



Informe de Pruebas: Pruebas Unitarias y Resultados

Introducción

Este documento detalla las pruebas realizadas al "Sistema de Gestión de Contactos" para verificar el cumplimiento de los requerimientos técnicos y funcionales establecidos. Se enfoca en validar la correcta implementación del paradigma de Programación Orientada a Objetos (POO) y la manipulación de datos.

Resumen del Entorno de Pruebas

Lenguaje	Plataforma	Librerías/Estructuras
Python	Google Colab	Uso de listas y diccionarios para el almacenamiento persistente en memoria durante la ejecución.

Pruebas Unitarias Detalladas

A continuación, se describen los casos de prueba ejecutados sobre las clases `Contacto` y `GestorContactos`:

Caso de Prueba (ID)	Objetivo	Entrada	Resultado Esperado	Resultado Obtenido
Caso de Prueba 01: Registro de Contactos	Validar que un nuevo objeto de la clase <code>Contacto</code> se cree con los atributos correctos.	Nombre: "Javier", Teléfono: "123456", Correo:	El objeto debe almacenar los 4 atributos correctamente.	EXITOSO. El constructor <code>__init__</code> asignó los valores de forma precisa.

Caso de Prueba (ID)	Objetivo	Entrada	Resultado Esperado	Resultado Obtenido
		"javier@email.com", Dirección: "Copiapó".		
Caso de Prueba 02: Búsqueda de Contactos	Verificar que el método <code>buscar_contacto</code> localice un elemento existente.	Buscar el nombre "Javier".	Mostrar la información detallada del contacto.	EXITOSO. El sistema realizó la iteración sobre la lista y aplicó el filtro correctamente.
Caso de Prueba 03: Edición de Datos	Comprobar la capacidad de modificar los atributos de un contacto.	Cambiar el teléfono de "Javier" a "9999999".	El atributo <code>telefono</code> del objeto debe actualizarse.	EXITOSO. Se verificó la mutabilidad del objeto dentro de la lista del gestor.
Caso de Prueba 04: Eliminación de Contactos	Asegurar que un contacto sea removido de la estructura de datos.	Eliminar contacto "Javier".	La lista de contactos debe reducir su tamaño en 1.	EXITOSO. Se utilizó el método <code>.remove()</code> validando la identidad del objeto.

```
Menú de Agenda:
1. Agregar Contacto
2. Lista de Contactos
3. Buscar Contacto
4. Eliminar Contacto
5. Editar Contacto
6. Salir
Selecciona una opción: 1
Nombre: Javier
Teléfono: 123456
Email: javier@email.com
Dirección: Copiapó

Menú de Agenda:
1. Agregar Contacto
2. Lista de Contactos
3. Buscar Contacto
4. Eliminar Contacto
5. Editar Contacto
6. Salir
Selecciona una opción: 1)
```

```
Menú de Agenda:
... 1. Agregar Contacto
     2. Lista de Contactos
     3. Buscar Contacto
     4. Eliminar Contacto
     5. Editar Contacto
     6. Salir
Selecciona una opción: 3
Nombre del contacto a buscar: Javier
```

2)

```
Buscando a Javier...
Nombre: Javier
Teléfono: 123456
Email: javier@email.com
Dirección: Copiapó
```

```
Menú de Agenda:
1. Agregar Contacto
2. Lista de Contactos
3. Buscar Contacto
4. Eliminar Contacto
5. Editar Contacto
6. Salir
Selecciona una opción: 5
Nombre del contacto a editar: Javier
Nuevo nombre: Javier
Nuevo teléfono: 999999
Nuevo email: javier@email.com
Nueva dirección: Copiapó
```

3)

```
Editando a Javier...
Javier ha sido editado.
```

```
Menú de Agenda:
1. Agregar Contacto
2. Lista de Contactos
3. Buscar Contacto
4. Eliminar Contacto
5. Editar Contacto
6. Salir
Selecciona una opción: 4
Nombre del contacto a eliminar: Javier
```

4)

```
Eliminando a Javier...
Javier ha sido eliminado.
```

Conclusión y Próximos Pasos

El prototipo funcional responde de manera satisfactoria a todas las operaciones CRUD (Crear, Leer, Actualizar, Borrar). La arquitectura basada en clases permitió una separación clara de responsabilidades, garantizando la escalabilidad y legibilidad del código fuente.