

Evidencia de Aprendizaje N° 2

Módulo de innovación de datos

Índice de contenidos

Miembros del equipo	4
Brief de funcionalidad	4
Descripción	4
Objetivos	4
Alcance esperado	5
Diagramas	5
Diagrama de Clases	5
Diagrama de Datos	6
Funcionamiento	6
Agregar Contacto	6
Precondiciones	6
Flujo	6
Mensajes	7
Actualizar Contacto	7
Precondiciones	7
Flujo	7
Mensajes	7
Eliminar Contacto	8
Precondiciones	8
Flujo	8
Mensajes	8
Listar Contactos	8
Flujo	8
Mensajes	8
Base de datos en SQLite	8
Conexión y creación de la BD	8
Sentencias DML según botón	9
Agregar	9
Actualizar	9
Eliminar	9
Listar	10
Interfaz Tkinter	10
Componentes principales	10
Habilitar/Deshabilitar botones	10
Limpieza de campos	10
Carga de formulario desde la grilla	11
Validaciones en Tkinter	11
POO (Programación Orientada a Objetos)	11
Clases	11
Principios POO aplicados	12
Posibles mejoras	12
Referencias	12
Anexo	13

Miembros del equipo

Nombre	DNI	Correo
Fontana Agostina	33.382.658	fontaago@gmail.com
Montiel Matias Ariel	42.474.994	matypollo10@gmail.com
Negro Clarisa	37.522.352	clarisanegro2@gmail.com
Roldan Yanina Maria Eva	33.602.217	yanina88roldan@gmail.com
Verdú Lucía Belen	42.051.559	lucia.verdu@mi.unc.edu.ar

Brief de funcionalidad

Descripción

El sistema es una aplicación de escritorio desarrollada en Python que permite realizar ABM (Alta, Baja y Modificación) de contactos con una interfaz Tkinter y con una BBDD en SQLite.

La solución permite registrar y consultar datos básicos de cada contacto (Nombre, Apellido, Teléfono, Email) y nos permite gestionar el ciclo de vida completo desde una GUI simple, con una grilla para listar resultados y botones de acción para cada operación.

A nivel técnico, el proyecto aplica POO con una clase principal Contacto y una capa de acceso a datos que ejecuta sentencias SQL sobre la tabla contactos. La base de datos es local (contactos.db).

Objetivos

- Implementar la clase Contacto con atributos (nombre, apellido, teléfono, email) y métodos asociados.
- Desarrollar operaciones CRUD contra SQLite.
- Construir una GUI en Tkinter que permita:
 - Alta de contactos,
 - Baja sobre un registro seleccionado,
 - Modificación de datos,
 - Listado/recarga de información en una grilla.
- Incorporar validaciones mínimas en el flujo.

- Entregar modelo de clases, modelo de datos y descripción del proceso de conexión a SQLite.
- Preparar datos de prueba y un breve video demo que evidencie el cumplimiento del requerimiento.

Alcance esperado

- Alta (Agregar): Crear contacto con nombre, apellido, teléfono y email. Dicho contacto se debe insertar en tabla contactos.
- Baja (Eliminar): Eliminar contacto seleccionado (por id) desde la grilla.
- Modificación (Actualizar): Actualizar campos del contacto seleccionado por id.
- Consulta (Listar): Mostrar contactos en una Grilla desde un comando SELECT *

Diagramas

Diagrama de Clases

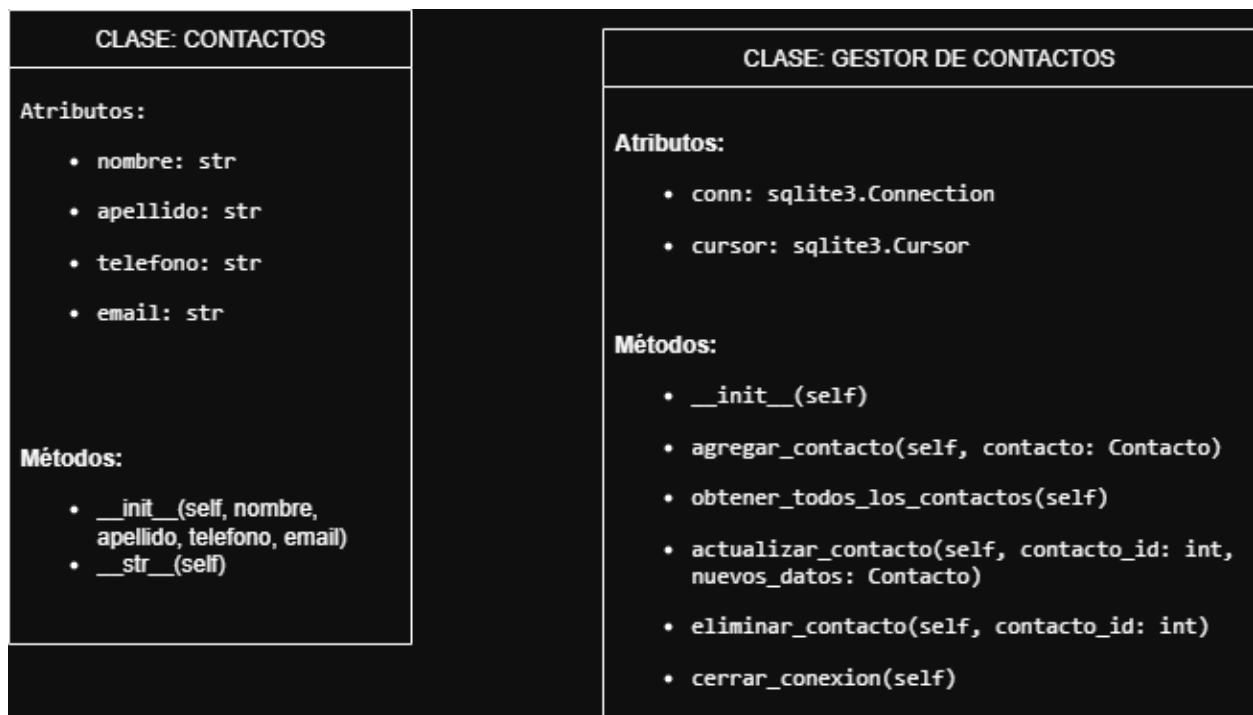


Diagrama de Datos

CONTACTOS			
INTEGER	id	PK	AUTOINCREMENT
TEXT	nombre		NOT NULL
TEXT	apellido		
TEXT	telefono		
TEXT	email		

Funcionamiento

Agregar Contacto

Precondiciones

- Campos completos: nombre, apellido, teléfono, email.
- Validaciones de formato:
 - Teléfono: solo dígitos (integer).
 - Email: debe contener “@”.
- La interfaz valida y habilita el botón “Agregar” solo si se cumplen las reglas.

Flujo

- El usuario completa los campos.
- La Interfaz válida si todo está OK, entonces habilita el botón “Agregar”.
- Al hacer clic en Agregar:
 - Se crea el objeto Contacto.
 - Se ejecuta INSERT en la tabla contactos.
 - SQLite asigna un ID autoincremental.
 - Se muestra pop-up de éxito: “Contacto creado (ID: {id}).”

- Se limpian los campos del formulario.
- El usuario presiona Listar para validar.

Mensajes

- Éxito: "Contacto creado ID "N"."
- Error de validación:
 - "El teléfono sólo admite números."
 - "El email debe contener '@'."
 - "Completá los campos obligatorios."
- Error BD: "No se pudo crear el contacto. Intente nuevamente."

Actualizar Contacto

Precondiciones

- Haber presionado Listar y seleccionado un contacto en la grilla.
- La selección carga los datos al formulario para edición.
- Validaciones idénticas a Agregar (teléfono solo dígitos, email con "@", campos obligatorios).

Flujo

- Listar para ver registros.
- Seleccionar una fila, automáticamente se rellenan los campos.
- Editar los campos.
- Clic en Actualizar:
 - Se ejecuta UPDATE ... WHERE id = {id}.
 - Pop-up: "Contacto actualizado ID "N"."
 - Limpieza de campos del formulario.
 - Presionar Listar para verificar los cambios.

Mensajes

- Éxito: "Contacto actualizado ID "N"."
- Validación: mismos mensajes que en Agregar.
- Error BD: "No se pudo actualizar el contacto."

Eliminar Contacto

Precondiciones

- Haber presionado Listar y seleccionado un contacto en la grilla.

Flujo

- Seleccionar el contacto en la grilla.
- Clic en Eliminar:
 - DELETE FROM contactos WHERE id = {id}.
 - Pop-up: "Contacto eliminado (ID: n)."
 - Limpieza de campos del formulario.
 - Presionar Listar para validar que ya no aparece.

Mensajes

- Éxito: "Contacto eliminado (ID: n)."
- Sin selección: "Seleccioná un contacto de la tabla."
- Error BD: "No se pudo eliminar el contacto."

Listar Contactos

Flujo

- Ejecuta SELECT * FROM contactos.
- Pinta resultados en la grilla con las columnas: ID, Nombre, Apellido, Teléfono, Email.
- Al inicio, la grilla puede estar vacía y recién mostrar datos al presionar Listar.

Mensajes

- Sin datos: la grilla muestra 0 filas.
- Error BD: "No se pudieron recuperar los contactos."

Base de datos en SQLite

Conexión y creación de la BD

- Conexión: conn = sqlite3.connect('contactos.db')

- Cursor: cursor = conn.cursor()
- DDL (una sola vez al iniciar):

SQL

```
CREATE TABLE IF NOT EXISTS contactos (  
    id INTEGER PRIMARY KEY AUTOINCREMENT,  
    nombre TEXT NOT NULL,  
    apellido TEXT NOT NULL,  
    telefono TEXT NOT NULL,  
    email TEXT NOT NULL  
);
```

- Commit tras operaciones de escritura: conn.commit()
- Cierre ordenado: conn.close()

Sentencias DML según botón

Agregar

SQL

```
INSERT INTO contactos (nombre, apellido, telefono, email)  
VALUES (?, ?, ?, ?);
```

Actualizar

SQL

```
UPDATE contactos  
SET nombre = ?, apellido = ?, telefono = ?, email = ?  
WHERE id = ?;
```

Eliminar

SQL

```
DELETE FROM contactos  
WHERE id = ?;
```

Listar

SQL

```
SELECT id, nombre, apellido, telefono, email  
FROM contactos  
ORDER BY id DESC;
```

Interfaz Tkinter

Componentes principales

- Entradas (Entry): txtNombre, txtApellido, txtTelefono, txtEmail.
- Botones: btnAgregar, btnActualizar, btnEliminar, btnListar.
- Grilla (ttk.Treeview): gridContactos con Scrollbar.

Habilitar/Deshabilitar botones

- btnAgregar: habilitado sólo cuando las validaciones de los cuatro campos pasan.
- btnActualizar / btnEliminar: habilitados solo cuando hay una fila seleccionada.
- btnListar: siempre habilitado.

Limpieza de campos

- Se limpian tras Agregar y Actualizar (para acelerar carga de nuevos registros).
- Se limpian tras Eliminar (evita editar un registro inexistente).
- Función: limpiar_campos() llamada al final de esas operaciones.

Carga de formulario desde la grilla

- Evento: Selección de contacto en la grilla, carga los datos en los Entry:
 - idSeleccionado, txtNombre, txtApellido, txtTelefono, txtEmail.

Validaciones en Tkinter

- Teléfono solo dígitos: `validate='key' + validatecommand` que acepte `str.isdigit()` o vacío.
- Email con "@": validación al perder foco o antes del INSERT/UPDATE.
- Límites de longitud: cortar entrada si supera el máximo en cada Entry.

POO (Programación Orientada a Objetos)

¿Qué hace que este programa sea POO?

La solución organiza el dominio "contactos" en clases y colaboraciones entre capas.

Clases

- Contacto (Modelo de dominio)
 - Representa a una persona en el sistema. Cada instancia mapea 1-a-1 con una fila de la tabla contactos.
 - Atributos: nombre, apellido, teléfono, email.
 - Métodos: constructor `__init__` (creación del objeto con sus datos) y `__str__` (representación legible para logs/UI).
 - Rol: encapsula el estado del contacto. Nos permite razonar sobre "un contacto" como entidad, independientemente de cómo se persista.
- GestorDeContactos (Capa de datos/servicio)
 - Expone operaciones de negocio sobre contactos y encapsula el acceso a SQLite.
 - Métodos: `agregar_contacto`, `obtener_todos_los_contactos`, `actualizar_contacto`, `eliminar_contacto`, `cerrar_conexion`.
 - Rol: concentra la lógica de persistencia (CRUD) y evita que la GUI conozca detalles SQL.
- GUI (Tkinter)
 - Maneja el flujo de interacción.
 - Rol: Es la capa de presentación.

Principios POO aplicados

- Abstracción: tratamos “contactos” como objetos (Contacto) con datos y comportamiento; la GUI no necesita saber “cómo” se guardan.
- Encapsulación: el SQL se aloja en GestorDeContactos; la GUI ve métodos de alto nivel.
- Separación de responsabilidades:
 - Modelo (Contacto) es un estado del dominio.
 - Repositorio GestorDeContactos añade persistencia.
 - Interfaz mejora la experiencia de usuario.
- Reciclar y mantenibilidad: cualquier cambio de BD se hace en una sola capa sin romper la GUI.

Posibles mejoras

1. Refresco automático del listado (quitar botón Listar).
2. El botón Agregar abre un modal con formulario, de esta forma se ve más minimalista.
3. Agregar íconos a los botones.
4. Acciones Editar y Eliminar como botones dentro de la tabla.
5. Importar contactos desde CSV.
6. Exportar contactos a CSV.
7. Exportar listado a PDF.
8. Añadir un Buscador por nombre/apellido/email.
9. Paginado en la grilla cuando la lista de contactos sea muy grande.
10. Encabezados interactivos que permiten ordenar por asc/desc.
11. Interfaz modernizada (JS/React/Angular).

Referencias



Interfaz de gestor de contactos

The screenshot shows a simple web application window titled "Gestor de Contactos". It features four input fields for "Nombre:", "Apellido:", "Teléfono:", and "Email:". Below these fields are four buttons: "Agregar", "Listar", "Eliminar", and "Actualizar". At the bottom, there is a table with the following data:

Id	Nombre	Apellido	Telefono	Email
6	Matias	Montiel	3513679866	maty@gmail.com
2	Luciano	Cuffia	2996745939	lucianocuffia@gmail.com

Interfaz de gestor de contactos mejorada

The screenshot displays a more modern, dark-themed interface for a contact manager. The title "Gestor de contactos" is prominently displayed at the top. Below it, a subtitle reads: "El nuevo sistema de gestor de contactos para la Evidencia 2 de innovación de datos siendo un hito de la tecnología de vanguardia. Cualquier sistema comparado con este pareciera que fuese de Windows 95". The interface includes a navigation bar with buttons: "Agregar Contacto" (highlighted in pink), "Importar", "Exportar", and "Dictado por voz". A search bar with the placeholder "Buscar" is also present. Below the navigation bar is a table with the following data:

Nombre	Apellido	Teléfono	Email	Acciones
Matias	Montiel	3513679866	matypollo10@gmail.com	 

Anexo

- IDE Utilizada: [Visual Studio Code](#)
- Link de la carpeta en Drive: [Video explicativo](#)
- Herramienta de diagramas: [Lucidchart](#) - [Draw.io](#)