Sistema de Gestión de Contactos

Descripción del Proyecto

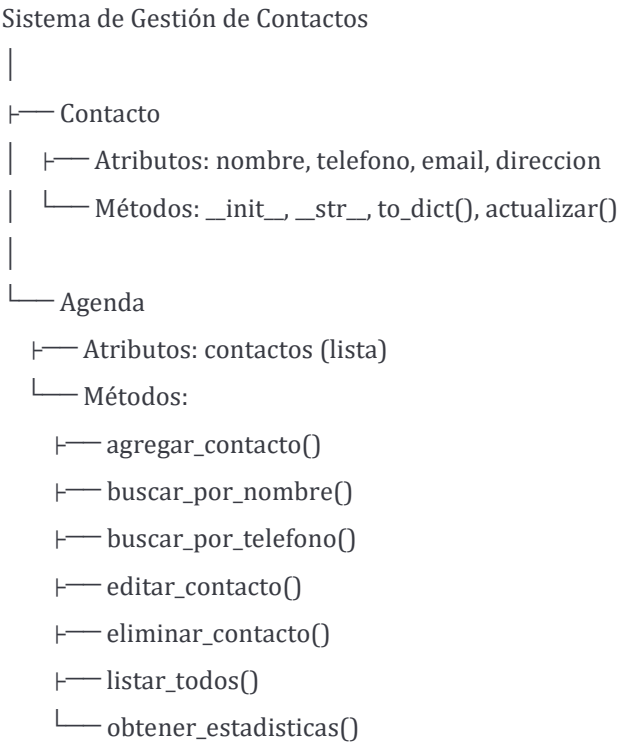
Sistema completo de gestión de contactos desarrollado en Python que permite almacenar, buscar, editar y eliminar información de contactos personales. El proyecto implementa principios de Programación Orientada a Objetos (POO) y buenas prácticas de desarrollo.

Objetivos Cumplidos

Registro de contactos: Agregar nuevos contactos con información completa
Edición y eliminación: Modificar o eliminar contactos existentes
Búsqueda avanzada: Buscar por nombre o teléfono
Estructuras de datos: Uso de listas y diccionarios
POO: Implementación con clases y encapsulación
Pruebas unitarias: Cobertura de funcionalidades principales

Arquitectura del Sistema

Estructura de Clases



Instalación y Ejecución

Requisitos

- Python 3.7 o superior
- Jupyter Notebook o JupyterLab
- No requiere librerías externas (usa solo la biblioteca estándar)

Ejecución en Jupyter

1. **Abrir Jupyter Notebook**

bash

```
jupyter notebook
```

2. **Ejecutar las celdas en orden**:
    - **Celda 1**: Clase Contacto
    - **Celda 2**: Clase Agenda
    - **Celda 3**: Menú Principal (opcional)
    - **Celda 4**: Pruebas Unitarias
    - **Celda 5**: Sistema Completo Integrado
3. **Iniciar el sistema interactivo**:

python

```
ejecutar_sistema()
```

Guía de Uso

Operaciones Básicas

1. Agregar un Contacto

python

```
agenda = Agenda()
agenda.agregar_contacto(
    nombre="Juan Pérez",
    telefono="+56 9 1234 5678",
    email="juan@email.com",
    direccion="Calle 123, Santiago"
)
```

2. Buscar Contactos

python

```
# Buscar por nombre
resultados = agenda.buscar_por_nombre("Juan")

# Buscar por teléfono
contacto = agenda.buscar_por_telefono("+56 9 1234 5678")
```

3. Listar Todos los Contactos

python

```
agenda.listar_todos()
```

4. Ver Estadísticas

python

```
agenda.obtener_estadisticas()
```

Pruebas Unitarias

El proyecto incluye pruebas exhaustivas para garantizar la calidad:

Cobertura de Pruebas

- Creación de contactos
- Conversión a diccionario
- Actualización de datos
- Agenda vacía
- Agregar contactos
- Validación de nombres vacíos
- Búsqueda por nombre
- Búsqueda por teléfono
- Contactos no existentes

Ejecutar Pruebas

python

```
ejecutar_pruebas()
```

Resultados Esperados

```
EJECUTANDO PRUEBAS UNITARIAS
==================================

test_actualizar_contacto ... ok
test_creacion_contacto ... ok
test_to_dict ... ok
test_agenda_vacia ... ok
test_agregar_contacto ... ok
test_buscar_por_nombre ... ok
...

RESUMEN DE PRUEBAS
Pruebas exitosas: 10
Pruebas fallidas: 0
Total de pruebas: 10
```

Estructura del Código

Clase Contacto

Responsabilidad: Representar un contacto individual

Atributos:

- `nombre` (str): Nombre completo del contacto
- `telefono` (str): Número telefónico
- `email` (str): Correo electrónico
- `direccion` (str): Dirección física

Métodos principales:

- `__init__()`: Constructor de la clase
- `__str__()`: Representación en texto
- `to_dict()`: Conversión a diccionario
- `actualizar()`: Actualización de datos

Clase Agenda

Responsabilidad: Gestionar la colección de contactos

Atributos:

- `contactos` (list): Lista de objetos Contacto

Métodos principales:

- `agregar_contacto()`: Añade nuevos contactos
- `buscar_por_nombre()`: Búsqueda por nombre (parcial)
- `buscar_por_telefono()`: Búsqueda por teléfono
- `editar_contacto()`: Modificación de contactos
- `eliminar_contacto()`: Eliminación de contactos
- `listar_todos()`: Listado ordenado
- `obtener_estadisticas()`: Estadísticas de la agenda

Características Destacadas

1. Búsqueda Inteligente

- Búsqueda parcial por nombre (no case-sensitive)
- Búsqueda exacta por teléfono
- Manejo de múltiples resultados

2. Validaciones

- Nombres no pueden estar vacíos
- Confirmación antes de eliminar
- Verificación de duplicados

3. Interfaz Amigable

- Menú interactivo con emojis
- Mensajes claros de éxito/error
- Formateo visual atractivo

4. Organización de Datos

- Listado alfabético automático
- Estadísticas útiles
- Conversión a diccionario para interoperabilidad

Principios de Diseño Aplicados

Programación Orientada a Objetos

1. **Encapsulación**: Los datos están protegidos dentro de las clases
2. **Abstracción**: Las clases ocultan la complejidad interna
3. **Separación de responsabilidades**: Cada clase tiene un propósito único

Estructuras de Datos

- **Listas**: Almacenamiento dinámico de contactos
- **Diccionarios**: Representación estructurada de datos

Buenas Prácticas

- Docstrings en todas las clases y métodos
- Nombres descriptivos de variables
- Manejo de excepciones
- Validación de entradas
- Código modular y reutilizable

Casos de Uso

Escenario 1: Gestión Personal

Usuario necesita organizar contactos de amigos y familia

→ Agregar contactos con toda la información

→ Buscar rápidamente por nombre

→ Mantener actualizada la información

Escenario 2: Pequeño Negocio

Empresa pequeña necesita gestionar clientes

→ Registro de clientes con datos de contacto

→ Búsqueda rápida durante llamadas

→ Estadísticas de base de datos de clientes

Manejo de Errores

El sistema incluye manejo robusto de errores:

- Validación de entradas vacías
- Manejo de búsquedas sin resultados
- Confirmaciones antes de eliminaciones
- Excepciones generales capturadas
- Mensajes informativos al usuario

Posibles Mejoras Futuras

1. **Persistencia de datos**
    - Guardar contactos en archivo JSON
    - Base de datos SQLite
2. **Funcionalidades adicionales**
    - Grupos de contactos
    - Etiquetas/categorías
    - Búsqueda avanzada con filtros
    - Exportación a CSV/Excel
3. **Interfaz gráfica**
    - GUI con Tkinter
    - Web app con Flask/Django
4. **Validaciones avanzadas**
    - Validación de formato de email
    - Validación de números telefónicos
    - Prevención de duplicados exactos

Conclusiones

Este proyecto demuestra:

Comprensión sólida de POO en Python
Uso efectivo de estructuras de datos
Implementación de pruebas unitarias
Código limpio y bien documentado
Interfaz de usuario intuitiva
Aplicación práctica y funcional

Autor

John Cabello

Proyecto desarrollado como parte del módulo 2 de Ciencia de Datos