UNEMI
UNIVERSIDAD ESTATAL DE MILAGRO

FACULTAD DE CIENCIAS E INGENIERÍA
CARRERA DE INGENIERÍA DE SOFTWARE

TEMA:

DETECCIÓN DE HERNIA

AUTORES:

ADAN ALI ESCANDÓN ROCA

ASIGNATURA:

MODELO MATEMÁTICOS Y SIMULACIÓN

DOCENTE:

ISIDRO FABRICIO MORALES TORRES

FECHA DE ENTREGA:

20/05/2025

PERIODO:

ABRIL 2025 JULIO 2025

MILAGRO-ECUADOR

Contenido

INTRODUCCIÓN

El presente informe técnico describe el desarrollo de una calculadora científica modular diseñada con PyQt5. El objetivo principal del proyecto es ofrecer una herramienta educativa y funcional que permita realizar operaciones matemáticas complejas de forma intuitiva y visualmente atractiva. Esta calculadora está orientada a estudiantes, docentes e investigadores que requieren una solución digital para resolver problemas relacionados con álgebra lineal, cálculo simbólico, métodos numéricos, generación de números aleatorios, entre otros.

El sistema se ha construido bajo una arquitectura modular, lo que permite su mantenimiento, escalabilidad y personalización. Cada módulo representa un área matemática específica y se integra de forma coherente dentro de una interfaz gráfica unificada. Además, se han implementado mecanismos de validación, gestión de errores y soporte para temas visuales, garantizando una experiencia de usuario fluida y robusta.

OBJETIVO GENERAL

- Desarrollar una calculadora científica modular con interfaz gráfica utilizando PyQt5, que permita realizar operaciones matemáticas simbólicas y numéricas en diferentes áreas, integrando una experiencia de usuario clara, funcional y visualmente atractiva.

OBJETIVOS ESPECÍFICOS

- Diseñar e implementar módulos independientes para operaciones con matrices, polinomios, vectores, gráficas, cálculo simbólico, métodos numéricos, álgebra lineal y simulaciones con Montecarlo.
- Crear una interfaz gráfica intuitiva, uniforme y responsiva que unifique todos los módulos mediante un sistema de navegación claro.
- Aplicar técnicas de validación y manejo de errores que garanticen la integridad de los datos ingresados por el usuario.
- Incorporar representaciones visuales (gráficas y tablas) que ayuden a la

comprensión de los resultados.

- Facilitar la escalabilidad del sistema mediante una arquitectura modular que permita añadir o modificar funcionalidades fácilmente.
- Garantizar la portabilidad del software mediante un empaquetado que genere un ejecutable (.exe) independiente del entorno de desarrollo.

Alcance del sistema

La calculadora científica desarrollada abarca un conjunto amplio de operaciones matemáticas distribuidas en módulos especializados. Cada módulo ha sido diseñado para resolver un tipo específico de problema, facilitando tanto el aprendizaje como la aplicación práctica de conceptos matemáticos.

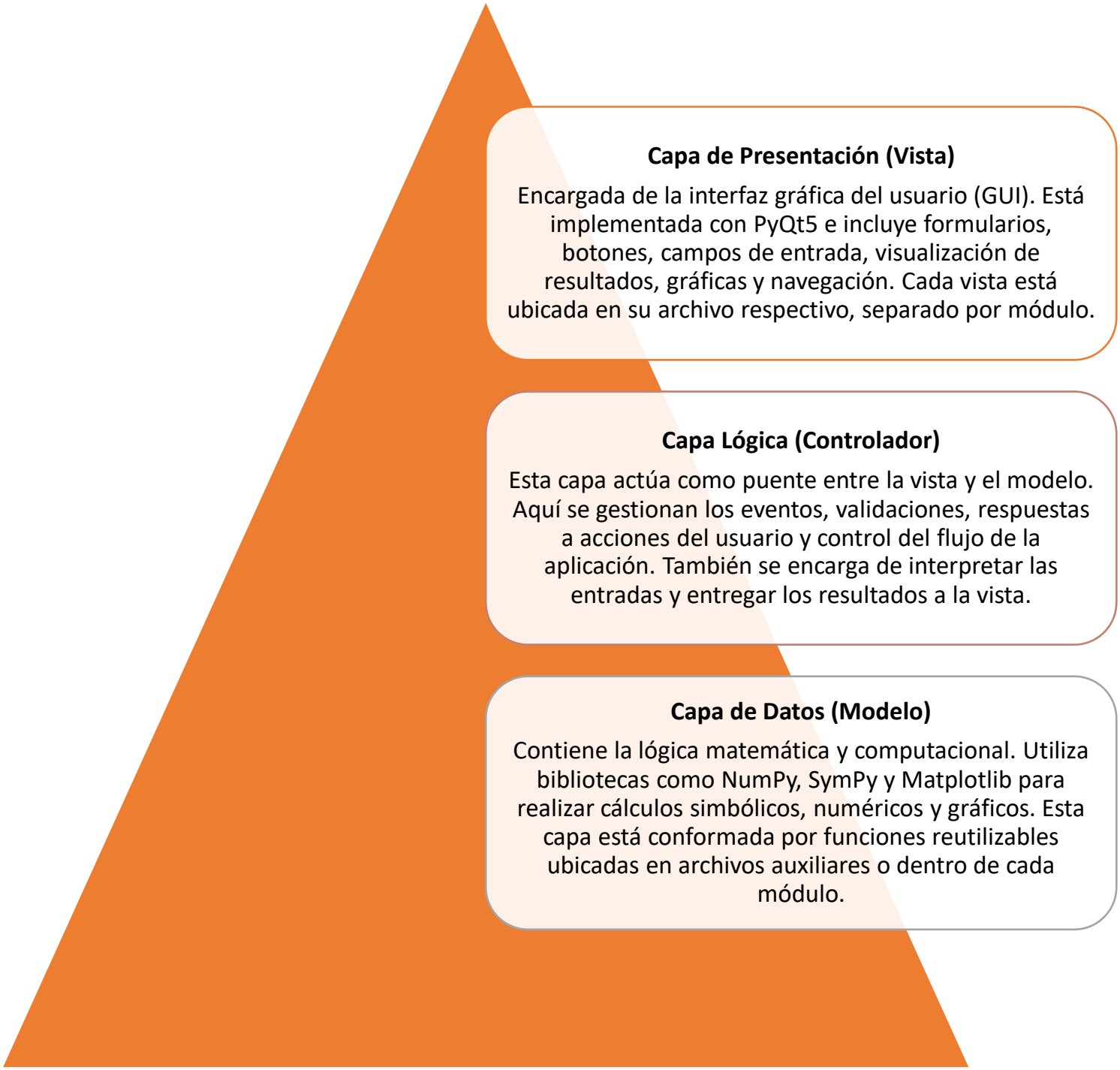
El sistema permite realizar:

- Operaciones básicas y avanzadas con matrices, como suma, resta, multiplicación, determinante, inversa y resolución de sistemas lineales.
- Manipulación y análisis de polinomios, incluyendo suma, multiplicación, derivación, integración y evaluación.
- Cálculos con vectores, como suma, resta, magnitud, producto punto y producto cruzado.
- Representación gráfica de funciones en 2D y 3D.
- Procesos de derivación e integración simbólica, tanto definida como indefinida.
- Resolución de ecuaciones diferenciales mediante métodos numéricos como Euler, Heun, Runge-Kutta y Taylor.
- Cálculo de valores y vectores propios, con visualización gráfica si la dimensión lo permite.
- Simulación de fenómenos mediante el método de Montecarlo, utilizando diversas distribuciones de probabilidad y generadores de números aleatorios.
- El sistema está pensado para ejecutarse en equipos con sistema operativo Windows, sin necesidad de instalar Python ni dependencias externas, gracias a su empaquetado como ejecutable independiente.

Arquitectura del Software

La arquitectura del sistema está basada en el patrón modelo-vista-controlador (MVC) adaptado a un entorno de desarrollo con PyQt5. Cada módulo funciona de manera independiente, pero sigue una estructura común, lo que facilita su integración, mantenimiento y escalabilidad.

El sistema se compone de tres capas principales:



Componentes Principales

- **main.py**: Punto de entrada de la aplicación.
- **main_window.py**: Controla el menú principal, el sistema de navegación y el cambio de vistas.
- **settings.py**: Define temas visuales y parámetros globales del sistema.
- **Archivos por módulo**: Cada módulo tiene su propia vista y lógica separadas, siguiendo una estructura organizada en carpetas (por ejemplo: matrices/, polinomios/, graficas/, etc.).
- **utilidades.py y importaciones.py**: Contienen funciones comunes y configuraciones globales reutilizadas.

Descripción de Módulos

Aquí vamos a describir brevemente cada uno de los módulos funcionales de tu calculadora. Empezamos por el primero:

Módulo de Matrices

Este módulo permite realizar operaciones con matrices de manera interactiva. El usuario puede especificar el número de matrices y sus dimensiones, y luego ingresar los valores manualmente. Entre las operaciones disponibles se encuentran:

- **Suma y Resta de Matrices**: Admite múltiples matrices de igual dimensión. La vista permite agregar matrices dinámicamente y muestra el resultado con scroll y diseño centrado.
- **Multiplicación de Matrices**: Permite multiplicar matrices de dimensiones compatibles. Cada matriz puede tener su propia dimensión, y la interfaz valida automáticamente los tamaños.
- **Determinante e Inversa**: Permite calcular el determinante e inversa de matrices cuadradas. Se utilizan validaciones simbólicas mediante SymPy.
- **Resolución de Sistemas Lineales**: Permite ingresar una matriz de coeficientes y un vector de resultados para resolver sistemas mediante métodos algebraicos.

La interfaz se adapta al tema visual seleccionado, mostrando colores personalizados, botones estilizados y resultados bien organizados.

Módulo de Polinomios

El módulo de polinomios permite trabajar con expresiones algebraicas de una o más variables. Su diseño está orientado tanto a la manipulación simbólica como a la evaluación numérica de polinomios.

Las funcionalidades principales son:

- **Suma y Multiplicación de Polinomios**: El usuario puede ingresar múltiples polinomios de forma dinámica. La operación seleccionada se aplica a todos y se muestra el resultado simbólico de forma clara.
- **Derivación de Polinomios**: Permite seleccionar la variable respecto a la cual se desea derivar. La interfaz adapta los campos según las variables presentes en los polinomios ingresados.
- **Integración de Polinomios**: Se puede calcular la integral indefinida o definida. En el caso de integración definida, se ingresan los límites de integración y el resultado se muestra de forma simbólica.
- **Evaluación de Polinomios**: Una vez obtenida una expresión, se puede evaluar para un valor numérico específico de la variable.

Todas las vistas están diseñadas con estilo visual coherente con el resto del sistema, incluyendo inputs validados, scroll para resultados y presentación visual clara de los resultados simbólicos (incluyendo raíces, fracciones, etc.).

Módulo de Vectores

El módulo de vectores permite realizar operaciones básicas y avanzadas entre vectores en R2\mathbb{R}^2R2 y R3\mathbb{R}^3R3, con una interfaz amigable y visualmente atractiva. Está diseñado para validar las dimensiones y entradas numéricas, asegurando resultados precisos y confiables.

Las funcionalidades disponibles incluyen:

- **Suma y Resta de Vectores**: Permite ingresar dos o más vectores de igual dimensión y realizar operaciones aritméticas básicas.
- **Magnitud (Norma)**: Calcula la longitud de un vector utilizando la fórmula de la norma euclidiana.
- **Producto Punto (Escalar)**: Devuelve un valor escalar como resultado del producto interno entre dos vectores del mismo tamaño.
- **Producto Cruzado**: Solo disponible para vectores de dimensión 3. Calcula un nuevo vector perpendicular a los dos vectores ingresados.

La interfaz valida automáticamente el tamaño de los vectores y notifica cualquier inconsistencia. Además, las entradas incorrectas se resaltan visualmente, y se utilizan controles como QGroupBox, botones estilizados y scroll para una mejor experiencia.

Módulo de Gráficas

El módulo de gráficas permite representar funciones matemáticas de una o más variables mediante visualizaciones en 2D y 3D. Está orientado a mejorar la comprensión visual de los comportamientos funcionales, con un diseño claro y estilizado.

Las principales funcionalidades son:

- **Gráficas 2D**: Representación de funciones de una sola variable (por defecto, xxx). El usuario puede ingresar la función simbólica y el rango de valores, y la gráfica se genera automáticamente usando Matplotlib.
- **Gráficas 3D**: Representación de funciones de dos variables (x,y). La vista permite definir los rangos de ambas variables, ingresando una expresión simbólica para z=f(x,y) y se muestra la superficie resultante en 3D.

El diseño de la interfaz adapta los elementos mostrados según el tipo de gráfica elegida. Las expresiones se validan simbólicamente para evitar errores, y los gráficos se generan en tiempo real, integrados en la misma ventana mediante FigureCanvas de Matplotlib.

Módulo de Cálculo

El módulo de cálculo simbólico permite realizar operaciones de derivación e integración sobre funciones ingresadas por el usuario, usando la biblioteca SymPy para manipulación simbólica. Este módulo está diseñado para mostrar los resultados con notación matemática clara y bien estructurada.

Funciones disponibles:

- **Derivación**: El usuario puede ingresar una función simbólica y seleccionar la variable respecto a la cual desea derivar. El resultado se muestra de forma simbólica, con soporte para funciones compuestas y trigonométricas.
- **Integración Indefinida**: Calcula la integral simbólica sin límites. Se permite integrar respecto a cualquier variable, y el resultado se presenta con constante de integración incluida.
- **Integración Definida**: Permite ingresar los límites inferior y superior, así como la variable de integración. El resultado es una expresión evaluada numéricamente.

La interfaz incluye validadores para evitar caracteres no permitidos, errores de sintaxis o funciones inválidas. Además, los resultados se presentan de forma visualmente comprensible, con notación de raíces, fracciones y funciones comunes estilizadas.

Módulo de Métodos Numéricos

Este módulo permite resolver ecuaciones diferenciales ordinarias (EDOs) utilizando distintos métodos numéricos paso a paso. Está orientado a apoyar la comprensión de las aproximaciones iterativas mediante tablas y gráficas generadas en tiempo real.

Métodos implementados:

- **Método de Euler**: Calcula la solución aproximada de una EDO utilizando incrementos lineales. Es útil como punto de partida para métodos más precisos.
- **Método de Heun (Euler Mejorado)**: Realiza una corrección sobre la predicción inicial para mejorar la precisión.
- **Método de Runge-Kutta de orden 4 (RK4)**: Proporciona una solución de alta precisión al combinar múltiples evaluaciones en cada paso.
- **Método de Taylor de orden 2**: Utiliza derivadas adicionales para mejorar la aproximación local de la solución.

Cada método muestra una **tabla con las iteraciones**, los valores intermedios (k1, k2, etc.) y una **gráfica** comparativa del crecimiento de la solución. También se incluye validación de expresiones simbólicas, detección de errores de sintaxis y prevención de cierres inesperados.

Módulo de Álgebra Lineal

Este módulo se enfoca en el cálculo de **valores propios (autovalores)** y **vectores propios (autovectores)** de matrices cuadradas. Está diseñado para apoyar el análisis de sistemas lineales, estabilidad y transformaciones lineales.

Funciones principales:

- **Cálculo de Valores y Vectores Propios**: A partir de una matriz cuadrada ingresada por el usuario, se determinan sus autovalores y los vectores propios correspondientes, utilizando la factorización simbólica de SymPy.
- **Visualización Gráfica**:
  - Si la matriz es de dimensión 2×2, se muestra una representación gráfica en el plano 2D.
  - Si la matriz es 3×3, se habilita la opción de graficar los vectores propios en 3D.

- Si la dimensión no permite graficar, los botones de visualización se desactivan automáticamente.

La interfaz está completamente validada: verifica que la matriz sea cuadrada, que los elementos sean numéricos, y que no existan campos vacíos. Además, mantiene el estilo visual uniforme del resto de la aplicación, con QGroupBox, botones estilizados, y scroll para visualizar los resultados largos.

Módulo de Montecarlo

El módulo de Montecarlo permite realizar simulaciones estadísticas utilizando números aleatorios generados con distintos métodos, y aplicando sobre ellos varias distribuciones de probabilidad. Este módulo es útil para modelar fenómenos estocásticos, validar algoritmos de generación y observar comportamientos probabilísticos.

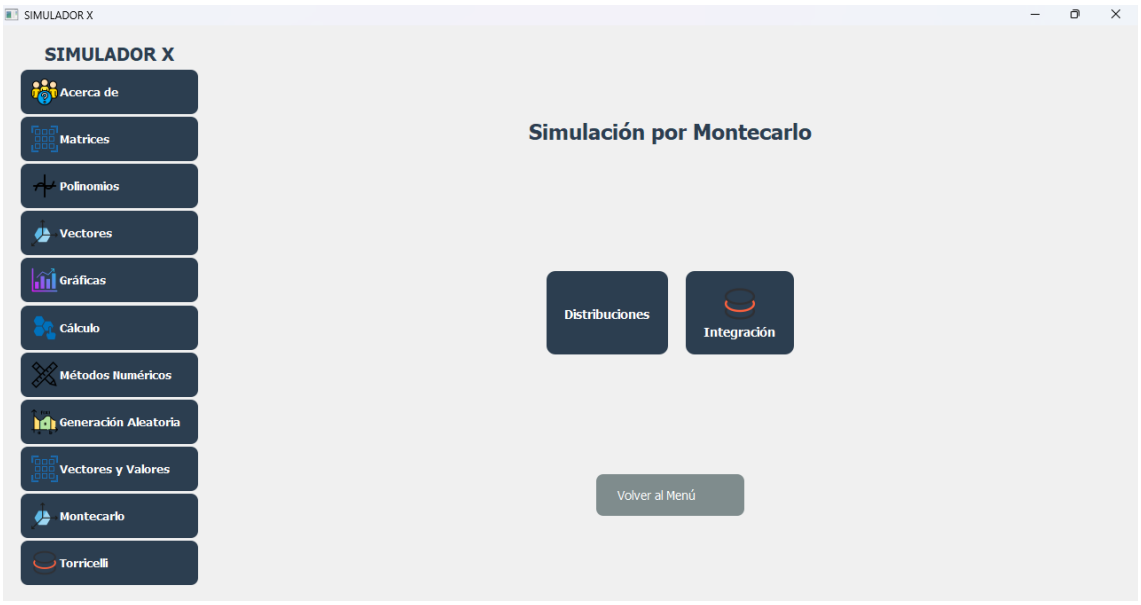
Funciones y características:

- **Métodos de Generación**:
  - Mersenne Twister (nativo de Python)
  - Congruencial Mixto y Multiplicativo
  - Productos Medios y Cuadrado Medio
  - Xorshift, PCG, Tausworthe (LFSR), WELL

- Ruido Físico (placeholder si se implementa con hardware real)

- **Distribuciones Aplicadas**:
  - Uniforme
  - Normal (Gaussiana)
  - Binomial
  - Poisson
  - Exponencial

- **Salida Visual**:
  - Tabla con los valores generados y sus transformaciones.
  - Gráfica de distribución para visualizar los resultados.
  - Scroll principal para navegar fácilmente entre la tabla y la gráfica.



Además, la interfaz permite seleccionar dinámicamente el método y la distribución desde un combo box, adaptando los campos del formulario según corresponda. El módulo incluye validaciones completas para asegurar entradas numéricas correctas y evitar errores de ejecución.

Caso de Estudio: Aplicación del Teorema de Torricelli

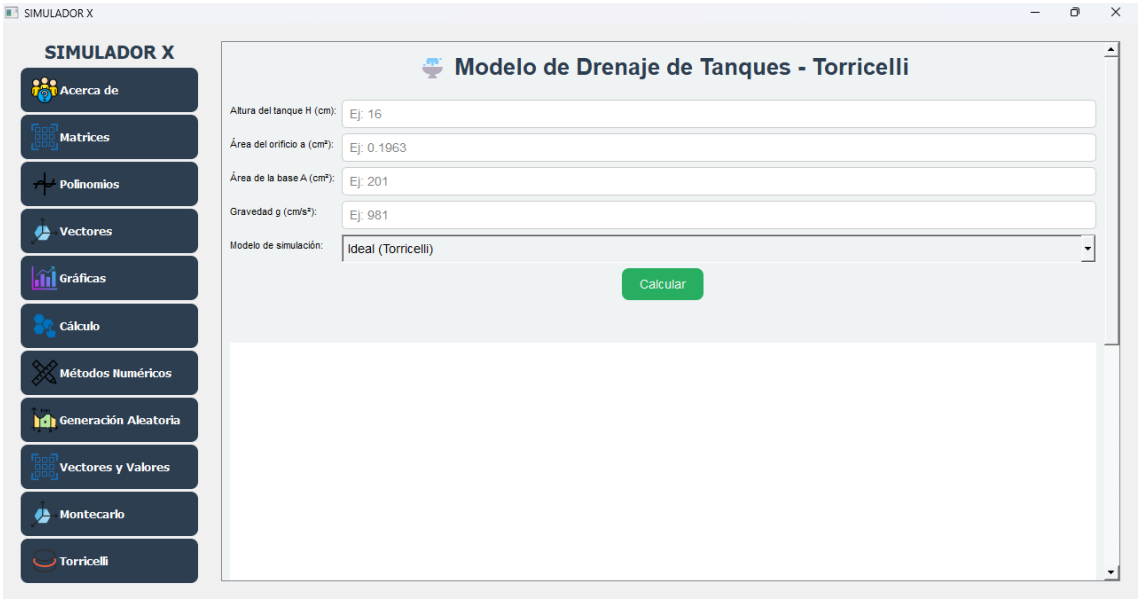
Además de los módulos principales, la calculadora incluye un caso de estudio práctico basado en el **Teorema de Torricelli**, aplicado a un experimento documentado en el archivo EJP-2021Torricelli.pdf.

Este módulo fue diseñado como una herramienta de apoyo para interpretar resultados experimentales y comparar simulaciones con la teoría física. Permite calcular la velocidad de salida de un fluido desde un orificio, utilizando la fórmula:

$$v = \sqrt{2gh}$$

Donde:

- **v:** velocidad de salida del líquido,
- **g**: aceleración gravitacional,
- h: altura del líquido respecto al orificio.



Este apartado sirve como ejemplo de cómo la calculadora puede ser utilizada en aplicaciones interdisciplinarias, integrando física, matemática y programación para resolver problemas reales.

Diseño de la Interfaz de Usuario

La interfaz gráfica de la calculadora ha sido diseñada con un enfoque moderno, modular y amigable, priorizando la experiencia del usuario. El diseño mantiene una estructura coherente entre módulos, con uso de colores personalizados, elementos estilizados y navegación intuitiva. Se aplicaron buenas prácticas de usabilidad para garantizar claridad, orden y respuesta visual ante acciones del usuario.

Herramienta y Framework

La interfaz fue desarrollada utilizando **PyQt5**, un binding de Python para la biblioteca Qt, ampliamente utilizada en el desarrollo de interfaces gráficas robustas.
Se eligió PyQt5 por las siguientes razones:

- Permite construir interfaces altamente personalizables.
- Se integra fácilmente con bibliotecas matemáticas como NumPy, SymPy y Matplotlib.
- Soporta el uso de estilos personalizados con CSS (Qt Stylesheets).
- Facilita la separación entre lógica y vista.

Flujo de Navegación

El sistema utiliza una arquitectura basada en `QStackedWidget`, donde todas las vistas se cargan en una única ventana principal (`main_window.py`). El usuario puede navegar entre los módulos a través de un **menú lateral izquierdo**, el cual despliega submódulos con sangría debajo del botón principal correspondiente.

Cada vista:

- Se presenta en pantalla completa con botones grandes, íconos personalizados y bordes redondeados.
- Incluye un botón “Volver al Menú” que regresa al usuario al menú anterior.
- Aplica scroll cuando el contenido supera el tamaño visible, especialmente útil en módulos con resultados largos (como métodos numéricos o Montecarlo).

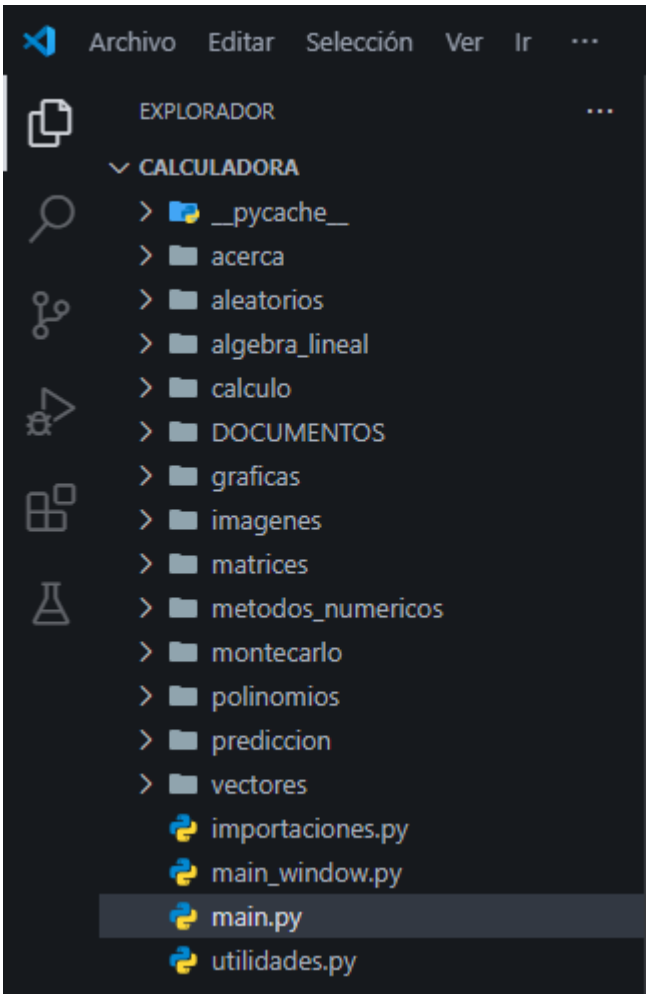
El diseño mantiene la uniformidad entre módulos, asegurando que la experiencia del usuario sea consistente en toda la aplicación.

Detalles de Implementación

Esta sección describe cómo está estructurado el proyecto internamente, las tecnologías empleadas y cómo se gestiona el entorno de desarrollo para garantizar la portabilidad y el mantenimiento del sistema.

Estructura de Carpetas y Archivos

El proyecto está organizado en una arquitectura modular y limpia. La carpeta principal es CALCULADORA, y dentro de ella se encuentran subcarpetas por cada módulo funcional, así como archivos auxiliares y de configuración:



Tecnologías y Librerías Utilizadas

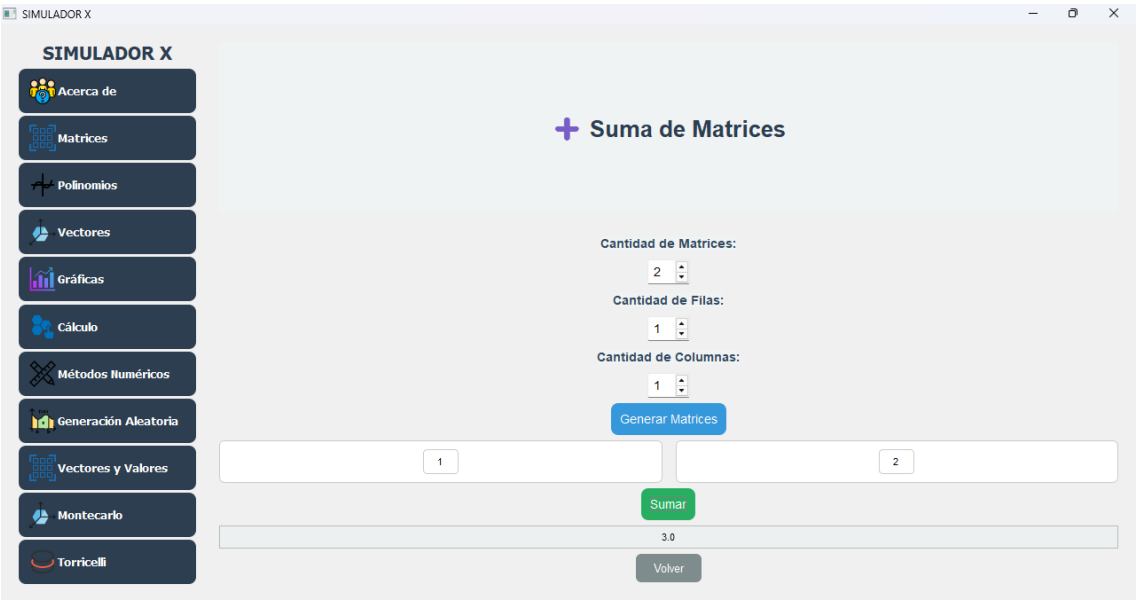
- **Python 3.x**: Lenguaje principal del proyecto.
- **PyQt5**: Para la creación de la interfaz gráfica (GUI).
- **SymPy**: Manipulación simbólica (derivadas, integrales, matrices simbólicas).
- **NumPy**: Operaciones matemáticas con arrays y vectores.
- **Matplotlib**: Generación de gráficas en 2D y 3D.
- **random / math**: Generación básica de números y operaciones matemáticas.

Pruebas Unitarias

Se realizaron pruebas unitarias manuales sobre funciones individuales, especialmente en los módulos matemáticos. Se verificaron casos típicos y extremos para asegurar que los resultados de operaciones como suma de matrices, derivadas, integrales y productos vectoriales sean correctos.

Ejemplos:

- Suma de matrices de dimensiones iguales y distintas (verificación de error).



- Derivadas de funciones polinómicas, trigonométricas y racionales.

- Evaluación de límites definidos en integrales.

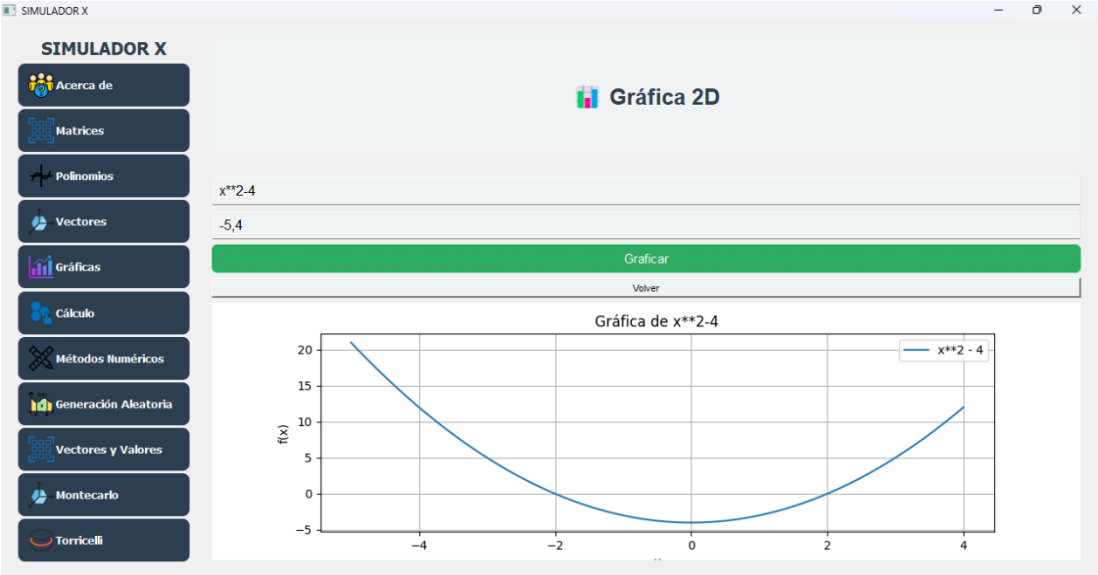
- Productos cruzados en vectores 3D.

Pruebas de Integración

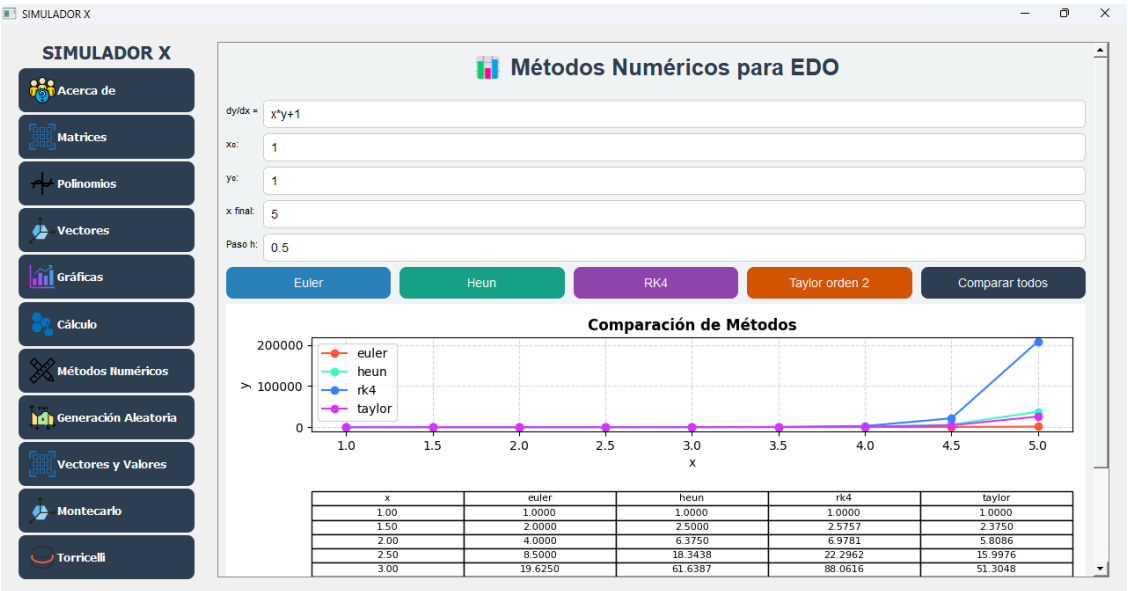
Se evaluó la interacción entre los diferentes módulos y su integración con la interfaz gráfica. Estas pruebas validaron que los datos fluyan correctamente entre vistas, que se mantenga el estado del sistema al cambiar de módulo, y que el sistema reaccione adecuadamente ante entradas inválidas.

Incluye:

- Paso correcto de datos desde la interfaz hacia los cálculos simbólicos.



- Visualización sincronizada entre tabla y gráfica en métodos numéricos.



x	euler	heun	rk4	taylor
1.00	1.0000	1.0000	1.0000	1.0000
1.50	2.0000	2.5000	2.5757	2.3750
2.00	4.0000	6.3750	6.9781	5.8086
2.50	8.5000	18.3438	22.2962	15.9976
3.00	19.6250	61.6387	88.0616	51.3048

- Navegación fluida entre submódulos desde el menú lateral.

Pruebas de Usabilidad

Se aplicaron pruebas informales con usuarios reales (compañeros de clase y docentes) para evaluar:

- Claridad de las vistas y etiquetas.
- Facilidad de navegación entre módulos.
- Respuestas del sistema ante errores y validaciones.

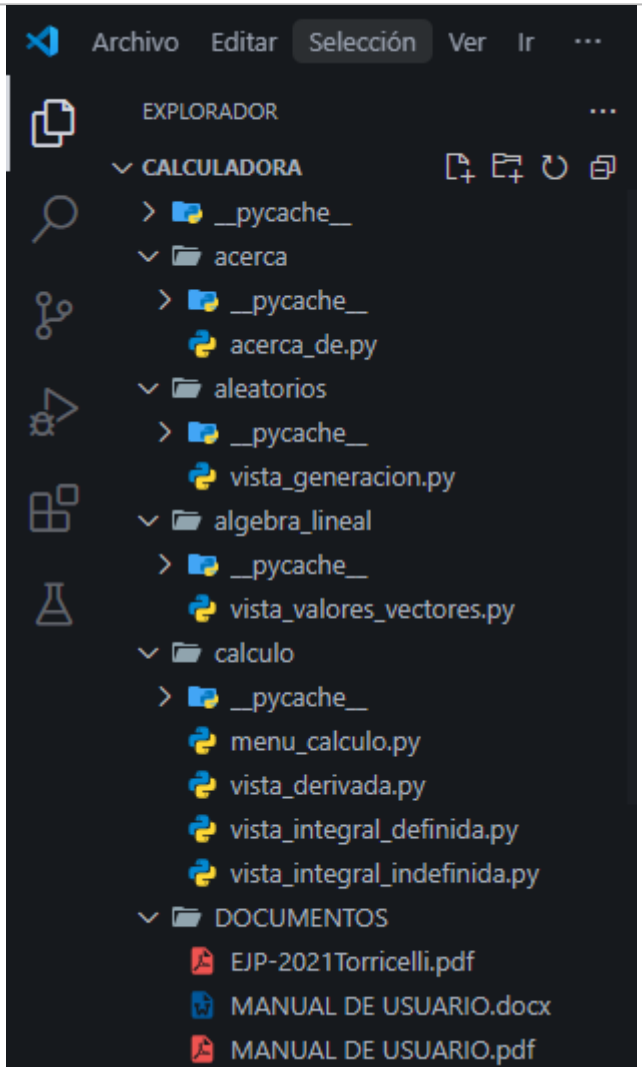
Además, se implementaron mejoras visuales y mensajes amigables para los casos en los que el usuario comete errores comunes, como dejar campos vacíos o ingresar expresiones inválidas.

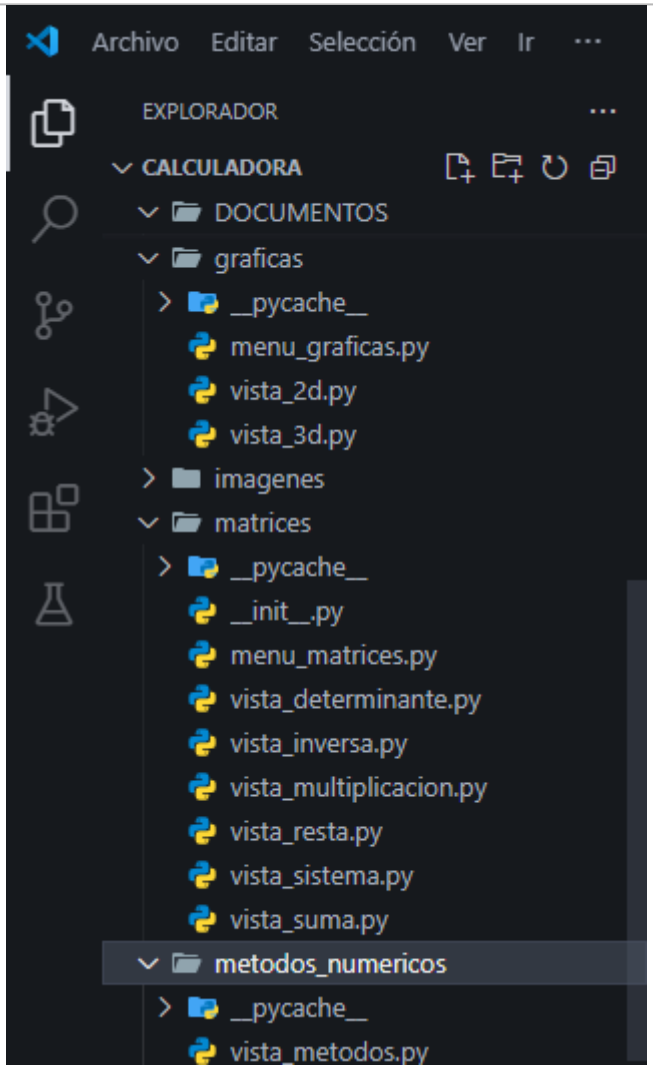
Mantenimiento y Escalabilidad

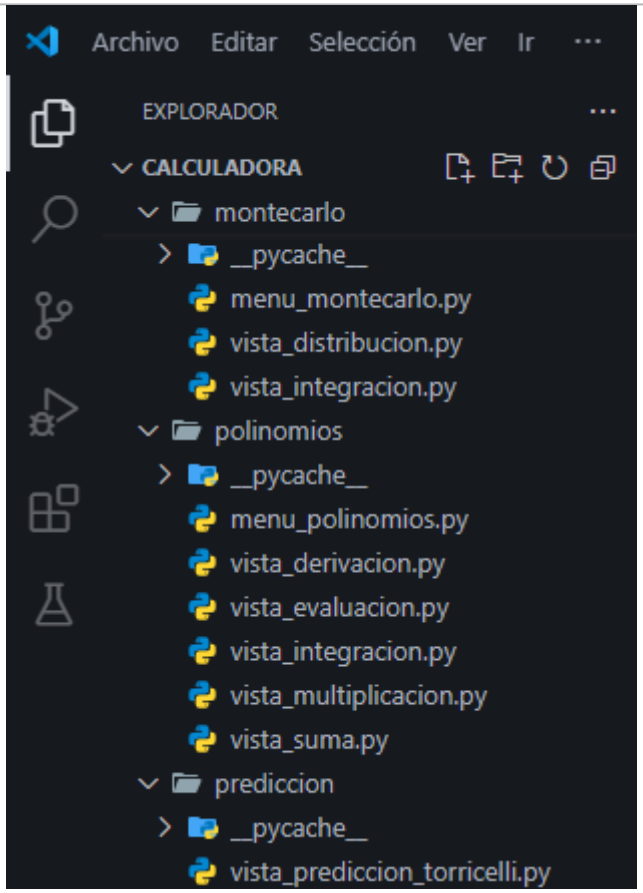
El proyecto ha sido diseñado con un enfoque modular y estructurado, lo que facilita su mantenimiento a corto y largo plazo, así como su escalabilidad para futuras extensiones.

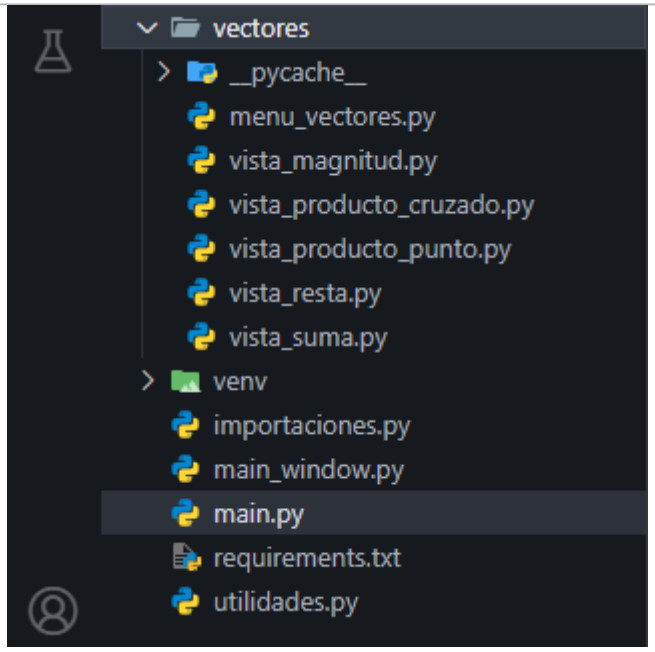
Mantenimiento

- Cada módulo está contenido en archivos separados, organizados por carpetas (matrices/, vectores/, calculo/, etc.), lo que permite localizar y modificar fácilmente cualquier parte del sistema.
- Se han utilizado funciones reutilizables y centralizadas en archivos como utilidades.py e importaciones.py, reduciendo la redundancia de código.
- El sistema de temas visuales está gestionado desde settings.py, permitiendo ajustar colores y estilos sin modificar múltiples archivos.
- Las rutas a imágenes, íconos y recursos están configuradas como **relativas**, lo que evita errores comunes al mover el proyecto entre carpetas o al compilarlo como ejecutable.


Archivo Editar Selección Ver Ir ···
EXPLORADOR ···
CALCULADORA
__pycache__
acerca
__pycache__
acerca_de.py
aleatorios
__pycache__
vista_generacion.py
algebra_lineal
__pycache__
vista_valores_vectores.py
calculo
__pycache__
menu_calculo.py
vista_derivada.py
vista_integral_definida.py
vista_integral_indefinida.py
DOCUMENTOS
EJP-2021Torricelli.pdf
MANUAL DE USUARIO.docx
MANUAL DE USUARIO.pdf


Archivo Editar Selección Ver Ir ···
EXPLORADOR ···
CALCULADORA
DOCUMENTOS
graficas
__pycache__
menu_graficas.py
vista_2d.py
vista_3d.py
imagenes
matrices
__pycache__
__init__.py
menu_matrices.py
vista_determinante.py
vista_inversa.py
vista_multiplicacion.py
vista_resta.py
vista_sistema.py
vista_suma.py
metodos_numericos
__pycache__
vista_metodos.py





Escalabilidad

- La arquitectura basada en QStackedWidget y control central (main_window.py) permite agregar nuevos módulos con facilidad, simplemente añadiendo su vista y conectándola al menú.
- La lógica de validación y procesamiento simbólico o numérico puede extenderse fácilmente integrando más funciones de bibliotecas como SymPy o NumPy.
- El sistema ya contempla soporte para diferentes tipos de entrada (símbolos, matrices, funciones), lo que facilita implementar futuros módulos como sistemas dinámicos, álgebra booleana o estadística avanzada.
- Se ha planificado soporte para simulaciones más complejas (como modelos epidemiológicos), reutilizando el módulo Montecarlo y la infraestructura gráfica ya implementada.

Seguridad y Manejo de Errores

Aunque se trata de una aplicación de escritorio sin conexión a internet, se han tomado medidas para garantizar la **seguridad funcional** del sistema y la **robustez frente a errores del usuario o del entorno**.

Validación de Entradas

- Todas las vistas cuentan con validadores que restringen el ingreso a datos numéricos donde corresponde.
- Se previene el uso de caracteres no permitidos en expresiones simbólicas y se verifica la estructura matemática antes de ejecutar operaciones.
- En el módulo de vectores, matrices y sistemas de EDOs, se valida la dimensionalidad y completitud de los datos antes del procesamiento.
- En integrales definidas y gráficas, se comprueba que los límites y rangos sean numéricamente válidos.

Manejo de Errores

- Se implementaron bloques try-except en puntos críticos del código para evitar cierres inesperados de la aplicación.
- Cuando ocurre un error (por ejemplo, una división por cero o una expresión inválida), se muestra un **mensaje amigable en pantalla**, sin cerrar la aplicación.
- En cada módulo, se ha diseñado un sistema de detección de errores específicos (como matrices no cuadradas, funciones no derivables, o entradas vacías) que permite advertir al usuario de manera clara.
- Los botones o funcionalidades se **desactivan automáticamente** si no se cumplen las condiciones necesarias (por ejemplo, graficar vectores propios solo si la matriz es 2×2 o 3×3).

Seguridad General

- No se manipulan archivos del sistema ni se realizan operaciones que comprometan datos del usuario.

- El sistema funciona localmente y no requiere privilegios especiales para ejecutarse.
- El uso de rutas relativas evita errores por acceso denegado o rutas mal configuradas.

Conclusiones

El desarrollo de esta calculadora científica modular ha permitido integrar múltiples áreas de las matemáticas aplicadas dentro de una sola herramienta interactiva, funcional y visualmente atractiva. Su arquitectura clara y su diseño modular hacen que sea fácilmente mantenible, reutilizable y escalable para futuras mejoras o extensiones.

Entre los logros más destacados se encuentran:

- La implementación exitosa de módulos para álgebra lineal, cálculo simbólico, métodos numéricos y generación de números aleatorios, todos conectados a través de una interfaz unificada en PyQt5.
- El uso efectivo de bibliotecas como SymPy, NumPy y Matplotlib para realizar cálculos avanzados y representar resultados de forma simbólica y gráfica.
- Un diseño de interfaz amigable que facilita la navegación, mejora la experiencia del usuario y proporciona retroalimentación clara ante errores.
- La incorporación de validaciones estrictas y manejo de errores que aumentan la confiabilidad del sistema.