

Mockups y Estructura MediCare Desk



Cada dosis, una muestra de cuidado

Presentado por:

Yamid Alfonso Gonzalez Torres

Jenny Catherine Herrera Garzon

Edwin Andres Marin Vanegas

Diego Steven Pinzon Yossa

Profesor:

Oscar Eduardo Alvarez Rodriguez

Universidad Nacional de Colombia

Facultad de Ingeniería

Ingeniería de software

2025



1. Prom

Pantalla de inicio de sesión

- Campos: Correo electrónico y contraseña
- Botones: Iniciar sesión, Olvidé mi contraseña, Registrarse como cuidador

MediCareDesk

Correo Electronico

Password

¿Olvidaste tu contraseña?

Login

Registrarse como cuidador

Pantalla de registro de cuidador

- Formulario con: Nombre, Correo, Teléfono, Relación con el paciente, Contraseña, Confirmar contraseña
- Casilla: Acepto términos
- Botón: Registrarse





Registro de cuidador

Nombre Completo

Relacion con el paciente

Telefono (opcional)

Correo

Contraseña

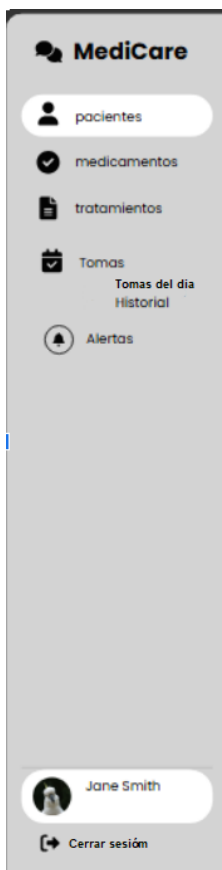
Confirmar contraseña

☐ Acepto terminos y condiciones

registrarse

Ya estas registrado? [ingresa aqui](#)

Menú de navegación



Botones que llevan a las vistas de pacientes, medicamentos, tratamientos, tomas (tomas del día e historial) y alertas.

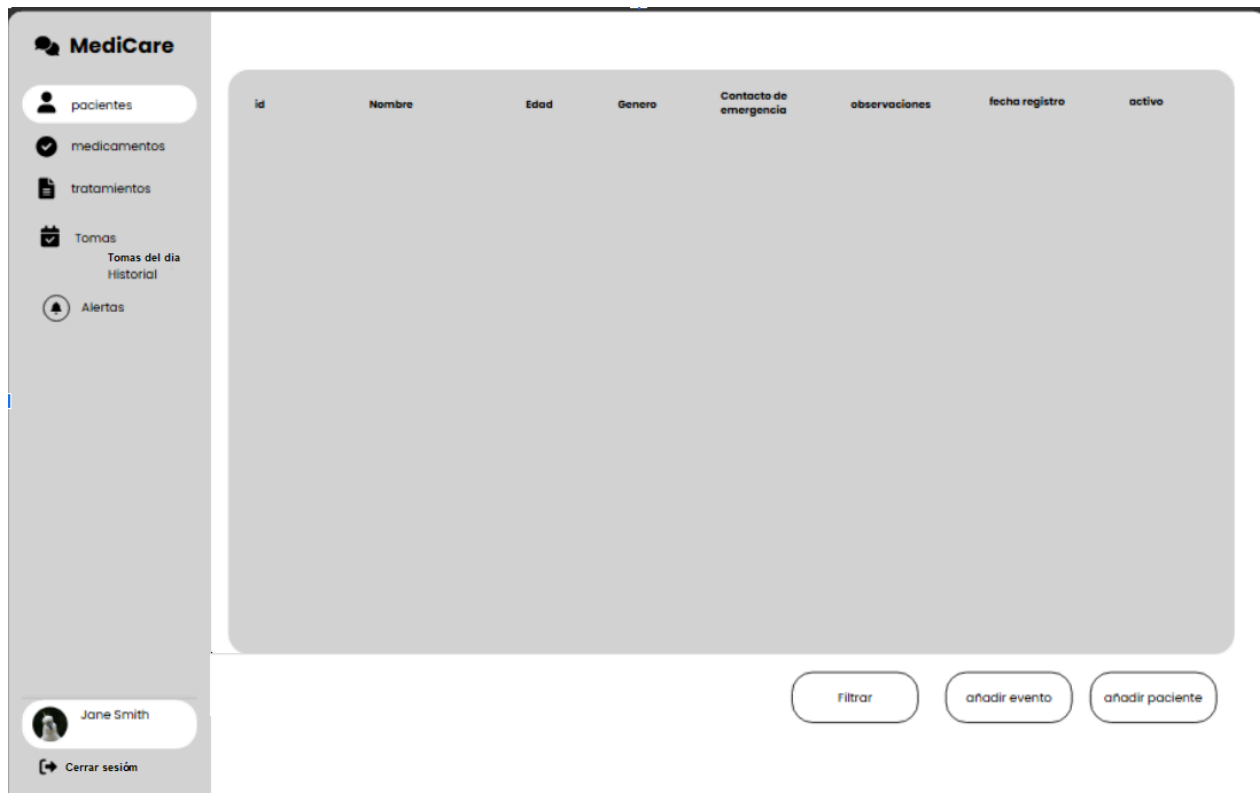
En la parte inferior va el nombre del usuario y el botón para cerrar sesión.

Notas: Mantener la letra mayúscula al inicio de cada palabra.

Pantalla de pacientes:

Cuenta con un formulario y 3 botones Filtrar, Añadir evento y Añadir paciente

Esta vista (pacientes) cuenta con el formulario para llenar la tabla de pacientes, para añadir paciente, el formulario debe tener permanentemente un espacio (en blanco) para completar la información, en la siguiente vista hay un pequeño ejemplo.



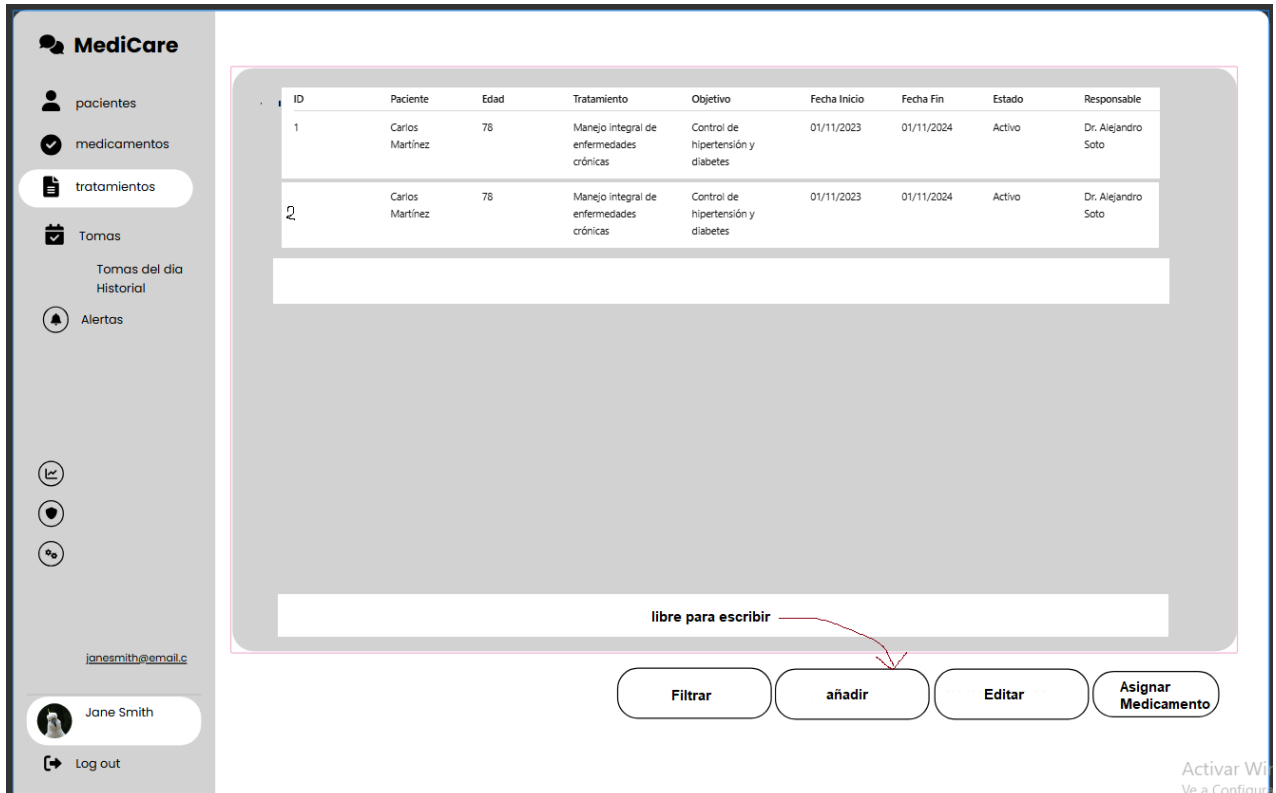
The screenshot displays the 'MediCare' application interface. On the left is a sidebar with navigation options: 'pacientes' (selected), 'medicamentos', 'tratamientos', 'Tomas' (with sub-options 'Tomas del día' and 'Historial'), and 'Alertas'. At the bottom of the sidebar, there is a user profile for 'Jane Smith' and a 'Cerrar sesión' button. The main area features a table with the following headers: 'id', 'Nombre', 'Edad', 'Genero', 'Contacto de emergencia', 'observaciones', 'fecha registro', and 'activo'. The table body is currently empty. At the bottom right of the main area, there are three buttons: 'Filtrar', 'añadir evento', and 'añadir paciente'.

Al pulsar añadir eventos se debe abrir una **ventana emergente** (esta pertenece a la tabla de Eventos), la idea es que este botón se active al pulsar sobre un paciente. la filtración para encontrar un paciente debe ser por nombre o id.

Pantalla medicamentos:

Sigue la misma lógica que la anterior, pero corresponde a la tabla de Medicamentos, pero solo con los botones de añadir medicamentos y filtrar por nombre o id.

Pantalla de tratamientos:



ID	Paciente	Edad	Tratamiento	Objetivo	Fecha Inicio	Fecha Fin	Estado	Responsable
1	Carlos Martínez	78	Manejo integral de enfermedades crónicas	Control de hipertensión y diabetes	01/11/2023	01/11/2024	Activo	Dr. Alejandro Soto
2	Carlos Martínez	78	Manejo integral de enfermedades crónicas	Control de hipertensión y diabetes	01/11/2023	01/11/2024	Activo	Dr. Alejandro Soto

libre para escribir

Filtrar añadir Editar Asignar Medicamento

Activar Wi Ve a Configuración

Sigue la misma lógica que la anterior, **editar no es obligatorio** pero se ve bien en caso de actualizar un tratamiento de un paciente . Al pulsar asignar medicamento, debe salir una **ventana emergente** con:

3. Botón "Asignar Medicamento":

- Abre un formulario modal con:
 - Medicamento (desplegable) con los nombres disponibles en la tabla `Medicamento` .
 - Dosis (texto)
 - Frecuencia (desplegable): con valores del ENUM.
 - Vía de administración (desplegable).
 - Hora preferida
 - Fecha inicio / Fecha fin
 - Estado del medicamento en el tratamiento

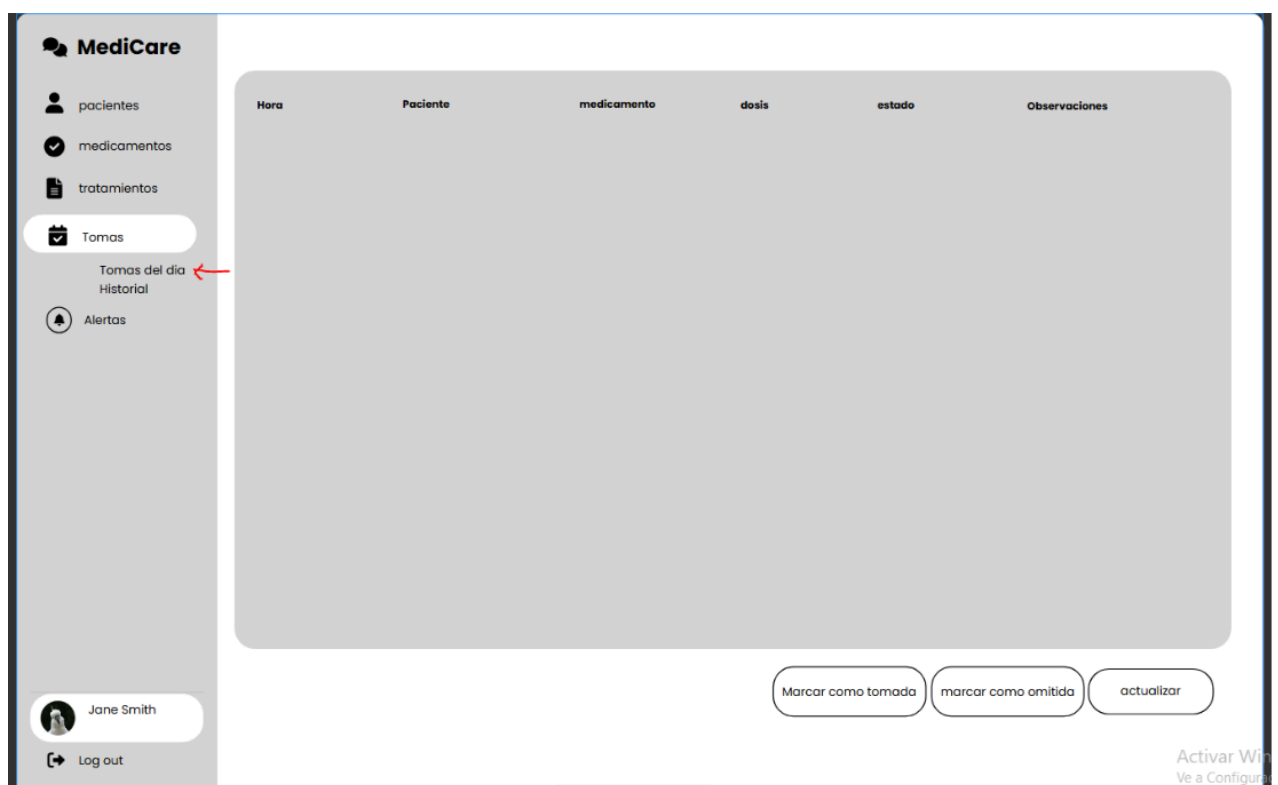
Al llenar este formulario estamos trabajando sobre la tabla `tratamiento_medicamento`, pero es muy importante porque ejecuta `asignarmedicamento` y se generan las tomas de manera automática

Pantalla de Tomas:

a. Tomas del día:

Después de que se generan las tomas (se llena la tabla de tomás, pero al final solo se vera o una parte de esta, que es tomas al día).

Cuenta con 3 botones: Marcar como tomada, Marcar como omitida y actualizar.



Al pulsar botón de actualizar es para ver el cambio en los estados, los marcadores actualizan el estado de programada a verificada u omitida



Resumen flujo ideal entre la interacción de tratamiento, asignar medicamento y la generación de las tomas:

None

```
[Usuario hace clic en "Asignar Medicamento"]
↓
[Formulario con datos del medicamento, dosis, frecuencia, vía, fechas, hora preferida]
↓
[Backend llama a: AsignarMedicamentoATratamiento(...)]
↓
[Obtener ID del nuevo tratamiento_medicamento]
↓
[Llamar a: GenerarTomasTratamiento(id, fecha_inicio, fecha_fin)]
↓
[Las tomas se insertan automáticamente en la tabla Toma]
↓
[Vista Tomas Hoy se actualiza y muestra estas tomas]
```

b. Historial de tomas

MediCare

- pacientes
- medicamentos
- tratamientos
- Tomas**
- Alertas

Tomas del día
Historial

Jane Smith 01/04/2024 30/04/2024

Fecha	Hora	Estado	Medicamento	Tratamiento	Observaciones
29/04/2024	08:00	verificada	Amoxicilina	Tratamiento A	Tomarlo con alimentos
21/04/2024	20:00	omitida	Amoxicilina	Tratamiento A	
29/04/2024	20:00	verificada	Ibuprofeno	Tratamiento B	Tomarlo con alimentos
28/04/2024	18:00	omitida	Amoxicilina	Tratamiento B	
27/04/2024	20:00	omitida	Ibuprofeno	Tratamiento A	
24/04/2024	10:00	verificada	Ibuprofeno	Tratamiento B	

mostrar historial Borrar

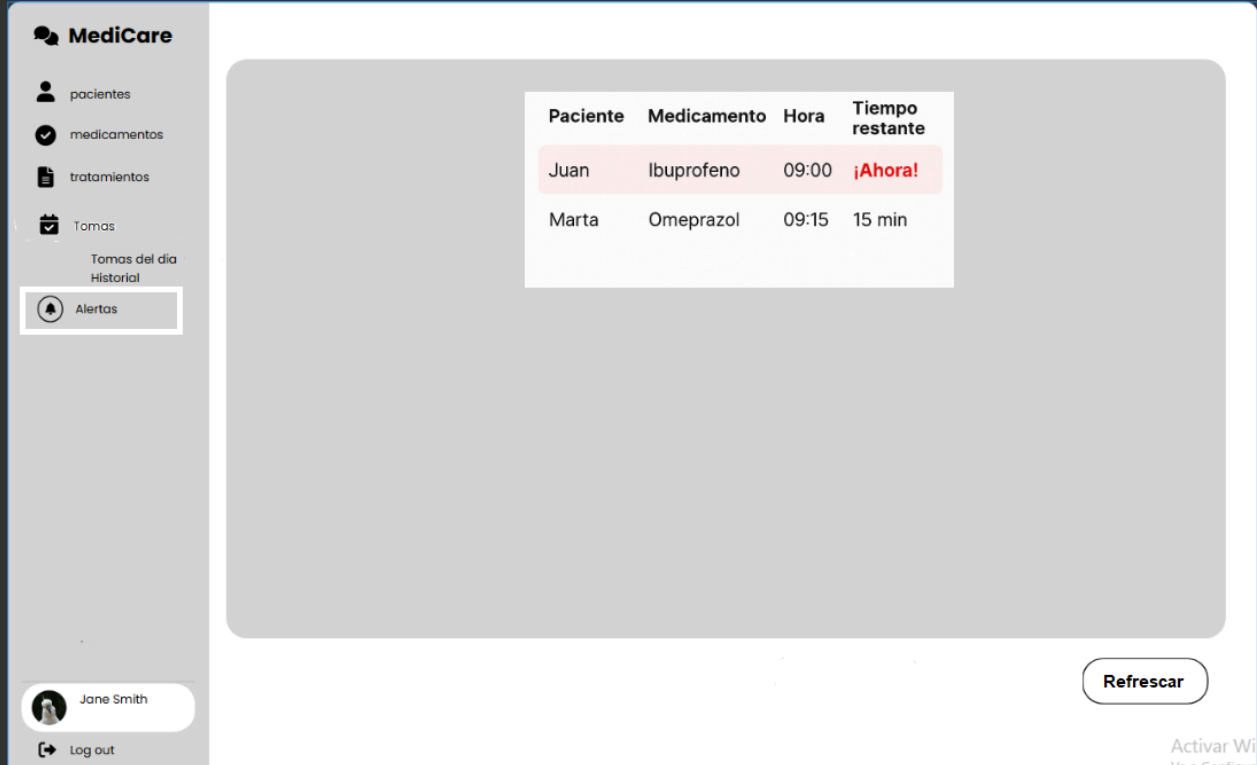
Jane Smith Log out

Activar Win
Ve a Configuración

Debe contar con un formulario para hacer la búsqueda del historial de tomas de un paciente, con parámetros de nombre o id y delimitadores de fechas

maneja dos botones, uno para mostrar el historial generado y otro para borrarlo

Alertas

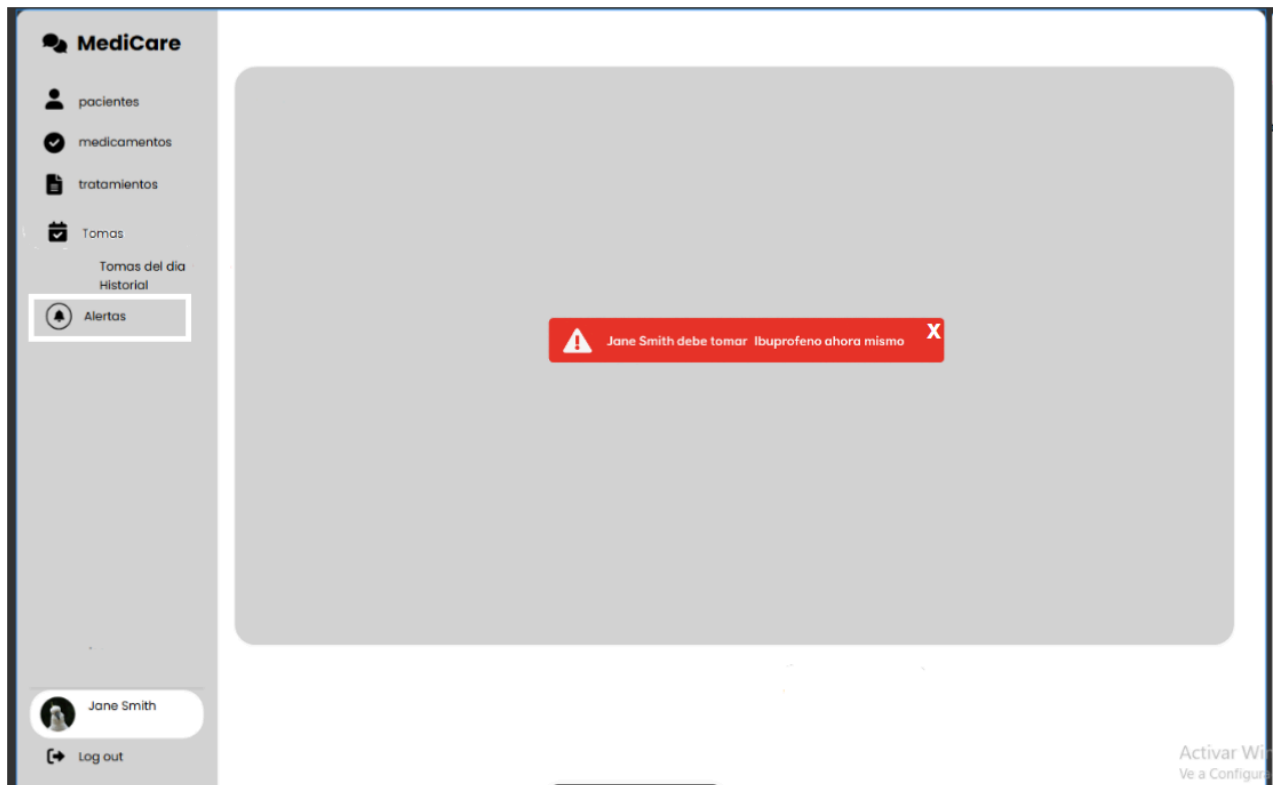


Paciente	Medicamento	Hora	Tiempo restante
Juan	Ibuprofeno	09:00	¡Ahora!
Marta	Omeprazol	09:15	15 min

Debe de mostrar una tabla con la información de las tomas más cercanas, por ejemplo a una hora, no es necesario que sea mucho porque en la vista de toma de hoy se verán todas las que hay a lo largo del día, esta tiene como fin mostrar las más cercanas a determinado tiempo, se debe actualizar en tiempo real, mostrando el tiempo restante, al marcar una toma como verificada u omitida esta debe desaparecer de esta vista.

El botón de refrescar actualiza la información, se requiere oprimir para actualizar.

El sistema debe contar con una ventana emergente de notificación, esta debe aparecer a cualquier vista y debe tener su respectivo botón para cerrar la notificación.



ESTRUCTURA

El proyecto está organizado en carpetas y archivos distribuidos por función. Esto se hace para separar claramente:

- La interfaz gráfica (lo que ve el usuario)
- La lógica de negocio (cómo funciona el sistema)
- El acceso a la base de datos
- Las funciones auxiliares o reutilizables
- Las pruebas
- La configuración general del proyecto

A continuación se dará una breve explicación de cada componente:

app/ — Código fuente principal

Contiene todo el código que hace funcionar la aplicación. Se subdivide en:


app/db/ — Acceso y estructura de la base de datos

Aquí se encuentra la lógica para interactuar con la base de datos local (SQLite). Incluye:

- **conexion.py**: abre la conexión con la base de datos **MediCareDesk.db**




- **modelos.py**: contiene funciones para guardar, consultar, actualizar o eliminar información (CRUD)
- **vistas_sql.py**: contiene scripts SQL para crear vistas que resumen la información (como tomas del día o historial)

 Para guardar o consultar datos del sistema.

app/logic/ — Lógica del negocio

Aquí se define el “cerebro” de la aplicación: cómo deben funcionar los procesos internos.


- **auth.py**: valida el inicio de sesión de cuidadores
- **tomas.py**: genera automáticamente las tomas según la frecuencia, y permite marcarlas como tomadas u omitidas
- **tratamientos.py**: asigna medicamentos a pacientes y gestiona su tratamiento
- **validaciones.py**: aplica reglas como evitar medicamentos duplicados o detectar vencimientos

 Para trabajar en la lógica del sistema: qué debe pasar y cuándo.

app/ui/ — Interfaz gráfica del usuario

Contiene todas las pantallas o ventanas desarrolladas con Tkinter.

- **login.py**: pantalla de inicio de sesión
- **registro.py**: Pantalla de registro
- **pacientes.py**: pantalla para registrar y consultar pacientes
- **medicamentos.py**: pantalla para registrar medicamentos
- **tratamientos.py**: pantalla para asignar tratamientos
- **tomas.py**: vista de tomas del día e historial
- **alertas.py**: vista de alertas inmediatas
- **menu_lateral.py**: vista menu lateral a la izquierda
- **base_view.py**: Vista que carga el **MenuLateral** y una zona de contenido (para las otras vistas)


 Para modificar o mejorar la interfaz que ve el usuario.

app/utils/ — Funciones auxiliares y patrones de diseño




Aquí se colocan herramientas que se pueden usar en diferentes partes del programa.

- **builder.py**: facilita la construcción de objetos (como pacientes o tratamientos) de forma ordenada
- **observer.py**: implementa el patrón Observer para que, por ejemplo, una alerta se active automáticamente si se marca una toma

 Para mejorar la calidad del código y reducir errores.

data/ — Base de datos local


Contiene el archivo **MediCareDesk.db**, que es la base de datos SQLite donde se guarda toda la información (pacientes, tomas, tratamientos, etc.).

 No se necesita un servidor externo: este archivo funciona localmente en cualquier computador.

resources/ — Recursos visuales y sonoros

Aquí van:


- Imágenes o íconos usados en la interfaz
- Sonidos para notificaciones o alertas

 Para cambiar los íconos o sonidos del sistema.

tests/ — Pruebas automáticas del sistema

Contiene pruebas para asegurarse de que el sistema funcione correctamente. Por ejemplo:

- **test_auth.py**: verifica que el login funcione bien
- **test_tomas.py**: prueba que la generación y actualización de tomas sea correcta
- **test_modelos.py**: prueba funciones de acceso a datos

 Úsalo si haces cambios y quieres asegurarte de que no rompiste nada.



main.py — Archivo principal del proyecto

Es el punto de entrada del sistema. Al ejecutar este archivo, se inicializa todo:

```
Python
python main.py
```

 Es la puerta de entrada a la aplicación.

requirements.txt — Lista de dependencias

Contiene los nombres de las librerías que deben instalarse para que el proyecto funcione.


Ejemplo:

```
Python
tk
schedule
reportlab
```

 Úsalo con `pip install -r requirements.txt` para preparar el entorno.

MediCareDesk_SQLite.sql — Script para crear la base de datos

Si por alguna razón necesitas crear una nueva base de datos vacía desde cero, este archivo contiene las instrucciones SQL para hacerlo.


 Úsalo si borraste la base y quieres regenerarla.

README.md — Descripción del proyecto



Archivo de texto que se muestra en GitHub. Contiene:

- Qué hace el sistema
- Cómo se instala
- Cómo se usa
- Cómo colaborar

 Es lo primero que debe leer alguien nuevo en el proyecto.

¿Cómo se conecta todo?

- El usuario abre `main.py`.
- Se carga la pantalla de login (de `ui/login.py`).
- Al iniciar sesión, puede ir a otras vistas: pacientes, tratamientos, tomas.
- La lógica que ejecutan esas vistas está en `logic/`.
- Cualquier cambio o consulta a la base de datos pasa por `db/`.
- Los sonidos, íconos y alertas se toman de `resources/`.

Flujo entre SQLite y Python

1. Agregar paciente → se inserta en tabla `Paciente`
 - Usas funciones Python como `insertar_paciente(...)` en `modelos.py`
 - Se puede buscar por nombre o ID (te explico cuál conviene abajo)
 - se puede agregar eventos relacionados al paciente, usa la tabla de Eventos pero no requiera vista en la base
2. Agregar medicamento → se inserta en la tabla `Medicamento`
 - Similar al anterior
3. Agregar tratamiento → se inserta en la tabla de `Tratamiento`
4. Asignar medicamento a tratamiento:
 - Se inserta en `Tratamiento_Medicamento`
 - Inmediatamente se llama una función Python equivalente a `GenerarTomasTratamiento` para generar todas las tomas en la tabla `Toma`
5. Vista de tomas del día:
 - Se filtra la tabla `Toma` por la fecha actual (`WHERE fecha = DATE('now')`)
 - Se muestra el estado (`programada, tomada, omitida`)



- Al marcar una toma, se ejecutan funciones Python que hacen lo mismo que los procedimientos `MarcarTomaComoVerificada` y `MarcarTomaComoOmitida`

6. Historial de tomas:

- Puede buscarse por nombre o ID del paciente
- Conviene usar ID internamente, pero permitir al usuario buscar por nombre para usabilidad

7. Zona de alertas:

- Filtra las tomas `programadas` cuya `hora_programada` está cerca de `time('now')`

Nota: las funcionalidades se deben implementar en el backend, son las misma en el script de Mysql :p

Organización de los integrantes

Compañero 1 : Catherine Herrera – Líder técnico y encargado del sistema base

Responsabilidades clave

- Organización general del proyecto y estructura del repositorio
- Implementación del **Login y Registro de cuidadores** (`login.py`, `registro.py`, `auth.py`)
- Implementación del archivo `main.py` y carga de vistas
- Adaptación de la base de datos a SQLite y conexión (`conexion.py`)
- Plantilla para vistas y estructura común de menús
- Asistencia técnica al equipo y validación final de integraciones

Compañero 2 – Módulo de Pacientes + Medicamentos + Menú lateral

| Vista/UI | `pacientes.py`, `medicamentos.py`, `menu_lateral.py` |

| Backend | CRUD en `modelos.py` |

| Tareas específicas |

- Registro, edición y filtrado de pacientes y medicamentos
- Ventana emergente para **eventos del paciente**



- Diseño y codificación del **menú lateral reutilizable** (único para todo el sistema)
Asegurarse de que el menú se pueda importar fácilmente desde otras vistas
(MenuLateral(Frame))

Consideraciones

- Tiene **más vistas** que los demás, pero son **repetitivas y visuales**
Requiere un diseño claro, pero **poca lógica de negocio**
- La estructura que desarrolle (menú + base_view) será usada por todos

Compañero 3 – Módulo de Tratamientos + Asignación de Medicamento

| Vista/UI | `tratamientos.py` |

| Backend | `tratamientos.py` (lógica), `modelos.py` (inserción) |

| Tareas específicas |

- Registro de tratamientos asociados a pacientes
- Ventana emergente para **asignar un medicamento a un tratamiento**
- Lógica para llamar a la función `GenerarTomasTratamiento()` en Python
- Validaciones de conflicto, fechas, y estado

Consideraciones

- Aunque tiene **menos vistas**, trabaja con **lógica encadenada** y múltiples tablas
- Debe coordinarse con lo que se haya registrado en pacientes y medicamentos
- El módulo tiene impacto directo en la integridad del sistema (genera tomas)

Compañero 4 – Módulo de Tomas + Alertas

| Vista/UI | `tomas.py`, `alertas.py` |

| Backend | `tomas.py`, `modelos.py` |

| Tareas específicas |

- Mostrar **tomas del día** con estados (programada, tomada, omitida)
- Botones para marcar como tomada u omitida
- **Botón "Refrescar"** que actualiza la tabla y muestra el **tiempo restante** hasta la hora programada
- Mostrar historial de tomas por paciente
- Vista de **alertas** para tomas próximas a vencer
- Implementar **ventana emergente de notificación** para las tomas inminentes

Consideraciones

- Aunque visualmente parece simple, requiere:
 - Cálculo de tiempos
 - Filtrado eficiente
 - Manejo de estado de la toma
- Interactúa con tomas generadas por Compañero 3

Estructura **app/** con asignación final de responsabilidades

None

app/

```
|— ui/
|   |— login.py           ← Compañero 1
|   |— registro.py       ← Compañero 1
|   |— pacientes.py      ← Compañero 2
|   |— medicamentos.py   ← Compañero 2
|   |— menu_lateral.py    ← Compañero 2 ✓
|   |— base_view.py       ← Compañero 2 (estructura común para vistas) ✓
|   |— tratamientos.py    ← Compañero 3
|   |— tomas.py           ← Compañero 4
|   |— alertas.py         ← Compañero 4
|
|— logic/
|   |— auth.py            ← Compañero 1
|   |— validaciones.py    ← Compañero 2 o general
|   |— tratamientos.py    ← Compañero 3
|   |— tomas.py           ← Compañero 4
|
|— db/
|   |— conexion.py        ← Compartido (con acceso centralizado a SQLite)
|   |— modelos.py         ← Compartido (con funciones bien documentadas y
separadas por entidad)

main.py                   ← Compañero 1
```