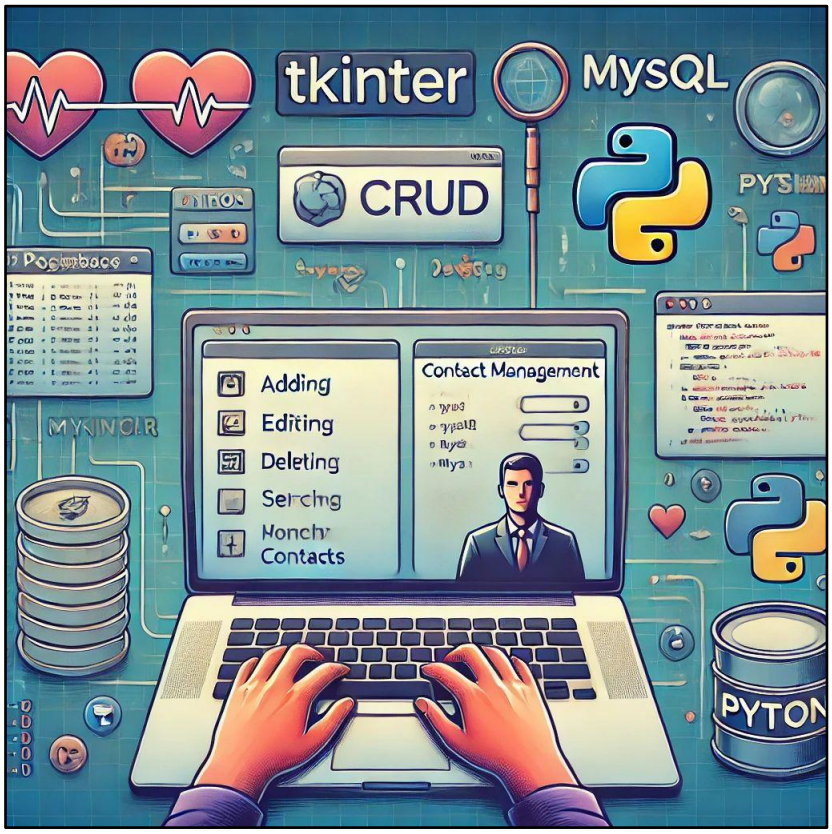

Aplicación CRUD para la Gestión de Contactos con Python, Tkinter y MySQL

UT - 4. Agenda de contactos

Apellidos: XXXXXX

Nombre: XXXXXXX

Nº PC: XX

Centro: I.E.S. La Marisma (Huelva)

Curso: 2º Técnico Superior en Desarrollo de Aplicaciones Web

Asignatura: Horas de Libre Configuración – Python

Profesor: Gonzalo Cañadillas Rueda

Fecha: DD/MM/AA

1. Introducción

La gestión eficiente de datos es un aspecto fundamental en el desarrollo de aplicaciones. En este documento, se detallará el proceso completo de creación de una aplicación de escritorio para la **gestión de contactos**, utilizando el lenguaje de programación **Python**, el sistema de gestión de bases de datos **MySQL** y la librería de interfaces gráficas **Tkinter**.

Esta aplicación permitirá al usuario realizar operaciones **CRUD** (**Create, Read, Update, Delete**), es decir, agregar, visualizar, modificar y eliminar contactos en una base de datos. Además, contará con características avanzadas como **validación de datos**, **búsqueda en tiempo real** y una **interfaz gráfica optimizada**.

1.1. ¿Por qué elegir Python y MySQL?

Python es un lenguaje de programación de alto nivel, ampliamente utilizado en aplicaciones de desarrollo web, inteligencia artificial, análisis de datos y automatización de procesos. Su facilidad de uso y su gran ecosistema de bibliotecas lo convierten en una opción ideal para proyectos de software.

Por otro lado, **MySQL** es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto, conocido por su robustez y eficiencia en la manipulación de grandes volúmenes de datos. Su combinación con Python mediante el conector `mysql-connector-python` permite una integración sencilla y potente.

Tkinter, incluido en la instalación estándar de Python, es una de las bibliotecas más utilizadas para la creación de interfaces gráficas en aplicaciones de escritorio. Su flexibilidad y facilidad de uso permiten diseñar ventanas interactivas sin necesidad de instalar paquetes adicionales.

---

2. Objetivos del Proyecto

Este proyecto tiene como objetivo principal el desarrollo de una aplicación funcional y fácil de usar, que permita gestionar una lista de contactos de manera eficiente. Se trabajará en la implementación de las siguientes funcionalidades:

1. **Registrar nuevos contactos**, almacenando información como nombre, teléfono y correo electrónico.
2. **Mostrar la lista de contactos en una interfaz gráfica**, utilizando una tabla interactiva.
3. **Actualizar información existente**, permitiendo modificaciones en los datos de los contactos.
4. **Eliminar contactos** de la base de datos de manera controlada.
5. **Implementar un sistema de búsqueda en tiempo real**, que permita filtrar los contactos según el nombre o el número de teléfono.
6. **Validar los datos ingresados**, garantizando que los números de teléfono sean válidos y que los correos electrónicos tengan un formato correcto.
7. **Optimizar la interfaz gráfica**, mejorando la experiencia de usuario con estilos personalizados mediante `ttk.Style`.

---

3. Requisitos Previos

Antes de comenzar con el desarrollo de la aplicación, es necesario contar con un entorno de trabajo adecuado y asegurarse de que todas las dependencias estén correctamente instaladas.

3.1. Instalación de Python

Si aún no está instalado, se debe descargar e instalar la última versión de **Python** desde su sitio web oficial:
https://www.python.org/downloads/

Para verificar la instalación, se puede ejecutar el siguiente comando en la terminal o en el símbolo del sistema:

```
python --version
```

Si la instalación fue exitosa, se mostrará la versión de Python instalada.

Nota: Si ya está instalado previamente y es funcional obviar este apartado.

3.2. Instalación de MySQL Server y MySQL Workbench

MySQL es el motor de bases de datos que se utilizará en este proyecto. Para instalarlo, se debe descargar desde:
https://dev.mysql.com/downloads/

Además, MySQL Workbench es una herramienta gráfica que facilita la administración de bases de datos. Se recomienda instalarlo junto con MySQL Server para gestionar los datos de manera más eficiente.

Nota: Si ya está instalado con alguna distribución tipo AMP obviar este apartado.

3.3. Instalación del Conector MySQL para Python

Para que Python pueda interactuar con MySQL, es necesario instalar el módulo `mysql-connector-python`. Esto se hace con el siguiente comando:

```
pip install mysql-connector-python
```

Para confirmar que la instalación fue exitosa, se puede abrir una terminal de Python e importar el módulo:

```
import mysql.connector
print("Conexión exitosa a MySQL")
```

Si no se generan errores, significa que el conector se instaló correctamente.

4. Creación de la Base de Datos

La base de datos es el componente central de la aplicación. Se diseñará una base de datos llamada **`contact_db`**, que contendrá una tabla para almacenar los contactos.

4.1. Diseño de la Base de Datos

Se define la estructura de la tabla **`contacts`** con los siguientes campos:

Campo	Tipo de Dato	Restricción
`id`	`INT`	Clave primaria, autoincremental
`name`	`VARCHAR(100)`	Obligatorio
`phone`	`VARCHAR(20)`	Obligatorio, debe contener solo números
`email`	`VARCHAR(100)`	Opcional, pero debe tener formato válido

4.2. Creación de la Base de Datos en MySQL

El siguiente código SQL permite crear la base de datos y la tabla:

```
CREATE DATABASE contact_db;
USE contact_db;

CREATE TABLE contacts (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    email VARCHAR(100)
);
```

Después de ejecutar este script en MySQL Workbench o similar o en la línea de comandos de MySQL, la base de datos estará lista para usarse.

---

5. Desarrollo de la Interfaz Gráfica con Tkinter

La interfaz gráfica será desarrollada con **Tkinter**, utilizando widgets de `ttk` para mejorar la estética y funcionalidad.

5.1. Componentes de la Interfaz

La ventana principal contendrá los siguientes elementos:

1. **Campos de entrada**: para nombre, teléfono y correo electrónico.

2. **Botones de acción**: para agregar, actualizar y eliminar contactos.
3. **Un campo de búsqueda**: que permitirá filtrar los contactos en tiempo real.
4. **Una tabla de visualización (`Treeview`)**: donde se mostrarán los contactos almacenados en la base de datos.

5.2. Organización de la Interfaz

- **Parte superior**: Campo de búsqueda.
- **Parte central**: Campos de entrada y botones de acción.
- **Parte inferior**: Tabla de contactos.

5.3. Mejora en la Apariencia con ttk.Style

Para mejorar la apariencia visual, se aplicarán estilos personalizados mediante `ttk.Style()`.

Ejemplo de configuración de estilos:

```
style = ttk.Style()
style.configure("TButton", padding=6, font=("Arial", 10), background="#007ACC",
foreground="white")
style.configure("Treeview", font=("Arial", 10))
```

Esto permitirá que los botones tengan un diseño más moderno y profesional.

---

6. Implementación de las Funciones CRUD

Cada una de las funciones CRUD debe estar correctamente implementada y validada.

6.1. Implementación de la Función "Añadir Contacto"

Esta función capturará los datos ingresados y los insertará en la base de datos, asegurándose de que no haya campos obligatorios vacíos.

6.2. Implementación de la Función "Actualizar Contacto"

Permite modificar los datos de un contacto seleccionado y guardar los cambios en la base de datos.

6.3. Implementación de la Función "Eliminar Contacto"

Antes de eliminar un contacto, se solicitará confirmación al usuario para evitar eliminaciones accidentales.

---

7. Validación de Datos en la Aplicación

Uno de los aspectos más importantes en el desarrollo de cualquier aplicación que maneje datos de usuarios es la validación de la información ingresada. Si no se aplican reglas estrictas para validar los datos, es posible que se almacenen registros incorrectos o inseguros en la base de datos. En esta aplicación, se implementarán validaciones para **nombre, teléfono y correo electrónico**.

7.1. Validación del Nombre

El campo de **nombre** es obligatorio y debe contener solo letras y espacios. No se debe permitir que quede vacío o que contenga caracteres especiales o números.

Reglas de validación para el nombre:

- No puede estar vacío.
- Solo debe contener letras y espacios (sin números ni caracteres especiales).

Para implementar esta validación en Python, se puede utilizar una expresión regular (regex).

7.2. Validación del Teléfono

El número de teléfono debe cumplir con ciertos criterios:

- Solo puede contener números.
- Debe tener entre 7 y 15 dígitos.
- No se permiten caracteres especiales ni espacios en blanco.

Para aplicar esta validación, se puede utilizar otra expresión regular.

Si el número de teléfono ingresado no cumple con estas condiciones, la aplicación mostrará un mensaje de error y no permitirá guardar el contacto.

7.3. Validación del Correo Electrónico

El campo de **correo electrónico** es opcional, pero si el usuario ingresa una dirección, esta debe tener un formato válido. Para ello, se utilizará otra expresión regular:

Si el usuario ingresa un correo en formato incorrecto, se mostrará un mensaje de error.

7.4. Implementación de la Validación en la Aplicación

Las funciones de validación deben integrarse con la lógica de la aplicación para asegurarse de que los datos sean correctos antes de enviarlos a la base de datos. Se deben combinar todas las validaciones antes de agregar un contacto con una comprobación previa.

8. Implementación de la Búsqueda en Tiempo Real

Para mejorar la experiencia del usuario, se implementará un **sistema de búsqueda**. Este permitirá filtrar los contactos a medida que el usuario escriba en un campo de búsqueda.

8.1. Funcionamiento de la Búsqueda

- El usuario ingresará un nombre o número en el campo de búsqueda.
- La aplicación filtrará automáticamente la lista de contactos y solo mostrará aquellos que coincidan con el texto ingresado.
- Se utilizará la cláusula `LIKE` de SQL para realizar la búsqueda en la base de datos.

8.2. Implementación de la Función de Búsqueda

Se implementará una función en Python que se ejecutará cada vez que el usuario escriba en el campo de búsqueda.

---

9. Mejora de la Apariencia de la Aplicación con ttk.Style

Por defecto, la interfaz de Tkinter tiene un diseño básico. Para mejorar la apariencia visual, se utilizará `ttk.Style`, que permite personalizar los estilos de los botones, la tabla y otros elementos gráficos.

9.1. Aplicación de Estilos Personalizados

Se configurará la apariencia de los botones y la tabla de contactos:

```
style = ttk.Style()
style.configure("TButton", padding=6, font=("Arial", 10), background="#007ACC",
foreground="white")
style.configure("Treeview", font=("Arial", 10))
style.configure("Treeview.Heading", font=("Arial", 10, "bold"))
```

9.2. Implementación de una Barra de Desplazamiento

Si la lista de contactos es larga, es recomendable agregar una barra de desplazamiento (`Scrollbar`) para mejorar la navegación:

```
scrollbar = ttk.Scrollbar(root, orient="vertical", command=tree.yview)
tree.configure(yscrollcommand=scrollbar.set)
scrollbar.grid(row=5, column=4, sticky="ns")
```

---

10. Pruebas y Evaluación de la Aplicación

Una vez implementadas todas las funciones, es fundamental realizar **pruebas exhaustivas** para asegurarse de que la aplicación funciona correctamente.

10.1. Pruebas a Realizar

1. **Prueba de Inserción de Datos**
   - Intentar añadir contactos con diferentes formatos de nombre, teléfono y email.
   - Verificar que los datos se almacenen correctamente en la base de datos.
2. **Prueba de Validaciones**
   - Intentar ingresar un número de teléfono con letras o caracteres especiales.
   - Intentar guardar un contacto sin nombre.
   - Comprobar que los correos electrónicos inválidos no se guarden.
3. **Prueba de Actualización y Eliminación**
   - Modificar un contacto y verificar que los cambios se reflejen en la base de datos.
   - Intentar eliminar un contacto y confirmar la eliminación.
4. **Prueba de Búsqueda**
   - Buscar contactos por nombre y por número de teléfono.
   - Comprobar que la lista se filtra correctamente sin necesidad de presionar un botón.

---

11. Conclusión

El desarrollo de esta aplicación permitirá implementar una solución robusta y funcional para la gestión de contactos en una base de datos relacional. A lo largo de este documento, se ha descrito en detalle cada uno de los pasos necesarios para su implementación, desde la configuración del entorno hasta la optimización de la interfaz gráfica.

Además de cumplir con las funcionalidades básicas de un sistema CRUD, se han incorporado características avanzadas como **validaciones estrictas**, **búsqueda en tiempo real** y **mejoras en la experiencia de usuario**, lo que convierte a esta aplicación en una herramienta completa y bien estructurada.

Como mejoras propuestas, se podrían agregar funcionalidades como **exportación de contactos a CSV**.