Guía de Calc 7.5

Capítulo 13
Macros de Calc

Automatizar tareas repetitivas

Derechos de Autor

Colaboradores

De esta edición

Olivier Hallot

De esta edición (traducción y revisión)

David Mamani Castro
J. Carlos Sanz Cabrero
B. Antonio Fernández

De ediciones previas

Skip Masonsmith
Andrew Pitonyak
Barbara Duprey
Jean Hollis Weber
Simon Brydon
Steve Fanning
Leo Moons
Felipe Viggiano
Olivier Hallot
Kees Kriek
Vasudev Narayanan
Rafael Lima

De ediciones previas (traducción y revisión)

José María López Sáez
J. Carlos Sanz Cabrero

Comentarios y sugerencias

Puede dirigir cualquier comentario o sugerencia acerca de este documento al foro del equipo de documentación en español (es necesario registrarse).

Nota

Todo lo que publique en este foro, incluyendo su dirección de correo y cualquier otra información personal que escriba en el mensaje se archiva públicamente y no puede ser borrada.

Fecha de publicación y versión del programa

Versión en español publicada en noviembre de 2023. Basada en la versión 7.5 de LibreOffice.

Uso de LibreOffice en macOS

Algunas pulsaciones de teclado y opciones de menú en macOS, son diferentes de las usadas en Windows y Linux. La siguiente tabla muestra algunas sustituciones comunes para las instrucciones dadas en este capítulo. Para una lista detallada vea la ayuda de la aplicación.

Windows o Linux	*macOS*	*Efecto*
Menú, **Herramientas > Opciones**	**LibreOffice > Preferences**	Acceso a las opciones de configuración
Clic derecho	*Ctrl*+clic o clic derecho	Abre un menú contextual
Ctrl	⌘ , *Cmd* o *Command*	Usado con otras teclas
Alt	⌥, Option o Alt	Usado con otras teclas

CONTENIDO

Introducción

El «Capítulo 13» de la *Guía de iniciación* es una introducción a las características de las macros disponibles en LibreOffice. El presente capítulo aporta más información introductoria sobre el uso de las macros dentro de una hoja de cálculo de Calc.

Una macro es un conjunto de órdenes o comandos que son almacenadas para un uso futuro. Un ejemplo de una simple macro es una en la que se ingresa una dirección postal en una celda de una hoja de cálculo. También se pueden usar las macros para automatizar tareas, tanto simples como complejas y permiten introducir nuevas funciones que no están incluidas en Calc.

La manera más sencilla de crear una macro es grabar una serie de acciones por medio de la *Grabadora de macros*. Calc guarda las macros grabadas empleando el lenguaje de programación de código abierto LibreOffice Basic, que es un derivado del famoso lenguaje de programación BASIC. Estas macros pueden editarse y mejorarse con el Entorno de Desarrollo Integrado LibreOffice Basic (IDE, por sus siglas en inglés) después ser grabadas.

Las macros de Calc más poderosas se crean escribiendo código en uno de los cuatro lenguajes de programación compatibles (LibreOffice Basic, BeanShell, JavaScript y Python). Este capítulo ofrece un resumen de las características de las macros en Calc y se enfoca principalmente al lenguaje de programación de macros predeterminado: LibreOffice Basic. Se incluyen algunos ejemplos para los lenguajes de programación *BeanShell*, *JavaScript* y *Python*, pero una descripción más completa de las características para estos lenguajes están fuera del alcance de este documento.

Compatibilidad con Visual Basic para aplicaciones (VBA)

El lenguaje de programación LibreOffice Basic y el lenguaje de programación VBA, que se encuentra en muchos documentos de Microsoft Office, incluidas las hojas de cálculo de Excel, son dialectos del lenguaje BASIC. Si desea usar macros en LibreOffice escritas en Microsoft Excel usando el código de macro VBA, primero debe editar el código en el IDE para Basic de LibreOffice.

Al final de este capítulo se detallan algunos consejos para convertir macros de Excel escritas en VBA.

Cómo utilizar la grabadora de macros

El «Capítulo 13» de la *Guía de iniciación* incluye ejemplos que muestran cómo utilizar la grabadora de macros y ayudan a comprender las secuencias de comandos generadas por LibreOffice Basic. Los siguientes pasos lo enfocan de manera más específica a una hoja de cálculo Calc, sin las explicaciones tan detalladas de la *Guía de iniciación*. Se crea y se guarda una macro que lleva a cabo un pegado especial con la operación multiplicar a lo largo de un rango de celdas de hojas de cálculo.

Nota

Use **Herramientas > Opciones > LibreOffice > Avanzado** en el menú y seleccione la opción *Activar grabación de macros (limitada)* para habilitar la grabadora de macros.

1) Seleccione **Archivo > Nuevo > Hoja de cálculo** del menú para crear una nueva hoja de cálculo.
2) Ingrese los números que se muestran en la figura 1 en las celdas `A1:C3` en la primera hoja del libro de cálculo.

	A	B	C
1	1	8	9
2	2	7	10
3	3	6	11

Figura 1: Ingrese los valores en el intervalo A1:C3

3) Seleccione la celda `A3`, que contiene el número `3` y seleccione **Editar > Copiar** en el menú para copiar dicho valor en el portapapeles.

4) Seleccione todas las celdas en el rango `A1:C3`.

5) Seleccione **Herramientas > Macros > Grabar macro** del menú para comenzar a grabar la macro. Calc muestra el diálogo *Grabar macro*, que incluye un botón *Finalizar grabación* (figura 2).

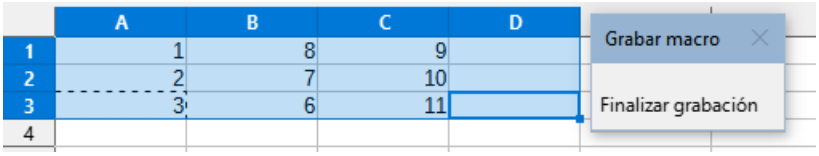

Figura 2: Grabar macro

6) Seleccione **Editar > Pegado especial > Pegado especial** del menú para abrir el diálogo *Pegado especial* (figura 3).

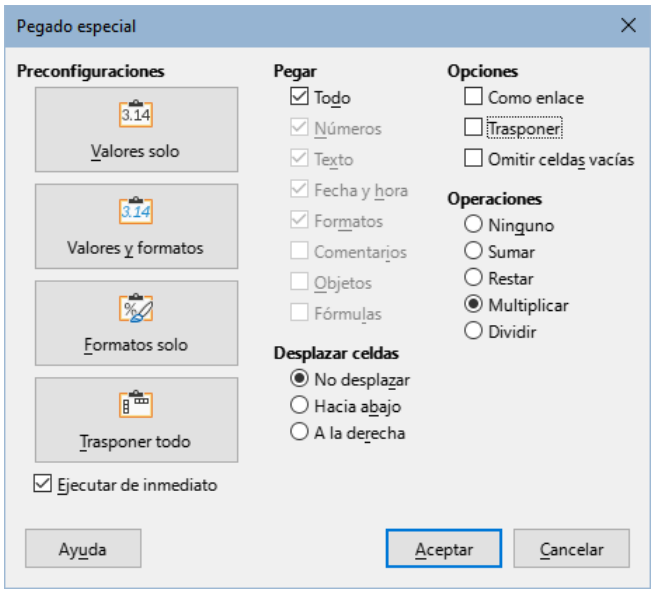

Figura 3: Diálogo Pegado especial

7) Seleccione la opción *Todo* en el área *Pegar* y *Multiplicar* en el área *Operaciones* y haga clic en *Aceptar*. Los valores en las celdas `A1:C3` ahora se multiplican por `3` (figura 4).

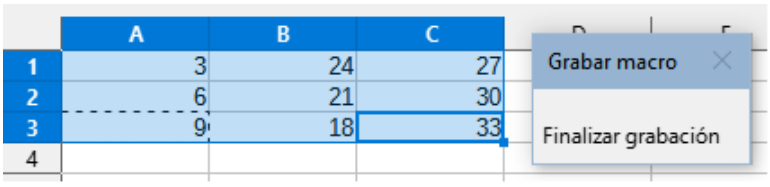

Figura 4: Celdas del intervalo A1:C3 multiplicadas por 3

8) Haga clic en el botón *Finalizar grabación* para finalizar la grabación de la macro. Calc muestra una variante del diálogo *Macros Basic* (figura 5).

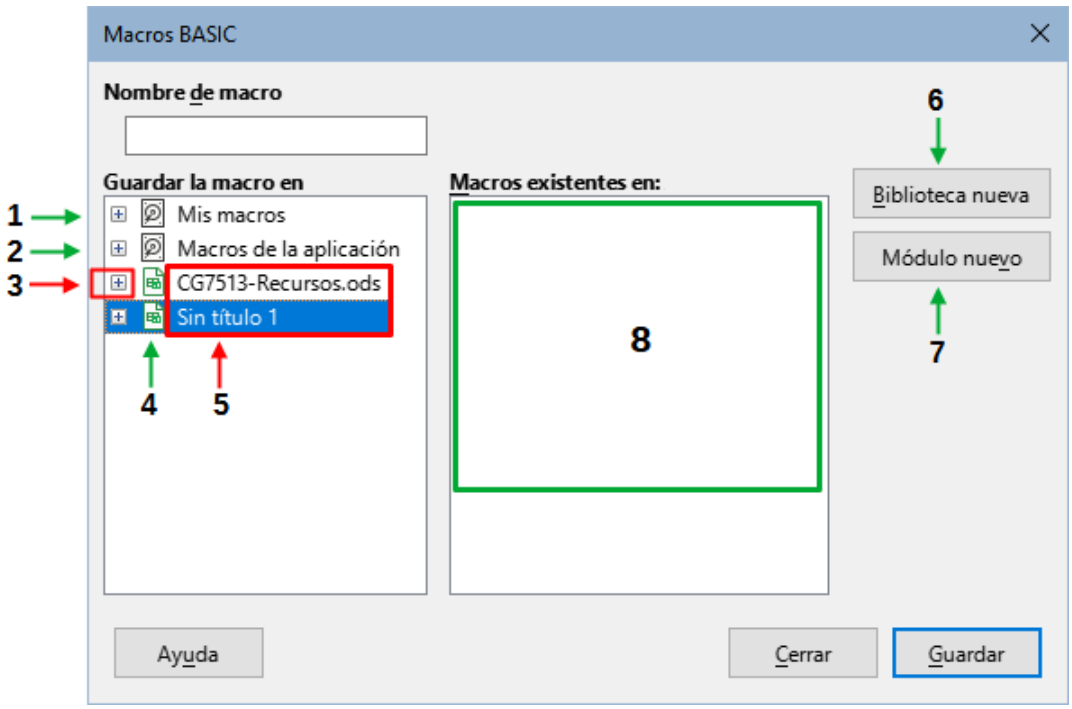

Figura 5: Partes del diálogo Macros Basic

1) Mis Macros
2) Macros de la aplicación
3) Ícono expandir/colapsar
4) Documentos abiertos
5) Documento actual
6) Crear nueva biblioteca
7) Crear nuevo módulo en la biblioteca
8) Macros en el módulo seleccionado

Nota

El área *Guardar la macro en* del diálogo Macros BASIC muestra las macros existentes de LibreOffice Basic, estructuradas jerárquicamente en contenedores, bibliotecas, módulos y macros como se describe en el «Capítulo 13» de la *Guía de iniciación*. La figura 5 muestra el contenedor *Mis macros*, el contenedor *Macros de la aplicación* y el contenedor para el archivo sin título creado en el paso 1). Utilice los iconos de expandir/contraer a la izquierda del nombre de cada contenedor para ver las bibliotecas, módulos y macros dentro de ese contenedor.

9) Seleccione la entrada para el documento actual en el área *Guardar la macro en*. Como el documento actual en este ejemplo que aún no se ha guardado, se lo llama por su nombre predeterminado: *Sin título 1*.
10) Los documentos ya guardados incluyen una biblioteca de macros llamada *Standard*. Esta biblioteca no se crea hasta que se guarda el documento o se necesita la biblioteca, así que en este punto del procedimiento de ejemplo, el nuevo documento no contiene una biblioteca. Se puede crear una nueva biblioteca que contenga la macro recién creada, pero no es necesario, pues se creará automáticamente al guardar la macro.
11) Haga clic en el botón *Módulo nuevo*. Calc muestra el diálogo *Módulo nuevo* (figura 6). Ingrese un nombre para el nuevo módulo o mantenga el asignado de manera predeterminada: *Module1*.

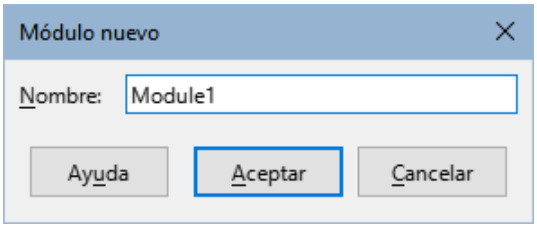

Figura 6: Diálogo Módulo nuevo

Nota

Las bibliotecas, los módulos y los nombres de macros deben seguir algunas reglas estrictas. De acuerdo a las reglas principales, los nombres deben:

– Comenzar con una letra o un guion bajo …
– Constar de letras minúsculas (a…z), letras mayúsculas (A…Z), dígitos (0…9) o guiones bajos (_)
– No deben contener ningún espacio, símbolos de puntuación o caracteres especiales (incluidas tildes).

12) Haga clic en el botón *Aceptar* para crear un nuevo módulo. Como no hay ninguna biblioteca en el documento actual, Calc crea y utiliza la biblioteca *Standard* automáticamente.

13) En el diálogo *Macros Basic*, seleccione la entrada para el módulo recientemente creado en el área *Guardar la macro en*, ingrese el texto *PegadoMultiplicacion* en el cuadro *Nombre de macro* y haga clic en el botón *Guardar* (figura 7).

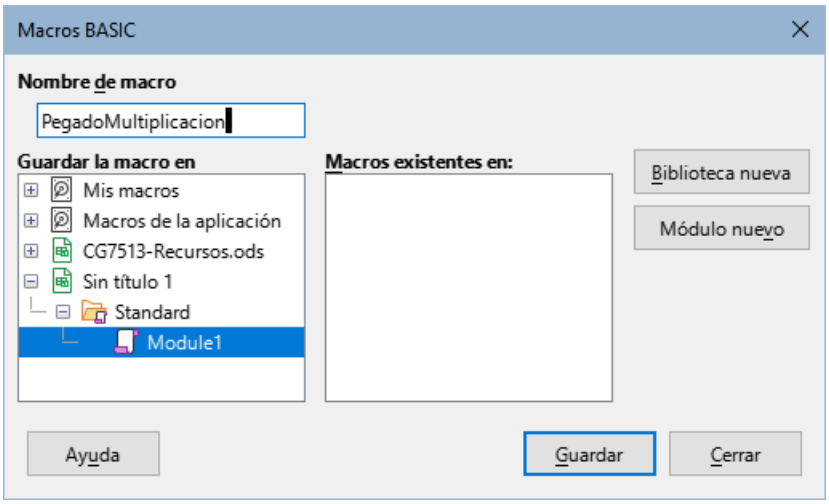

Figura 7: Asignar nombre a una macro (Macros Basic)

La macro se guarda con el nombre *PegadoMultiplicacion* en el módulo recientemente creado dentro de la biblioteca *Standard* del documento *Sin título 1*. El listado 1 muestra su contenido.

Listado 1. PegadoMultiplicacion - Pegado especial con la macro multiplicar

```
sub PegadoMultiplicacion
 rem ----------------------------------------------------------------
 rem define variables
 dim document  as object
 dim dispatcher as object
 rem ----------------------------------------------------------------
 rem get access to the document
 document   = ThisComponent.CurrentController.Frame
 dispatcher = createUnoService("com.sun.star.frame.DispatchHelper")
```

```
 rem ----------------------------------------------------------------------
 dim args1(5) as new com.sun.star.beans.PropertyValue
 args1(0).Name = "Flags"
 args1(0).Value = "A"
 args1(1).Name = "FormulaCommand"
 args1(1).Value = 3
 args1(2).Name = "SkipEmptyCells"
 args1(2).Value = false
 args1(3).Name = "Transpose"
 args1(3).Value = false
 args1(4).Name = "AsLink"
 args1(4).Value = false
 args1(5).Name = "MoveMode"
 args1(5).Value = 4
 dispatcher.executeDispatch(document, ".uno:InsertContents", "", 0, args1())
end sub
```

Nota

la macro es sólo un ejemplo del uso de la grabadora de macros y organización de macros, No tiene una aplicación práctica. Cada vez que se ejecute multiplicará por `3` las celdas del intervalo `A1:C3`

Nota

El modelo de componentes utilizado en LibreOffice es Universal Network Objects (UNO) y la grabadora de macros utiliza el despachador UNO para la mayoría de los comandos. Sin embargo, hay dos problemas asociados con este enfoque técnico. Una es que los envíos no están completamente documentados y pueden estar sujetos a cambios. Otra es que la grabadora ignora algunos valores de los diálogos que se abren mientras se graba una macro; por lo tanto, es posible que grabe una macro complicada que en realidad no se ejecutará todo como se esperaba. Para más información, busque «Limitaciones de la grabadora de macros» en la ayuda.

Cómo escribir funciones propias

Crear una macro como función

Se puede escribir una macro y luego nombrarla como se nombraría a una función Calc. Siga los siguientes pasos para crear una macro función sencilla:

1) Cree una nueva hoja de cálculo, guárdela con el nombre PruebasMacrosCalc.ods y déjela abierta en Calc.
2) Seleccione **Herramientas > Macros > Organizar Macros > BASIC** del menú para abrir el diálogo *Macros BASIC* (figura 8). Tenga en cuenta que el diálogo *Macros BASIC* en este contexto es diferente de la versión que se ve en Calc cuando el usuario hace clic en el botón *Finalizar grabación* en el diálogo *Grabar macro* (figura 5).
3) El área *Macro de* muestra una lista de todos los contenedores disponibles, incluyendo aquellos relacionados con cualquier documento de LibreOffice que esté abierto. *Mis Macros* contiene macros que pueden escribirse o agregarse a LibreOffice y están disponibles en más de un documento. *Macros de la aplicación* contiene macros que se incluyen con la instalación de LibreOffice y no deben cambiarse.

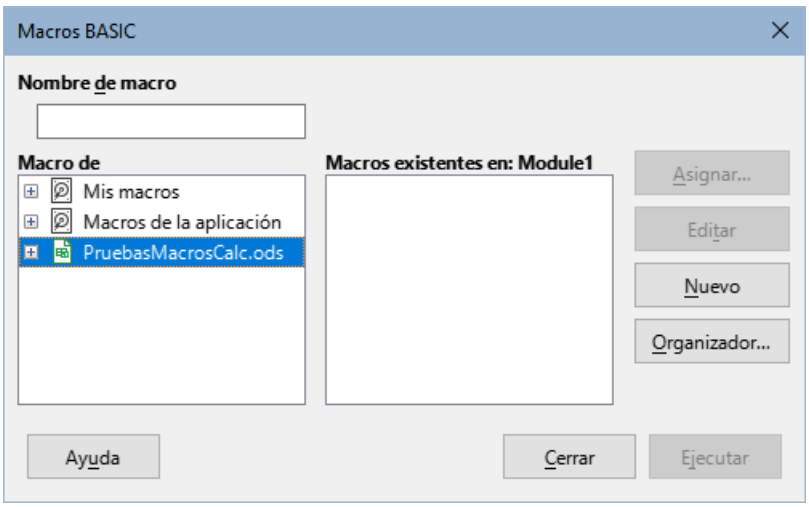

Figura 8: Macros de Basic

4) Haga clic en *Organizador* para abrir el *Organizador de macros de BASIC* (figura 9). Seleccione el archivo `PruebasMacrosCalc.ods` en la lista *Ubicación*

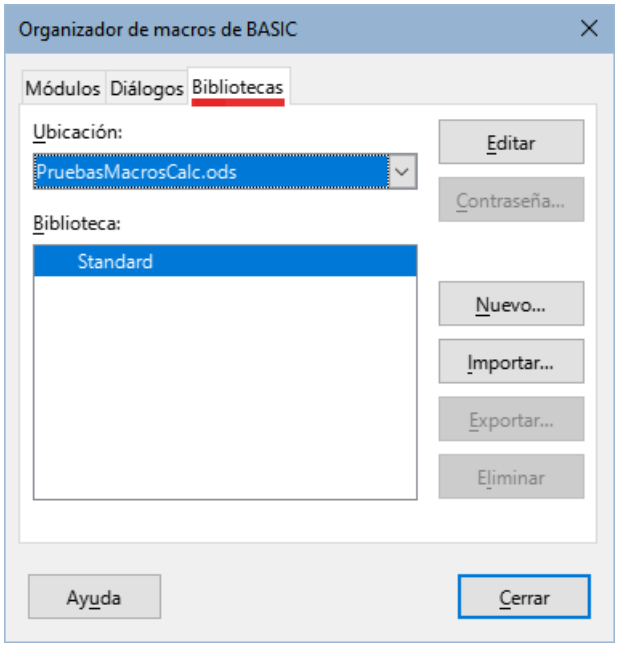

Figura 9: Organizador de macros de Basic

Haga clic en la pestaña *Bibliotecas* y, en el área *Ubicación*, seleccione la entrada para el nombre del documento actual. El área *Biblioteca* mostrará las bibliotecas de macros que contiene el archivo. Siempre encontrará la biblioteca *Standard*, aunque no tenga ninguna, es la asignada de manera predeterminada y no se puede eliminar.

5) Haga clic en *Nuevo* para abrir el diálogo *Biblioteca nueva* para crear una nueva biblioteca para este documento (figura 10).

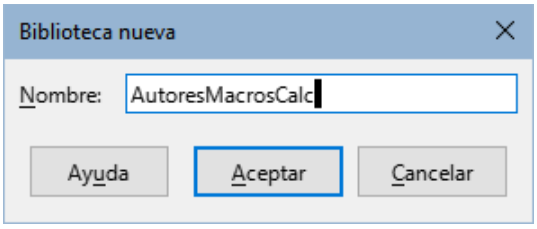

Figura 10: Diálogo Biblioteca nueva

6) Ingrese un nombre de biblioteca descriptivo (por ejemplo, *AutoresMacrosCalc*) y haga clic en *Aceptar* para crear la biblioteca. El área *Biblioteca:* del diálogo del *Organizador de macros de BASIC* se actualiza para incluir el nombre de la biblioteca creada recientemente. Un nombre de biblioteca puede tener hasta 30 caracteres. Tenga en cuenta que, en algunos casos, el diálogo podría mostrar solo parte del nombre.

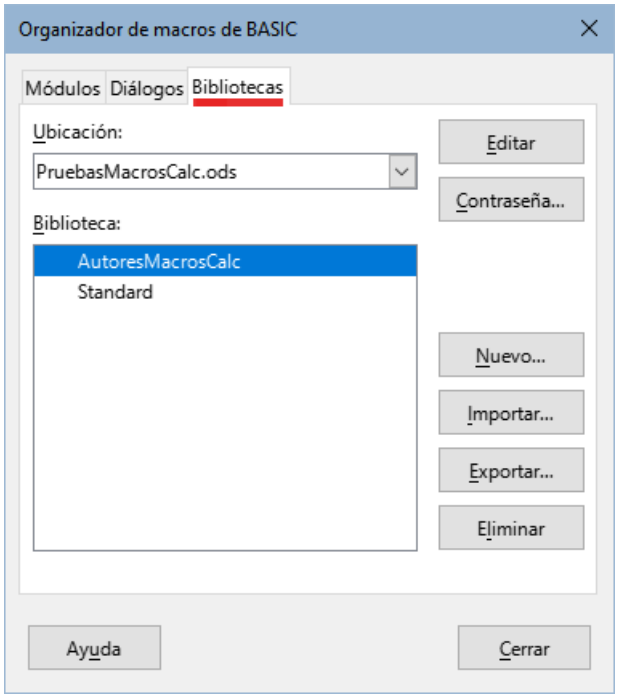

Figura 11: Nueva biblioteca en el área Biblioteca:

7) Asegúrese que la entrada *AutoresMacrosCalc* esté seleccionada en el área *Biblioteca:* y haga clic en *Editar* para editar la biblioteca. Calc automáticamente crea un módulo llamado *Module1* y una macro llamada *Main*. Calc abre el «Entorno de desarroyo integrado (IDE)», como se muestra en la figura 12.

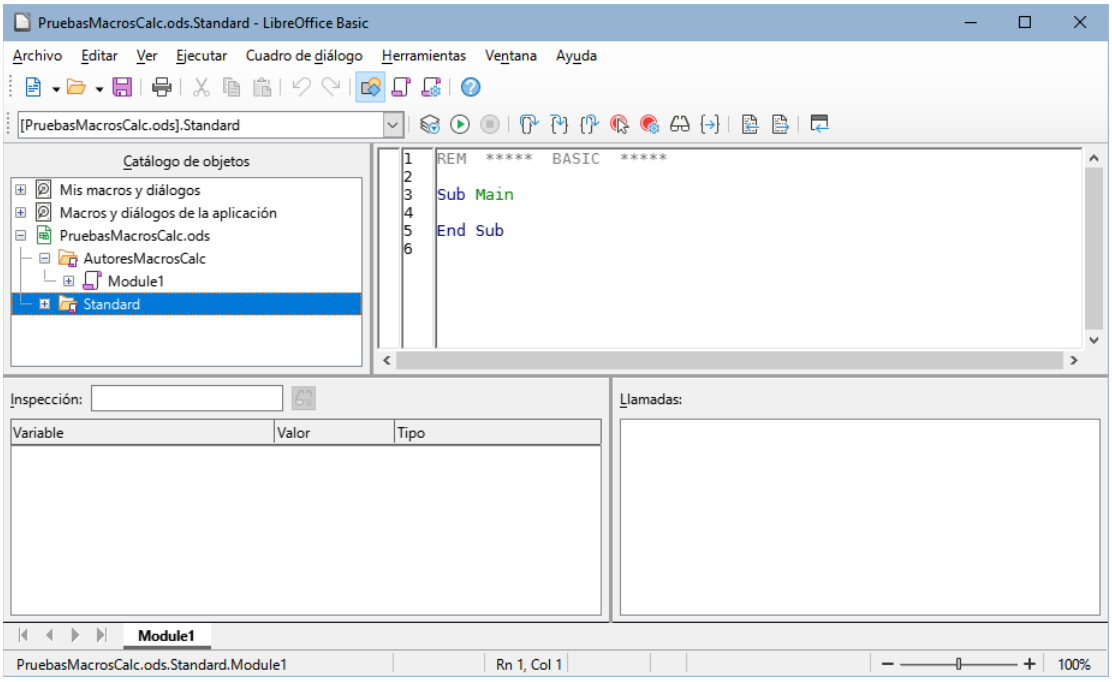

Figura 12: Entorno de desarrollo integrado de LibreOffice Basic

En la figura 12 se muestra el *IDE de LibreOffice Basic* con la a configuración predeterminada, que consta de:

- La barra de menú
- Tres barras de herramientas (*Idioma*, *Macro* y *Estándar*). La barra de herramientas *Macro* contiene varios íconos para editar y probar programas.
- El *Catálogo de objetos* permite la selección del contenedor, biblioteca, módulo y macro requeridos.
- La *ventana del editor*, en el que se edita el código de las macros *LibreOffice Basic*. La columna primera columna del lado izquierdo se usa para establecer puntos de ruptura en el código del programa.
- La ventana *Observador* (a la izquierda, bajo el *Catálogo de objetos* y la *ventana del editor*) muestra los contenidos de las variables o matrices durante un paso del proceso en particular.
- La ventana *Llamadas* (ubicada a la derecha, de la ventana *Observador*) proporciona información sobre la pila de llamadas de procedimientos y funciones cuando se ejecuta un programa.
- Un área de control de pestañas para acceder a las mácros dentro de un módulo.
- La *Barra de estado*.

El *IDE* de *LibreOffice Basic* proporciona características poderosas para el desarrollo y la depuración de las macros de *LibreOffice Basic*. Una descripción más completa de estas características está fuera del alcance de este documento, pero se puede encontrar mayor información en le sistema de Ayuda.

8) En la *ventana del editor*, cambie el código para que sea el mismo que se muestra en el listado 2. La adición importante es la creación de la función *NumeroCinco*, que devuelve el valor numérico `5`.

Sugerencia

La instrucción `Option Explicit` obliga a que se declaren todas las *variables* antes de que puedan ser usadas. Si se omite la `Option Explicit`, las variables se definen automáticamente de tipo `Variant`.

Listado 2. Función que devuelve el valor numérico 5.

```
REM ***** BASIC *****
Option Explicit

Sub Main

End Sub

Function NumeroCinco()
   NumeroCinco = 5
End Function
```

9) Haga clic en el botón *Guardar* de la barra de herramientas *Estándar* dentro del *IDE* de *LibreOffice Basic* o pulse *Ctrl+G* para guardar el módulo (*Module1*) modificado.

Cómo utilizar la macro como una función

Utilice la hoja de cálculo `PruebasMacrosCalc.ods` recien creada, seleccione una celda e ingrese la fórmula `= NumeroCinco()` (figura 13). Calc encuentra la macro, la nombra y muestra el resultado (`5`) en esa celda.

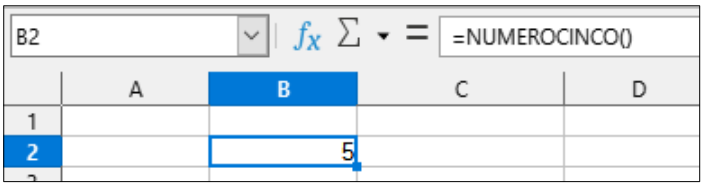

Figura 13: Utilice la macro NumeroCinco como una función

Sugerencia

Los nombres de las funciones no son sensibles a mayúsculas y minúsculas. En la figura 13, el nombre de la función se ingresa como `NumeroCinco()`, pero Calc lo muestra como *NUMEROCINCO()* en la barra de *Fórmulas*.

Advertencia de seguridad de las macros

Ahora debería guardar el documento Calc, cerrarlo y abrirlo nuevamente. Dependiendo de las configuraciones en el diálogo *Seguridad de macros* al que se accede seleccionando **Herramientas > Opciones > LibreOffice > Seguridad > Seguridad de macros** del menú, se podría ver una de las advertencias que muestra Calc (figuras 14 y 15).

En el caso de la advertencia de la figura 14, deberá hacer clic en *Activar macros*, en caso contrario, Calc no permitirá que se ejecute ninguna macro en el documento. Si un documento contiene macros, es más seguro hacer clic en *Desactivar macros* para evitar infecciones en caso de que la macro contenga un virus.

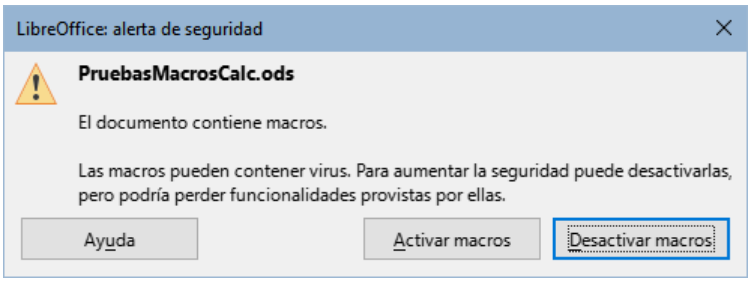

Figura 14: Advertencia de macros en un documento

En el caso de la advertencia que se muestra en la figura 15, Calc no permitirá que se ejecute ninguna macro en el documento y deberá hacer clic en el botón *Aceptar* para eliminar la advertencia de su pantalla.

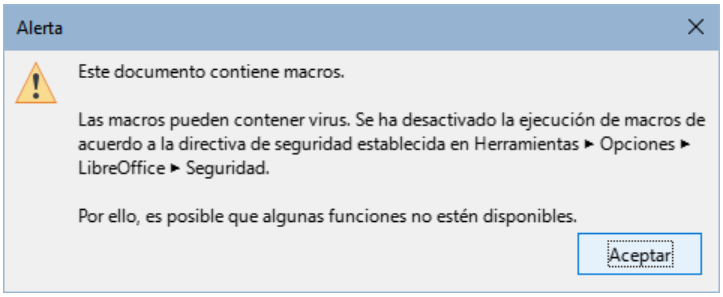

Figura 15: Advertencia de macros deshabilitadas.

Cuando el documento se carga con las macros desactivadas, Calc no podrá encontrar ninguna macro de tipo función e indicará un error en las celdas afectadas con el texto `#¿NOMBRE?`.

Bibliotecas cargadas / descargadas

Al abrir una hoja de cálculo, Calc no abre todas las bibliotecas que encuentra en los contenedores disponibles porque consumen recursos aunque no se utilicen. En su lugar, Calc automáticamente carga solamente la biblioteca *Standard* dentro del contenedor *Mis Macros* y la propia biblioteca *Standard* del documento. Ninguna otra biblioteca se carga de manera automática.

Al abrir la hoja de cálculo `PruebasMacrosCalc.ods` nuevamente, Calc revisa todas las bibliotecas visibles y cargadas para la función ya que no contiene una función llamada `NumeroCinco()`. Las bibliotecas cargadas en *Macros de la aplicación*, *Mis Macros* y el documento se revisan para que la función sea nombrada correctamente. En la primera implementación, la función `NumeroCinco()` se almacena en la biblioteca *AutoresMacrosCalc*, que no se carga automáticamente cuando se abre el documento. Por lo tanto, no se encuentra la función `NumeroCinco()` y aparece una condición de error en la celda donde se la nombra (figura 16).

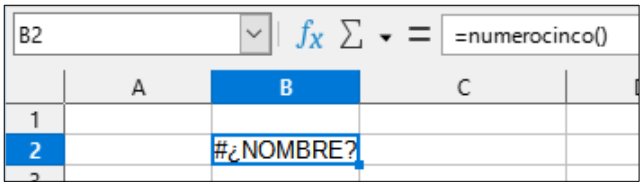

Figura 16: La función de la macro no está disponible

Seleccione **Herramientas > Macros > Organizar Macros > BASIC** del menú para abrir el diálogo *Macros BASIC* (figura 17). El ícono de una biblioteca cargada (por ejemplo, *Standard*) tiene un aspecto diferente al ícono de una biblioteca que no está cargada (por ejemplo, *AutoresMacrosCalc*).

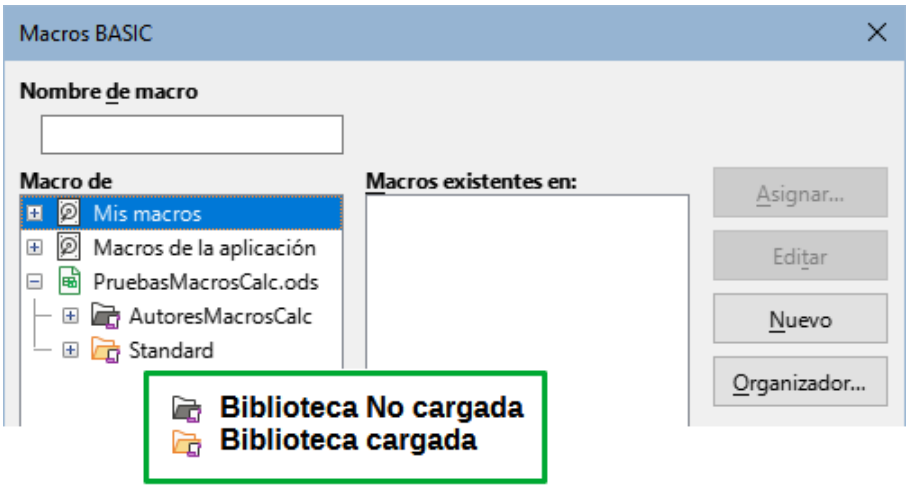

Figura 17: Diferentes símbolos para bibliotecas cargadas y no cargadas

Haga clic en el ícono de expansión al lado de *AutoresMacrosCalc* para cargar la biblioteca. El ícono cambia de apariencia para indicar que la biblioteca ya está cargada. Haga clic en *Cerrar* para cerrar el diálogo *Macros BASIC*.

Desafortunadamente, la celda que contiene la función `= NumeroCinco()` todavía marca error. Calc no recalcula las celdas que marcan error a menos que se editen o cambien de alguna manera. La solución más habitual es almacenar las macros utilizadas como funciones en la biblioteca *Standard*. Otra solución para este problema es crear una macro auxiliar (*stub*) en la biblioteca *Standard*.

La macro auxiliar carga la biblioteca que contiene la función y luego llama a la macro. Los siguientes pasos ilustran el método.

1) Seleccione **Herramientas > Macros > Organizar Macros > BASIC** en el menú para abrir el diálogo *Macros BASIC*. Seleccione la macro *NumeroCinco* y haga clic en *Editar* para abrir la macro y editarla (figura 18).

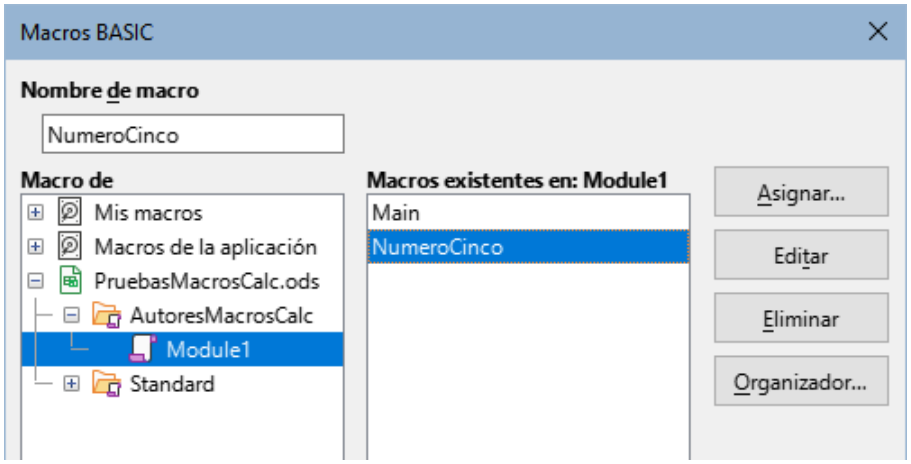

Figura 18: Macros Basic.

2) Calc muestra el *IDE* de *LibreOffice Basic* (figura 12), con el cursor de entrada en la ventana del editor en la línea `Function NumeroCinco()`. Cambie el nombre de `NumeroCinco` a `NumeroCinco_Implementacion` para que el código de la función coincida con el del Listado 3.

Listado 3. Cambie el nombre de NumeroCinco a NumeroCinco_Implementacion

```
Function NumeroCinco_Implementacion()
   NumeroCinco_Implementacion = 5
End Function
```

3) Haga clic en el botón *Seleccionar Macro* en la barra de herramientas *Standard* del *IDE de LibreOffice Basic* para abrir el diálogo *Macros BASIC* (figura 18).

4) Seleccione la biblioteca *Standard* en el documento PruebasMacrosCalc.ods y haga clic en el botón *Nuevo* para crear un nuevo módulo. Ingrese un nombre significativo tal como *FuncionesCalc* y haga clic en *Aceptar*. Calc crea una macro llamada *Main* de manera automática y abre el módulo para su edición.

5) Cree una macro en el módulo *FuncionesCalc* de la biblioteca *Standard* que cargue la biblioteca *AutoresMacrosCalc* si no está ya cargada y luego llame la función de la implementación. Vea el listado 4. Aquí la función *NumeroCinco* es el la macro auxiliar resguardo (stub).

Listado 4. Cree una nueva función «NumeroCinco» para llamar a NumeroCinco_Implementacion

```
Function NumeroCinco()
  If NOT BasicLibraries.isLibraryLoaded("AutoresMacrosCalc") Then
    BasicLibraries.LoadLibrary("AutoresMacrosCalc")
  End If
  NumeroCinco = NumeroCinco_Implementacion()
End Function
```

6) Guarde, cierre y abra el documento Calc nuevamente. Esta vez, si las macros están habilitadas, la función `NumeroCinco()` funcionará como se espera.

Cómo pasar argumentos a una macro

Para ilustrar una función que acepte argumentos, escribiremos una macro que calcule la suma de los argumentos que son positivos. Los argumentos que sean inferiores a cero no se toman en cuenta (vea listado 5).

Listado 5. SumaPositivos. Calcula la suma de los argumentos positivos

```
Function SumaPositivos(Optional x)
 Dim Suma As Double
 Dim iRow As Integer
 Dim iCol As Integer

 Suma = 0.0
  If NOT IsMissing(x) Then
    If NOT IsArray(x) Then
      If x > 0 Then Suma = x
    Else
      For iRow = LBound(x, 1) To UBound(x, 1)
        For iCol = LBound(x, 2) To UBound(x, 2)
          If x(iRow, iCol) > 0 Then Suma = Suma + x(iRow, iCol)
        Next
      Next
    End If
  End If
  SumaPositivos = Suma
End Function
```

La macro que se encuentra en el Listado 5 muestra algunas técnicas importantes:

1) El argumento `x` es opcional. Cuando un argumento no es opcional y se llama a la función sin este, Calc genera un mensaje de advertencia cada vez que se llama a la macro. Cada vez que se llame a la función, aparecerá el mensaje de error.
2) La función `IsMissing` revisa si se ha pasado un argumento antes de utilizar la macro.
3) La función `IsArray` revisa si hay varios valores dentro del argumento pasado (si es una matriz con varios valores). Por ejemplo, `= PositivoSum(7)` o `= PositivoSum(A4)`. En el primer caso, el número `7` se pasa como argumento y en el segundo caso, el valor de la celda `A4` es el argumento. En estos casos, `IsArray` devuelve `Falso` porque sólo se ha pasa un valor.
4) Si se pasa un intervalo como argumento a la función, se considera una matriz de valores; por ejemplo `= PositivoSum(A2:B5)`. Las variables `LBound` (límite inferior) y `UBound` (límite superior) se utilizan para determinar los límites de la matriz utilizada. Aunque el límite inferior (Lbound) normalmente es `1` , en lugar de utilizar `1`, se considera más seguro utilizar `Lbound` en caso de que cambie en otra ocasión.

Sugerencia

La macro que se encuentra en el Listado 5 es cuidadosa y verifica que el argumento sea una matriz o un único valor. La macro no verifica que cada valor sea numérico. Puede ser tan cuidadoso como desee. Cuanto más verifique, más robusta será la macro, pero también algo más lenta.

Pasar dos argumentos es tan sencillo como pasar uno: agregue otro argumento a la definición de la función (vea Listado 6). Cuando llame una función con dos argumentos, separe los argumentos con una coma: `= PruebaMax(3, -4)`.

Listado 6. PruebaMax acepta dos argumentos y devuelve el mayor de los argumentos

```
Function PruebaMax(x, y)
  If x >= y Then
    PruebaMax = x
  Else
    PruebaMax = y
  End If
End Function
```

Los argumentos se pasan como valores

De manera predeterminada, los argumentos que se pasan a una macro desde Calc son siempre valores. No es posible saber el nombre de las celdas que se utilizan. Por ejemplo, `= SumaPositivos(A3)` pasa el contenido de la celda "`A3`", pero `SumaPositivos` no tiene forma de saber que ese valor está en la celda "`A3`". Si necesita utilizar los nombres de las celdas referenciadas en vez de su valores, pase los argumentos como nombres de las celdas (cadena de caracteres, entre comillas), analice los nombres de las celdas y obtenga el contenido o valor de las celdas.

Cómo escribir macros que actúen como funciones incorporadas

Aunque Calc encuentre y llame a las macros como funciones normales, en realidad no se comportan como lasfunciones ya incorporadas. Por ejemplo, las macros no aparecen en las listas de funciones. Se pueden escribir funciones que se comporten como las funciones de Cal si se crea una extensión o complemento. Este es un tema avanzado destinado a programadores con experiencia y está más allá del alcance de esta guía. En la Ayuda puede encontrar alguna información, junto con enlaces a lecturas más detalladas.

Eliminar macros de LibreOffice Basic

Siga los siguientes pasos para eliminar una macro:

1) Utilice **Herramientas > Macros > Organizar macros > BASIC** en el menú para abrir el diálogo *Macros BASIC* (figura 18).
2) Seleccione la macro que desea eliminar y haga clic en el botón *Eliminar*.
3) Calc muestra un diálogo de confirmación. Haga clic en *Sí* para continuar.
4) Haga clic en el botón *Cerrar* para cerrar el diálogo *Macros BASIC*.

Siga los siguientes pasos para eliminar un módulo:

1) Utilice **Herramientas > Macros > Organizar macros > BASIC** en el menú para abrir el diálogo *Macros BASIC* (figura 18).
2) Haga clic en el botón *Organizador* para abrir el diálogo *Organizador de macros de BASIC* (figura 19).
3) Asegúrese de que la pestaña *Módulos* esté seleccionada.
4) Seleccione el módulo que quiera eliminar en la lista *Módulo*.
5) Haga clic en el botón *Eliminar*.
6) Calc muestra un diálogo de confirmación. Haga clic en *Sí* para continuar.
7) Haga clic en el botón *Cerrar* para cerrar el diálogo *Organizador de macros de BASIC*.
8) Haga clic en el botón *Cerrar* para cerrar el diálogo *Macros BASIC*.

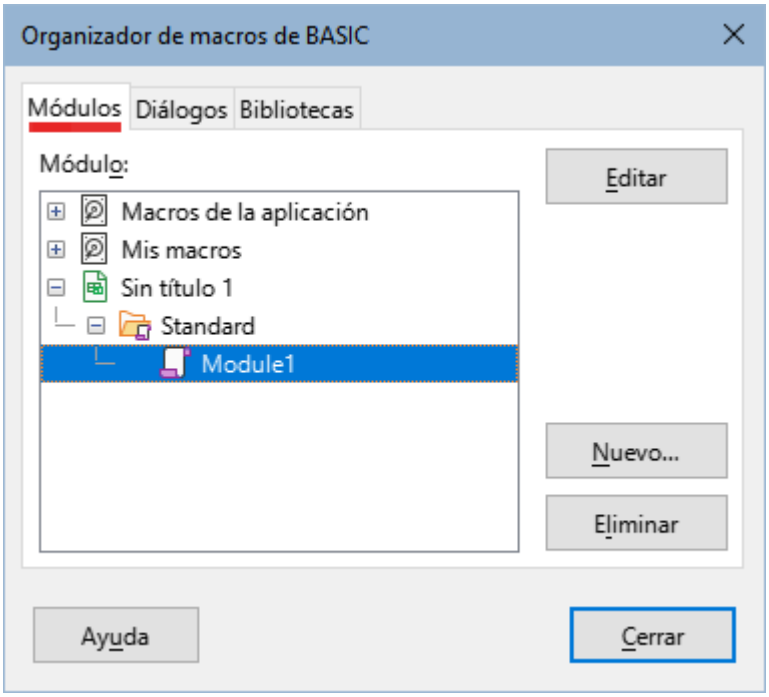

Figura 19: Organizador de macros de BASIC, página Módulos.

Cómo acceder a las celdas de manera directa

Para manipular un documento Calc se puede acceder directamente a los objetos internos de LibreOffice. Por ejemplo, la macro en el Listado 7 suma los valores en la celda `A2` de cada hoja en el documento actual. `ThisComponent` hace referencia al documento actual cuando se activa la macro. Un libro de Calc contiene hojas de cálculo y la macro accede a estas a través de la llamada a `ThisComponent.getSheets()`. Seleccione `getCellByPosition(col, row)` para indicar una celda en una fila y columna específicas.

Listado 7. SumaCeldaTodasHojas suma los valores de la celda A2 de cada hoja del libro

```
Function SumaCeldaTodasHojas()
 Dim Suma As Double
 Dim i As integer
 Dim oHojas, oHoja, Dim oCelda

 Suma = 0
 oHojas = ThisComponent.getSheets()
 For i = 0 To oHojas.getCount() - 1
  oHoja = oHojas.getByIndex(i)
  oCelda = oHoja.getCellByPosition(0, 1) 'celda A2
  Suma = Suma + oCelda.getValue()
 Next
 SumaCeldaTodasHojas = Suma
End Function
```

Sugerencia

Un objeto de celda acepta los métodos `getValue()`, `getString()` y `getFormula()` para obtener el valor numérico, el valor de la cadena de caracteres o la fórmula utilizada en una celda. Utilice los métodos adecuados para obtener los valores correspondientes.

Utilice `oSheet.getCellRangeByName("A2")` obtener un intervalo de celdas por su nombre. Si se especifica una única celda, devuelve un objeto celda. Si se especifica un rango de celdas, devuelve un intervalo de celdas completo (vea el Listado 8). Tenga en cuenta que un intervalo devuelve la información como una matriz de matrices, que es más complicado de gestionar que si se tratar de una matriz de dos dimensiones como se muestra en el Listado 5.

Listado 8. SumaCeldasTodasHojas suma todos los valores en el intervalo A2:C5 de cada hoja

```
Function SumaCeldasTodasHojas()
 Dim Suma As Double
 Dim iFila As Integer, iCol As Integer, i As Integer
 Dim oHojas, oHoja, oCeldas
 Dim oFila(), oFilas()

  REM El método getDataArray() devuelve cadenasde caracteres y números,
  REM pero no se usa en esta función.
  REM El método getData() solo devuelve números y es el aplicado en esta función

  Suma = 0
  oHojas = ThisComponent.getSheets()
  For i = 0 To oHojas.getCount() - 1
    oHoja = oHojas.getByIndex(i)
    oCeldas = oHoja.getCellRangeByName("A2:C5")
    oFilas() = oCeldas.getData()
    For iFila = LBound(oFilas()) To UBound(oFilas())
      oFila() = oFilas(iFila)
        For iCol = LBound(oFila()) To UBound(oFila())
          Suma = Suma + oFila(iCol)
        Next
    Next
  Next
 SumaCeldasTodasHojas = Suma
End Function
```

Sugerencia

Cuando se llama a una macro como una función de Calc, la macro no puede modificar ningún valor en la hoja desde la que se llama a la macro, excepto el valor de la celda que contiene la función.

Clasificación

Considere clasificar los datos que se muestran en la figura 20. Primero, clasifíquelos por la columna `B` de forma descendente y luego por la columna `A` de forma ascendente.

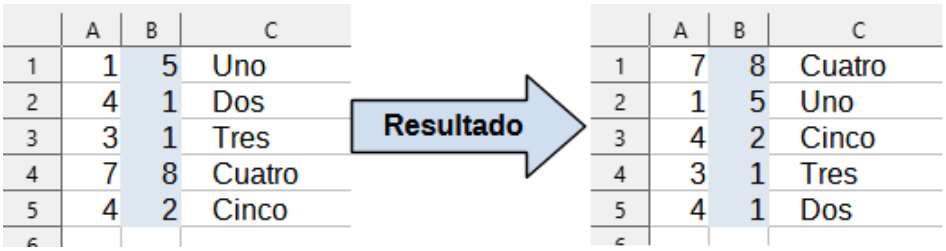

Figura 20: Clasificación: columna B descendente y columna A ascendente

El ejemplo del Listado 9 muestra cómo clasificar en estas dos columnas.

Listado 9. SortRange clasifica las celdas A1:C5 de la Hoja 1

```
Sub SortRange
 Dim oSheet     ' Hoja de cálculo que contiene datos para ordenar.
 Dim oCellRange  ' Intervalo de datos para ordenar.

 REM Una matriz de ordenación de campos determina las columnas que se ordenan.
 REM Esta es una matriz con dos elementos, 0 y 1.
 REM Para ordenar en una sola columna, use:
 REM Dim oSortFields(0) As New com.sun.star.util.SortField
 Dim oSortFields(1) As New com.sun.star.util.SortField

 REM El descriptor de clasificación es una matriz de propiedades.
 REM La propiedad principal contiene los campos de clasificación.
 Dim oSortDesc(0) As New com.sun.star.beans.PropertyValue

 REM Obtener la hoja llamada "Hoja1"
 oSheet = ThisComponent.Sheets.getByName("Hoja1")

 REM Obtener el rango de celdas para ordenar
 oCellRange = oSheet.getCellRangeByName("A1:C5")

 REM Seleccione el rango para ordenar.
 REM El único propósito sería enfatizar los datos ordenados
 'ThisComponent.getCurrentController.select(oCellRange)

 REM Las columnas están numeradas comenzando con 0, por lo que
 REM columna A es 0, columna B es 1, etc.
 REM Ordenar la columna B (columna 1) de forma descendente.
 oSortFields(0).Field = 1
 oSortFields(0).SortAscending = FALSE

 REM Si la columna B tiene dos celdas con el mismo valor,
 REM usar la columna A ascendente para decidir el orden.
 oSortFields(1).Field = 0
 oSortFields(1).SortAscending = TRUE

 REM Configurar el descriptor de clasificación.
 oSortDesc(0).Name = "SortFields"
 oSortDesc(0).Value = oSortFields()

 REM Ordenar el rango.
 oCellRange.Sort(oSortDesc())
End Sub
```

Resumen de las macros de BeanShell, JavaScript y Python

Introducción

Es posible que muchos programadores no estén familiarizados con *LibreOffice Basic*, Calc acepta macros escritas en otros tres lenguajes que les resulten más conocidos. Estos lenguajes son BeanShell, JavaScript y Python.

El lenguaje de programación principal para Calc es *LibreOffice Basic* y en la instalación LibreOffice estándar ofrece un poderoso entorno de desarrollo integrado (IDE) junto con otras opciones para este lenguaje.

Las macros se organizan de la misma manera para los cuatro lenguajes de programación. El contenedor *Macros de la aplicación* tiene todas las macros suministradas en la instalación de LibreOffice. El contenedor *Mis Macros* tiene todas sus macros que están disponibles para cualquiera de sus documentos de LibreOffice. Cada documento también puede contener sus propias macros, no disponibles para otros documentos.

Cuando se utiliza la función de Grabar macros, Calc crea la macro en *LibreOffice Basic*. Para utilizar los otros lenguajes de programación disponibles, debe escribir el código por sí mismo.

Cuando ejecute una macro desde **Herramientas > Macros > Ejecutar macro** en el menú, Calc muestra el diálogo *Selector de macros*. El diálogo permite seleccionar y ejecutar cualquier macro escrita en cualquiera de los lenguajes disponibles (figura 21).

Cuando edite una macro desde **Herramientas > Macros > Editar macros** en el menú, Calc muestra el *IDE de LibreOffice Basic*. Este diálogo permite seleccionar y editar cualquier macro disponible de *LibreOffice Basic*, pero no macros en otros lenguajes.

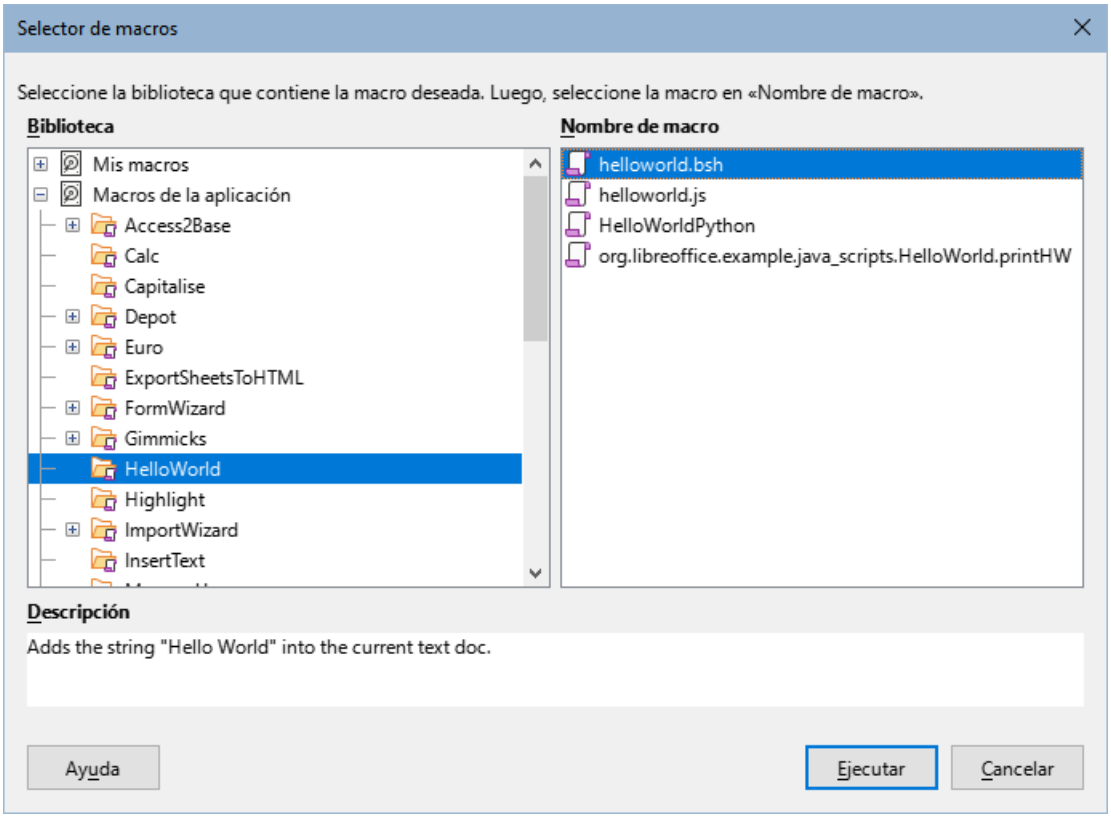

Figura 21: Selector de macros

El modelo de componente utilizado en LibreOffice se conoce como *Universal Network Objects* o *UNO*. Las macros de LibreOffice en cualquier lenguaje de programación utilizan una interfaz *UNO* de ejecución de programación de aplicación *(API)*. La interfaz `XSCRIPTCONTEXT` se ofrece a las secuencias de comandos para los cuatro lenguajes y proporciona un medio de acceso a las varias interfaces que se pueden necesitar para realizar cualquier acción en un documento.

Macros de BeanShell

BeanShell es un lenguaje de programación similar a *Java*, lanzado por primera vez en 1999.

Cuando seleccione **Herramientas > Macros > Organizar macros > BeanShell** desde el menú, Calc mostrará el diálogo *Macros en BeanShell* (figura 22).

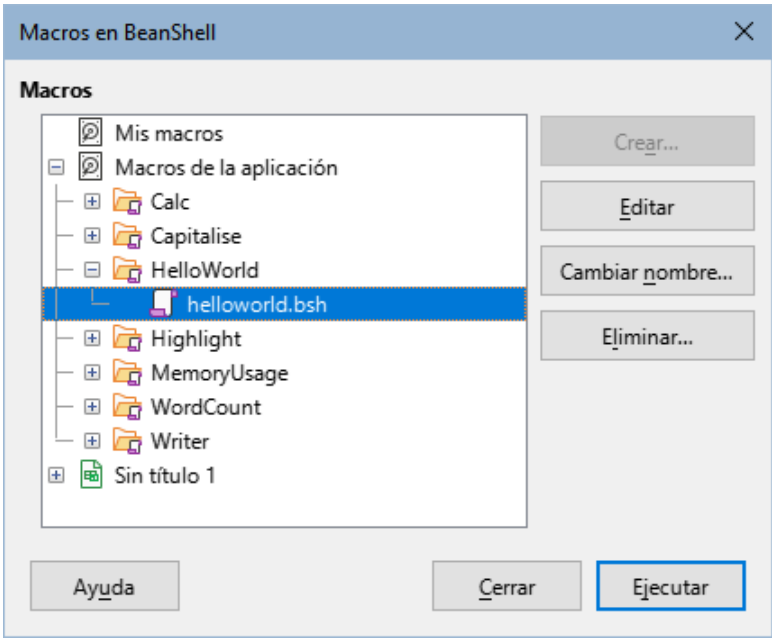

Figura 22: Macros en BeanShell

Haga clic en el botón *Editar* del diálogo *Macros en BeanShell* para acceder a la *ventana de depuración BeanShell* (figura 23).

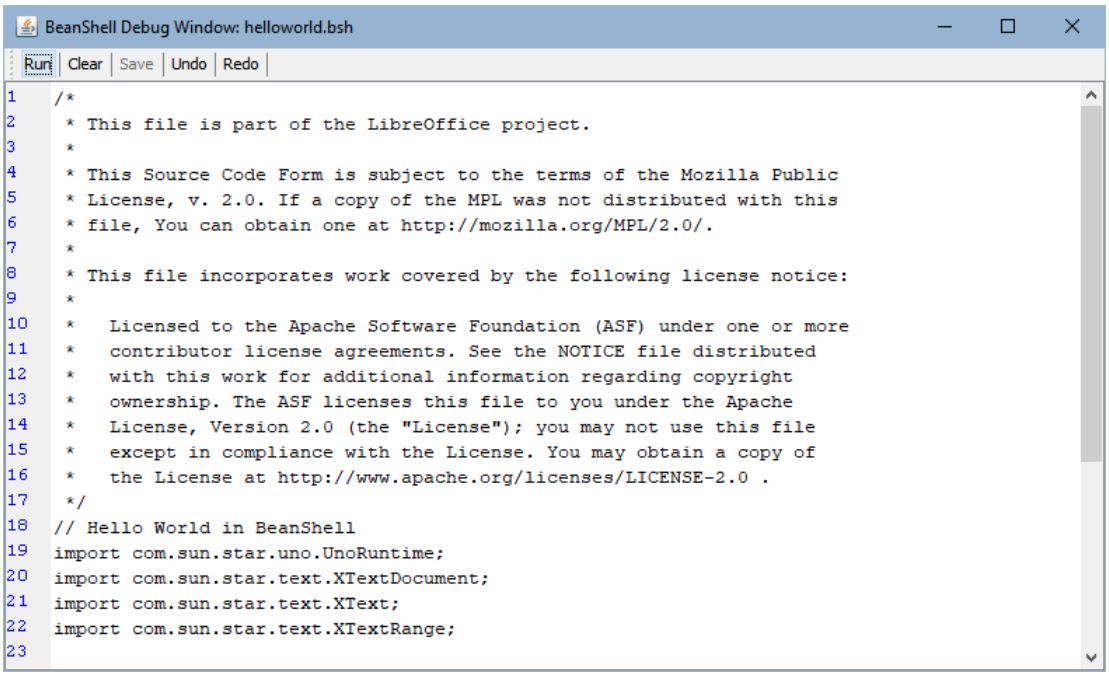

Figura 23: Ventana de depuración BeanShell.

El Listado 10 es un ejemplo de una macro *BeanShell* que inserta el texto «*Hola mundo desde BeanShell*» en la celda `A1` de la hoja de cálculo Calc activa.

Listado 10. Ejemplo de una macro en BeanShell

```
import com.sun.star.uno.UnoRuntime;
import com.sun.star.sheet.XSpreadsheetView;
import com.sun.star.text.XText;
model = XSCRIPTCONTEXT.getDocument();

controller = model.getCurrentController();
```

```
view = UnoRuntime.queryInterface(XSpreadsheetView.class, controller);

sheet = view.getActiveSheet();

cell = sheet.getCellByPosition(0, 0);

cellText = UnoRuntime.queryInterface(XText.class, cell);

textCursor = cellText.createTextCursor();

cellText.insertString(textCursor, "Hola mundo desde BeanShell", true);

return 0;
```

Macros de JavaScript

JavaScript es un lenguaje de programación de alto nivel, lanzado por primera vez en 1995.

Cuando seleccione **Herramientas > Macros > Organizar Macros > JavaScript** en el menú, Calc muestra el diálogo de *Macros en JavaScript* (figura 24).

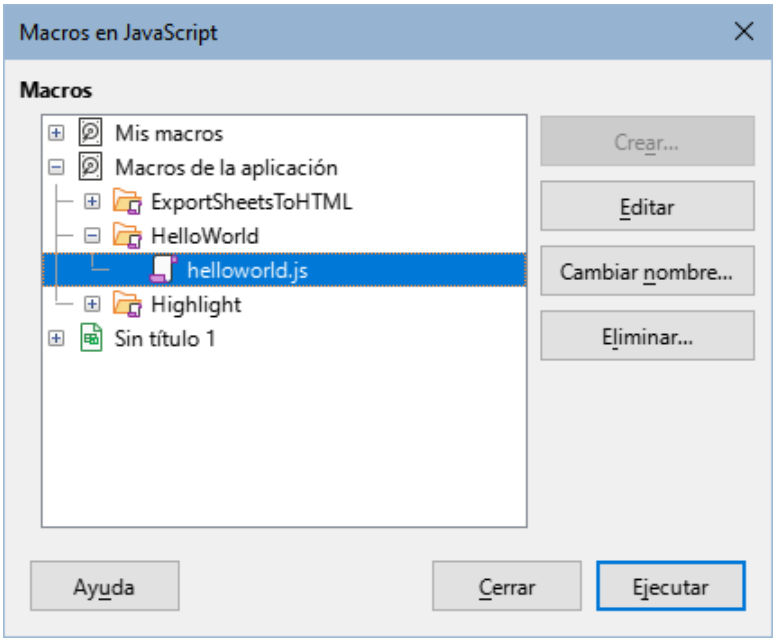

Figura 24: Macros en JavaScript

Haga clic en el botón *Editar* en el diálogo *Macros en JavaScript* para acceder al *depurador Rhino de JavaScript* (figura 25).

Encuentrará instrucciones más detalladas para utilizar esta herramienta en el sitio web de Mozilla en https://github.com/mozilla/rhino.

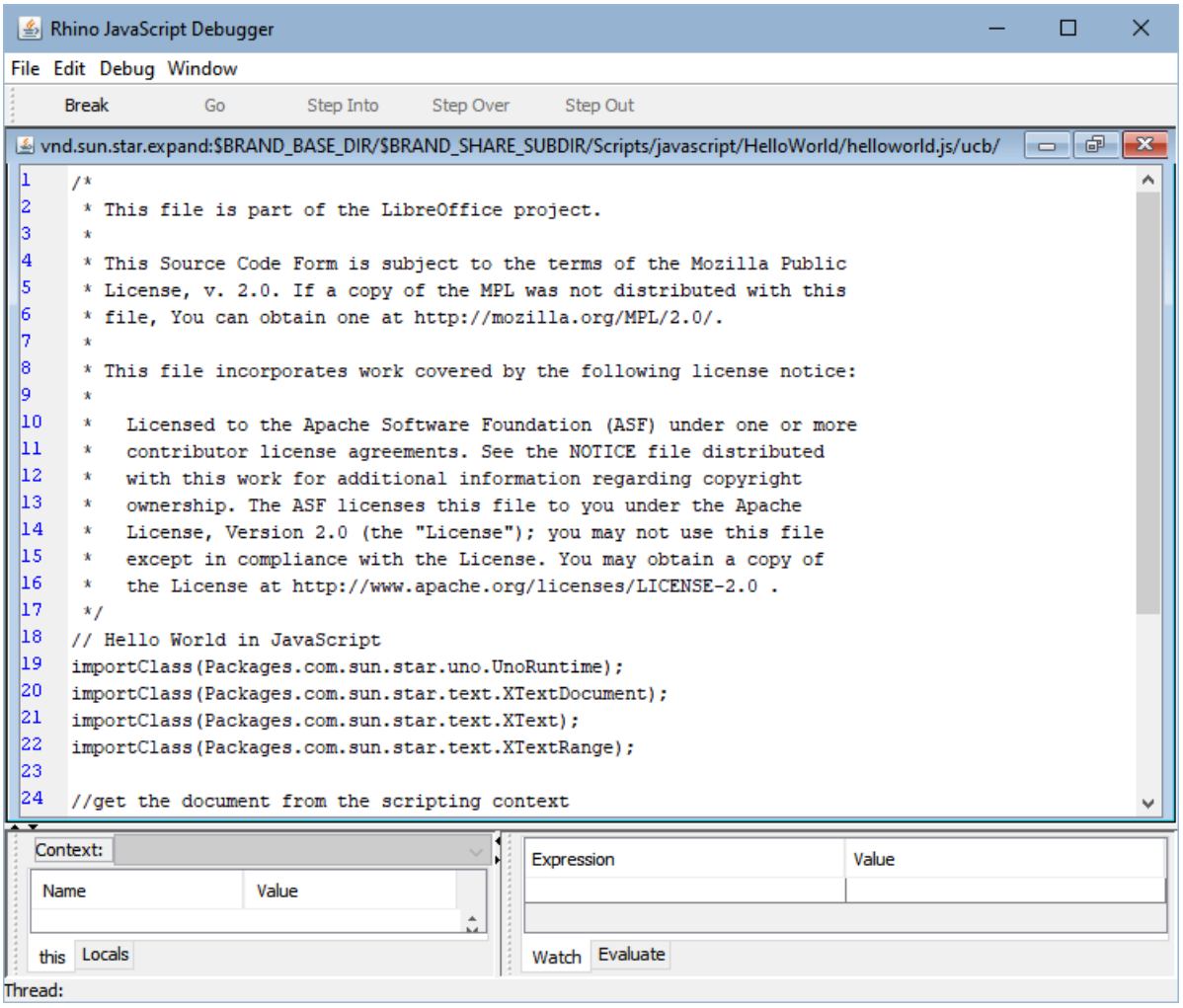

Figura 25: Depurador Rhino de JavaScript

El Listado 11 es un ejemplo de una macro JavaScript que inserta el texto «*Hola mundo desde JavaScript*» en la celda `A1` de la primera hoja en la hoja de cálculo Calc.

Listado 11. Ejemplo de una macro en JavaScript.

```
importClass(Packages.com.sun.star.uno.UnoRuntime);
importClass(Packages.com.sun.star.sheet.XSpreadsheetDocument);
importClass(Packages.com.sun.star.container.XIndexAccess);
importClass(Packages.com.sun.star.table.XCellRange);
importClass(Packages.com.sun.star.table.XCell);

documentRef = XSCRIPTCONTEXT.getDocument();

spreadsheetInterface = UnoRuntime.queryInterface(XSpreadsheetDocument,
documentRef);

allSheets = UnoRuntime.queryInterface(XIndexAccess,
spreadsheetInterface.getSheets());

theSheet = allSheets.getByIndex(0);

Cells = UnoRuntime.queryInterface(XCellRange,theSheet);

cellA1 = Cells.getCellByPosition(0,0);
```

```
theCell = UnoRuntime.queryInterface(XCell,cellA1);

theCell.setFormula("Hola mundo desde JavaScript");
```

Macros de Python

Python es un lenguaje de programación general de alto nivel, lanzado por primera vez en 1991.

Cuando seleccione **Herramientas > Macros > Organizar Macros > Python** del menú, Calc muestra el diálogo *Macros en Python* (figura 26).

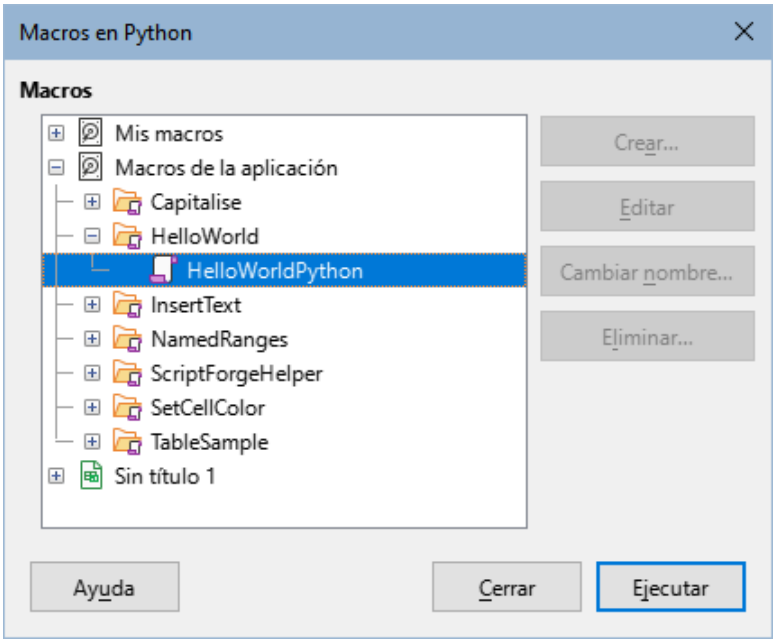

Figura 26: Macros en Python

Las funciones para editar y depurar las secuencias de comandos Python no están por el momento incorporadas en la interfaz de usuario de LibreOffice. No obstante, puede editar las secuencias de comandos Python con el editor de texto que prefiera o en un IDE externo. La extensión *Alternative Python Script Organizer (APSO)* facilita la edición de las secuencias de comandos Python, especialmente cuando están incrustadas en un documento. Al utilizar APSO, puede configurar el editor de código fuente que desee, ejecutar el shell de Python integrado y depurar secuencias de comandos Python.

Para más información, busque Python en el sistema Ayuda de LibreOffice y visite la sección *Diseño y desarrollo de aplicaciones en Python* de la wiki de *The Document Foundation* (https://wiki.documentfoundation.org/Macros/Python_Design_Guide).

El Listado 12 es un ejemplo de una macro *Python* que utiliza la celda `A1` de la primera hoja en una hoja de cálculo Calc para insertar el texto «*Hola mundo desde Python*».

Listado 12. Ejemplo de una macro enPython

```
import uno

def HelloWorld():
  doc = XSCRIPTCONTEXT.getDocument()
  cell = doc.Sheets[0]['A1']
  cell.setString('Hola mundo desde Python')
  return
```

Biblioteca ScriptForge

Los programadores de macros necesitan realizar tareas como crear y abrir archivos, acceder a controles de formulario, leer datos de bases de datos incrustadas en documentos de LibreOffice Base, etc. El objetivo de la biblioteca ScriptForge es simplificar la creación de macros facilitando la ejecución de dichos comandos sin tener que aprender las API (interfaces de programación de aplicaciones) y los comandos necesarios de LibreOffice.

La biblioteca ScriptForge es compatible con LibreOffice Basic y Python. Está organizado en un conjunto de servicios, cada uno de los cuales proporciona métodos y propiedades relacionados con un tema específico. Por ejemplo, el servicio de diálogo brinda acceso a los diálogos disponibles en los módulos de script y el servicio de base de datos permite la ejecución de comandos SQL en documentos base.

El «Capítulo 11, Primeros pasos con macros», de la *Guía de iniciación* contiene material introductorio adicional sobre la biblioteca ScriptForge e incluye un ejemplo simple. Se puede encontrar información más detallada y muchos ejemplos en el sistema de ayuda de LibreOffice, buscando el término "ScriptForge" en el índice.

Inspector de objetos incorporado

LibreOffice tiene una API (interfaz de programación de aplicaciones) extensa que los programadores de macros pueden usar para automatizar casi cualquier aspecto de sus aplicaciones. Sin embargo, uno de los principales desafíos para los programadores es descubrir los tipos de objetos UNO (Universal Network Objects), así como los servicios, métodos y propiedades admitidos.

El inspector de objetos integrado se puede utilizar para ayudar en la inspección de objetos y descubrir cómo se puede acceder a ellos y utilizarlos. Para acceder a esta herramienta, vaya a **Herramientas > Herramientas de desarrollo** en el menú y se abrirá una ventana de inspección de objetos (figura 27). De manera predeterminada, esta ventana está anclada en la parte inferior de la interfaz de usuario.

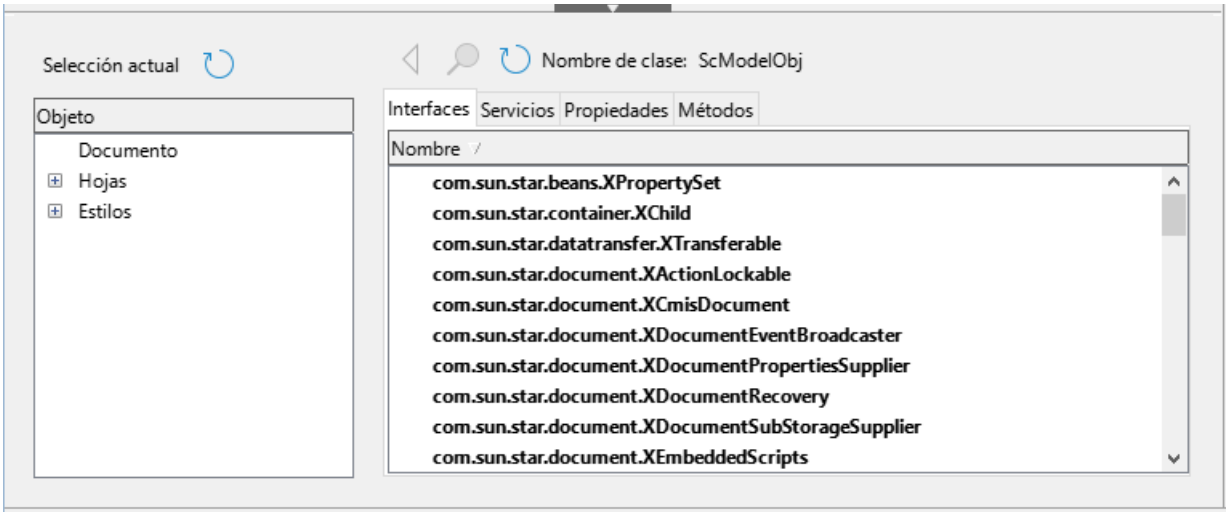

Figura 27: Ventana del inspector de objetos.

La parte izquierda de la ventana consta del navegador del modelo de objetos del documento (DOM), que permite al usuario navegar por todos los objetos del documento. Cuando se selecciona un objeto, la siguiente información sobre el objeto se muestra en pestañas dentro de la parte derecha de la ventana:

- Los nombres de todas las interfaces implementadas.

- Los nombres de todos los servicios soportados por el objeto.
- Los nombres y tipos de todas las propiedades disponibles en el objeto.
- Los nombres, argumentos y tipos de retorno de todos los métodos que puede llamar el objeto.

En lugar de inspeccionar objetos utilizando el navegador DOM, es posible inspeccionar directamente un objeto seleccionado en el documento al pulsar el botón *Selección actual*.

El título «Iniciación a las macros» de la *Guía de iniciación* contiene información adicional sobre el inspector de objetos integrado. Puede encontrar información más detallada y ejemplos en el sistema de ayuda de LibreOffice, buscando el término «Herramientas de desarrollo» en el índice de ayuda.

Trabajando con macros VBA

Para el programador de Excel/VBA, LibreOffice Basic es un lenguaje de programación muy similar a VBA. Aunque Calc puede leer un libro de Excel, la razón principal por la que VBA no funciona en Calc es que Calc utiliza un mecanismo diferente para acceder a los componentes del libro, como las celdas de la hoja de trabajo. Específicamente, los objetos, atributos y métodos usan diferentes nombres y el comportamiento correspondiente es ligeramente diferente en algunas ocasiones.

Para convertir el código VBA, primero debe cargar el código VBA en LibreOffice.

Cargando código VBA

En la página de Propiedades de VBA (**Herramientas > Opciones > Cargar/Guardar > Propiedades de VBA**), puede elegir si desea mantener las macros en los documentos de Microsoft Office que se abren en LibreOffice.

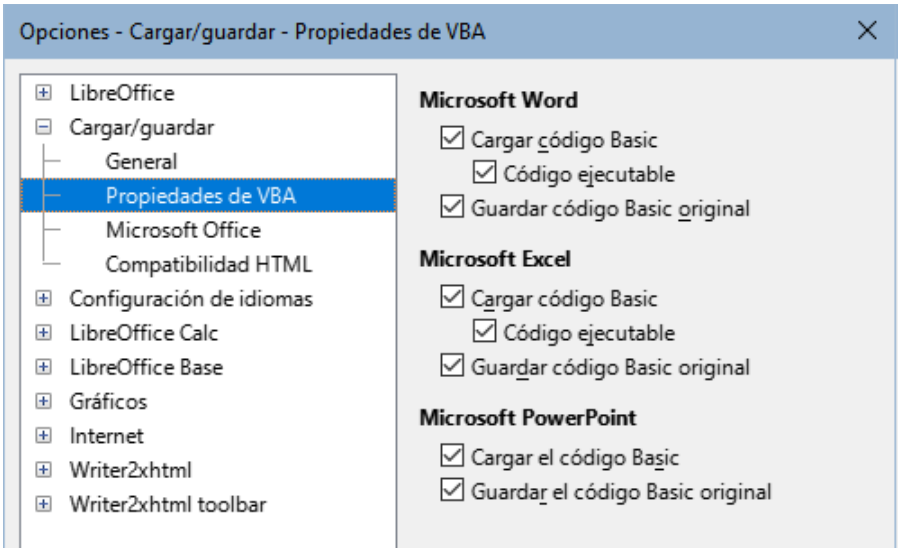

Figura 28: Opciones - Cargar/Guardar - Propiedades de VBA.

Si elige *Cargar código Basic*, puede editar las macros en LibreOffice. El código modificado se guarda en un documento ODF pero no se conserva si lo guarda en un formato de Microsoft Office.

Si elige *Guardar código Basic original*, las macros no funcionarán en LibreOffice pero se mantendrán sin cambios si guarda el archivo en formato de Microsoft Office.

Si está importando un archivo de Microsoft Word o Excel que contiene código VBA, puede seleccionar la opción *Código ejecutable*. Mientras que el código se conserva pero se vuelve

inactivo (si lo inspecciona con el IDE básico notará que está todo comentado), con esta opción el código está listo para ejecutarse.

Guardar código Basic original tiene prioridad sobre *Cargar código Basic*. Si se seleccionan ambas opciones y edita el código deshabilitado en LibreOffice, el código VBA original se guardará al guardar en un formato de Microsoft Office.

Para eliminar cualquier posible virus de macro del documento de Microsoft Office, desmarque *Guardar código Basic original*. El documento se guardará sin el código VBA.

Instrucción Option VBASupport

La instrucción Option VBAsupport especifica que LibreOffice Basic admitirá algunas instrucciones, funciones y objetos de VBA. La declaración debe agregarse antes del código del programa ejecutable en un módulo.

Nota

El soporte para VBA no está completo pero cubre una gran parte de los patrones de uso comunes.

Cuando VBASupport está habilitado, los argumentos de función de LibreOffice Basic y los valores devueltos son los mismos que sus contrapartes de VBA. Cuando está deshabilitado, las funciones de LibreOffice Basic pueden aceptar argumentos y devolver valores diferentes a los de sus contrapartes de VBA.

Listado 13. Uso de la Option VBASupport

```
Option VBASupport 1
Sub Ejemplo
  Dim sVar As Single
  sVar = Worksheets("Sheet1").Range("A1")
  Print sVar
End Sub
```

Sin la instrucción Option VBASupport, el código del Listado 13 debe convertirse a LibreOffice Basic en el Listado 14.

Listado 14. Código VBA convertido

```
Sub Ejemplo
  Dim sVar As Single
  Dim oSheet as Object
  Dim oCell as Object
  ' Worksheets(«Sheet1»).
  oSheet = ThisComponent.getSheets().getByIndex(0)
  ' Range("A1")
  oCell = oSheet.getCellByPosition(0, 0)
  sVar = oCell.getValue()
  Print sVar
End Sub
```

`Option VBASupport` puede afectar negativamente o ayudar en las siguientes situaciones:

- Permitir caracteres especiales como identificadores. Todos los caracteres definidos como letras en el juego de caracteres Latin-1 (ISO 8859-1) se aceptan como parte de los identificadores. Por ejemplo, variables con caracteres acentuados en sus nombres.
- Crear constantes VBA que incluyan caracteres no imprimibles (`vbCrLf`, `vbNewLine`,...).
- Admite las sentencias `Private/Public` para los procedimientos.
- Declaración `Set` obligatoria para objetos.

- Valores predeterminados para parámetros opcionales en los procedimientos.
- Argumentos con nombre cuando existen múltiples parámetros opcionales.
- Precarga de librerías de LibreOffice Basic.

VBA UserForms (diálogos de LibreOffice Basic)

Los formularios de usuario (diálogos) aparecen con frecuencia en macros que exigen su interacción y selección de parámetros. El fragmento de código que sigue es una receta para estas conversiones, que no se manejan automáticamente por las opciones de VBA.

Listado 15. Visualización VBA de un UserForm [Diálogo] llamado «MiForm»

```
Sub MyProc
  MiForm.Show
End Sub
```

Listado 16. Pantalla de LibreOffice Basic de un UserForm [Diálogo] llamado «MiForm»

```
' oDlg debe ser visible a nivel del módulo
Dim oDlg As Object
Sub MyProc
  DialogLibraries.LoadLibrary("Standard")
  oDlg = CreateUnoDialog(DialogLibraries.Standard.MiForm)
  oDlg.execute()
End Sub
```

Nota

La variable oDlg es visible a nivel de módulo para todos los demás procedimientos que acceden a los controles del diálogo. Esto significa que todos los procedimientos que manipulan o acceden a los controles en el diálogo están alojados en un solo módulo.

Conclusión

Este capítulo proporciona un resumen de cómo crear bibliotecas y módulos, cómo utilizar la grabadora de macros, cómo utilizar las macros como funciones de Calc y cómo escribir sus propias macros sin la grabadora de macros. Cada tema merece al menos un capítulo y aprender a escribir sus propias macros para Calc podría fácilmente requerir una guía completa. En resumen, esto es solo el comienzo de todo lo que puede aprender.

Si ya está familiarizado con el lenguaje BASIC (o con un lenguaje de programación), La página web de la «Documentación oficial en español» ofrece un conjunto de tarjetas de referencia rápida de LibreOffice Basic que pueden serle útiles.

Puede obtenerse más información sobre las características de las macros de Calc en el sistema *Ayuda*, las páginas de *wiki* de *The Document Foundation* (por ejemplo, https://wiki.documentfoundation.org/Macros) y otras fuentes de Internet (por ejemplo, el sitio de preguntas y respuestas http://ask.libreoffice.org/).