
 'version': '1.0.0',  
 'database': 'connected'  
 }), 200  
 except Exception as e:  
 return jsonify({  
 'status': 'unhealthy',  
 'error': str(e)  
 }), 500

#### 1.12.2.2 Métricas de Sistema:

* **Uptime:** >99.5%
* **Tiempo de respuesta:** <2 segundos
* **CPU:** <80% promedio
* **Memoria:** <85%
* **Disco:** >20% libre

### 1.12.3 10.3 Troubleshooting Común

#### 1.12.3.1 Problemas Frecuentes:

**1. Error de Conexión BD:**

sudo systemctl status mysql  
sudo tail -f /var/log/mysql/error.log  
sudo systemctl restart mysql

**2. Error 500 Aplicación:**

tail -f /home/clinica/sistema-clinica/logs/clinica.log  
sudo systemctl status clinica  
sudo nginx -t

**3. Alto Uso de Memoria:**

top -o %MEM  
mysql -e "SHOW PROCESSLIST;"  
sudo systemctl restart clinica

## 1.13 11. ESCALABILIDAD Y FUTURO

### 1.13.1 11.1 Optimizaciones de Rendimiento

#### 1.13.1.1 Paginación Eficiente:

@bp.route('/pacientes')  
def lista\_pacientes():  
 page = request.args.get('page', 1, type=int)  
 pacientes = Paciente.query.paginate(  
 page=page, per\_page=20, error\_out=False  
 )  
 return render\_template('pacientes/lista.html', pacientes=pacientes)

#### 1.13.1.2 Índices de Base de Datos:

# migrations/add\_indexes.py  
def upgrade():  
 op.create\_index('idx\_paciente\_dni', 'paciente', ['dni'])  
 op.create\_index('idx\_consulta\_fecha', 'consulta', ['fecha\_consulta'])  
 op.create\_index('idx\_consulta\_paciente', 'consulta', ['paciente\_id'])

### 1.13.2 11.2 Arquitectura Escalable

#### 1.13.2.1 Load Balancing:

upstream clinica\_backend {  
 server 127.0.0.1:5000;  
 server 127.0.0.1:5001;  
 server 127.0.0.1:5002;  
}  
  
server {  
 location / {  
 proxy\_pass http://clinica\_backend;  
 }  
}

#### 1.13.2.2 Cache con Redis:

import redis  
from flask import current\_app  
  
redis\_client = redis.Redis(host='localhost', port=6379, db=0)  
  
def cache\_patient\_data(patient\_id, data):  
 key = f"patient:{patient\_id}"  
 redis\_client.setex(key, 3600, json.dumps(data)) # 1 hora  
  
def get\_cached\_patient\_data(patient\_id):  
 key = f"patient:{patient\_id}"  
 cached = redis\_client.get(key)  
 return json.loads(cached) if cached else None

### 1.13.3 11.3 Roadmap de Desarrollo

#### 1.13.3.1 Próximas Funcionalidades:

1. **Módulo de Inventario:** Medicamentos y suministros médicos
2. **Telemedicina:** Consultas virtuales con videollamada
3. **API REST Completa:** Para integración con sistemas externos
4. **Aplicación Móvil:** Para médicos y pacientes
5. **Inteligencia Artificial:** Asistente de diagnóstico
6. **Integración DICOM:** Para manejo de imágenes médicas
7. **Analytics Avanzado:** Dashboard con métricas en tiempo real

#### 1.13.3.2 Tecnologías Futuras:

* **FastAPI:** Para API de alto rendimiento
* **React/Vue.js:** Frontend moderno
* **Docker:** Containerización
* **Kubernetes:** Orquestación
* **GraphQL:** API flexible
* **Machine Learning:** Predicción de diagnósticos

## 1.14 12. CONSIDERACIONES ESPECIALES

### 1.14.1 12.1 Funcionalidad Gineco-Obstétrica Condicional

Esta es una característica única del sistema que merece explicación detallada:

#### 1.14.1.1 Implementación Frontend:

// Control dinámico de visibilidad  
function toggleAntecedentesSection(showForFemale) {  
 const antecedentesSection = document.getElementById('antecedentes-section');  
 if (antecedentesSection) {  
 antecedentesSection.style.display = showForFemale ? 'block' : 'none';  
 }  
}  
  
// Llamada automática al seleccionar paciente  
function updateGeneralDataFields(patient) {  
 // ... otros campos  
 toggleAntecedentesSection(patient.genero === 'Femenino');  
}

#### 1.14.1.2 Implementación Backend:

# Guardado condicional en routes.py  
if 'guardar\_motivo\_consulta' in request.form:  
 # ... campos comunes  
   
 # Solo para pacientes femeninos  
 if paciente.sexo == 'Femenino':  
 consulta.gestas = request.form.get('gestas')  
 consulta.partos = request.form.get('partos')  
 consulta.abortos = request.form.get('abortos')  
 consulta.hijos\_vivos = request.form.get('hijos\_vivos')  
 consulta.hijos\_muertos = request.form.get('hijos\_muertos')  
 consulta.fecha\_ultima\_regla = request.form.get('fecha\_ultima\_regla')

#### 1.14.1.3 Template Condicional:

<!-- Sección visible solo para pacientes femeninos -->  
<div class="mb-3" id="antecedentes-section" style="display: none;">  
 <label class="form-label">Antecedentes</label>  
 <div class="border p-3">  
 <h5 class="mb-3">  
 <span class="border-bottom border-3 border-primary">Gineco-Obstetricos</span>  
 <span class="ms-2">Fisiológicos</span>  
 <span class="ms-2">Patológicos</span>  
 <!-- ... más categorías -->  
 </h5>  
 <div class="row g-3 mb-3" id="antecedentes-gineco-section">  
 <!-- Campos específicos para mujeres -->  
 <div class="col-md-2">  
 <label class="form-label">Gestas</label>  
 <input type="text" class="form-control" name="gestas">  
 </div>  
 <!-- ... más campos -->  
 </div>  
 </div>  
</div>

### 1.14.2 12.2 Autoguardado Inteligente

#### 1.14.2.1 Implementación Completa:

// Sistema de autoguardado cada 30 segundos  
let autoSaveInterval;  
  
function initAutoSave() {  
 if (autoSaveInterval) {  
 clearInterval(autoSaveInterval);  
 }  
   
 autoSaveInterval = setInterval(() => {  
 if (window.currentPatient) {  
 saveFormData();  
 }  
 }, 30000);  
}  
  
function saveFormData() {  
 const formData = {};  
 const allFields = document.querySelectorAll('input, textarea, select');  
   
 allFields.forEach(field => {  
 if (field.name && field.name !== 'csrf\_token') {  
 if (field.type === 'checkbox') {  
 formData[field.name] = field.checked;  
 } else {  
 formData[field.name] = field.value;  
 }  
 }  
 });  
   
 // Guardar en localStorage  
 const storageKey = `consulta\_${window.currentPatient.id}`;  
 localStorage.setItem(storageKey, JSON.stringify(formData));  
   
 showAutosaveIndicator();  
}  
  
function showAutosaveIndicator() {  
 const indicator = document.getElementById('autosave-indicator');  
 if (indicator) {  
 indicator.style.display = 'block';  
 setTimeout(() => {  
 indicator.style.display = 'none';  
 }, 2000);  
 }  
}

## 1.15 13. CONCLUSIONES

### 1.15.1 13.1 Logros del Sistema

Este Sistema de Gestión Clínica representa una solución completa y moderna para la digitalización de procesos médicos en entornos universitarios. Los principales logros incluyen:

* **Arquitectura Sólida:** MVC con Flask, escalable y mantenible
* **Seguridad Robusta:** Autenticación, autorización y protección de datos
* **Experiencia de Usuario:** Interfaz intuitiva con tecnologías modernas
* **Funcionalidades Especiales:** Antecedentes condicionales por género
* **Documentación Completa:** Facilita mantenimiento y desarrollo futuro

### 1.15.2 13.2 Impacto Esperado

* **Eficiencia:** Reducción del 70% en tiempo de registro
* **Precisión:** Eliminación de errores de transcripción
* **Trazabilidad:** Historial médico completo y accesible
* **Reportes:** Análisis estadístico para mejor toma de decisiones
* **Escalabilidad:** Capacidad para crecer con las necesidades

### 1.15.3 13.3 Recomendaciones de Implementación

1. **Fase Piloto:** Implementar en una clínica pequeña primero
2. **Capacitación:** Entrenar exhaustivamente a usuarios finales
3. **Migración Gradual:** Mantener sistemas paralelos temporalmente
4. **Monitoreo Continuo:** Supervisar rendimiento y usabilidad
5. **Feedback Loop:** Incorporar sugerencias de usuarios

## 1.16 14. INFORMACIÓN DE CONTACTO

### 1.16.1 14.1 Equipo de Desarrollo

* **Desarrollador Principal:** [Tu Nombre]
* **Email:** [tu-email@ejemplo.com](mailto:tu-email@ejemplo.com)
* **Institución:** Universidad de San Carlos de Guatemala
* **Facultad:** Ciencias Médicas

### 1.16.2 14.2 Repositorio y Documentación

* **GitHub:** <https://github.com/tu-usuario/sistema-clinico>
* **Documentación Online:** <https://docs.tu-dominio.com>
* **Issues y Support:** <https://github.com/tu-usuario/sistema-clinico/issues>

### 1.16.3 14.3 Licencia

Este proyecto está licenciado bajo la Licencia MIT. Ver el archivo LICENSE para más detalles.

**© 2025 Universidad de San Carlos de Guatemala - Facultad de Ciencias Médicas**

*Documentación técnica completa generada el: $(date '+%Y-%m-%d %H:%M:%S')*

**NOTA:** Esta documentación es un documento vivo que debe actualizarse con cada versión del sistema. Para reportar errores en la documentación o sugerir mejoras, por favor crear un issue en el repositorio del proyecto.