

UNIVERSIDAD  
**NACIONAL**  
DE COLOMBIA

---

# ***Manual Técnico***

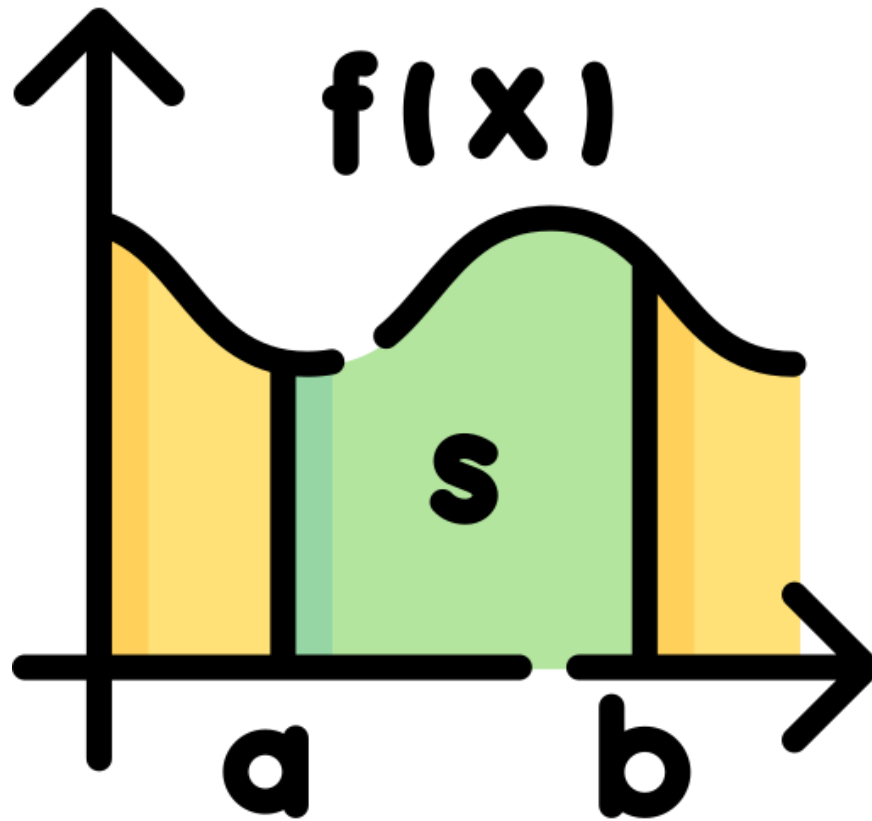
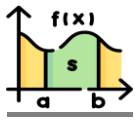
**Proyecto: Calculatory**

**Desarrollado por:**  
**Santiago Camargo Molina**

Programación de computadores

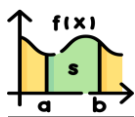
Universidad Nacional de Colombia  
Facultad de Ingeniería

---



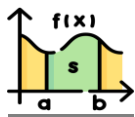
# Calculatory

## MANUAL TÉCNICO



## CONTENIDO

CONTENIDO .....	3
1. Objetivos .....	4
1.1 Objetivos Específicos.....	4
2. Alcance .....	4
3. Requerimientos Técnicos .....	4
3.1 Requerimientos Mínimos de Hardware.....	4
3.2 Requerimientos Mínimos de Software.....	5
4. Herramientas Utilizadas para el Desarrollo.....	5
5. Instalación .....	5
6. Configuración.....	7
7. Usuarios .....	12
7.1 Usuario de base de datos.....	12
7.2 Usuario de aplicación .....	12
7.3 Usuario Administrador .....	12
8. Estructura de la aplicación .....	12
8.1 Arquitectura de diseño.....	13
8.2 Análisis del diseño .....	14
9. Limitaciones de la aplicación .....	25
10. Casos de uso .....	26
11. Errores frecuentes.....	27
12. Referencias.....	28



## 1. Objetivos

El Manual se ha creado con el fin de dar soporte al usuario en cuanto a la estructura de la aplicación, sus funcionalidades, limitaciones y mantenimiento requerido en caso de un fallo.

En este documento, podrá encontrar adicionalmente, la descripción del código fuente y el diseño implementado para la aplicación, como parte de una ruta específica para el operador de sistema.

### 1.1 Objetivos Específicos.

A continuación, se enlista los objetivos que se planean cumplir con la creación de este manual.

- Requisitos mínimos del equipo
- Guía de instalación
- Conexión con la base de datos
- Estructura de la aplicación
- Funcionalidades específicas
- Limitaciones de la aplicación
- Casos de uso
- Errores frecuentes

## 2. Alcance

Este documento está dirigido al: Usuario y Operador del sistema

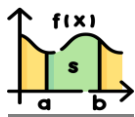
Conocimientos básicos en: Programación orientada a objetos y bases de datos

## 3. Requerimientos Técnicos

A continuación, se presentan los requerimientos mínimos en cuanto a hardware y software, para la implementación y correcto funcionamiento de la aplicación.

### 3.1 Requerimientos Mínimos de Hardware.

Procesador:	Dual-Core
Arquitectura:	x86/x64
Memoria RAM (Mínimo):	4 GB
Disco Duro:	20 GB (Adicional al SO)
Tarjeta Gráfica:	Opcional



## 3.2 Requerimientos Mínimos de Software.

Privilegios de Administrador: Si

Sistema Operativo: Windows7 / MacOS / Linux

## 4. Herramientas Utilizadas para el Desarrollo.

A continuación, se enlistan las diversas herramientas, que permitieron el desarrollo del programa que desea implementar.

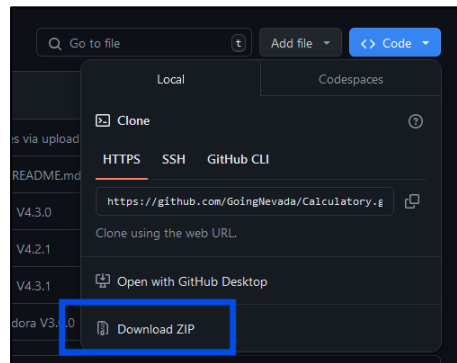
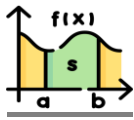
- **Python 3.11 (Tkinter integrado)**  
*Lenguaje de programación interpretado*  
Utilizado para la programación de la funcionalidad de la aplicación
- **Navegador Web**  
*Conexión con aplicación web de gestor de base de datos*  
Utilizado como medio para la conexión entre el gestor de base de datos y la aplicación de escritorio.
- **Firebase**  
*Gestor de base de datos*  
Aplicación web, para la gestión de datos en tiempo real, utilizando una arquitectura de bases no relacionales.
- **Vscode**  
*Entorno de desarrollo de software*  
Utilizado para el desarrollo de la aplicación, permitiendo gestionar de forma rápida el lenguaje Python.

## 5. Instalación

En el siguiente enlace, podrá encontrar los archivos que debe descargar, para realizar la instalación de la aplicación (Debe contar con conexión a internet).

- GitHub: <https://github.com/GoingNevada/Calculatory.git>

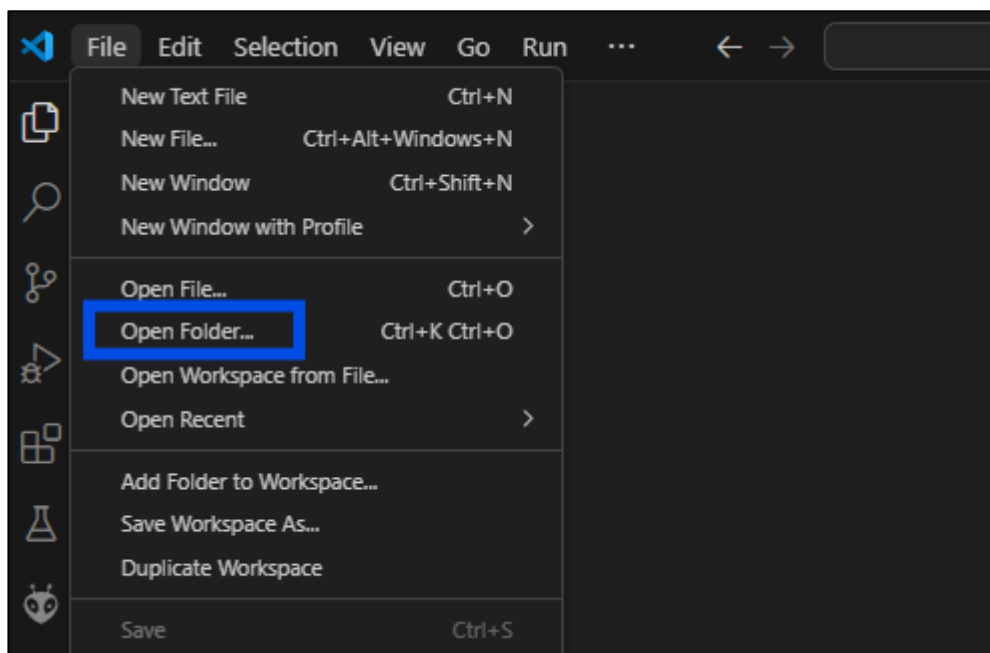
En la pantalla, podrá encontrar la opción “<> código”, en él, se desplegará un menú que le permitirá descargar el archivo comprimido que contine todos los elementos necesarios para la instalación.



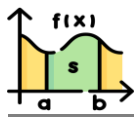
Una vez realizada la descarga, proceda a descomprimir el archivo “.zip”. En la carpeta que se ha creado, aparecerán cinco archivos y una carpeta llamada “resources”.

Nombre	Tipo
resources	Carpeta de archivos
calc_ctrl	Archivo de origen Python
db_ctrl	Archivo de origen Python
gui	Archivo de origen Python
main	Archivo de origen Python
README	Archivo de origen Markd...

Ingresa al entorno de desarrollo Vscod, y en el menú de “File”, se desplegarán diferentes opciones, seleccione “Open folder” y seleccione la carpeta creada al haber descomprimido el archivo “.zip”



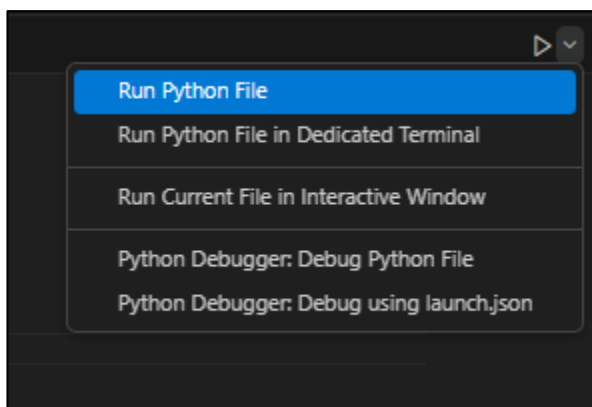
Una vez abierta la carpeta de la aplicación, proceda a instalar las librerías necesarias para su correcto funcionamiento, para esto, puede dirigirse dentro del mismo entorno de desarrollo a la terminal, y ejecutar la siguiente línea de comandos.



```
pip install -r ".\resources\requirements.txt"
```

Este comando instalara automáticamente los requerimientos de la aplicación, si existe algún problema, por favor revise si ha instalado Python correctamente, o en su defecto, revisar la documentación de Python y Vscode.

Una vez finalizada la instalación de las librerías, proceda a abrir el archivo nombrado “**main.py**” y ejecútelo como un archivo Python.

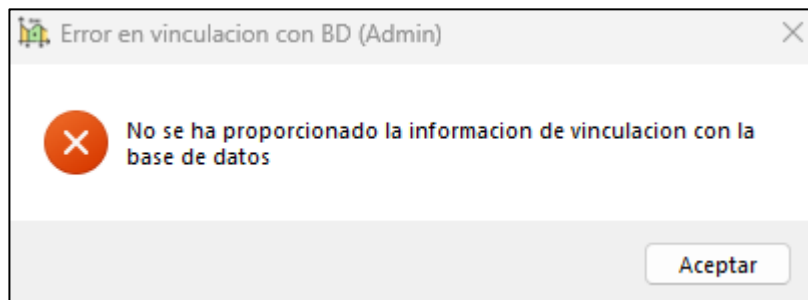
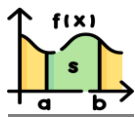


Al ejecutar el script “**main.py**”, se abrirá la ventana principal de la aplicación, como se ejemplifica en la siguiente imagen.



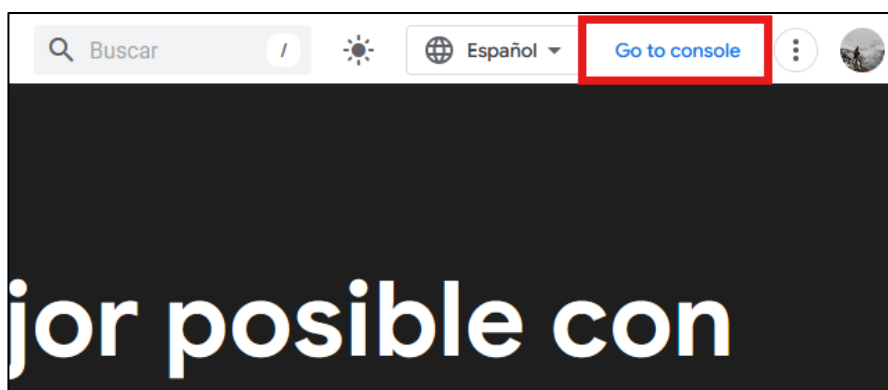
## 6. Configuración

Al iniciar la aplicación por primera vez, le saldrá un mensaje de error, que le indica que “no se ha proporcionado la información de vinculación con la base de datos”, como lo muestra la siguiente imagen.



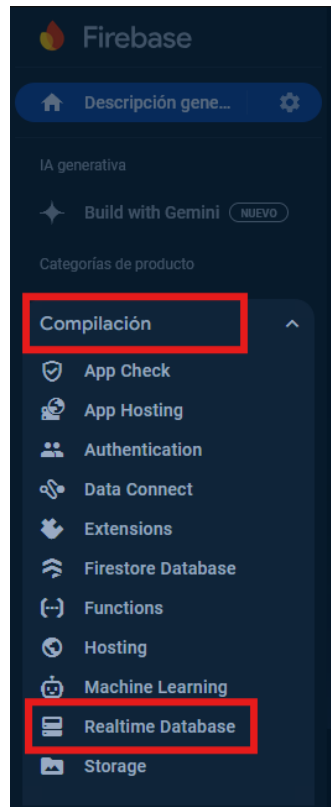
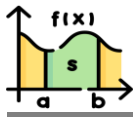
Este mensaje de error se debe a que recién instalada la aplicación, no se cuenta con una clave y dirección de acceso a la base de datos, por consiguiente, debe realizar los siguientes pasos.


1. Cree un nuevo proyecto en la aplicación web **Firestore**, de la siguiente manera:
  - a. Diríjase al siguiente enlace: <https://firebase.google.com/?hl=es>, utilizando su navegador web.
  - b. Inicie sesión con una cuenta de correo electrónico dentro de la plataforma.
  - c. Una vez ingresado a la plataforma con su usuario, diríjase a la opción de “**Go to console**”.

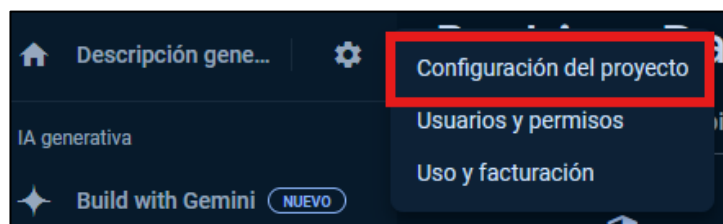


- d. Lo redireccionaran a la consola principal, donde podrá seleccionar la opción de “**Crear un proyecto**”.
  - e. Añada un nombre de su preferencia a su proyecto y en acepte los servicios de Google Analytics
  - f. Una vez realizado el proceso anterior, ingresará a la página principal del proyecto, donde podrá seleccionar en el menú de **Compilación**, la opción de **Realtime Database**.

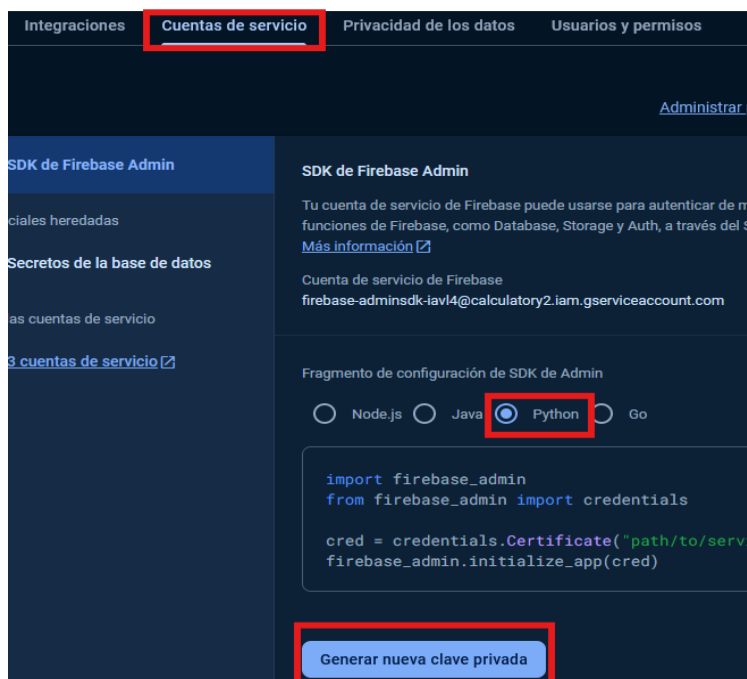
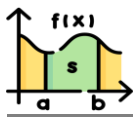




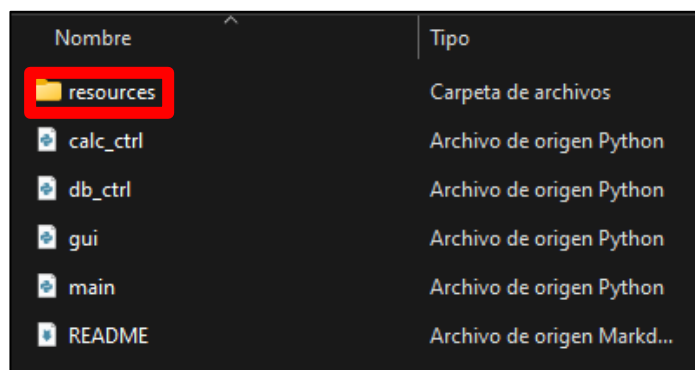
- g. Seleccione la opción de **Crear una base de datos**, deje la configuración de ubicación por defecto en “Estados Unidos”, y posteriormente, seleccione la opción de **Comenzar en modo prueba**.
  - h. Una vez realizadas estas acciones, se creará su base de datos en la aplicación web de **Firebase**
2. Desde el menú de su base de datos en **Firebase**, que se encuentra señalizado por el icono , seleccione la opción de **Configuración del proyecto**.



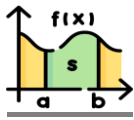
Allí encontrara diferentes configuraciones del proyecto, seleccione el apartado de **Cuentas de servicio**. En esta sección podrá descargar la llave de acceso a su base de datos, para ello, seleccione la opción **Python** que aparece en el lado derecho del submenú, y seleccione la opción de **Generar nueva clave privada**.



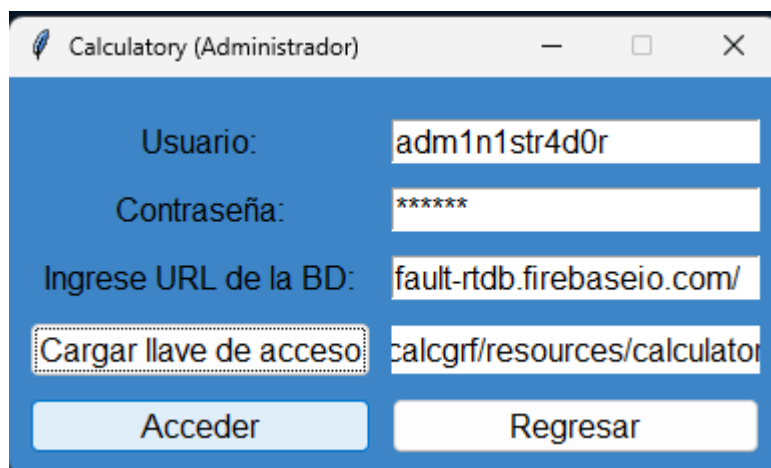
3. Al haber generado una clave privada de su base de datos, se descargará un archivo “.json” el cual, debe guardar en la carpeta nombrada “resources” que se encuentra dentro de la carpeta que descomprimió inicialmente de la aplicación.



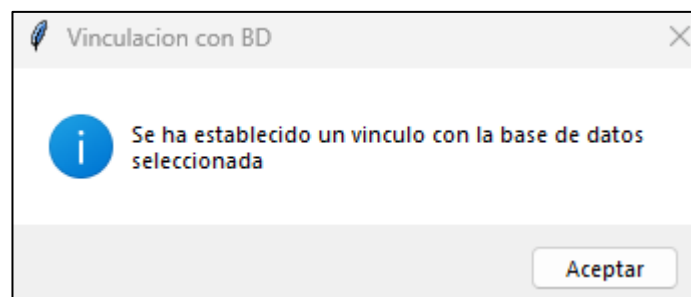
4. En la ventana principal de la aplicación, seleccione la opción “**Acceso Administrador**”, se desplegará otra ventana donde debe autenticarse, utilizando el usuario y clave otorgados por el Autor de la aplicación. Una vez realizada la autenticación, aparecerán 2 opciones que corresponden a la dirección web de la base de datos y el archivo de clave generado anteriormente.
5. En el apartado de “**Ingrese URL de la BD**” pegue el enlace que puede generar directamente desde su proyecto en **Firestore**



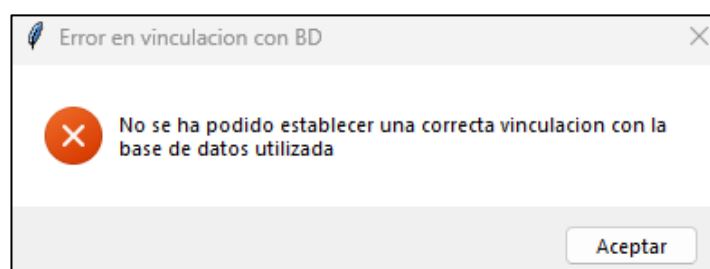
6. En la opción de “**Cargar llave de acceso**” seleccione el archivo guardado en la carpeta “**resources**” y proceda a seleccionar el botón de “**Acceder**”.

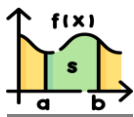


7. Una vez realizado el proceso, le saldrá un mensaje donde se menciona que se ha establecido un vínculo con la base de datos seleccionada.



En caso de que no se haya podido realizar dicha acción, le saltara un mensaje de error, por favor confirme la información suministrada de su base de datos en ese caso.





8. Por último, regístrese como usuario, seleccionando el botón “**Registrarse**” para que pueda comenzar a guardar sus datos en **Firestore**, o si desea, puede iniciar como Anónimo a la aplicación (No se guardaran los datos con esta opción)

## 7. Usuarios

A continuación, se describen los distintos tipos de usuarios posibles, se delimitan su límite de privilegios y demás información que se crea oportuna.

### 7.1 Usuario de base de datos

Los usuarios de base de datos son todas aquellas personas que tienen acceso a la aplicación, y tienen permiso de guardar su información, así como el historial de su uso de la aplicación en la base de datos:

Tipo de usuario: Cualquiera

Formación: Ninguna específica

Privilegios generales a nivel de base de datos: Guardar información

### 7.2 Usuario de aplicación

Los usuarios de aplicación son todas aquellas personas que tienen acceso a la aplicación, y pueden utilizarla para cumplir su propósito (Realizar cálculos matemáticos y graficar funciones).

Tipo de usuario: Cualquiera

Formación: Ninguna específica

Privilegios dentro de la aplicación: Realizar cálculos matemáticos y graficas de funciones

### 7.3 Usuario Administrador

El usuario Administrador es la persona con acceso a la base de datos para crear, leer, actualizar o borrar información dentro de la misma, así como se le concede el privilegio de cambiar la base de datos a utilizar para la recolección de información.

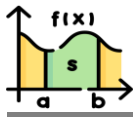
Tipo de usuario: Administrador

Formación: Desarrollo de software, manejo de bases de datos

Privilegios generales a nivel de Administrador: CRUD y configuración de la aplicación

## 8. Estructura de la aplicación

Calculatory fue diseñada para ser una aplicación de escritorio, por ende, sus componentes de arquitectura están orientados a entregar una interfaz y



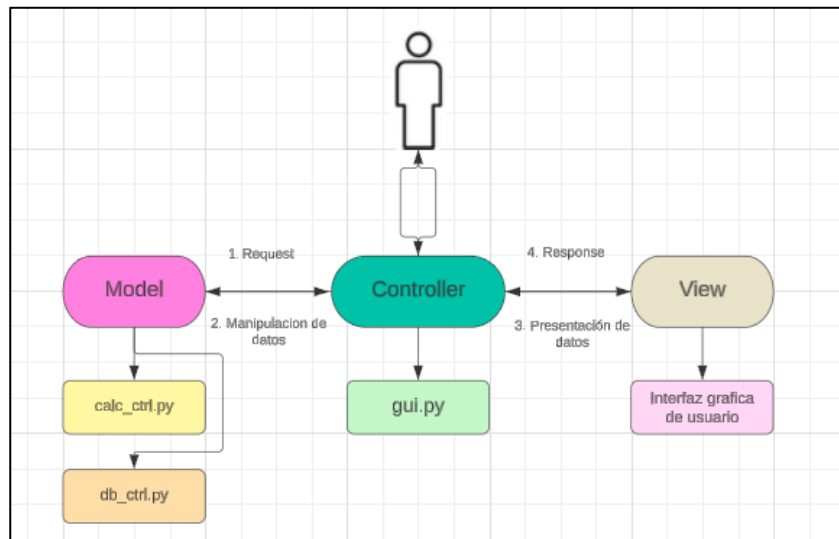
experiencia de usuario de computadora. A continuación, se realizará la revisión técnica de la arquitectura implementada en la aplicación, así como el diagrama de flujo, que ejemplifica el proceso que realiza la aplicación con cada interacción del usuario.

## 8.1 Arquitectura de diseño

Para el desarrollo de Calculatory, se implementó la arquitectura **MVC (Model – View – Controller)** por sus siglas en inglés, en la cual se separarán los componentes de un programa basándose en la responsabilidad de cada uno, por ende, si se requiere hacer una modificación en una parte específica del código, el resto permanece intacto. La arquitectura **MVC** se caracteriza por utilizar 3 componentes.

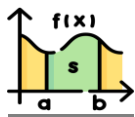
- **Model:** Es el componente que se encarga de dar una estructura a los datos, proveyendo en esta sección la lógica que se debe utilizar para que los datos que ingresen del **Controller**, puedan ser tratados de forma correcta según el comportamiento especificado para ese modelo.
- **View:** El “View” o la vista, se refiere a la interfaz que utiliza el usuario para interactuar con la aplicación de forma directa, en este sentido, puede ser una interfaz gráfica de usuario (**GUI**).
- **Controller:** Este componente se encarga de comunicar las solicitudes entre **View** y **Model**, pues permite la interacción entre estos dos componentes, gestionando las solicitudes para que se dé una respuesta rápida y correcta al usuario.

La arquitectura **MVC** que se implementó en la aplicación es la siguiente.



Como se puede ver en la gráfica, la arquitectura **MVC** de la aplicación, consta del archivo “**gui.py**” el cual se utiliza como el **Controller** de la aplicación, gestionando las solicitudes del usuario por medio de la interfaz gráfica. Por otro lado, en el **Model**, existen 2 archivos con los cuales la aplicación, realiza el tratamiento de los datos. A continuación, se realiza una descripción de cada uno.

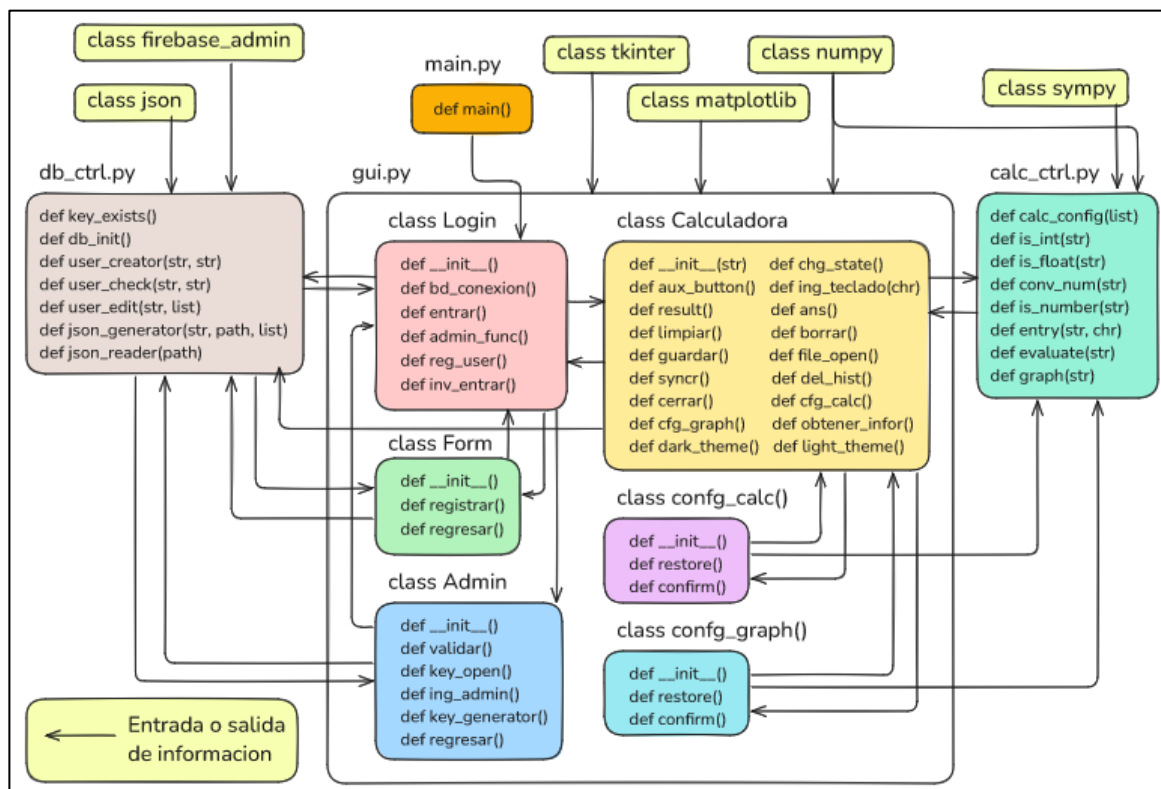
- **calc\_ctrl.py:** En este Script, se establece la lógica utilizada para tratar los datos de tipo numérico, respondiendo a las solicitudes de cálculo y graficación, respondiendo del mismo modo con los resultados obtenidos.
- **db\_ctrl.py:** En este Script, se establece la lógica utilizada en cuanto al



tratamiento de los datos para su gestión en la base de datos.

## 8.2 Análisis del diseño

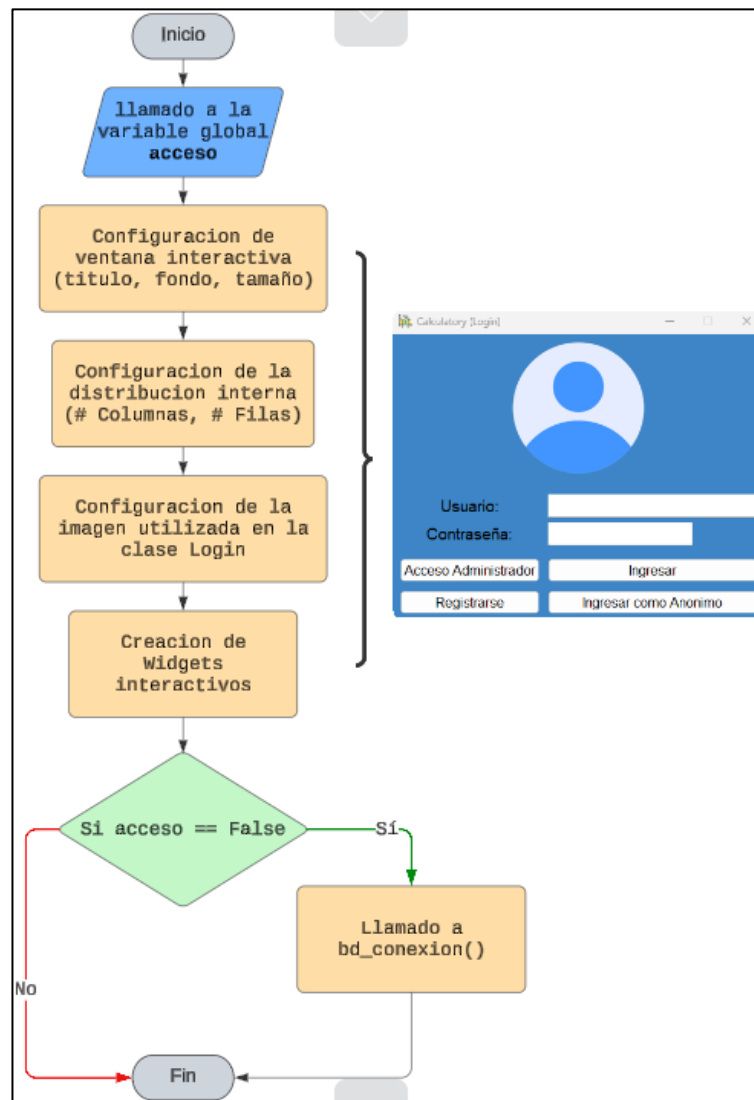
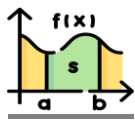
La aplicación utiliza el modelo de clases para distribuir las funcionalidades de acuerdo con el comportamiento que se requiere, a continuación, se realiza la descripción grafica del diseño implementado.



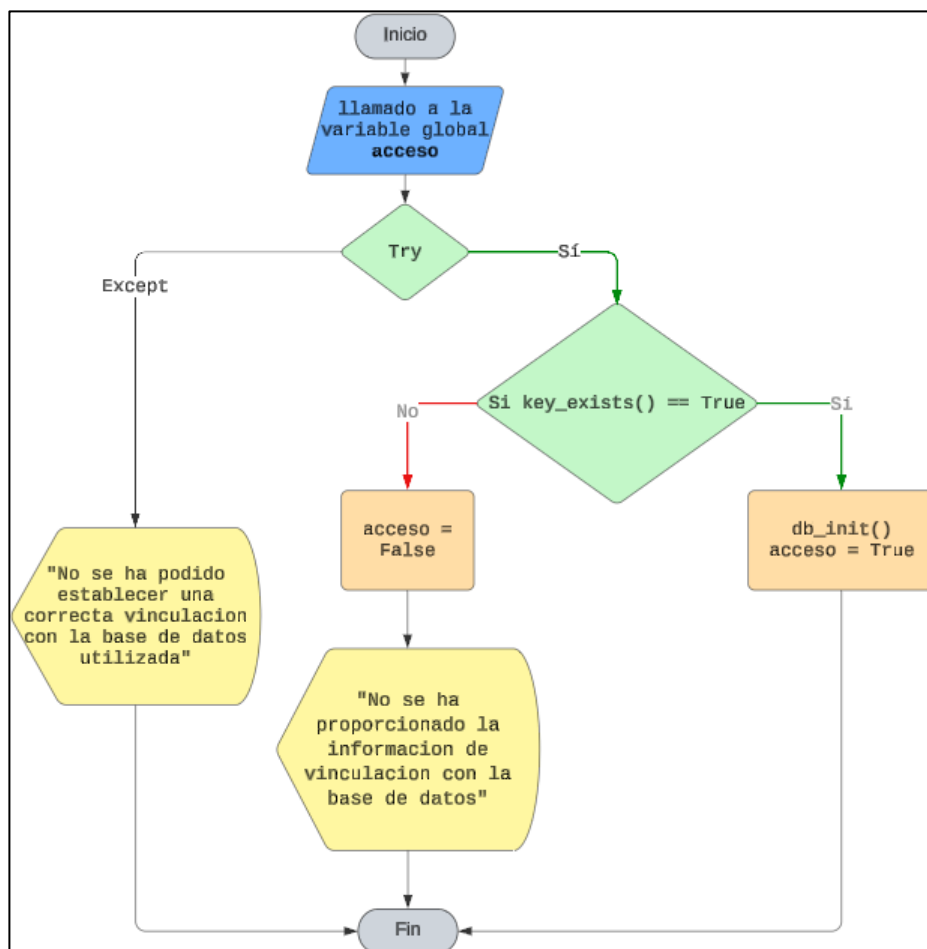
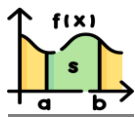
Como se puede observar en la figura anterior, la aplicación trabaja con las librerías **Tkinter**, **Numpy**, **Matplotlib**, **Sympy**, **Firestore\_admin** y **Json**, para lograr una funcionalidad complementaria, además utiliza 6 clases principales ubicadas en “**gui.py**” con las cuales el usuario puede interactuar de forma directa a través de la interfaz de la aplicación.

❖ **Clase Login:** Esta clase, se utiliza principalmente como puente entre la base de datos de **Firestore** y la aplicación, ya que permite administrar el ingreso de los usuarios registrados, o en su defecto, registrar un nuevo usuario, también incluye la funcionalidad para el administrador de la aplicación en cuanto a la opción de proporcionar el registro de la base de datos a utilizar. Contiene diferentes métodos que permiten realizar estas acciones. A continuación, se realizará su descripción.

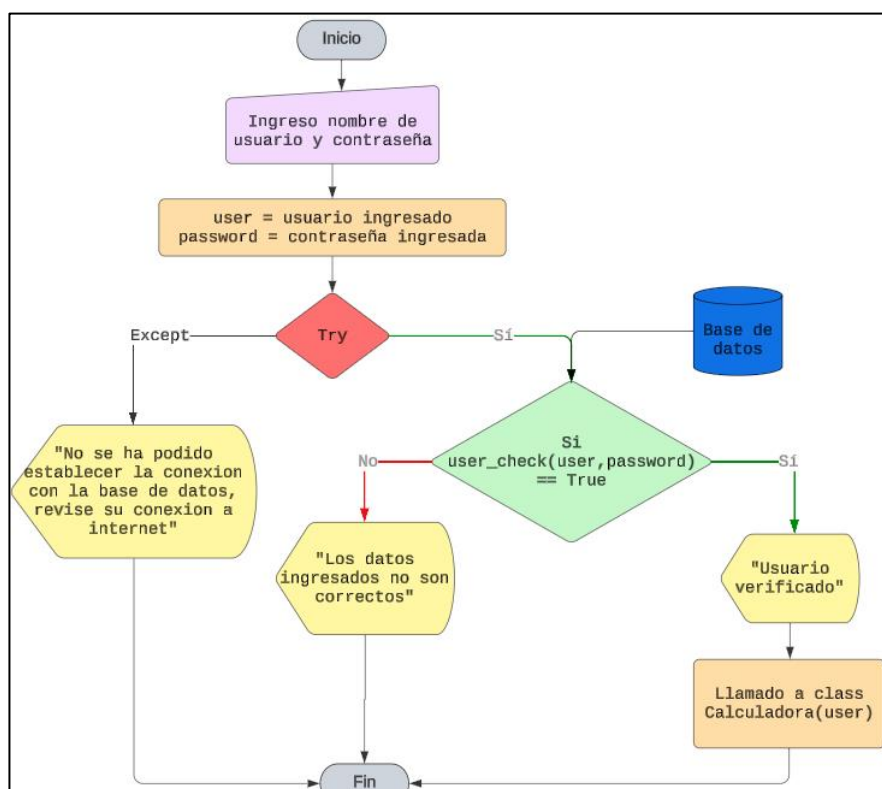
- **Método “\_\_init\_\_”:** Este método es un método constructor de la clase, lo que significa que realiza las inicializaciones de variables, además de las configuraciones de la ventana y widgets respectivamente.



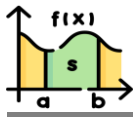
- **Método “bd\_conection”:** Este método utiliza la función **key\_exists** para comprobar que exista una clave de ingreso a la base de datos, cuando se ha iniciado la aplicación, si la clave existe, utilizara la función **db\_init()** para establecer la conexión con la base de datos, de lo contrario, suministrara mensajes de error según sea el caso.



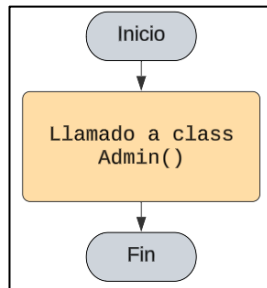
- **Método “entrar”:** Este método permite acceder a la cuenta del usuario, que se encuentra guardada en la base de datos, utilizando la función **user\_check()** para realizar la comprobación de los datos del usuario. Es llamado a través del botón de **Ingresar** de la ventana grafica.



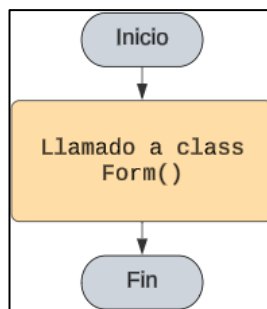




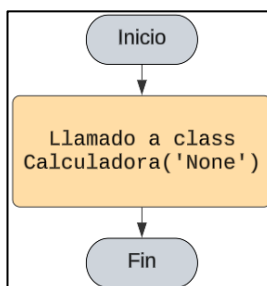
- **Método “admin\_func”:** Este método permite acceder a la ventana de configuración de la base de datos, el usuario debe acceder con el nombre y clave del administrador para realizar dicha acción. Es llamado a través del botón de **Acceso Administrador** de la ventana grafica.



- **Método “reg\_user”:** Este método permite abrir una nueva venta para que el usuario pueda registrarse en la base de datos del Administrador. Es llamado a través del botón de **Registrarse** de la ventana grafica.

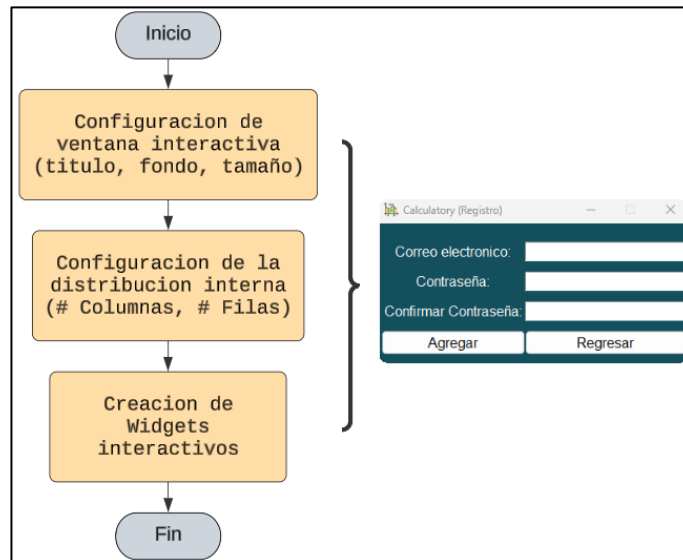
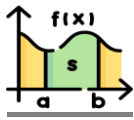


- **Método “inv\_entrar”:** Este método permite ingresar al usuario como Anónimo sin la necesidad de estar registrado en la aplicación. Es llamado a través del botón de **Ingresar como Anónimo** de la ventana grafica.

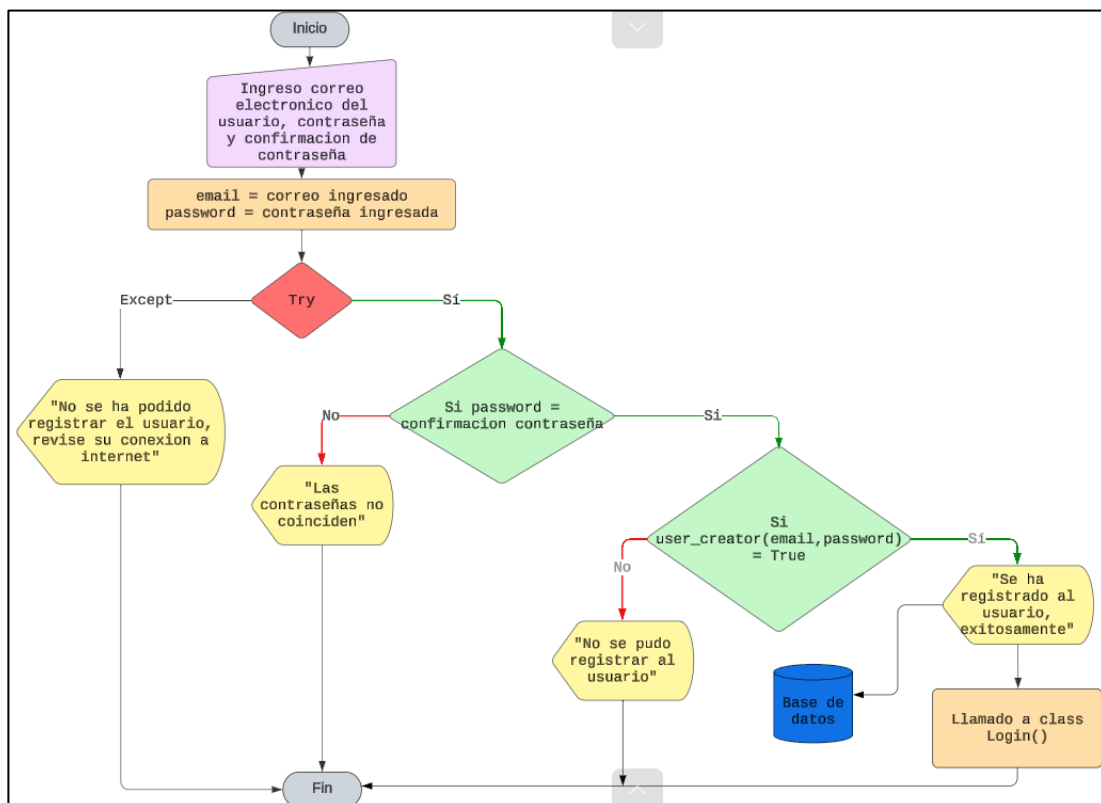


❖ **Clase Form:** Esta clase se utiliza para que el usuario pueda registrarse en la base de datos del Administrador, y puede guardar su información directamente. Utiliza diferentes métodos para obtener y comprobar los datos ingresados antes de realizar el registro.

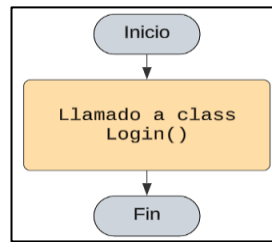
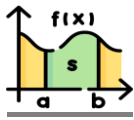
- **Método “\_\_init\_\_”:** Al igual que el método del mismo nombre en la clase Login, el método “\_\_init\_\_” permite realizar la inicialización de las variables, configurar la ventana y sus respectivos widgets.



- **Método “registrar”:** Este método se utiliza para registrar al usuario en la base de datos, obteniendo los datos de correo electrónico, contraseña y confirmación de contraseña, para posteriormente establecerlo como usuario de la aplicación. Es llamado a través del botón de **Agregar** de la ventana grafica.

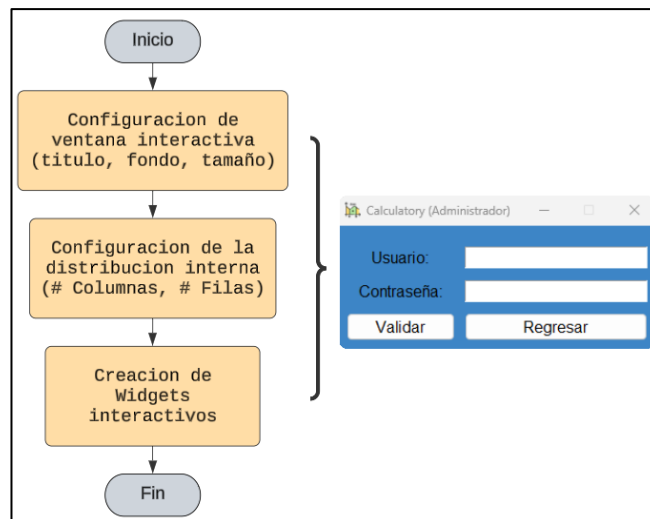


- **Método “regresar”:** Este método permite regresar a la ventana anterior, la cual sería la clase Login. Es llamado a través del botón de **Regresar** de la ventana grafica.

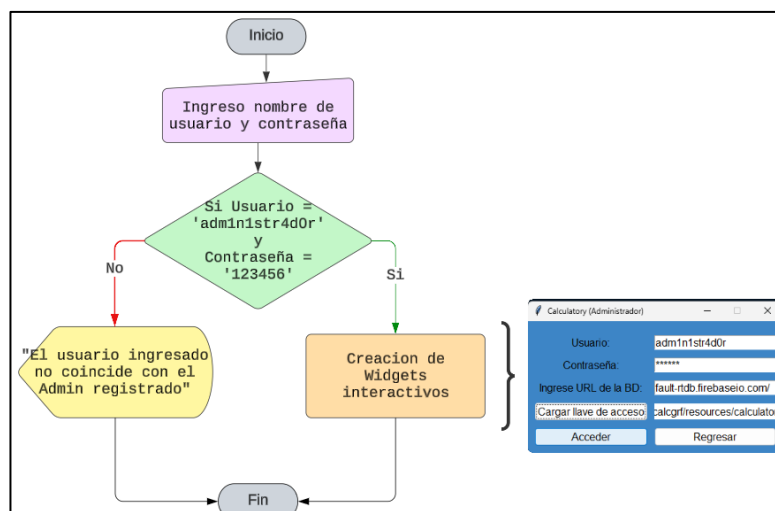


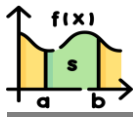
❖ **Clase Admin:** Esta clase se utiliza para realizar la autenticación manual con una base de datos en **Firestore**, utilizando los métodos pertenecientes a esta clase, es posible validar la información de la base de datos como lo es su URL y su Clave de Acceso.

- **Método “\_\_init\_\_”:** Al igual que en los casos anteriores, este método se utiliza como constructor de la clase, realizando las inicializaciones de variables y configurando la ventana y widgets.

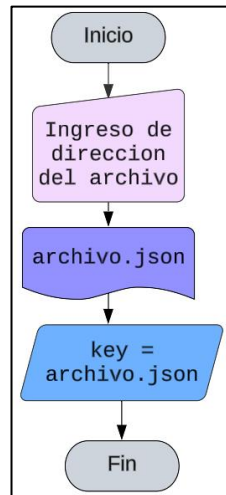


- **Método “validar”:** Este método se utiliza para validar la información ingresada en los campos de Usuario y Contraseña que aparecen en la ventana, para así comprobar que se trate de un usuario Administrador que desea cambiar la configuración de la base de datos. Si los datos son correctos, este mismo método permitirá mostrar otros dos widgets donde se puede añadir la información de la base de datos a utilizar. Es llamado a través del botón de **Validar** de la ventana gráfica.

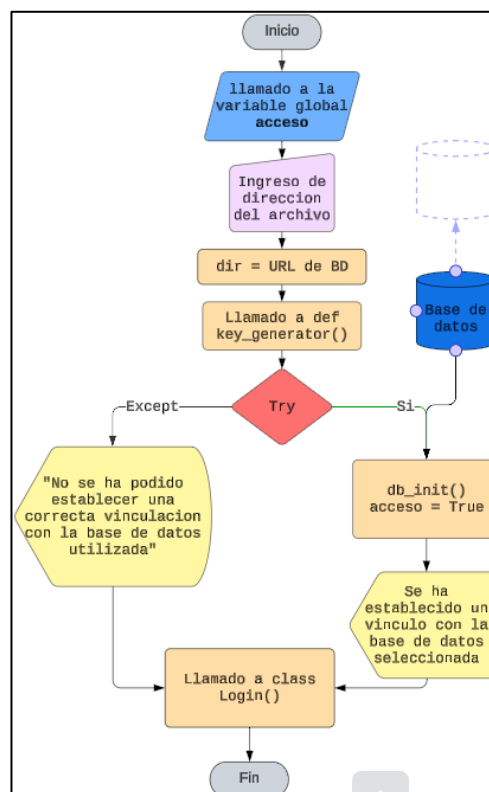




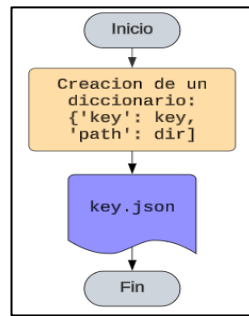
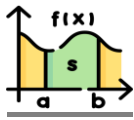
- **Método “key\_open”:** Este método se utiliza para obtener el **Path** del archivo “.json” generado por **Firestore** para la respectiva base de datos a utilizar. Es llamado a través del botón de **Cargar llave de acceso** de la ventana grafica.



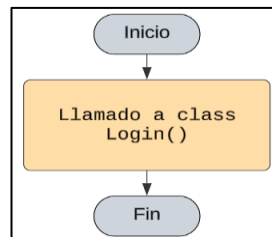
- **Metodo “ing\_admin”:** Este método es utilizado para realizar la vinculacion con la base de datos ingresada, creando un archivo de registro con los datos de la base de datos para vincularse automáticamente a ella, al abrir nuevamente la aplicación. Es llamado a través del botón **Acceder** de la ventana grafica.



- **Método “key\_generator”:** Este método se utiliza para generar un archivo “.json” que contiene la información de la base de datos establecida para guardar los datos de los usuarios

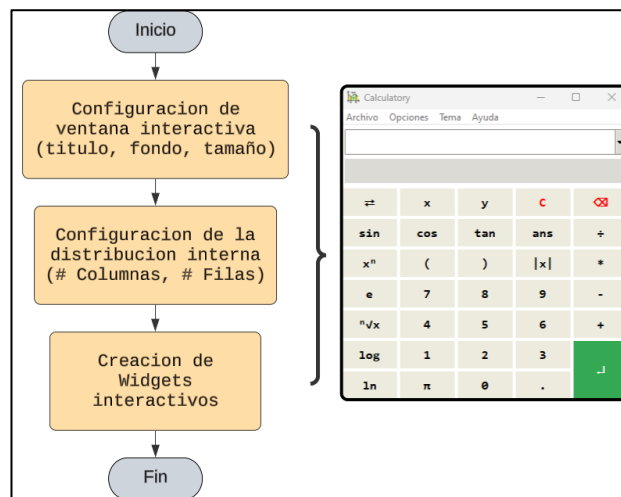


- **Método “regresar”:** Este método permite regresar a la ventana anterior, la cual sería la clase Login. Es llamado a través del botón de **Regresar** de la ventana grafica.

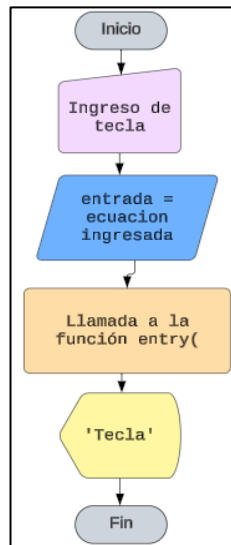
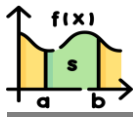


- ❖ **Clase Calculadora:** En esta clase están todos los métodos que permiten realizar las principales funcionalidades de la aplicación, las cuales serían realizar cálculos matemáticos y graficar funciones de dos dimensiones. Esta clase recibe un parámetro al ser llamada, el cual se refiere al nombre del usuario que accede.

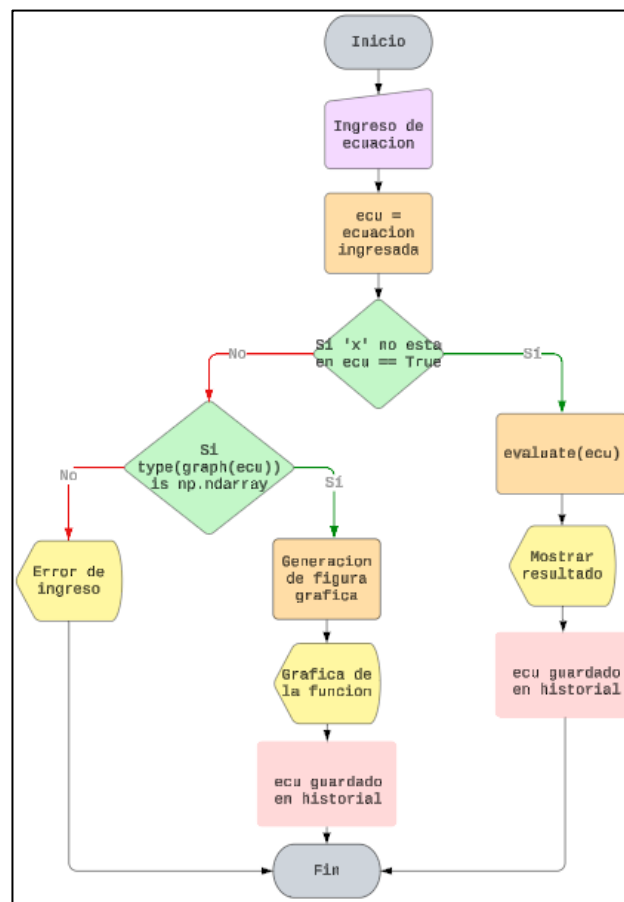
- **Método “\_\_init\_\_”:** Este método no es diferente a los anteriores, como se ha establecido, sirve como método constructor de la clase, para que se realicen las inicializaciones de variables y configuración de la ventana con sus widgets.



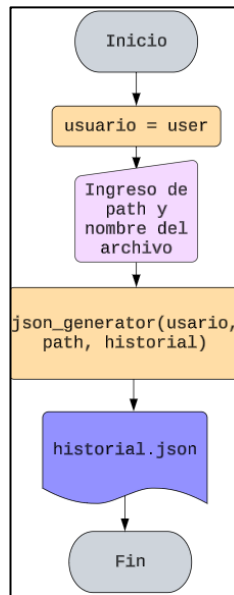
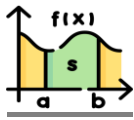
- **Método “ing\_teclado”:** Este método llama a la función **entry()** del modelo **Calc\_ctrl** para verificar que la sintaxis de la ecuación que se esté ingresando sea válido.



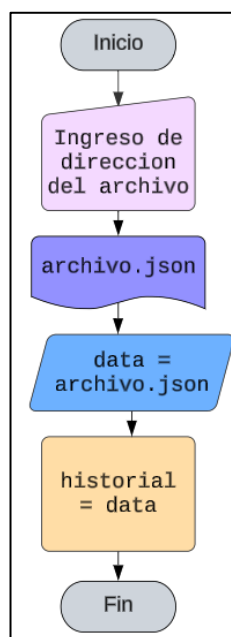
- **Método “result”**: Este método se utiliza para realizar el respectivo calculo o graficacion de la ecuación ingresada, y así mismo, obtener el resultado. En el caso de una funcion a graficar, el método creara un canvas para mostrar la figura generada. Es llamado a través del botón de **Enter** de la ventana grafica.



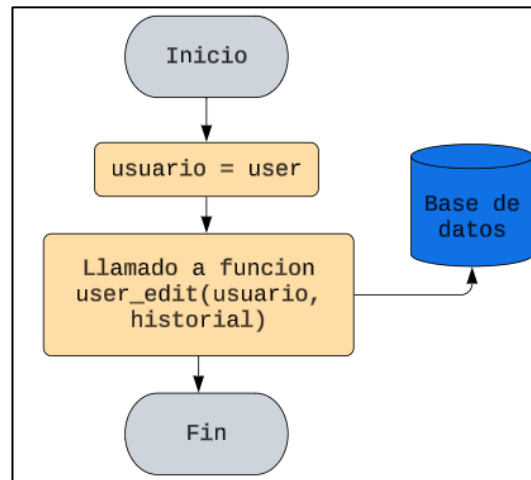
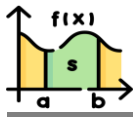
- **Método “guardar”**: Este método permite generar un archivo “.json” con el historial obtenido del usuario, el cual puede guardar en la computadora de forma personalizada. Es llamado a través del botón de **Guardar json** de la ventana grafica.



- **Método “file\_open”:** Este método permite abrir un archivo “.json” para cargar el historial de dicho archivo, y poder continuar desde el último punto de guardado. Es llamado a través del botón de **Abrir desde json** de la ventana grafica.

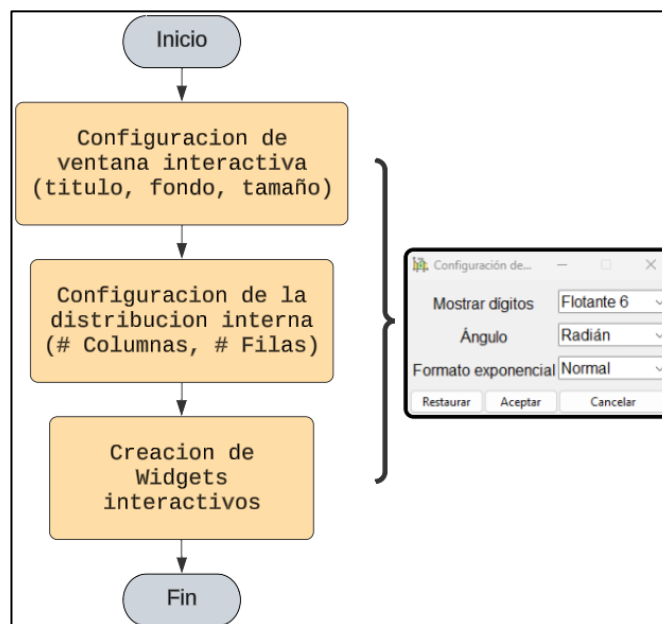


- **Método “syncr”:** Este método se utiliza para sincronizar el historial actual del usuario con su cuenta en la base de datos, subiendo la información de manera automática para que el Administrador pueda tener el control de sus datos. Es llamado a través del botón de **Sincronizar en DB** de la ventana grafica.



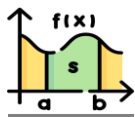
❖ **Clase `config_calc`:** Esta clase es utilizada para abrir una ventana donde se puede cambiar la configuración de la forma en que se muestran los resultados de la calculadora.

- **Método “`__init__`”:** Al igual que en los casos anteriores, este método se utiliza como constructor de la clase, realizando las inicializaciones de variables y configurando la ventana y widgets.

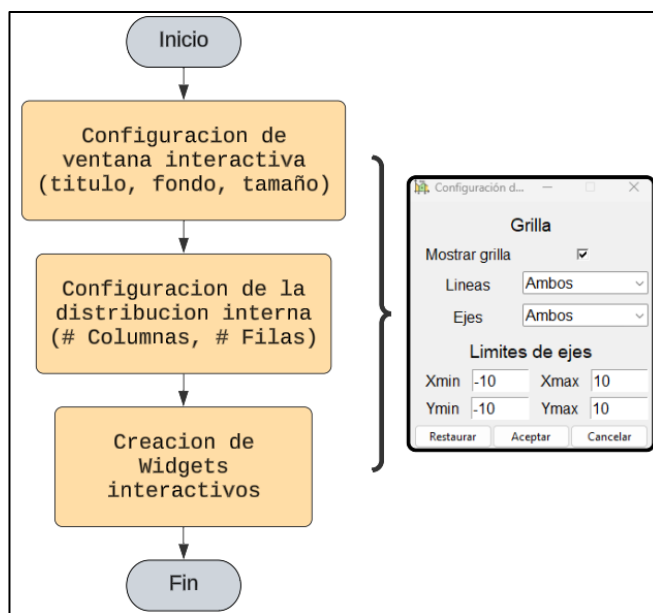


- **Método “`restore`”:** Este método se utiliza para restablecer los valores de configuración de calculo a su estado inicial. Es llamado desde el botón de **Restaurar** en la ventana grafica.
  - **Método “`confirm`”:** Este método se utiliza para establecer los nuevos valores de configuración de calculo para las futuras operaciones que realice el usuario dentro de una misma sesión. Es llamado desde el botón de **Aceptar** en la ventana grafica.
- ❖ **Clase `config_graph`:** Esta clase es utilizada para abrir una ventana donde se puede cambiar la configuración de las gráficas.
- **Método “`__init__`”:** Este método no es diferente a los anteriores, como se ha establecido, sirve como método constructor de la clase, para que se





realicen las inicializaciones de variables y configuración de la ventana con sus widgets

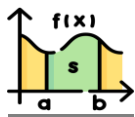


- **Metodo “restore”:** Este método se utiliza para restablecer los valores de configuración de graficas a su estado inicial. Es llamado desde el botón de **Restaurar** en la ventana grafica.
- **Metodo “confirm”:** Este método se utiliza para establecer los nuevos valores de configuración de graficas para las futuras funciones que ingrese el usuario dentro de una misma sesión. Es llamado desde el botón de **Aceptar** en la ventana grafica.

## 9. Limitaciones de la aplicación

**Calculatory** es una aplicación de calculadora que utiliza el modulo Sympy para realizar los cálculos matemáticos, sin embargo, no está exenta de algunas limitaciones en cuanto al nivel de calculo que puede realizar, constituyendo así, una desventaja frente a otras aplicaciones de calculadora, sin embargo, esta desventaja se presenta más en el ámbito de la sintaxis que se utilice para escribir la ecuación. A continuación, se realiza una lista de las limitaciones que presenta la aplicación tanto en sintaxis, como en calculo

- ❖ **Escritura de la ecuación:** Esta limitación se refiere a que la aplicación no es capaz de reconocer una ecuación que no este escrita de la forma en que la puede interpretar, como el siguiente caso:
  - **Multiplicacion (forma correcta):**  $a * b$  |  $a * (b)$  |  $(a) * b$  |  $(a) * (b)$
  - **Multiplicacion (forma incorrecta):**  $ab$  |  $a(b)$  |  $(a)b$  |  $(a)(b)$
- ❖ **Escritura de la función:** Esta limitación se refiere a que la aplicación no es capaz de reconocer una función que no esté escrita de la forma en que la puede interpretar, como el siguiente caso:
  - **Funcion (forma correcta):**  $x$  |  $x^2$  |  $\tan(x)$  |  $\ln(x)$  |
  - **Funcion (forma incorrecta):**  $y = x$  |  $y = x^2$  |  $y = \tan(x)$  |  $y = \ln(x)$

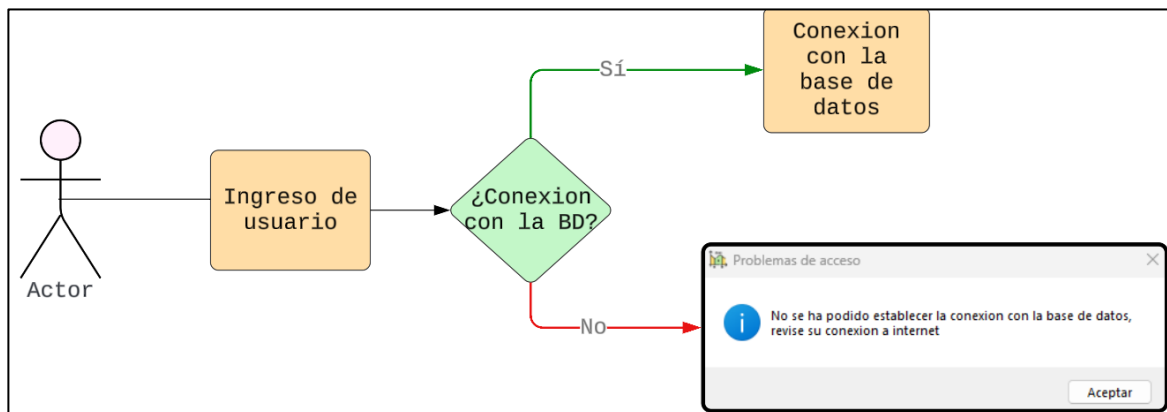


Cualquier tipo de escritura de forma incorrecta, arrojará un error que le informará que no se puede realizar el cálculo para evitar bloqueos en la aplicación.

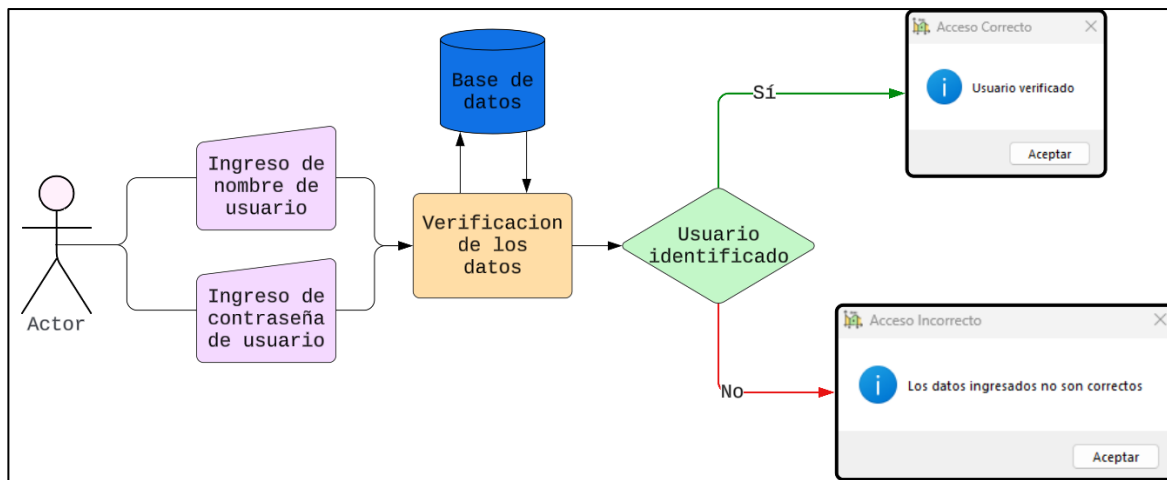
## 10. Casos de uso

Los casos de uso de la aplicación varían de acuerdo con el comportamiento que puede tener el usuario con la aplicación, sin embargo, a continuación, se realiza la descripción de los casos mas comunes para visibilizar la ruta que puede tener dicha interacción del usuario dentro de la aplicación.

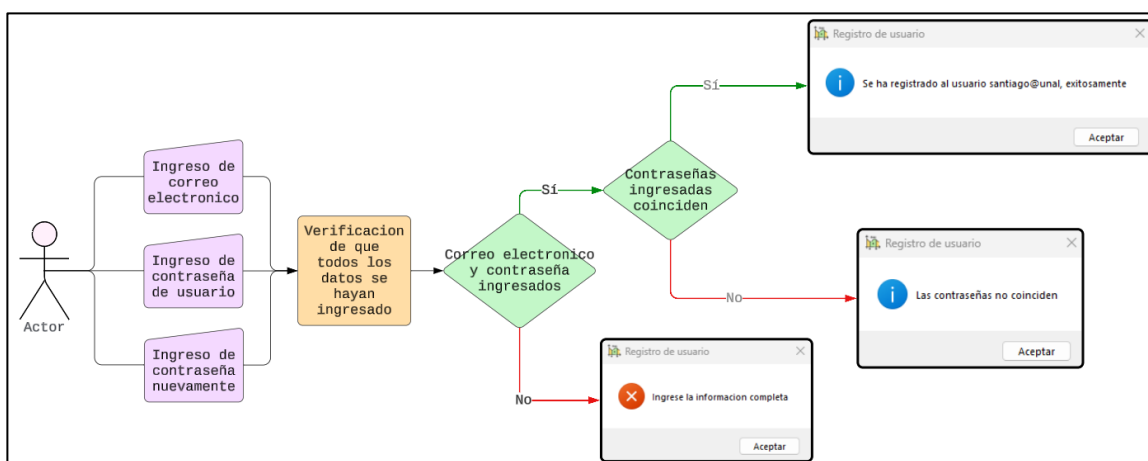
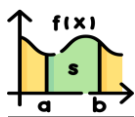
### ❖ Ingreso sin conexión a internet



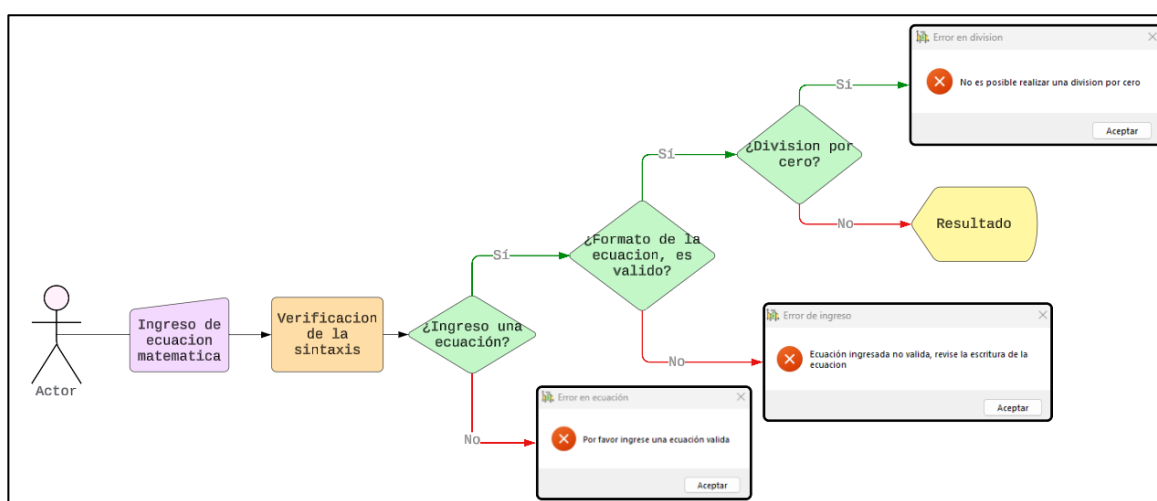
### ❖ Ingresar como usuario



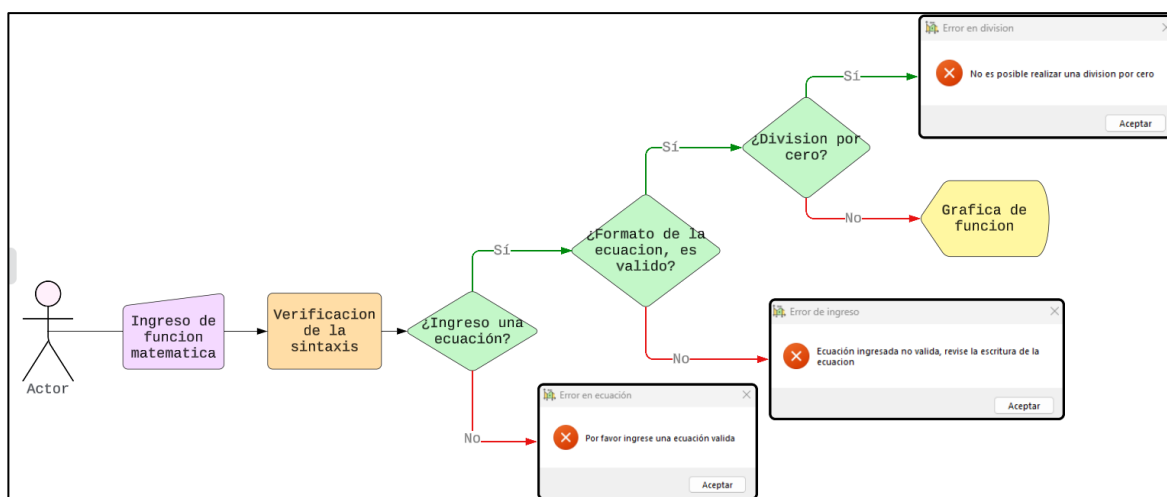
### ❖ Registrarse como usuario



### ❖ Ingreso de una ecuación matemática



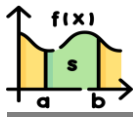
### ❖ Ingreso de una función matemática.



## 11. Errores frecuentes.

A continuación, se presentan una serie de errores frecuentes, y su solución en caso de presentarse.

- ❖ **Cambio de información de la base de datos:** Si usted ya había ingresado con anterioridad la información de una base de datos y requiere cambiarlos a otra, lo que debe realizar para evitar errores de inicialización es borrar el



archivo que se genera en la carpeta “**resources**” de la aplicación, al realizar esta acción, podrá suministrar nuevamente la información de la nueva base de datos.

- ❖ **Error en la graficación:** Para solucionar este error, utilice la opción de cerrar que se encuentra en la barra superior de **Archivo**, si el error persiste, proceda a cerrar la aplicación y volver a ejecutarla.
- ❖ **Error de autenticación de usuario:** Asegúrese de estar registrado en la base de datos que esta utilizando al momento de querer iniciar sesión, ya que, si ha cambiado constantemente la información de la base de datos a utilizar es posible que no se haya registrado en dicha base de datos.

## 12. Referencias

- Escalante, M. (2024, 7 julio). *Por dónde comenzar para instalar Python*. abcXperts. <https://abcxperts.com/por-donde-comenzar-para-instalar-python/>
- MVC - Glosario de MDN Web Docs: Definiciones de términos relacionados con la Web | MDN. (2023, 13 noviembre). MDN Web Docs. <https://developer.mozilla.org/es/docs/Glossary/MVC>
- What Is a Database? (2020, 24 noviembre). <https://www.oracle.com/co/database/what-is-database/>
- Kurth, M. (2024, 26 abril). Tipos de arquitecturas de software: Cuáles hay y en qué se diferencian. Teclab. <https://www.teclab.edu.ar/tipos-de-arquitecturas-de-software-cuales-hay-y-en-que-se-diferencian/>