Proyecto modulo #2

Ficha técnica: Sistema de Gestión de Contactos

1. Descripción General

El sistema es una aplicación de consola robusta diseñada para la administración centralizada de información de contacto. Implementa un flujo de trabajo seguro que diferencia las capacidades de un usuario estándar de las de un administrador, garantizando la integridad de la base de datos mediante procesos de validación rigurosos.

2. Arquitectura del Software

El proyecto se basa en una arquitectura de N-Capas Lógicas dentro de un único script, separando la interfaz, la lógica de negocio y el modelado de datos.

A. Capa de Modelado (POO)

Se utiliza Programación Orientada a Objetos para representar las entidades del sistema:

- Clase usuario (Clase Base): Define el plano fundamental. Aplica Encapsulamiento mediante atributos protegidos (_nombre, _correo, _celular, _direccion), asegurando que el acceso a los datos sea controlado.
- Clase contacto (Heredera): Extiende a usuario para representar registros individuales en la agenda.
- Clase administrador (Heredera): Especialización que añade el atributo contraseña, permitiendo implementar un sistema de permisos para funciones críticas (Edición/Eliminación).

B. Capa de Lógica y Validación

Para evitar la corrupción de datos, se implementaron funciones de validación mediante bucles while True y métodos de cadena:

- Sanitización de Strings: Uso de .replace(" ","").isalpha() para asegurar nombres reales.
- Restricciones de Formato: Validación de longitud exacta (8 dígitos) para números celulares.
- Verificación de Sintaxis: Comprobación de existencia del carácter @ para correos electrónicos.

3. Especificaciones de los Módulos

- Módulo sys: Proporciona acceso a variables y funciones que interactúan con el intérprete. Se utiliza específicamente sys.exit() para forzar el cierre del programa ante brechas de seguridad (exceso de intentos de login).

- Gestión de Colecciones: Se emplea una Lista de Diccionarios. Esta estructura permite una iteración eficiente y una búsqueda dinámica mediante el algoritmo de coincidencia implementado en la función buscar_contacto.

4. Requerimientos del Entorno

- Lenguaje: Python 3.8 o superior.
- Dependencias: Ninguna (Uso exclusivo de la biblioteca estándar de Python para máxima portabilidad).
- Terminal: Compatible con cualquier consola (CMD, PowerShell, Bash, Terminal de VS Code).

5. Algoritmo de Búsqueda

La función buscar_contacto utiliza una búsqueda lineal con Normalización de Texto (.lower()). Esto permite que la búsqueda no sea sensible a mayúsculas o minúsculas, mejorando significativamente la experiencia del usuario (UX).

Código detallado:

1. Gestión de Validaciones e Integridad de Datos

El sistema implementa funciones de validación que actúan como "puertas de enlace" para asegurar que solo datos sanitizados ingresen a la estructura de almacenamiento.

- Validación de Cadenas (nombre_validar): Se utiliza un bucle while True para forzar la reentrada en caso de error. La expresión nom.replace(" ","").isalpha() es clave, ya que permite que el nombre contenga espacios (necesario para nombres compuestos) pero rechaza cualquier carácter numérico o símbolo especial, garantizando la calidad del dato de identidad.
- Validación de Longitud y Tipo (cel_validar): El algoritmo aplica una doble verificación. Primero, se asegura de que la entrada sea puramente numérica con .isdigit(). Segundo, mediante len(), se restringe la entrada a exactamente 8 dígitos. Esto previene errores comunes en la base de datos de contactos.
- Validación de Formato (correo_validar): Se aplica el método .count("@") sobre la cadena de entrada. Si el contador es menor a 1, el sistema asume que el formato de correo es inválido, impidiendo el registro de información incompleta.

2. Arquitectura de Objetos y Serialización (POO)

El diseño se basa en una estructura de clases que utiliza principios fundamentales de la POO:

- Encapsulamiento y Atributos Protegidos: En la clase base usuario, se observa el uso del prefijo _ (ej. self._nombre). Esto indica, por convención, que los atributos son protegidos y no deben ser manipulados directamente desde fuera de la clase, promoviendo el acceso controlado.
- Método de Serialización (diccionario): Este método es crítico para la persistencia en memoria. Transforma una instancia de clase (un objeto complejo) en un diccionario de Python. Esto permite que el sistema almacene los contactos en una lista estándar sin perder la organización de los datos.
- Herencia y Constructor super(): Tanto en la clase contacto como en administrador, se invoca a super().__init__. Esto permite que las clases hijas hereden la lógica de inicialización de la clase padre, evitando la duplicación de código y facilitando el mantenimiento.

3. Motor de Búsqueda y Comparación de Datos

La función buscar_contacto(termino, lista) implementa un algoritmo de búsqueda lineal optimizado:

- Normalización: Se utiliza .lower() tanto en el término de búsqueda como en los datos almacenados. Esto elimina la sensibilidad a mayúsculas y minúsculas (case-insensitivity), mejorando la experiencia del usuario.
- Multicriterio: El condicional if evalúa si el término coincide con el nombre o con el número celular, otorgando versatilidad a la búsqueda.
- Retorno de Índice: El hecho de retornar el índice (i) en lugar del dato mismo permite que las funciones de modificación y eliminación actúen directamente sobre la posición correcta en la lista.

4. Lógica de Seguridad y Control de Errores

El bloque principal del programa gestiona el flujo del usuario y la protección del sistema:

- Protocolo de Autenticación: En la opción 2, se implementa un sistema de intentos fallidos. El cálculo intentos - (i + 1) informa al usuario en tiempo real sobre su margen de error. La instrucción sys.exit() se ejecuta como una medida de seguridad definitiva para cerrar el proceso ante un posible intento de acceso por fuerza bruta.
- Validación del Menú: El uso de not opcion.isdigit() seguido de continue previene que el programa se detenga (crash) si el usuario ingresa accidentalmente una letra en lugar de un número, reiniciando el ciclo de manera controlada.

5. Gestión de la Lista de Almacenamiento:

El sistema utiliza una lista dinámica denominada lista_contactos. Las operaciones sobre esta lista son:

- Adición: Mediante .append(), se insertan los diccionarios generados por la clase.
- Eliminación: Se emplea .pop(ind_registro), el cual elimina el elemento de la memoria utilizando el índice específico proporcionado por el motor de búsqueda, asegurando que no se borren datos incorrectos.

Versiones y creación del programa por partes

El primer paso fue realizar el bucle base para que el menú se ejecutara hasta que el usuario no quiera registrar mas contactos, por esta razón, el inicio fue lo primero que se creó para generar orden en el código, se consideró también generar un mensaje de error en caso de que el usuario no ingrese uno de los valores aceptados por el sistema. En las imágenes adjuntas se puede visualizar la prueba de lo realizado.

```
# Proyecto Sistema de Gestión de Contactos -

#registro, la edición y eliminación de contactos, además de realizar
#búsquedas y organizar los datos de manera eficiente.

opciones = ["1","2","3","4"]

print("-- SISTEMA DE GESTIÓN DE CONTACTOS --\n 1.Registro de contacto \n
#opcion = input("Ingrese la acción que desea realizar (1-4): ")
while True:
    opcion = input("Ingrese la acción que desea realizar (1-4): ")

    if not opcion.isdigit():
        print("ERROR - debe ingresar un número: ")
        continue #Reiniciar el menú

    if opcion in opciones:
        if opcion == "1":
            print("REGISTRAR")
        elif opcion == "2":
            print ("MODIFICAR/ELIMINAR CONTACTO")
        elif opcion == "3":
            print("BUSCAR CONTACTO")
        elif opcion == "4":
            break
    else:
        print("ERROR - Debe seleccionar una opción del 1 al 4")
```

```
In [19]: runfile('C:/Users/56996/.spyder-py3/autosave/
untitled7.py', wdir='C:/Users/56996/.spyder-py3/autosave')
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): L
ERROR - debe ingresar un número:
Ingrese la acción que desea realizar (1-4): M
ERROR - debe ingresar un número:
Ingrese la acción que desea realizar (1-4): 5
ERROR - Debe seleccionar una opción del 1 al 4
Ingrese la acción que desea realizar (1-4): 6
ERROR - Debe seleccionar una opción del 1 al 4
Ingrese la acción que desea realizar (1-4): 2
MODIFICAR/ELIMINAR CONTACTO
Ingrese la acción que desea realizar (1-4): 1
REGISTRAR
Ingrese la acción que desea realizar (1-4): 3
BUSCAR CONTACTO
Ingrese la acción que desea realizar (1-4): 4
```

Luego se comenzó a generar las variables que se completaban por pantalla, pero de inmediato se consideró usar funciones, ya que, ingresar y verificar los datos se haría no solo en el registro de contactos, sino que también al modificarlos. Se crearon solo de los datos que debían ser validados para registrarlos (nombre, celular y correo) donde cada uno tenia sus especificaciones, como por ejemplo el correo que debía contener "@" o el numero de celular que debía tener 8 dígitos numéricos.

```
def nombre_validar():
    while True:
        nom = input("Ingrese nombre:")
        if nom.replace(" ","").isalpha():
            return nom
        else:
            print("ERROR - El nombre solo debe contener letras.")

def cel_validar():
    while True:
        cel = input("Ingrese número celular: ")
        if cel.replace(" ","").isdigit() and len(cel.replace(" ","")) == 8:
            return cel
        else:
            print("ERROR - Rectifique número celular")

def correo_validar():
    while True:
        email = input("Ingrese su correo electrónico: ")
        if email.count("@"):
            return email
        else:
            print("ERROR - Ingrese un correo válido.")
```

```
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 1
REGISTRAR
Ingrese nombre: 5
ERROR - El nombre solo debe contener letras.
Ingrese nombre:Mey Antonia
Ingrese número celular: 63
ERROR - Rectifique número celular
Ingrese número celular: f
ERROR - Rectifique número celular
Ingrese número celular: 963626 23
Ingrese su correo electrónico: mey
ERROR - Ingrese un correo válido.
Ingrese su correo electrónico: Mey@gmail.com
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 4
SESIÓN FINALIZADA
```

Listo el menú y el registro de contactos, se continuo con el almacenamiento de los contactos, por esta razón, se fue generando un diccionario por cada contacto registrado, creando así una lista de diccionarios. Una vez realizado esto, se continuo con la creación de clases para aplicar POO y se consideró mejor crear el diccionario luego de definir los datos según su clase y se modificó de la siguiente manera el código:

```
#REGISTRO- REGISTRO- REGISTRO- REGISTRO
    if opcion in opciones:
        if opcion == "1":
            print("REGISTRAR")
            nombre = nombre_validar()
            celular = cel_validar()
            correo = correo_validar()
            direccion = input("Ingrese la dirección: ")

            reg_contacto = {
                "nombre":nombre,
                "celular":celular,
                "correo": correo,
                "direccion":direccion
                }

            lista_contactos.append(reg_contacto)
            print(f"{nombre} registrado como contacto")
```

```
class usuario:   #clase base
    def __init__(self, nombre, correo, celular, direccion):
        self._nombre = nombre
        self._celular = celular
        self._correo = correo
        self._direccion = direccion

    def diccionario(self):
        return {
            "nombre":self._nombre,
            "correo":self._correo,
            "celular":self._celular,
            "direccion":self._direccion
            }


class contacto(usuario):
    def __init__(self, nombre, correo, celular, direccion):
        super().__init__(nombre, correo, celular, direccion)

class administrador(usuario):
    def __init__(self, nombre, correo, celular, direccion, contraseña):
        super().__init__(nombre, correo, celular, direccion)
        self.contraseña = contraseña
```

```
#REGISTRO- REGISTRO- REGISTRO- REGISTRO
    if opcion in opciones:
        if opcion == "1":
            print("REGISTRAR")
            nombre = nombre_validar()
            celular = cel_validar()
            correo = correo_validar()
            direccion = input("Ingrese la dirección: ")

            reg_contacto = contacto(nombre, correo, celular, direccion)


            lista_contactos.append(reg_contacto.diccionario())

            print(f"{nombre} registrado como contacto")
```

Esto además permite un código mas compacto y ordenado, ya que, de esta forma no es necesario volver a guardar los datos en el diccionario, solo basta con determinar su clase. Y a continuación un ejemplo de ejecución del código modificado.

```
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 1
REGISTRAR
Ingrese nombre:mey
Ingrese número celular: 98989898
Ingrese su correo electrónico: correo@gmial.com
Ingrese la dirección: casa 123
mey registrado como contacto
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 4
SESIÓN FINALIZADA
[{'nombre': 'mey', 'correo': 'correo@gmial.com', 'celular': '98989898',
'direccion': 'casa 123'}]
```

El código se fue realizando por partes, es por eso que luego se continuo con cada una de las partes del menú, de esta forma se hizo mucho mas sencillo ir desarrollando el programa.

La opción modificar es muy similar al registro de contactos, por esta razón se replicaron muchas líneas del código, pero se debía ir rectificando la existencia de los registros realizados.

```
#MODIFICAR- MODIFICAR- MODIFICAR- MODIFICAR
        elif opcion == "2":
            #SEGURIDAD DE INGRESO
            intentos = 4
            aceptar = False

            for i in range(1, intentos):
                ingreso = input("Ingrese contraseña de administrador: ")
                if ingreso == adm_base.contraseña:
                    print("Ingreso correcto")
                    aceptar = True
                    break  #SALIR DEL BUCLE
                else:
                    intentos_rest = intentos - i
                    if intentos_rest > 0:
                        print("ERROR - Contraseña incorrecta")
                        print(f"Te quedan {intentos_rest} intentos")
                    else:
                        print("Acceso bloqueado")

            if aceptar:
                print("MODIFICAR/ELIMINAR CONTACTO \n 1.Modificar contacto \n 2.Eliminar conta
                accion = input("Ingrese la opción que desea realizar (1-2): ")
                while accion not in ["1","2"]:
                    accion = input("ERROR - Ingrese la opción que desea realizar: ")


                if accion == "1":
                    print("modificar")

                elif accion == "2":
                    print("eliminar")
```

```
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 2
Ingrese contraseña de administrador: D
ERROR - Contraseña incorrecta
Te quedan 2 intentos
Ingrese contraseña de administrador: D
ERROR - Contraseña incorrecta
Te quedan 1 intentos
Ingrese contraseña de administrador: D
Acceso bloqueado
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4):
```

```
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 2
Ingrese contraseña de administrador: D
ERROR - Contraseña incorrecta
Te quedan 2 intentos
Ingrese contraseña de administrador: administrador.124
Ingreso correcto
MODIFICAR/ELIMINAR CONTACTO
 1.Modificar contacto
 2.Eliminar contacto
Ingrese la opción que desea realizar (1-2): 1
modificar
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4):
```

```
import sys
```

```
  else:
      print("ACCESO BLOQUEADO, el sistema se cerrará")
      sys.exit()  #SALIR DEL PROGRAMA -
```

```
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 2
Ingrese contraseña de administrador: f
ERROR - Contraseña incorrecta
Te quedan 2 intentos
Ingrese contraseña de administrador: f
ERROR - Contraseña incorrecta
Te quedan 1 intentos
Ingrese contraseña de administrador: f
ACCESO BLOQUEADO, el sistema se cerrará
```

```python
#INGRESO ACEPTADO

if aceptar:
    print("MODIFICAR/ELIMINAR CONTACTO \n 1.Modificar contacto \n 2.Eliminar conta
    accion = input("Ingrese la opción que desea realizar (1-2): ")
    while accion not in ["1","2"]:
        accion = input("ERROR - Ingrese la opción que desea realizar: ")

    if accion == "1":
        print("MODIFICAR CONTACTO")
        nom_buscar = nombre_validar()

        for i in range(len(lista_contactos)):
            if nom_buscar.lower() in lista_contactos[i]["nombre"].lower():
                print(f"{lista_contactos[i]} encontrado")

            elif nom_buscar.lower() not in lista_contactos[i]["nombre"].lower():
                print("El contacto no existe en los registros")

    elif accion == "2":
        print("eliminar")
```

```
        if aceptar:
            print("MODIFICAR/ELIMINAR CONTACTO \n 1.Modificar contacto \n 2.Eliminar conta
            accion = input("Ingrese la opción que desea realizar (1-2): ")
            while accion not in ["1","2"]:
                accion = input("ERROR - Ingrese la opción que desea realizar: ")

            if accion == "1":
                print("MODIFICAR CONTACTO")

                while True:
                    nom_buscar = nombre_validar()
                    if nom_buscar == "salir":
                        break

                    ind_registro = ""
                    for i in range(len(lista_contactos)):
                        if nom_buscar.lower() in lista_contactos[i]["nombre"].lower():
                            ind_registro = i
                            print(f"{lista_contactos[i]} encontrado")
                            break #DEJAR DE BUSCAR


                        else:
                            print("El contacto no existe en los registros")


            elif accion == "2":
                print("eliminar")
```

```
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 1
REGISTRAR
Ingrese nombre:mey
Ingrese número celular: 96969696
Ingrese su correo electrónico: mey@•gmial.com
Ingrese la dirección: casa 123
mey registrado como contacto
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 2
Ingrese contraseña de administrador: administrador.124
Ingreso correcto
MODIFICAR/ELIMINAR CONTACTO
 1.Modificar contacto
 2.Eliminar contacto
Ingrese la opción que desea realizar (1-2): 1
MODIFICAR CONTACTO
Ingrese nombre:mey
{'nombre': 'mey', 'correo': 'mey@•gmial.com', 'celular': '96969696',
'direccion': 'casa 123'} encontrado
```

```python
if accion == "1":
    print("MODIFICAR CONTACTO")

    while True:
        nom_buscar = nombre_validar()
        if nom_buscar == "salir":
            break

        ind_registro = -1
        for i in range(len(lista_contactos)):
            if nom_buscar.lower() in lista_contactos[i]["nombre"].lower():
                ind_registro = i
                print(f"{lista_contactos[ind_registro]['nombre']} encontrado")
                break #DEJAR DE BUSCAR

        if ind_registro != -1:
            print("Datos actuales")
            print("nombre: ", lista_contactos[ind_registro]['nombre'])
            print("correo: ", lista_contactos[ind_registro]['correo'])
            print("celular: ", lista_contactos[ind_registro]['celular'])
            print("direccion: ", lista_contactos[ind_registro]['direccion'])

            print(" INGRESE LOS NUEVOS DATOS")
            nom_nuevo = nombre_validar()
            cor_nuevo = correo_validar()
            cel_nuevo = cel_validar()
            dir_nueva = input("Ingrese dirección: ")

            lista_contactos[ind_registro] ={
                "nombre":nom_nuevo,
                "correo":cor_nuevo,
                "celular":cel_nuevo,
                "direccion":dir_nueva
                }
            print(f"El contacto {nom_nuevo} ha sido modificado con éxito.")
            break  #IMPOPRTANTE CIERRA EL BUCLE
        else:
            print("El contacto no existe en los registros")
```

```
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 1
REGISTRAR
Ingrese nombre:MEY
Ingrese número celular: 12345567
Ingrese su correo electrónico: @
Ingrese la dirección: CASA
MEY registrado como contacto
-- SISTEMA DE GESTIÓN DE CONTACTOS --
 1.Registro de contacto
 2.Modificar/Eliminar un contacto
 3.Buscar datos de un contacto
 4.Salir
Ingrese la acción que desea realizar (1-4): 2
Ingrese contraseña de administrador: administrador,124
ERROR - Contraseña incorrecta
Te quedan 2 intentos
Ingrese contraseña de administrador: administrador.124
Ingreso correcto
MODIFICAR/ELIMINAR CONTACTO
 1.Modificar contacto
 2.Eliminar contacto
Ingrese la opción que desea realizar (1-2): 1
MODIFICAR CONTACTO
Ingrese nombre:3
ERROR - El nombre solo debe contener letras.
Ingrese nombre:mey
MEY encontrado
Datos actuales
nombre:  MEY
correo:  @
celular:  12345567
direccion:  CASA
 INGRESE LOS NUEVOS DATOS
Ingrese nombre:4
ERROR - El nombre solo debe contener letras.
Ingrese nombre:meyline
Ingrese su correo electrónico: d
ERROR - Ingrese un correo válido.
Ingrese su correo electrónico: @
Ingrese número celular: 67676767
Ingrese dirección: d
El contacto meyline ha sido modificado con éxito.
```

```python
elif accion == "2":
    print(" -- ELIMINAR CONTACTO --")

    while True:
        nom_buscar = input("Ingrese nombre del contacto a eliminar (o salir): ")
        if nom_buscar.lower() == "salir":
            break

        ind_registro = -1
        for i in range(len(lista_contactos)):
            if nom_buscar.lower() in lista_contactos[i]["nombre"].lower():
                ind_registro = i
                break

        if ind_registro != -1:
            print(f"Contacto {lista_contactos[ind_registro]['nombre']} encontrado")

            confirmar = input(f"¿Está seguro que desea eliminar a {lista_contactos[ind_registro]['nombre']}? (s/n): ")

            if confirmar.lower() == "s":
                eliminado = lista_contactos.pop(ind_registro)
                print(f"El contacto '{eliminado['nombre']}' ha sido eliminado.")
                break # MENU PRINCIPAL
            else:
                print("Operación cancelada.")
                break # MENU PRINCIPAL
        else:
            print("El contacto no existe en los registros")
```

```
Ingrese la acción que desea realizar (1-4): 2
Ingrese contraseña de administrador: adinistrador.124
ERROR - Contraseña incorrecta
Te quedan 2 intentos
Ingrese contraseña de administrador: administrador.124
Ingreso correcto
MODIFICAR/ELIMINAR CONTACTO
 1.Modificar contacto
 2.Eliminar contacto
Ingrese la opción que desea realizar (1-2): 2
 -- ELIMINAR CONTACTO --
Ingrese nombre del contacto a eliminar (o salir): f
El contacto no existe en los registros
Ingrese nombre del contacto a eliminar (o salir): mey
Contacto MEY encontrado
¿Está seguro que desea eliminar a MEY? (s/n): s
El contacto 'MEY' ha sido eliminado.
```

```python
#BUSCAR- BUSCAR- BUSCAR- BUSCAR- BUSCAR- BUSCAR
        elif opcion == "3":
            print("--- BUSCAR CONTACTO ---")
            if not lista_contactos:
                print("La agenda está vacía actualmente.")
            else:
                nom_buscar = input("Ingrese el nombre del contacto a buscar (o 'salir'): ")
                if nom_buscar.lower() != "salir":

                    ind_registro = -1
                    for i in range(len(lista_contactos)):
                        # Usamos 'in' por si solo recuerdan parte del nombre
                        if nom_buscar.lower() in lista_contactos[i]["nombre"].lower():
                            ind_registro = i
                            break

                    if ind_registro != -1:
                        print("CONTACTO ENCONTRADO:")
                        print(f"Nombre:    {lista_contactos[ind_registro]['nombre']}")
                        print(f"Correo:    {lista_contactos[ind_registro]['correo']}")
                        print(f"Celular:   {lista_contactos[ind_registro]['celular']}")
                        print(f"Dirección: {lista_contactos[ind_registro]['direccion']}\n")
                    else:
                        print("El contacto no existe en los registros")
```

```python
def buscar_contacto(termino, lista):
    for i in range(len(lista)):
        if (termino.lower() in lista[i]["nombre"].lower() or
            termino in lista[i]["celular"]):
            return i
    return -1
```

```python
if aceptar:
    print("MODIFICAR/ELIMINAR CONTACTO \n 1.Modificar contacto \n 2.Eliminar contacto")
    accion = input("Ingrese la opción que desea realizar (1-2): ")
    while accion not in ["1","2"]:
        accion = input("ERROR - Ingrese la opción que desea realizar: ")

    if accion == "1":
        print("-- MODIFICAR CONTACTO --")

        while True:
            nom_buscar = nombre_validar()
            if nom_buscar == "salir":
                break

            ind_registro = buscar_contacto(nom_buscar, lista_contactos)
            #IMPORTANTE: FUNCION
            #ind_registro = -1
            #for i in range(len(lista_contactos)):
             #   if nom_buscar.lower() in lista_contactos[i]["nombre"].lower():
              #      ind_registro = i
               #     print(f"{lista_contactos[ind_registro]['nombre']} encontrado")
                #    break #DEJAR DE BUSCAR

            if ind_registro != -1:
                print("Datos actuales")
                print("nombre: ", lista_contactos[ind_registro]['nombre'])
                print("correo: ", lista_contactos[ind_registro]['correo'])
                print("celular: ", lista_contactos[ind_registro]['celular'])
                print("direccion: ", lista_contactos[ind_registro]['direccion'])

                print(" INGRESE LOS NUEVOS DATOS")
                nom_nuevo = nombre_validar()
                cor_nuevo = correo_validar()
                cel_nuevo = cel_validar()
                dir_nueva = input("Ingrese dirección: ")

                lista_contactos[ind_registro] ={
```