**Pontificia Universidad Católica del** **Ecuador - Sede Ambato**Logotipo, nombre de la empresa

Descripción generada automáticamente

**Carrera:** Ingeniería en Sistemas de la Información

**Asignatura:** Programación VI

**Docente:** Ing. Edison Meneses Mg.

**Estudiante:** Joaquín Bermeo

**Tema:** Programación Orientada a Objetos – Sistema de Consultas Médicas

**Fecha de Entrega:** 01/05/2025

**Introducción:**

1. **Objetivo:**

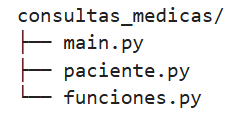
Este proyecto tiene como objetivo diseñar un sistema que permita facilitar el registro de pacientes y consultas médicas usando la "Programación Orientada a Objetos (POO)", dicha información y datos podrán ser consultados dentro del sistema ya que se irán almacenando. Este sistema fue diseñado para permitir que el guardado y manejo de información sea de manera ordenada y se pueda acceder fácilmente a la misma gracias a búsqueda individual o completa de pacientes.

1. **Contexto:**

El sistema fue diseñado de manera modular, es decir, se usaron varios archivos que trabajan en conjunto para que el programa funcione, esto permite que la organización y optimización del código se lleve de manera adecuada y sea más entendible a diferencia de manejar un solo archivo "main". El proyecto lo que busca es utilizar los conceptos básicos y fundamentales de la Programación Orientada a Objetos como el uso de clases, objetos, métodos/funciones, etc., para lograr cumplir el objetivo principal.

**Desarrollo:**

1. **Estructura del proyecto:**

****

**Archivo “main.py”:**

Este archivo contiene el menú principal y la lógica que gestiona la interacción con el usuario. El programa se mantiene activo mediante el uso de un ciclo \*while\*, lo que permite que el usuario interactue entre las distintas opciones que se manejan dentro del sistema sin la necesidad de ejecutarlo siempre.

**Archivo “paciente.py”:**

Aquí se define la clase “Paciente” la cual contiene los atributos y métodos relacionados a la información de un paciente. Los atributos implementados fueron: nombre, cédula, edad, tipo de sangre y consultas (lista de diccionarios), esta última permite que a un paciente se le pueda registrar varias consultas y se almacenen. Por otro lado, se implementaron métodos que trabajan en conjunto con el archivo de funciones “agregar\_consulta”, mostrar\_información y mostrar\_consultas.

**Archivo “funciones.py”:**

Contiene funciones auxiliares de los distintos procesos del sistema como registrar pacientes consultas, buscar pacientes por su cédula y mostrar información de los pacientes registrados. Al inicio se importa la clase Paciente del archivo “paciente.py”. Cada una de estas funciones cuentan con sus respectivas validaciones para que el programa se ejecute y utilice sin que haya errores. Además de las funciones, aquí se inicializa la lista que se encarga de ir almacenando todos los pacientes que se vayan registrando “lista\_pacientes”.

1. **Diseño de la clase “Paciente”:**

La clase “Paciente” es fundamental dentro del funcionamiento del sistema, ya que representa el objeto con el que interactúa el usuario. Los atributos que contiene la clase son los siguientes:

* **nombre:** Atributo que almacena el nombre del paciente.
* **cédula:** Atributo que almacena la cédula del paciente.
* **edad:** Atributo que almacena la edad del paciente.
* **tipo\_sangre:** Atributo que almacena el tipo de sangre del paciente.
* **lista\_consultas:** Lista de diccionarios vacía que almacena las consultas del paciente.

Texto

El contenido generado por IA puede ser incorrecto.

Los métodos que se implementaron dentro de a clase “Paciente” son:

* **agregar\_consulta:** Método que recibe como parámetros a fecha, diagnostico y tratamiento con los que se crea un diccionario que después es almacenado en la lista “lista\_consultas”.

Texto

El contenido generado por IA puede ser incorrecto.

* **mostrar\_información:** Método que se encarga de mostrar la información básica del paciente, es decir, su nombre, cédula, edad, y tipo de sangre.

Texto

El contenido generado por IA puede ser incorrecto.

* **mostrar\_consultas:** Método que muestra el historial de consultas del paciente, es decir, imprime cada uno de los elementos que han sido registrados en la lista “lista\_consultas”, si esta se encuentra vacía se imprime un aviso.

Texto

El contenido generado por IA puede ser incorrecto.

1. **Funcionamiento del menú y funciones:**

**Menú:**

Se creó una función llamada menu() que está dentro del archivo “main”. Dentro de la función se maneja un ciclo “while True:” que permite que el sistema se mantenga activo hasta que el usuario desee salir. Se lee una única entrada de un número entero que es la opción que ingresa el usuario para ir navegando entre las distintas opciones (1. Registrar nuevo paciente 2. Registrar una consulta médica 3. Mostrar datos completos de un paciente 4. Mostrar todos los pacientes registrados 5. Salir), este proceso de selección se lo realizó mediante el uso de if, elif y else donde se compara el valor ingresado por el usuario y permite acceder a las funciones principales del sistema que se exportan del archivo funciones.py. Además, se aplicó la respectiva validación con un try para que el sistema no colapse cuando se ingrese caracteres que no sean números.

Texto

El contenido generado por IA puede ser incorrecto.

**Funciones:**

1. **registrar\_paciente():**

Esta función se encarga de capturar la información necesaria para posteriormente crear el objeto “paciente” utilizando la clase creada en el archivo “paciente.py”. En la misma función se valida que el usuario no pueda repetir la cédula en varios pacientes con la implementación de la función “buscar\_paciente\_cedula(cedula)”, ya que se trata de un identificador único, de igual forma trabaja en conjunto con la función “validar\_cedula”. Por otro lado, la edad es validada para que el usuario no pueda ingresar números negativos y que solo pueda ingresar números enteros. Finalmente, se creó una lista con todos los tipos de sangre que existen lo que permite comparar la entrada del usuario con la lista y de esta manera validar que se ingrese un tipo de sangre real.

Texto

El contenido generado por IA puede ser incorrecto.

1. **validar\_cedula():**

Como su nombre lo indica, dentro de esta función se pide al usuario ingresar la cédula para validar la parte básica, es decir, que no haya letras y sean 10 dígitos. El dato ingresado se lo mantiene como tipo String, ya que existen números de cédula que empiezan por el número 0 y al momento de transformarlo a entero no se cuenta el número mencionado y se tomaría como inválida a la cédula ingresada.

Texto

El contenido generado por IA puede ser incorrecto.

1. **buscar\_paciente\_cedula(cedula):**

Esta función se encarga de buscar a un paciente por su número de cédula dentro de la lista que contiene a todos los pacientes registrados, si existe el paciente retorna al mismo sino retorna None. El único parámetro que recibe es la cédula.

Texto

El contenido generado por IA puede ser incorrecto.

1. **registrar\_consulta():**

En esta función se realiza el proceso de registrar consultas de cada paciente, al inicio se valida que por lo menos exista un paciente registrado para poder continuar, en el caso de que no existan pacientes lo único que hará es mostrar una alerta de que no existen pacientes registrados. Luego de la primera validación se pide al usuario ingresar una cedula utilizando la función “validar\_cedula” y se guarda en una función, después se busca el paciente con la función “buscar\_paciente\_cédula(cedula)” y en el caso de que exista se pasará a la siguiente parte, caso contrario se muestra que no se encontró al paciente. Finalmente, se captura la información para generar el diagnostico (fecha, diagnóstico y tratamiento). Se utilizó la librería datetime para que solo se pueda ingresar el formato de fecha DD/MM/AAAA y que tampoco se ingrese fechas futuras o inválidas.

Texto

El contenido generado por IA puede ser incorrecto.

1. **mostrar\_paciente():**

Al igual que la función anterior se valida que la lista de pacientes no esté vacía, luego se usa “validar\_cedula” para que el usuario ingrese una cédula válida para buscar el paciente con la función “buscar\_paciente\_cédula(cedula)”. Si se encuentra al paciente se muestra la información completa del mismo, es decir, su información básica y todo el historial de consultas utilizando métodos de la clase “Paciente”.

Texto

El contenido generado por IA puede ser incorrecto.

1. **mostrar\_todos\_pacientes():**

Esta función también valida que haya al menos un usuario registrado en la lista “lista\_pacientes”, después se utiliza un for para ir imprimiendo cada uno de los pacientes, pero en este caso solo mostrando su información básica (nombre, cédula, edad y tipo de sangre). Además, se muestra la cantidad total de usuarios registrados y se usa un contador para enumerar a los pacientes y la visualización de la información sea más organizada.

Texto

El contenido generado por IA puede ser incorrecto.

**Capturas de pantalla de la ejecución:**

1. **Menú:**

**Texto

El contenido generado por IA puede ser incorrecto.**

1. **Registrar Pacientes:**

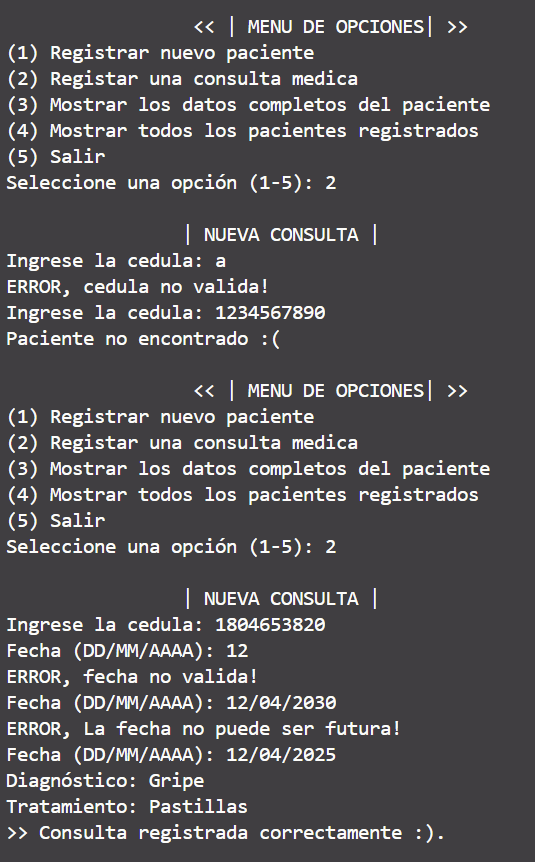
**Texto

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

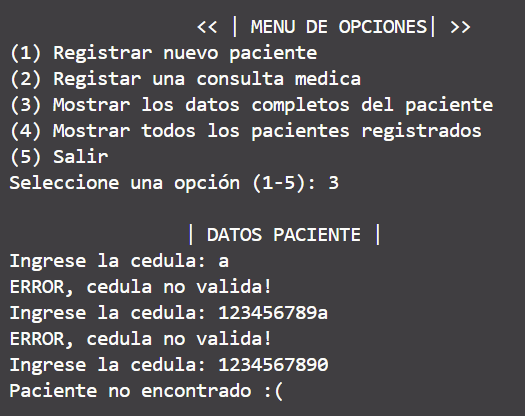
1. **Registrar Consulta:**

****

**Texto

El contenido generado por IA puede ser incorrecto.**

1. **Mostrar los datos completos del paciente:**

****

**Imagen de la pantalla de un celular de un mensaje en letras negras

El contenido generado por IA puede ser incorrecto.**

**Texto

El contenido generado por IA puede ser incorrecto.**

1. **Mostrar todos los pacientes registrados:**

**Texto

El contenido generado por IA puede ser incorrecto.**

1. **Salir**

**Texto

El contenido generado por IA puede ser incorrecto.**

**Conclusiones**

**¿Qué aprendí durante el desarrollo del proyecto?**

* Este proyecto me permitió aplicar y mejorar los conceptos de la Programación Orientada a Objetos, como el uso de clases, objetos y métodos. Por otro lado, aprendí a estructurar un proyecto de manera modular (separado en varios archivos) y a trabajar de manera mucho más optimizada y organizada.

**¿Qué partes fueron más difíciles o interesantes?**

* Personalmente una de las partes más difíciles e interesantes fue crear la clase Persona y usar de manera adecuada sus métodos, ya que no es igual que utilizar funciones que no se encuentran dentro de una clase, al inicio es complejo adaptarse, pero después se vuelve interesante porque el código se ve mucho más limpio. Otra de las partes más difíciles fue validar de manera adecuada todos los datos, no por complejidad sino porque se debe ir analizando a detalle que información no debe estar permitida y la que sí.

**Posibles mejoras o nuevas funcionalidades**

* Almacenar los datos de los pacientes y consultas en un archivo externo como un “.txt” o “. json” para que la información no sea volátil y se pueda tener una “base de datos”.
* Se podría añadir otra opción la cuál permita editar los datos del paciente o directamente eliminar un paciente.
* Validaciones más estrictas de los datos de entrada como por ejemplo que se valide si la cédula es verdadera.