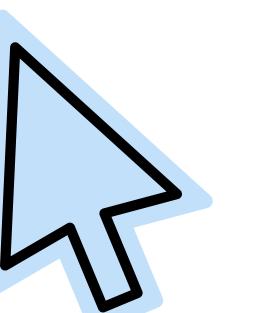


# SISTEMA MEDICO

Solucion integral administrativa

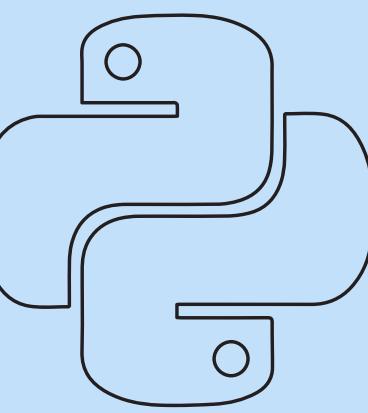
Integrantes:

Diego Enrique Arista  
Juan Vargas Rocha  
Job Yauri Leon



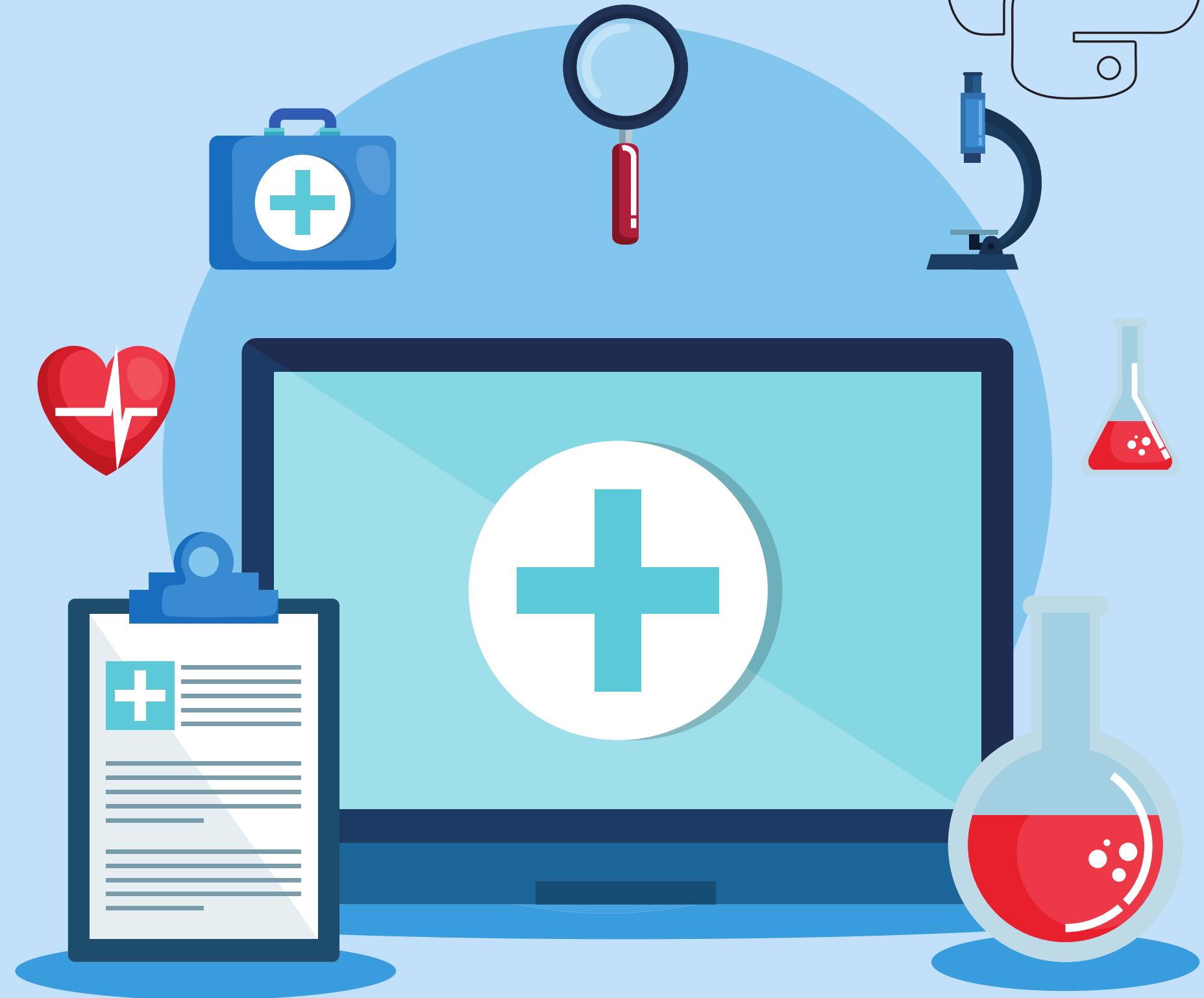
# CONTENIDOS

- 01** Introducción y contexto
- 02** Objetivos del proyecto
- 03** Diseño de la interfaz
- 04** Diagrama de Clases
- 05** Arquitectura del Sistema
- 06** Código Relevante

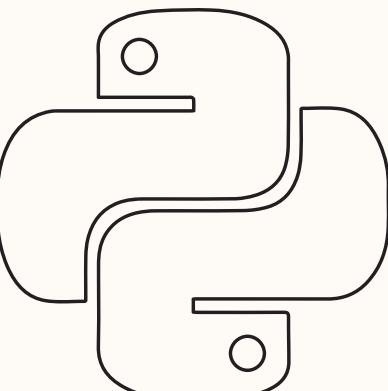


# CONTENIDOS

- 07** Manejo de Archivos
- 08** Beneficios del Sistema
- 09** Conclusión



# INTRODUCCIÓN Y CONTEXTO



## PROBLEMÁTICA

Actualmente, muchos centros médicos pequeños y medianos aún usan registros en papel o sistemas poco organizados.

Esto genera problemas como pérdida de información, duplicación de datos y demoras en la atención.

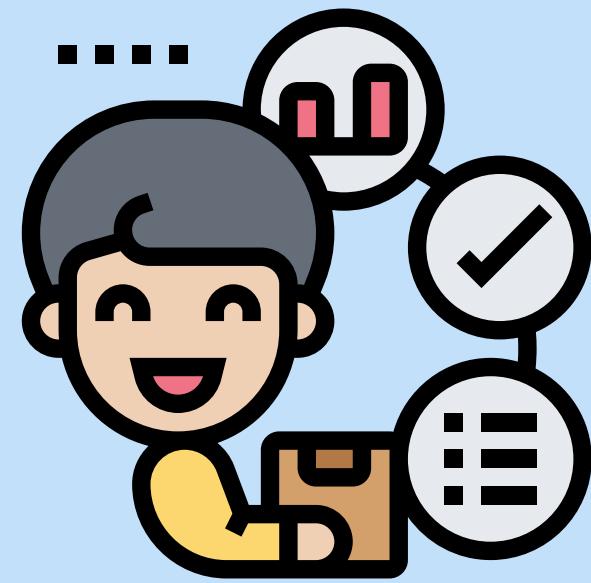
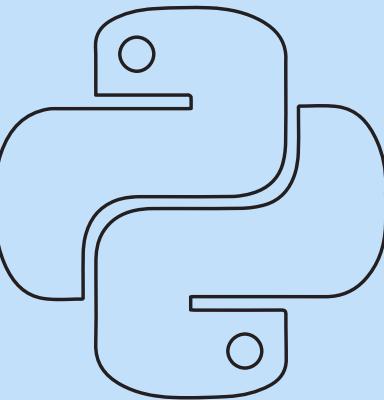
La creciente demanda de atención rápida y organizada obliga a implementar soluciones digitales.

Con la llegada de tecnologías como Python, PyQt5 y manejo de datos en JSON, es posible crear sistemas eficientes sin requerir grandes inversiones en infraestructura.

El objetivo es centralizar y agilizar la gestión de pacientes, doctores y citas en un solo sistema fácil de usar.



# OBJETIVOS DEL PROYECTO



## CALIDAD

- Desarrollar un sistema de gestión médica integral y fácil de usar, que optimice la administración de pacientes, doctores y citas, mejorando la eficiencia y la calidad del servicio.



## ORDEN

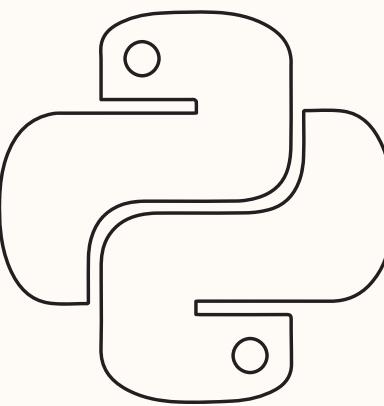
- Gestionar pacientes, doctores y citas de forma rápida y ordenada, permitiendo registrar, modificar y eliminar información



## CONFIABILIDAD

- Garantizar la seguridad y confiabilidad de los datos, utilizando almacenamiento estructurado y controlado.

# DISEÑO DE LA INTERFAZ



## VISTA PRINCIPAL

- Interfaz creada con **PyQt5**, clara y fácil de usar.

- Menú principal con accesos directos a cada módulo.

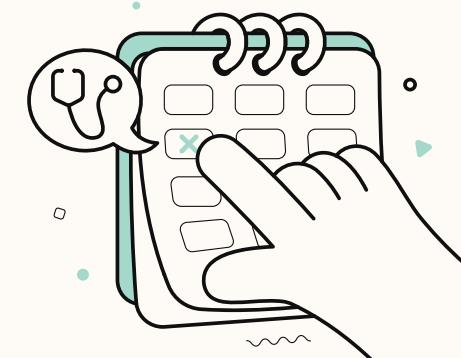
- Botones y campos organizados para navegación rápida.

## MODULOS



- **Pacientes:** registro y consulta de datos.

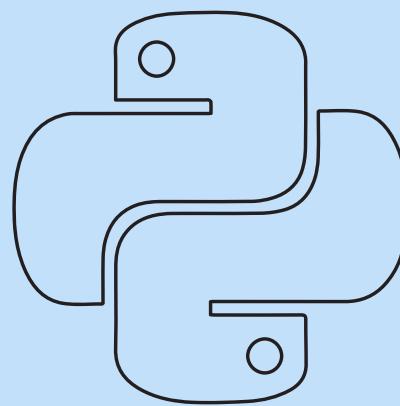
- **Doctores:** gestión de información profesional.



- **Citas:** programación y control de horarios.



# DISEÑO DE FORMULARIO



## Registro de pacientes

**Registro de Paciente**

Nombre:	Ej. Juan Pérez
DNI:	Ej. 12345678
Fecha de nacimiento:	dd-mm-aaaa

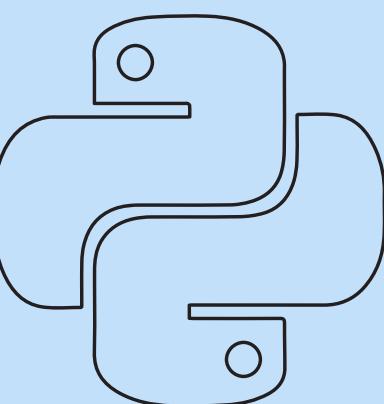
**Guardar**

## Lista de Pacientes

**Lista de Pacientes Registrados**

👤 ♂	Nombre: job   DNI: 61964437   Nacimiento: 17-05-2002
👤 ♂	Nombre: juan   DNI: 61964438   Nacimiento: 10-08-2004
👤 ♂	Nombre: joan   DNI: 52464635   Nacimiento: 10-02-2000

**Eliminar Paciente**



# DISEÑO DE LISTAS

## Registro de doctores

### Registro de Doctor

Nombre:

Ej. Dr. Carlos Ramos

Especialidad:

Ej. Cardiología

Guardar

## Lista de Doctores

### Doctores Registrados

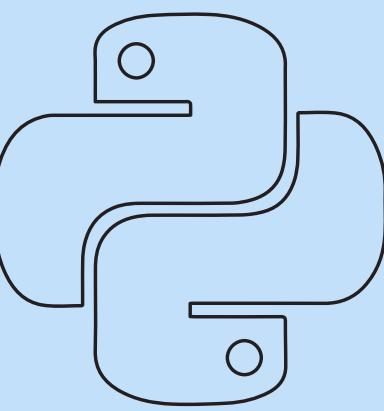
👤 Nombre: daniel | Especialidad: medicina general

👤 Nombre: Franco | Especialidad: Cirujano Plastico

👤 Nombre: Jonathan Rodriguez | Especialidad: Cardiologo

Eliminar Doctor

# DISEÑO DE CITAS



## Agendar Cita Medica

### Agendar Cita Médica

Paciente: Ej. Juan Pérez

Fecha: dd-mm-aaaa

Hora: hh:mm

Motivo: Motivo de la cita

**Agendar**

## Cita Agendadas

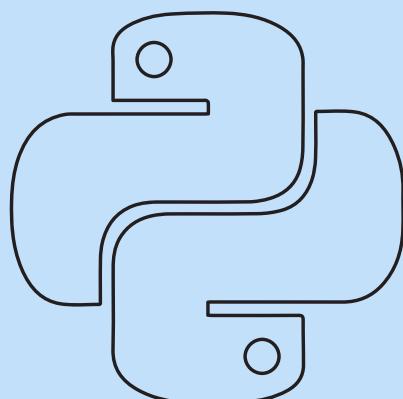
### Citas Agendadas

Paciente: job  
Fecha: 11-07-2025 Hora: 10:00  
Motivo: dolor de garganta

Paciente: Paola Chavez  
Fecha: 24-07-2025 Hora: 10:00

**Eliminar Cita**

# DISEÑO DE CONSULTAS E HISTORIAL



## Consulta Medica

**Nueva Consulta Médica**

Paciente:	Ej. Juan Pérez
Doctor:	Ej. Dra. Flores
Fecha:	dd-mm-aaaa
Síntomas:	Síntomas del paciente
Tratamiento:	Tratamiento recomendado

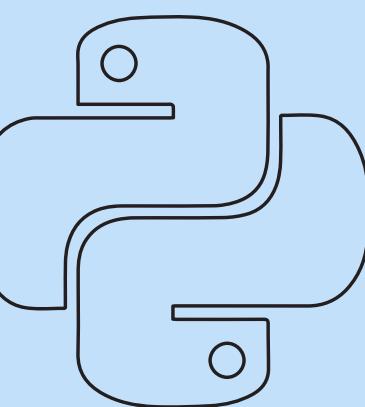
**Guardar Consulta**

## Historial Medico

**Historial Médico del Paciente**

61964437	<b>Buscar historial</b>
<p>📅 11-07-2025   Doctor: daniel 🕒 Síntomas: dolor de garganta 💊 Tratamiento: antiinflamatorio y panadol</p>	

**Exportar a Excel**



# DISEÑO DE NOTIFICACIONES Y CONFIG.

## Notificaciones

**Centro de Notificaciones**

Paciente registrado: Cesar Daniel Emilio

Doctor registrado: Octavio Morales

**Eliminar**

## Configuracion

**Configuración del sistema**

Opciones de configuración

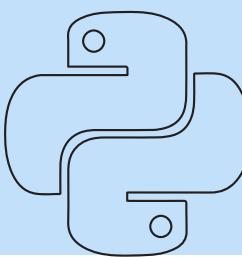
**Tema** Claro

**Notificaciones**  Activar notificaciones

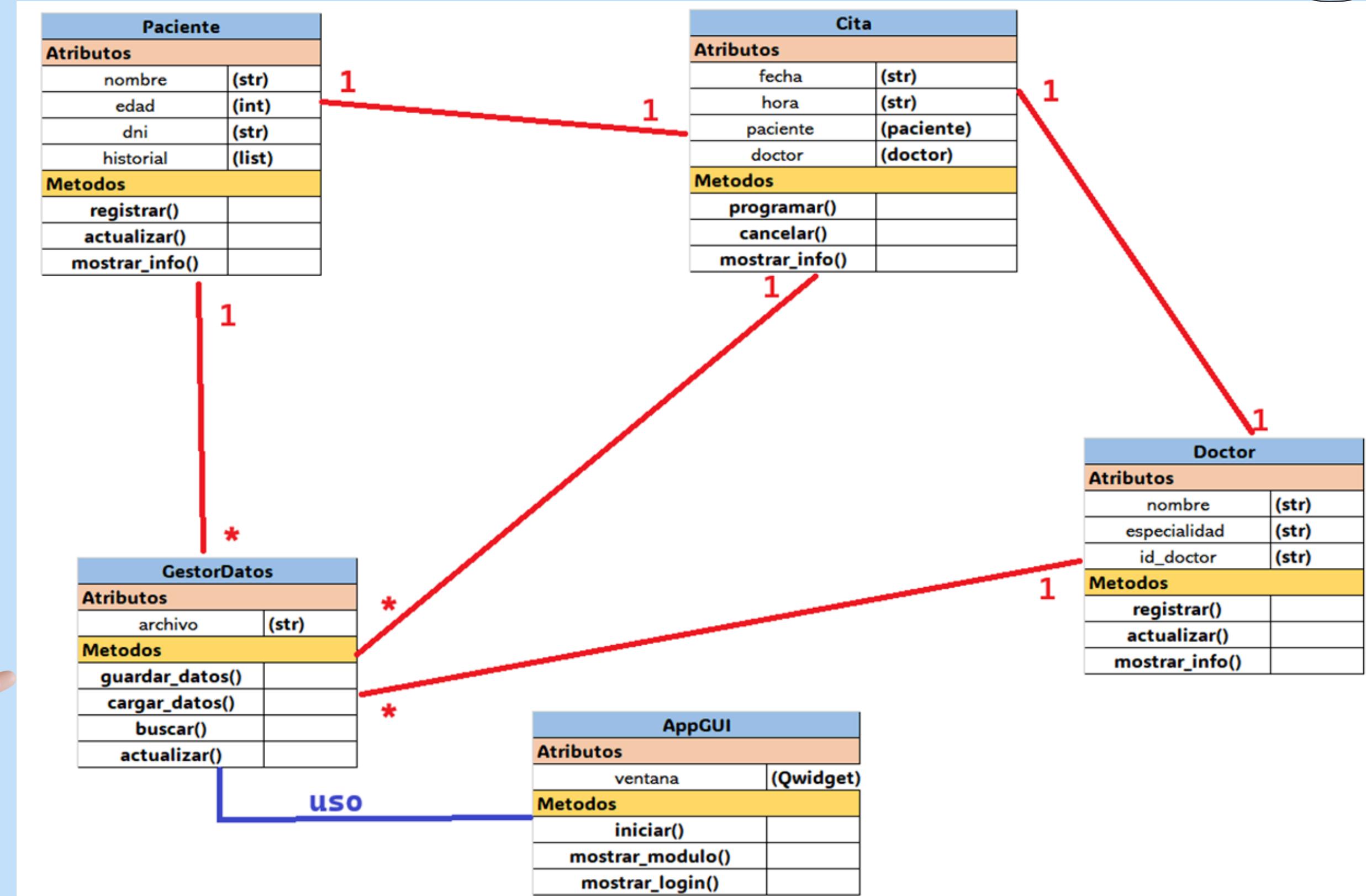
**Guardar configuración**

# DISEÑO DE LOGIN





# DIAGRAMA DE CLASES



# ARQUITECTURA DEL SISTEMA

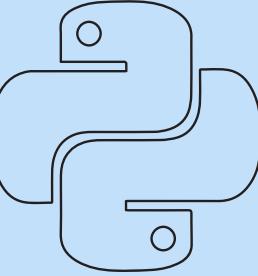
- Estructura de carpetas y archivos :
- /config
  - config.json
  - notificaciones.json
  - temas.py
- /form
  - form\_cita.py
  - form\_consulta.py
  - form\_doctor.py
  - form\_paciente.py
  - login\_window.py
- /historial
  - Exportaciones.xlsx
- /resources
  - Principal.jpg

/views  
Configuracion.py  
Historial\_medico.py  
Lista\_cita.py  
Lista\_doctores.py  
Lista\_pacientes.py  
Main\_window.py  
Notificaciones.py

app\_gui.py à Principal ejecución

confi.py  
data\_manager.py  
datos.json





# CODIGO RELEVANTE



## APP\_GUI.PY

```
app_gui.py x
app_gui.py > cargar_tema
1 import os
2 import sys
3 import json
4 from config.temas import TEMA_OSCURO, TEMA_CLARO
5 from PyQt5.QtWidgets import QApplication
6 from data_manager import cargar_datos
7 from forms.login_window import LoginWindow
8 from views.main_window import MainWindow

9
10 # Variables globales para los datos
11 pacientes, doctores, citas = cargar_datos()

12 # Referencias globales para mantener vivas las ventanas
13 app = None
14 ventana_principal = None
15 ventana_login = None

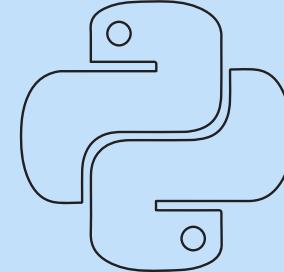
16
17 def iniciar_principal():
18     global ventana_principal
19     ventana_principal = MainWindow(on_logout=iniciar_login)
20     ventana_principal.show()

21
22 def iniciar_login():
23     global ventana_login
24     ventana_login = LoginWindow(on_login_success=iniciar_principal)
25     ventana_login.show()

26
27 CONFIG_FILE = "config/config.json"

28
29 def cargar_tema():
30     if os.path.exists(CONFIG_FILE):
31         with open(CONFIG_FILE, "r") as f:
32             config = json.load(f)
33             if config.get("tema") == "Oscurito":
34                 return TEMA_OSCURO
35             return TEMA_CLARO

36
37
38 if __name__ == "__main__":
39     app = QApplication(sys.argv)
40     app.setStyleSheet(cargar_tema())
41     iniciar_login()
42     sys.exit(app.exec_())
```



# CODIGO RELEVANTE

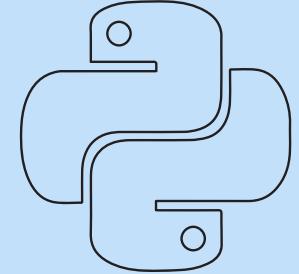


## DATA\_MANAGER.PY

```
# data_manager.py
import json
import os

DATA_FILE = "datos.json"

def guardar_datos(pacientes, doctores, citas):
    data = {
        "pacientes": [
            {
                "nombre": p.nombre,
                "dni": p._Paciente_dni,
                "fecha_nacimiento": p._Paciente_fecha_nacimiento,
                "historial": [
                    {
                        "fecha": c.fecha,
                        "doctor": c.doctor.get_nombre(),
                        "sintomas": c.sintomas,
                        "tratamiento": c.tratamiento
                    } for c in p._Paciente_historial
                ]
            } for p in pacientes
        ],
        "doctores": [
            {
                "nombre": d.get_nombre(),
                "especialidad": d._Doctor_especialidad
            } for d in doctores
        ],
        "citas": [
            {
                "paciente": c.paciente.nombre,
```



# CODIGO RELEVANTE



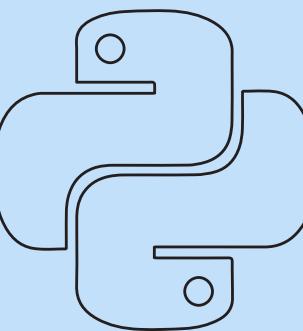
## LOGIN\_WINDOW.PY

```
class LoginWindow(QWidget):
    def __init__(self, on_login_success=None):
        super().__init__()
        self.setWindowTitle("Iniciar Sesión")
        self.resize(400, 250)
        self.centrar_ventana()
        self.on_login_success = on_login_success

    def centrar_ventana(self):
        qr = self.frameGeometry()
        cp = self.screen().availableGeometry().center()
        qr.moveCenter(cp)
        self.move(qr.topLeft())

    def verificar_login(self):
        usuario = self.user_input.text()
        clave = self.password_input.text()

        # Centro de credenciales
        if usuario == "yauri" and clave == "admin123":
            QMessageBox.information(self, "Éxito", "Inicio de sesión exitoso.")
            self.close()
            if self.on_login_success:
                self.on_login_success()
        else:
            QMessageBox.warning(self, "Error", "Usuario o contraseña incorrectos.")
```



# MANEJO DE ARCHIVOS

ESTE MÓDULO SE ENCARGA DE GUARDAR Y CARGAR TODA LA INFORMACIÓN DEL SISTEMA EN UN ARCHIVO DATOS.JSON, ASEGURANDO QUE LOS DATOS DE PACIENTES, DOCTORES Y CITAS SE MANTENGAN DESPUÉS DE CERRAR EL PROGRAMA.

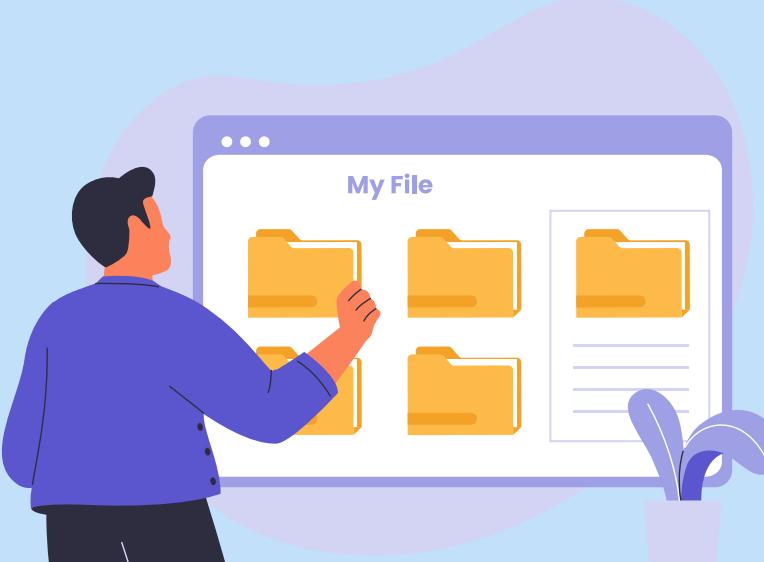
```
# data_manager.py
import json
import os

DATA_FILE = "datos.json"
```

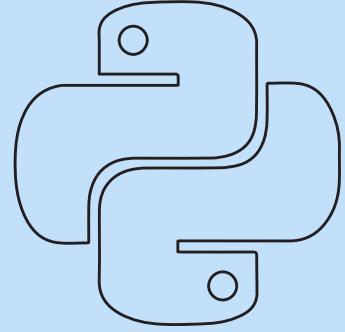
DEFINE EL NOMBRE DEL ARCHIVO DONDE SE ALMACENARÁ LA INFORMACIÓN.

```
def guardar_datos(pacientes, doctores, citas):
    data = {
```

- RECIBE LISTAS DE PACIENTES, DOCTORES Y CITAS.
- CONVIERTE LOS OBJETOS EN DICCIONARIOS LISTOS PARA GUARDAR EN FORMATO JSON.
- USA LISTAS POR COMPRENSIÓN PARA EXTRAER:
  - PACIENTES → NOMBRE, DNI, FECHA DE NACIMIENTO Y SU HISTORIAL MÉDICO (CONSULTAS PREVIAS).
  - DOCTORES → NOMBRE Y ESPECIALIDAD.
  - CITAS → PACIENTE, FECHA, HORA Y MOTIVO.



# BENEFICIOS DEL SISTEMA



Centralización de la información

Historial médico organizado

Reducción de errores humanos

Reportes personalizados

Ahorro de tiempo

Seguridad de datos

Accesibilidad

Escalabilidad

# CONCLUSIÓN

EL SISTEMA INTEGRA LA GESTIÓN DE PACIENTES, DOCTORES Y CITAS EN UNA INTERFAZ PYQT5 INTUITIVA, CON ALMACENAMIENTO EN JSON PARA DATOS SEGUROS Y PORTABLES, REDUCIENDO TIEMPOS Y ERRORES, Y OFRECIENDO ESCALABILIDAD PARA FUTURAS MEJORAS.

