

Especialización en Ciencia de Datos

Actividades

Estudiante: Yoseph Barrera
Profesor(a): Silvia Salinas Ayaviri
Fecha: 23 de diciembre de 2025

Trabajo Módulo 2: Informe de Pruebas

1. Motivación y objetivo

El sistema `agenda_contactos.py` implementa una agenda de contactos con operaciones CRUD (crear, leer/listar, actualizar y eliminar) y funciones de búsqueda. Dado que el funcionamiento del sistema depende de piezas lógicas pequeñas (por ejemplo, asignación de IDs, almacenamiento consistente y recuperación por criterios), se incluyó un conjunto de pruebas unitarias para verificar que los componentes esenciales se comporten como se espera.

El objetivo de este informe es describir (i) la librería utilizada para testeo y su justificación, (ii) la lógica detrás de los casos de prueba seleccionados, y (iii) los resultados obtenidos al ejecutar las pruebas.

2. Herramienta de testeo: unittest

Para el testeo se utilizó `unittest`, librería estándar de Python. Esta decisión es intencional por dos razones principales. Primero, evita dependencias externas y facilita la reproducibilidad (cualquier entorno con Python 3 puede ejecutar las pruebas sin instalar paquetes adicionales). Segundo, `unittest` provee una estructura clara para organizar pruebas en torno a una clase de test (`TestCase`) y un conjunto de aserciones simples (por ejemplo, igualdad de resultados y existencia de objetos), lo cual calza bien con el alcance del módulo.

En términos prácticos, `unittest` permite que el testeo sea parte del flujo normal de desarrollo: después de cada ajuste al código, correr las pruebas sirve como “chequeo de humo” que confirma que lo fundamental no se rompió.

3. Enfoque general: qué se testea y por qué

La estrategia seguida prioriza el funcionamiento de la aplicación desde la perspectiva de un usuario: si el sistema no puede *crear* contactos o no puede *encontrarlos* luego, entonces el resto de funcionalidades pierde valor. Por lo mismo, las pruebas se centran en dos pilares:

1. **Registro (crear contacto):** valida que un contacto se guarda efectivamente y que el sistema responde con un identificador consistente.
2. **Búsqueda (por nombre y por teléfono):** valida que el contacto previamente registrado puede ser recuperado por criterios razonables.

Adicionalmente, cada prueba se ejecuta sobre una agenda nueva gracias al método `setUp()`. Esto asegura independencia entre pruebas: cada test se ejecuta “desde cero”, evitando que el estado creado por una prueba influya sobre el resultado de otra (un principio básico para que el testeo sea confiable y reproducible).

4. Casos de prueba implementados

A continuación se describen los casos incluidos, indicando su rol dentro del sistema.

4.1. Caso 1: registro de contacto (`test_agregar_contacto`)

Este caso evalúa la función de registro como un test de consistencia mínima del sistema. Se agrega un contacto con datos válidos y se verifica que:

- el ID retornado coincide con el esperado (por ejemplo, el primer registro retorna $ID = 1$), y
- el número de contactos almacenados aumenta en una unidad.

La lógica detrás de este caso es simple: si la agenda no asigna IDs de forma coherente o no refleja el nuevo contacto al listar, la operación CRUD más básica falla.

4.2. Caso 2: búsqueda por nombre (`test_buscar_por_nombre`)

Este caso evalúa la búsqueda por nombre bajo un criterio de coincidencia parcial (substring), que es un comportamiento típico en agendas: un usuario suele buscar con fragmentos del nombre. Se registra un contacto y luego se consulta por un texto parcial; el test verifica que la cantidad de resultados coincide con lo esperado (un único match en este escenario).

Este caso es relevante porque también valida implícitamente que el almacenamiento en la estructura interna es consistente: si el contacto se “perdiera” o quedara mal guardado, la búsqueda no retornaría resultados.

4.3. Caso 3: búsqueda por teléfono (`test_buscar_por_telefono`)

La búsqueda por teléfono es especialmente útil en una agenda porque permite identificar de forma directa a una persona. En el sistema, el teléfono se normaliza y se utiliza para indexación y/o control de duplicados; por ello, este caso funciona también como un chequeo indirecto de consistencia del dato (teléfono).

El test agrega un contacto con un teléfono válido, luego busca por ese mismo teléfono, y verifica que:

- el resultado existe (no es nulo), y
- el contacto recuperado corresponde al registrado (por ejemplo, coincide el nombre).

5. Resultados

Las pruebas fueron ejecutadas con éxito, finalizando con estado `OK`. La evidencia del output (incluyendo el resumen de ejecución) se encuentra guardada en el repositorio en el archivo:

```
docs/evidencia_tests.txt
```

Este resultado respalda que, para el conjunto de escenarios testeados, el sistema cumple con el comportamiento esperado en registro y recuperación de contactos.

6. Discusión: cobertura actual y extensiones razonables

El set actual de pruebas está diseñado como una base sólida y minimalista: prueba lo esencial y permite detectar rápidamente regresiones en las funciones más críticas. Sin embargo, existen extensiones naturales para aumentar cobertura en versiones futuras. Por ejemplo:

- **Duplicados:** verificar que el sistema rechaza el registro de dos contactos con el mismo teléfono.
- **Validaciones negativas:** comprobar que correos mal formateados o teléfonos inválidos disparan `ValueError` como corresponde.
- **Edición:** asegurar que editar un contacto actualiza correctamente la información y, si cambia el teléfono, que se mantiene la coherencia del índice.
- **Eliminación:** verificar que un contacto eliminado deja de aparecer en listados y búsquedas.
- **Casos borde:** búsquedas sin resultados, IDs inexistentes y entradas vacías.

En síntesis, el enfoque actual cumple el objetivo del módulo y entrega evidencia verificable de funcionamiento. A futuro, aumentar la batería de pruebas permitiría robustecer el sistema ante escenarios menos frecuentes y validar con mayor detalle la integridad de las operaciones CRUD completas.