Proyecto: Sistema de Gestión de Contactos

Módulo 2: Fundamentos de programación Python para el análisis de datos

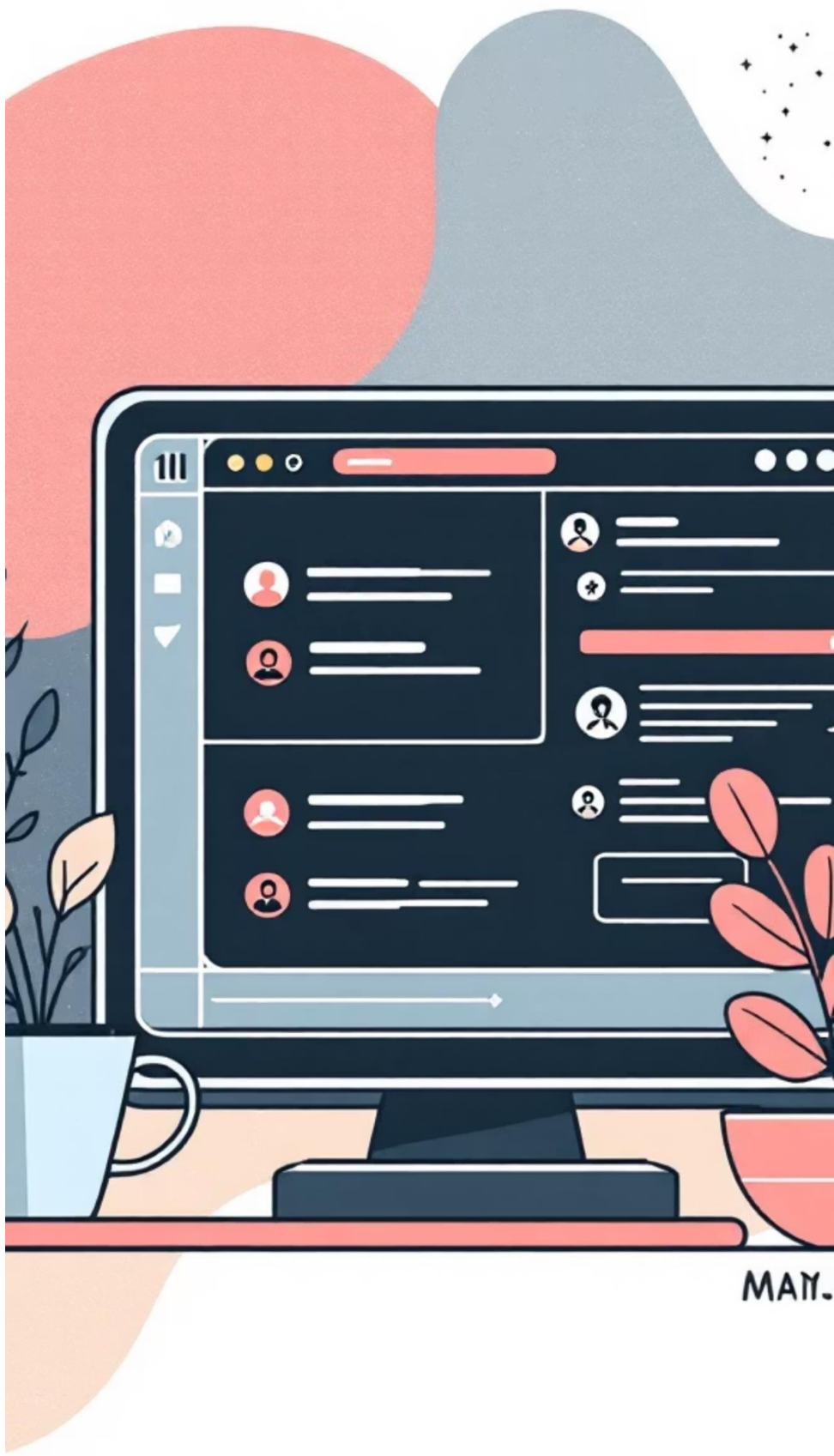
El objetivo principal de este proyecto es desarrollar una aplicación en Python que permita almacenar y gestionar información personal de manera eficiente. Esta aplicación esta enfocada en la intuición del usuario y la organización de los datos.

Interfaz Intuitiva

Diseño pensando en la facilidad de uso para el usuario.

Datos Organizados

Estructura clara para una fácil recuperación y gestión.



Requerimientos del Sistema

Funcionalidades Generales

- Registro de nuevos contactos.
- Edición y eliminación de registros existentes.
- Búsqueda por nombre o número de teléfono.

Requerimientos Técnicos

- Uso de Programación Orientada a Objetos .
- Almacenamiento mediante estructuras de datos, como listas.
- Implementación de pruebas unitarias para validar el correcto funcionamiento.

Arquitectura Técnica: Programación Orientada a Objetos

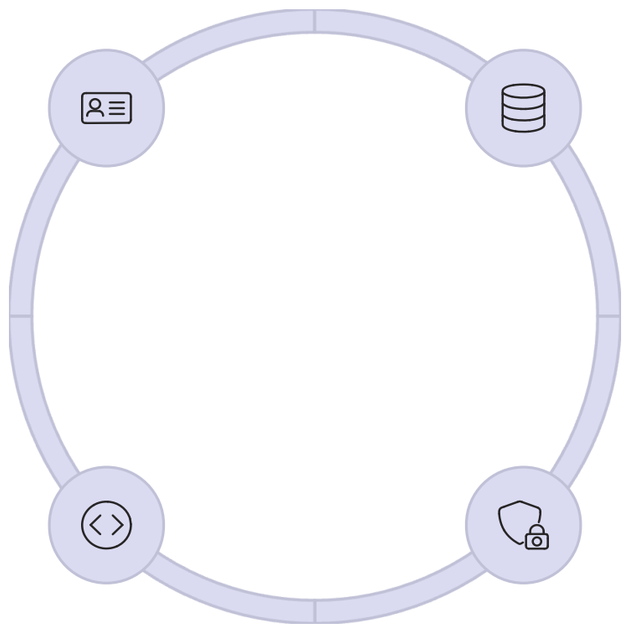
La OOP es fundamental para la modularidad y el mantenimiento del código de nuestro sistema de gestión de contactos.

Clase Contacto

Define atributos como nombre, teléfono, correo, dirección e incluye un método **actualizar**.

Encapsulamiento

Se agrupan datos y comportamientos, facilitando la modularidad y el mantenimiento del código.



Clase SistemaGestionContactos

Actúa como motor del sistema que gestiona la **lista** (`self.contactos`) para almacenar las instancias de la clase **Contacto**.

Modularidad y Mantenibilidad

Al separar la lógica en clases y métodos. el sistema es más fácil de extender y reutilizar sin afectar otras partes del programa.

Estructuras de Datos y Sentencias Iterativas

Listas Dinámicas

Empleadas para gestionar colecciones de contactos, permitiendo añadir y eliminar elementos dinámicamente.

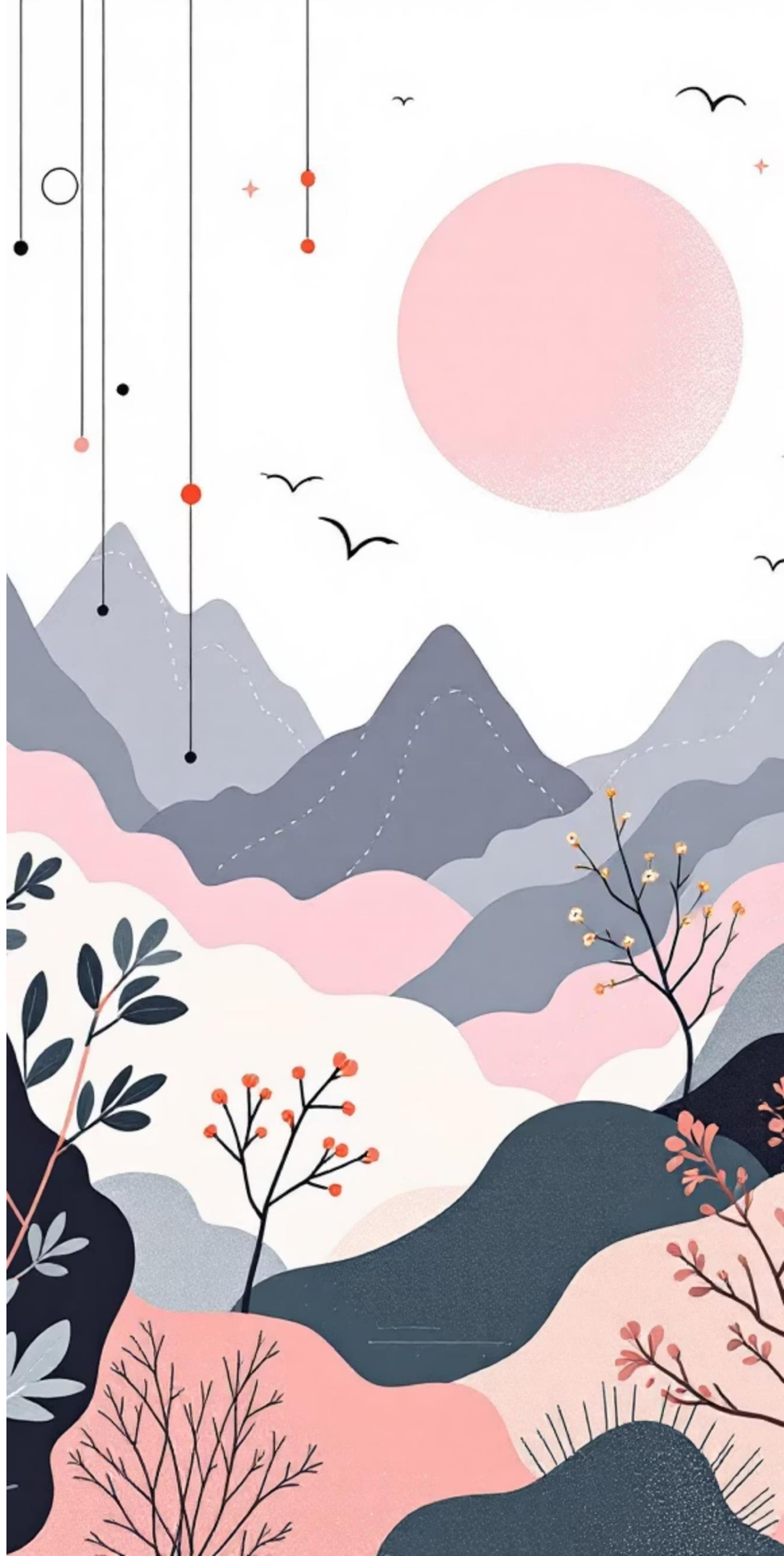
Búsqueda Eficiente

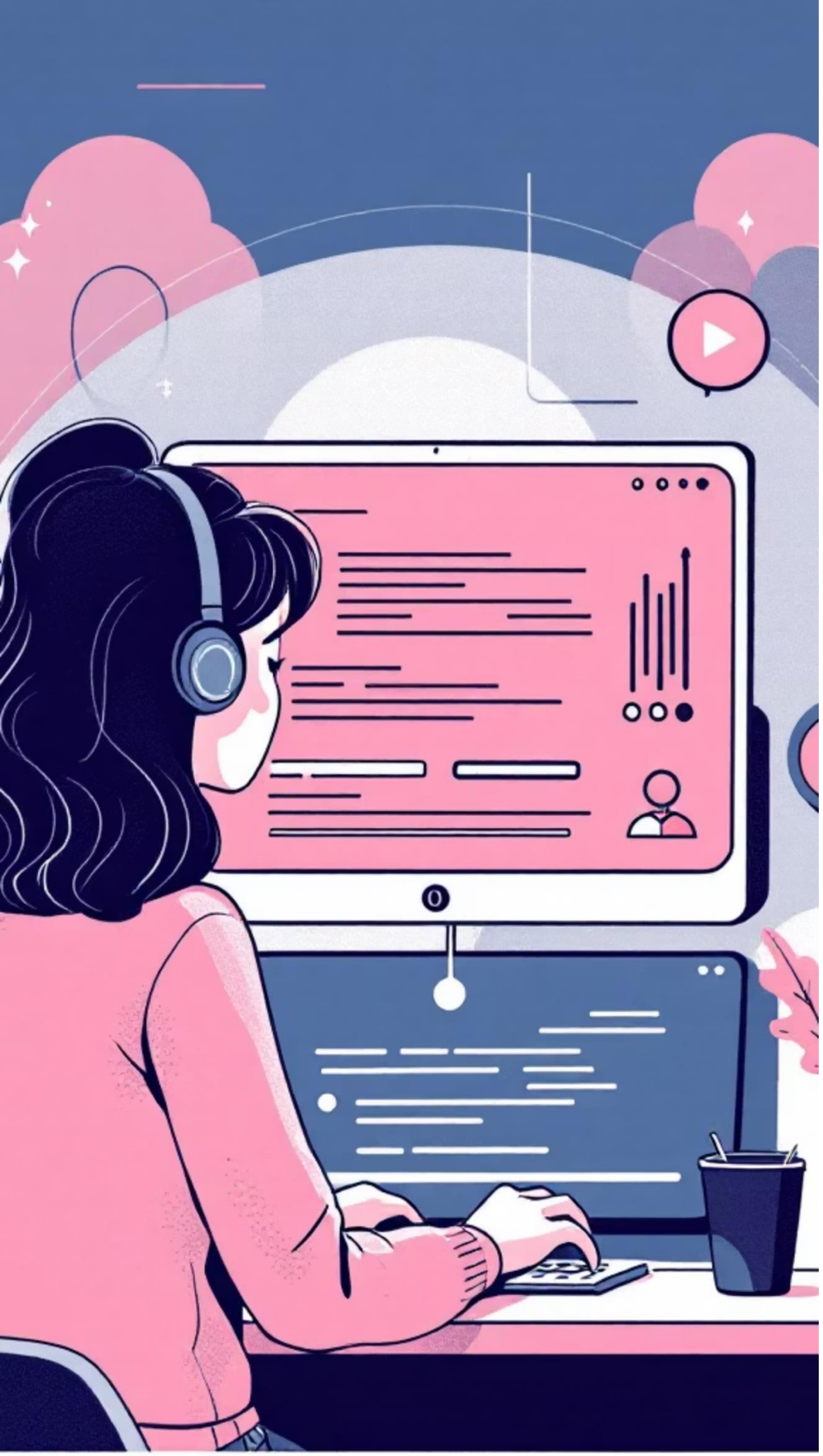
Uso de bucles `for` y comprensiones de listas para para buscar coincidencias de nombre o teléfono.

Validación

Uso de la función `any()` para verificar la existencia de un contacto antes de intentar editarlo o eliminarlo, evitando errores en tiempo de ejecución.

Estas herramientas garantizan la integridad y el rendimiento del sistema al manipular los datos de contacto.





Interfaz de Usuario e Interactividad

La interacción con el usuario es clave para la usabilidad del sistema.

Menú Principal

Implementado con un bucle `while True` mantiene la aplicación activa hasta que el usuario decida salir, ofreciendo una experiencia continua.

Captura de Datos Sencilla

La función `input()` permite al usuario ingresar instrucciones y datos de manera directa desde la consola.

Salida de Información Clara

El uso de **f-strings** y el método `__str__` en la clase `Contacto` asegura que la información se muestre de forma legible y elegante.

Guía de Uso del Sistema - Mini Manual

Esta sección detalla el flujo operativo del sistema, tal como se validó en el prototipo funcional, guiándole a través de cada opción disponible.

Opción 1: Agregar Contacto

Ingrese secuencialmente el **nombre, teléfono, correo y dirección**. Una vez completados, el contacto se almacena automáticamente en la lista del gestor

Opción 2: Editar Contacto

Ingrese el nombre exacto del contacto existente. Si existe, el sistema permite ingresar **nuevos datos** o presionar Enter para dejar el espacio en blanco y así conservar la información actual. Si no existe, aparece un mensaje de error.

Opción 3: Eliminar Contacto

Ingrese el nombre exacto del contacto para remover su registro de forma permanente de la lista. Si no existe, aparece un mensaje de error.

Opción 4: Buscar Contacto

Permite encontrar un contacto de la lista ingresando un nombre o número de teléfono para visualizar rápidamente las coincidencias.

Opción 5: Mostrar Todos los Contactos

Visualice el listado completo de todos los contactos registrados registrados actualmente en el sistema.

Opción 6: Salir

Al seleccionar esta opción, el programa muestra un mensaje de despedida y cierra el ciclo de ejecución de forma segura.

Desafíos y Soluciones Implementadas

Durante el desarrollo, existieron retos respecto a la robustez y la claridad del código.

Desafío: Lógica Repetida

Evitar duplicación de código y mejorar la legibilidad del sistema.

Solución: Métodos Específicos

Creación de métodos específicos en la clase gestora, como `editar_contacto` y `eliminar_contacto` para separar la lógica de negocio de la interfaz de usuario.

Desafío: Búsquedas sin Resultado

Manejar eficazmente escenarios donde no se encuentran coincidencias.

Solución: Condicionales `if-else`

Implementación de mensajes claros para informar al usuario cuando no se encuentran resultados, mejorando la experiencia.

Conclusión y Logros del Proyecto

Prototipo Funcional

Se ha logrado un prototipo funcional que cumple con todos los requisitos solicitados.

Calidad del Código

Prioricé la **legibilidad**, nombres descriptivos e indentación adecuada.

Escalabilidad Futura

El diseño modular (OOP) permite añadir fácilmente nuevos atributos o funcionalidades estadísticas.

Repositorio Oficial