Instituto Politécnico Nacional

Unidad Profesional Interdisciplinaria de Ingeniería y Ciencias Sociales y Administrativas

Ingeniera en informática

Calculadora

Alumnos:

García Nieto Darío Gabriel

Hernández Loyola Luis Enrique

Hernández Ramírez María Fernanda

Sandoval Romero Ángel Tiamath

Unidad de aprendizaje:

Ingeniera de pruebas

Secuencia:

4NV61

Profesor:

Ramon Cruz Martínez

Índice

Resumen

La Calculadora es una aplicación diseñada para usuarios que necesitan realizar operaciones matemáticas simples de manera rápida, eficiente y con la posibilidad de revisar y recuperar cálculos anteriores. Esta herramienta ofrece funcionalidades esenciales como suma, resta, multiplicación y división, junto con un sistema integrado de historial que almacena y permite consultar operaciones pasadas, facilitando la verificación y reutilización de resultados.

La aplicación está construida con una interfaz intuitiva y accesible, optimizada para ser una aplicación de escritorio, garantizando una experiencia fluida y sin complicaciones. Su arquitectura ligera y eficiente asegura un rendimiento rápido incluso en equipos con recursos limitados, mientras que el historial persistente mantiene los datos disponibles entre sesiones.

Además, la calculadora automatiza procesos repetitivos, reduce errores manuales y mejora la productividad en tareas cotidianas que requieren cálculos frecuentes. Con un enfoque en la simplicidad y usabilidad, está diseñada para ser utilizada por cualquier persona, sin necesidad de conocimientos técnicos avanzados, convirtiéndola en una herramienta indispensable para estudiantes, profesionales y usuarios en general.

Introducción

En la era digital, donde la agilidad y la precisión son esenciales en el día a día, las herramientas de cálculo básicas siguen siendo fundamentales para usuarios de todos los ámbitos. Este proyecto tiene como objetivo desarrollar una calculadora sencilla pero eficaz, que permita realizar operaciones matemáticas básicas (suma, resta, multiplicación y división) e incorpore un sistema de historial para almacenar y consultar operaciones anteriores.

El desarrollo de la aplicación se basa en una arquitectura ligera y optimizada, garantizando que los cálculos sean rápidos, exactos y que el historial se mantenga persistente entre sesiones. Además, la calculadora incluye funcionalidades intuitivas de gestión del historial, como la capacidad de revisar, seleccionar y eliminar operaciones previas, lo que brinda flexibilidad y adaptabilidad al usuario según sus necesidades.

Importancia del proyecto

La ejecución de operaciones matemáticas simples es una necesidad constante en entornos educativos, profesionales y domésticos. Una calculadora con historial no solo agiliza estos procesos, sino que también reduce errores humanos al permitir verificar cálculos anteriores rápidamente. Esta herramienta mejora la productividad y la experiencia del usuario, evitando la repetición de operaciones y facilitando el seguimiento de secuencias de cálculo.

Justificación del proyecto

En la actualidad, muchas calculadoras básicas disponibles en el mercado carecen de un sistema de historial funcional, lo que obliga a los usuarios a recordar o anotar manualmente resultados previos, generando interrupciones y posibles inexactitudes. Esta solución moderna permite almacenar y gestionar automáticamente el historial de operaciones, optimizando el flujo de trabajo y ofreciendo una experiencia más eficiente y confiable. Los beneficios incluyen:

1. Reducción de errores en cálculos repetitivos.
2. Ahorro de tiempo al evitar reingresar datos.
3. Mayor control y trazabilidad de las operaciones realizadas.
4. Accesibilidad para usuarios con distintos niveles de conocimiento técnico.

Esta calculadora no solo satisface una necesidad práctica inmediata, sino que también se convierte en una herramienta indispensable para estudiantes, profesionales y cualquier persona que requiera realizar cálculos cotidianos con precisión y comodidad.

Código

```
#LÓGICA DE LA CALCULADORA

import tkinter as tk #Se importa la librería Tkinter (librería estándar de Python para crear interfaces gráficas) con el alías tk

historial = [] #Lista para guardar el historial

def click_boton(valor):

    actual = str(pantalla.get()) #Obtiene lo que ya está en la pantalla

    pantalla.delete(0, tk.END) #Borra la pantalla

    pantalla.insert(0, actual + str(valor)) #Vuelve a escribir lo que tenía más el nuevo valor para ir construyendo la operación


def limpiar():

    pantalla.delete(0, tk.END) #All Clean que borra todo lo que hay en la pantalla


def calcular():  # evalúa la expresión matemática

    try:

        operacion = pantalla.get()

        # Reemplazamos símbolos por los que entiende Python

        operacion = operacion.replace("x", "*").replace("÷", "/")

        resultado = eval(operacion)

        historial.append(f"{pantalla.get()} = {resultado}")  # guardamos con los símbolos originales

        pantalla.delete(0, tk.END)

        pantalla.insert(0, str(resultado))

    except:
```

```
        pantalla.delete(0, tk.END)
        pantalla.insert(0, "Error")

def borrar_uno():
    actual = pantalla.get()
    if actual:  # Si no está vacío
        pantalla.delete(len(actual)-1, tk.END)

#VENTANA PRINCIPAL
ventana = tk.Tk() #crea la ventana principal.
ventana.title("Calculadora GOAT") #le pone título a la ventana
ventana.config(bg="#ff8777")  # fondo rosa

#PANTALLA
pantalla = tk.Entry(ventana, font=("Arial", 24), borderwidth=8, relief="ridge",
justify="right", bg="#CAA3A3", fg="black")
pantalla.grid(row=0, column=0, columnspan=4, padx=10, pady=10)

#LISTA DE BOTONES
botones = [
    ("7", 1, 0), ("8", 1, 1), ("9", 1, 2), ("÷", 1, 3),
    ("4", 2, 0), ("5", 2, 1), ("6", 2, 2), ("x", 2, 3),
    ("1", 3, 0), ("2", 3, 1), ("3", 3, 2), ("-", 3, 3),
    ("0", 4, 0), (".", 4, 1), ("=", 4, 2), ("+", 4, 3),
    ("↑", 5, 0), ("↓", 5, 1)
]
```

```
#CREACIÓN DE BOTONES
operadores = {"+", "-", "x", "÷"}

for (texto, fila, columna) in botones:
    if texto == "=":
        #Botón "=" → azul
        tk.Button(ventana, text=texto, width=10, height=2, font=("Arial", 14),
                  bg="#03627E", fg="white", command=calcular)\
            .grid(row=fila, column=columna, padx=5, pady=5)
    elif texto in operadores:
        #Botones de operación
        tk.Button(ventana, text=texto, width=10, height=2, font=("Arial", 14),
                  bg="#45a056", fg="white", command=lambda t=texto: click_boton(t))\
            .grid(row=fila, column=columna, padx=5, pady=5)
    else:
        #Botones de números y punto
        tk.Button(ventana, text=texto, width=10, height=2, font=("Arial", 14),
                  bg="#CAA3A3", fg="black", command=lambda t=texto: click_boton(t))\
            .grid(row=fila, column=columna, padx=5, pady=5)

#BOTÓN DE ALL CLEAN
tk.Button(ventana, text="AC", width=10, height=2, font=("Arial", 14),
          bg="#ff2c2c", fg="white", command=limpiar)\
    .grid(row=5, column=3, columnspan=4, padx=5, pady=5)

#BOTÓN BACKSPACE O CLEAR ENTRY PARA LOS CUATES
tk.Button(ventana, text="CE", width=10, height=2, font=("Arial", 14),
```

```
        bg="#0e604b", fg="white", command=borrar_uno)\
    .grid(row=5, column=1, columnspan=3, padx=5, pady=5)

#CICLO PRINCIPAL
ventana.mainloop() #Hace que la ventana se quede abierta y responda a los clics en los botones. Sin esto, la ventana se cerraría inmediatamente al ejecutar el programa
```

Desarrollo

Diseño de la Aplicación

La calculadora se desarrolla en Python utilizando un enfoque modular que incluye los siguientes componentes principales:

Lógica de Cálculo:

- Módulo encargado de realizar las operaciones matemáticas básicas (suma, resta, multiplicación, división).
- Validación de entradas para evitar errores (ej. división por cero).

Gestión del Historial:

- Almacenamiento de operaciones en una estructura de datos tipo lista o base de datos ligera (ej. SQLite).
- Cada registro del historial incluye: operación realizada, resultado.

Interfaz de Usuario:

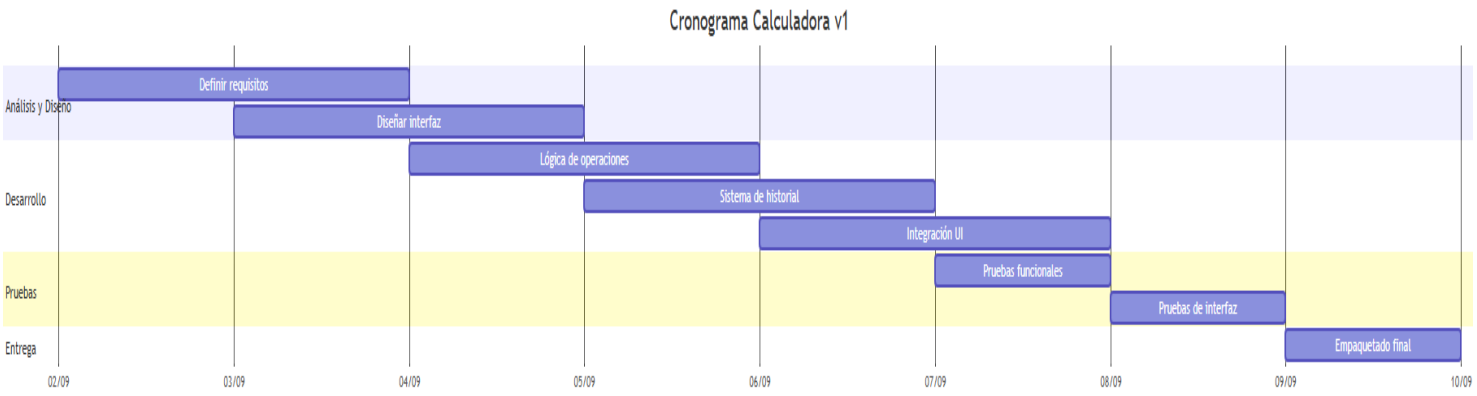
- Implementada con Tkinter (para escritorio) o frameworks web como Flask si es una aplicación web.
- Diseño intuitivo con botones numéricos, operadores y un área de visualización para el historial.

Cronograma

Período: 2 al 9 de septiembre de 2024

Días clave:

- **2-3 sept:** Análisis y diseño (requisitos + interfaz)
- **4-5 sept:** Desarrollo (lógica + historial)
- **6 sept:** Integración completa
- **7-8 sept:** Pruebas y ajustes
- **9 sept:** Entrega final



Funcionalidades Principales
La aplicación permite las siguientes operaciones:

Operaciones Básicas:

- Ejecutar cálculos en tiempo real con soporte para números enteros y decimales.
- Gestión de errores para entradas inválidas.

Gestión del Historial:

- **Crear**: Registrar automáticamente cada operación válida.
- **Leer**: Consultar y filtrar operaciones anteriores.
- **Actualizar**: Editar operaciones incorrectas (opcional).
- **Eliminar**: Borrar registros individuales o limpiar todo el historial.

Persistencia de Datos:

El historial se guarda en un archivo local (ej. JSON, CSV) o base de datos para persistir entre sesiones.

Ventajas de la Aplicación

- **Automatización de Cálculos**: Elimina la necesidad de recordar o anotar resultados manualmente.
- **Acceso Rápido al Historial**: Consulta inmediata de operaciones anteriores sin reingresar datos.
- **Reducción de Errores**: Verificación de cálculos previos para garantizar precisión.
- **Portabilidad y Ligereza**: Python permite ejecutar la aplicación en múltiples plataformas con mínimo consumo de recursos.

Integrantes y su rol

Analista – Hernandez Loyola Luis Enrique

Responsabilidades:

- *Definir los requisitos funcionales (operaciones básicas, gestión de historial) y no funcionales (rendimiento, usabilidad).*
- *Investigar necesidades del usuario y casos de uso típicos (ej: estudiantes, oficina).*
- *Planificar la estructura lógica de la aplicación: flujo de operaciones, persistencia de datos y manejo de errores.*
- *Validar que los requisitos cumplan con los objetivos del proyecto.*

Entregables:

- *Documento de requisitos (F y NF).*
- *Diagramas de flujo o casos de uso.*

Diseñador - Hernández Ramírez María Fernanda

Responsabilidades:

- *Diseñar la interfaz de usuario (UI) para que sea intuitiva y accesible.*
- *Crear prototipos de la calculadora (botones, disposición del historial, colores).*
- *Definir la experiencia de usuario (UX): cómo interactúa el usuario con el historial, retroalimentación visual.*
- *Seleccionar tecnologías para la UI (ej: Tkinter, PyQt, o web con Flask).*

Entregables:

- Prototipos de la interfaz (bocetos o wireframes).
- Guía de estilos (tipografía, colores, tamaños).

Programador - Sandoval Romero Ángel Tiamath

Responsabilidades:

- *Implementar la lógica de las operaciones matemáticas (suma, resta, multiplicación, división).*
- *Desarrollar el sistema de historial (almacenamiento en JSON, SQLite o memoria).*
- *Integrar la interfaz gráfica con la lógica de negocio.*
- *Asegurar que el código sea eficiente, legible y modular.*
- *Manejar excepciones (ej: división por cero, entradas inválidas).*

Tecnologías: *Python, Tkinter (UI), SQLite/JSON (datos).*

Entregables:

- *Código fuente de la aplicación.*
- *Documentación interna del código.*

Tester - García Nieto Darío Gabriel

Responsabilidades:

- *Crear casos de prueba para operaciones matemáticas (valores válidos e inválidos).*
- *Verificar el funcionamiento del historial (guardado, consulta, eliminación).*
- *Probarla interfaz de usuario (usabilidad, responsividad).*
- *Validar la persistencia de datos (que el historial sobreviva entre sesiones).*
- *Reportar bugs y colaborar con el programador para resolverlos.*

Entregables:

- *Plan de pruebas.*
- *Reportes de bugs y pruebas de regresión.*
- *Validación final de que cumple con los requisitos.*

Metodología Estructurada en el Proyecto

La metodología utilizada en este proyecto es estructurada y secuencial, con etapas claramente definidas:

1. **Análisis inicial**: Se identificaron las necesidades clave de la calculadora: operaciones básicas y gestión del historial.
2. **Diseño del sistema**: Se definió una arquitectura modular con lógica de cálculo, persistencia de datos (JSON/SQLite) e interfaz intuitiva (Tkinter).
3. **Definición de operaciones CRUD**: El sistema permite Crear, Leer, Actualizar y Eliminar registros del historial.
4. **Implementación**: Se desarrolló el código en Python integrando lógica, interfaz y almacenamiento.
5. **Pruebas y validación**: Se verificaron las operaciones matemáticas, el historial y la usabilidad.
6. **Documentación**: Todo el proceso fue documentado para garantizar mantenimiento y escalabilidad.

Diseño

Diagrama de secuencia

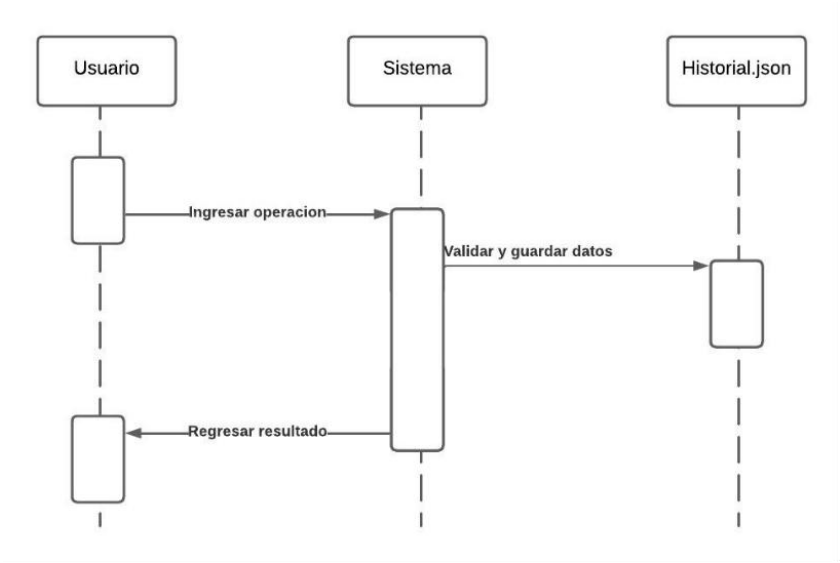
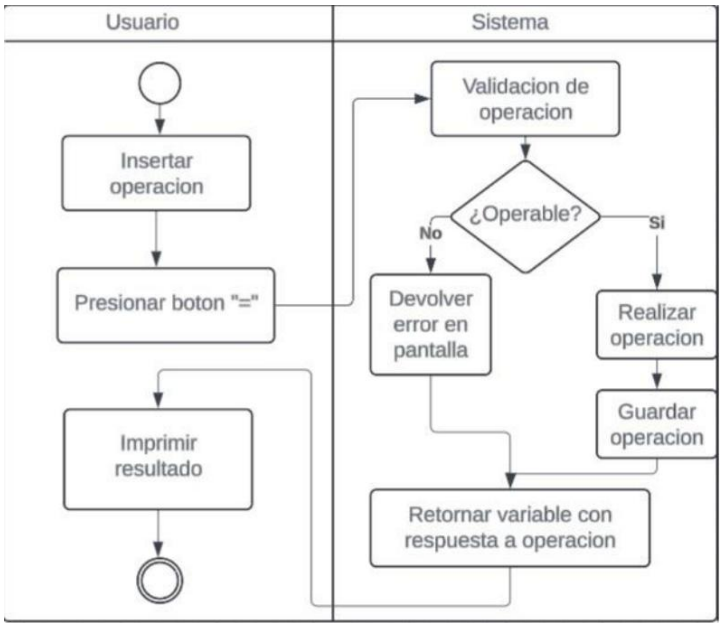


Diagrama de actividades



Requerimientos Funcionales (RF)

RF-01: Operaciones Básicas

1. Permitir la ejecución de sumas, restas, multiplicaciones y divisiones.
2. Validar entradas para evitar errores (ej: división por cero).

RF-02: Gestión del Historial

1. Almacenar automáticamente cada operación realizada (expresión y resultado).
2. Consultar el historial de operaciones anteriores.
3. Actualizar o corregir entradas incorrectas en el historial.
4. Eliminar registros individuales o limpiar todo el historial.

RF-03: Persistencia de Datos

1. Guardar el historial en un archivo (JSON/CSV) o base de datos (SQLite).
2. Cargar el historial automáticamente al iniciar la aplicación.

RF-04: Interfaz de Usuario

1. Mostrar una calculadora con botones numéricos y operadores.
2. Incluir un área de visualización para operaciones y resultados.
3. Mostrar el historial en una sección separable o emergente.

Requerimientos No Funcionales (RNF)

RNF-01: Rendimiento

1. Las operaciones matemáticas deben ejecutarse en menos de 100 ms.
2. El historial debe cargarse en menos de 1 segundo.

RNF-02: Usabilidad

1. Interfaz intuitiva y accesible para usuarios con distintos niveles técnicos.
2. Diseño responsive para adaptarse a diferentes tamaños de pantalla.

RNF-03: Portabilidad

1. La aplicación debe funcionar en Windows, macOS y Linux.
2. Uso de tecnologías ligeras (Python, Tkinter) para bajo consumo de recursos.

RNF-04: Mantenibilidad

1. Código modular y documentado para facilitar actualizaciones.
2. Estructura clara que permita agregar nuevas operaciones (ej: científicas).

RNF-05: Persistencia

1. El historial debe conservarse entre sesiones sin pérdida de datos.

RNF-06: Confiabilidad

1. Manejo de errores para entradas inválidas o fallos inesperados.
2. Disponibilidad del 100% en modo local (sin dependencia de internet).

RNF-07: Seguridad

1. Validación de entradas para prevenir inyección de código (si se usan bases de datos).
2. Protección contra corrupción de archivos de historial.

Pruebas

No. Prueba	Caso de prueba	Pasos	Resultado esperado	Cumple
1	Suma de números enteros	Ingresar 2 + 3 =	5	Si
2	Resta con decimales	Ingresar 5.5 – 2.2 =	3.3	Si
3	Multiplicación	Ingresar 7 x 3 =	21	Si
4	División entre 0	Ingresar 7 / 0 =	Error	Si
5	Limpiar	Ingresar 397 presionar AC	Pantalla vacía	Si
6	Error de sintaxis	Ingresar 2++3 =	Muestra error	Multiplica los signos (Mejor solución)

Conclusión

El desarrollo de esta calculadora básica con historial en Python demuestra cómo una planificación estructurada, con requisitos bien definidos y un enfoque modular, permite crear una herramienta eficaz y fácil de usar. La combinación de operaciones matemáticas esenciales con un sistema de historial persistente no solo resuelve necesidades inmediatas de cálculo, sino que también mejora la productividad y reduce errores. La metodología aplicada, junto con tecnologías ligeras y portables, garantiza un producto robusto, escalable y adaptable a futuras mejoras, cumpliendo con los estándares de calidad y usabilidad esperados por usuarios diversos.