Guías B4x

# Lenguaje Basic B4x

Copyright: © 2017 Anywhere Software

Edición 1.0

Última actualización: 2017.05.25

1 BASIC ..... 6
1.1 B4x ..... 6
1.1.1 B4A Android ..... 6
1.1.2 B4i iOS ..... 6
1.1.3 B4J Escritorio Java ..... 6
1.1.4 B4R Arduino ..... 6
2 Variables y objetos ..... 8
2.1 Tipos de variables ..... 8
2.2 Nombres de variables ..... 11
2.3 Declarando variables ..... 11
2.3.1 Variables simples ..... 11
2.3.2 Matriz de variables ..... 12
2.3.3 Matriz de views / nodes (objetos) ..... 14
2.3.4 Variables Type, solo B4A, B4i y B4J ..... 17
2.4 Conversión de tipos ..... 18
2.5 Ámbito ..... 19
2.5.1 Variables de proceso ..... 19
2.5.2 Variables de activity, solo B4A ..... 20
2.5.3 Variables locales ..... 20
2.6 Consejos ..... 20
3 Lenguaje Basic ..... 21
3.1 Expresiones ..... 21
3.1.1 Expresiones matemáticas ..... 21
3.1.2 Expresiones relacionales ..... 22
3.1.3 Expresiones booleanas ..... 22
3.2 Palabras clave estándar ..... 23
Abs (Number As Double) As Double ..... 25
ACos (Value As Double) As Double ..... 25
ACosD (Value As Double) As Double ..... 25
Array ..... 25
Asc (Char As Char) As Int ..... 25
ASin (Value As Double) As Double ..... 25
ASinD (Value As Double) As Double ..... 25
ATan (Value As Double) As Double ..... 25
ATan2 (Y As Double, X As Double) As Double ..... 25
ATan2D (Y As Double, X As Double) As Double ..... 25
ATanD (Value As Double) As Double ..... 25
BytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String ..... 26
CallSub (Component As Object, Sub As String) As Object ..... 26
CallSub2 (Component As Object, Sub As String, Argument As Object) As Object ..... 26
CallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object ..... 26
Catch ..... 26
cE As Double ..... 26
Ceil (Number As Double) As Double ..... 26
CharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String ..... 27
Chr (UnicodeValue As Int) As Char ..... 27
Continue ..... 27
Cos (Radians As Double) As Double ..... 27
CosD (Degrees As Double) As Double ..... 27
cPI As Double ..... 27
CRLF As String ..... 27

Dim....27
Exit....27
False As Boolean....28
Floor (Number As Double) As Double....28
For....28
GetType (object As Object) As String....28
If....28
IsNumber (Text As String) As Boolean....28
LoadBitmap (Dir As String, FileName As String) As Bitmap....29
LoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap....29
Log (Message As String)....29
Logarithm (Number As Double, Base As Double) As Double....29
LogColor (Message As String, Color As Int)....29
Max (Number1 As Double, Number2 As Double) As Double....29
Me As Object....29
Min (Number1 As Double, Number2 As Double) As Double....29
Not (Value As Boolean) As Boolean....29
Null As Object....29
NumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String....30
NumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String....30
Power (Base As Double, Exponent As Double) As Double....30
QUOTE As String....30
Regex As Regex....30
Return....30
Rnd (Min As Int, Max As Int) As Int....30
RndSeed (Seed As Long)....30
Round (Number As Double) As Long....30
Round2 (Number As Double, DecimalPlaces As Int) As Double....31
Select....31
Sender As Object....31
Sin (Radians As Double) As Double Ángulo medido en radianes....31
SinD (Degrees As Double) As Double....31
Calcula la función trigonométrica Seno Ángulo medido en grados....31
Sleep (Value As Double) As Double....31
Sqrt (Value As Double) As Double....31
Sub....32
SubExists (Object As Object, Sub As String) As Boolean....32
TAB As String....32
Tan (Radians As Double) As Double....32
TanD (Degrees As Double) As Double....32
True As Boolean....32
Try....32
Type....33
Until....33
While....33
3.3 Instrucciones condicionales....34
3.3.1 If – Then – End If....34
3.3.2 Select – Case....36
3.4 Estructuras Loop....38

3.4.1 For – Next ....38
3.4.2 For - Each ....39
3.4.3 Do - Loop ....40
3.5 Subs ....42
3.5.1 Declaración ....42
3.5.2 Llamadas ....42
3.5.3 Llamar a una Sub desde otro módulo ....42
3.5.4 Nombres ....43
3.5.5 Parámetros ....43
3.5.6 Valor devuelto ....43
3.6 Subs resumibles ....44
3.6.1 Sleep ....44
3.6.2 Wait For ....45
3.6.3 Flujo de código ....47
3.6.4 Esperar la terminación de una sub para reanudar otra ....48
3.6.5 DoEvents ....49
3.6.6 Cuadros de Diálogo ....50
3.6.7 SQL con Wait For ....51
3.6.7.1 Consultas ....51
3.6.7.2 B4J ....52
3.6.8 Notas y consejos ....53
3.7 Eventos ....54
3.7.1 B4A ....54
3.7.2 B4i ....57
3.7.3 B4J ....59
3.7.4 B4R ....62
3.7.5 Resumen del interfaz de usuario ....63
3.8 Librerías ....64
3.8.1 Librerías estándar ....64
3.8.2 Carpeta de librerías adicionales ....65
3.8.3 Cargar y actualizar una librería ....66
3.8.4 Mensaje de error "¿falta la referencia de una librería?" ....66
3.9 Manipulación de cadenas ....67
3.9.1 B4A, B4i, B4J ....67
3.9.2 B4R ....69
3.10 Formato de números ....72
3.10.1 B4A, B4i, B4J ....72
3.10.2 B4R ....72
3.11 Temporizadores ....73
3.12 Archivos B4A, B4i, B4J ....75
3.12.1 El objeto 'File' ....75
3.12.2 Nombres de archivo ....77
3.12.3 Subcarpetas ....77
3.12.4 TextWriter ....78
3.12.5 TextReader ....79
3.12.6 Codificación del texto ....80
3.13 Lists, solo B4A, B4i y B4J ....82
3.14 Maps, solo B4A, B4i y B4J ....84
4 Herramientas de ayuda ....86
4.1 Función de búsqueda en el foro ....86
4.2 Visor de ayuda de B4x (B4xHelp) ....88
4.3 Documentación de ayuda - B4A Object Browser ....91

Principales colaboradores: Klaus Christl (klaus), Erel Uziel (Erel)
Traducción: Manuel Martínez

**Para buscar por una palabra o sentencia, usa la función Búsqueda en el menú Editar.**

Actualizado para las siguientes versiones:
B4A versión 7.00
B4i versión 4.01
B4J versión 5.51
B4R versión 1.80

# 1 BASIC

En la programación de computadoras, **BASIC,** siglas de **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode (**Código simbólico de instrucciones de propósito general para principiantes** en español), es una familia de lenguajes de programación de alto nivel. El BASIC original, el Dartmouth BASIC, fue diseñado en 1964 por John George Kemeny y Thomas Eugene Kurtz en el Dartmouth College en Nuevo Hampshire, Estados Unidos, como un medio para facilitar la programación en ordenadores a estudiantes (y profesores) que no fueran de ciencias. En aquella época, casi todo el uso de los ordenadores requería codificar software hecho a medida, con lo cual quedaba restringido a personas con formación como científicos y matemáticos (fuente: Wikipedia).

## 1.1 B4x

**B4X** es un conjunto de IDEs (Entorno de desarrollo integrado) para desarrollo rápido de aplicaciones, que permite la creación de aplicaciones para las siguientes plataformas: Google Android, Apple iOS, Java, Raspberry Pi y Arduino. B4X usa un dialecto de Visual Basic (en adelante llamado "B4X") y también cuenta con un diseñador visual que simplifica el proceso de creación de la interfaz de usuario.

### 1.1.1 B4A Android

B4A – La forma más fácil de desarrollar apps nativas para Android.
B4A incluye todas las características necesarias para desarrollar rápidamente cualquier tipo de aplicación para Android.
B4A es utilizado por decenas de miles de desarrolladores de todo el mundo, incluyendo empresas como la NASA, HP, IBM y otros.
Junto con B4i ahora puedes desarrollar fácilmente aplicaciones para Android y iOS.

### 1.1.2 B4i iOS

B4i – La forma más fácil de desarrollar apps nativas para iOS...
B4i es una herramienta de desarrollo para aplicaciones nativas de iOS.
B4i sigue los mismos conceptos que B4A, lo que permite la reutilización de la mayor parte del código y crear aplicaciones para Android y iOS.
Es la única herramienta de desarrollo que permite desarrollar el 100% de apps nativas de iOS sin un local ordenador Mac.

### 1.1.3 B4J Java Desktop

B4J – Moderna herramienta, parecida a VB6, para el desarrollo de aplicaciones de escritorio, servidores y soluciones de IoT, independiente de la plataforma.
B4J es una herramienta de desarrollo **100% gratuita** para escritorio, servidores y soluciones de IoT.
Con B4J puedes crear fácilmente aplicaciones de escritorio (IU), programas de consola (sin interfaz de usuario) y soluciones de servidor.
Las aplicaciones compiladas se pueden ejecutar en Windows, Mac, Linux y placas ARM (como Raspberry Pi).

### 1.1.4 B4R Arduino

B4R – Crea fácilmente programas nativos para Arduino y ESP8266.

B4R es una herramienta de desarrollo **100% gratuita** para crear programas nativos para Arduino y ESP8266.
B4R sigue los mismos conceptos de otras herramientas B4X, proporcionando una sencilla y potente herramienta de desarrollo.
B4R, B4A, B4J y B4i juntos, conforman la mejor solución de desarrollo para el Internet de las Cosas (IoT - Internet of Things).

# 2 Variables y objetos

Una **variable** es un nombre simbólico dado a alguna, conocida o no, cantidad de información, con el fin de permitir que sea usada de forma independiente de la información que representa. Un nombre de variable, en código fuente de ordenador, generalmente se asocia con una ubicación de almacenamiento de datos y por lo tanto también su contenido, y estos pueden cambiar durante el curso de la ejecución del programa (fuente: Wikipedia).

Hay dos tipos de variables: primitivas y no-primitivas.
Las primitivas incluyen los tipos de variables numéricas: Byte, Short, Int, Long, Float y Double.
También incluyen: Boolean y Char.

## 2.1 Tipos de variables

**B4A, B4i, B4J**

Lista de tipos de variables y sus rangos:

| B4 | Tipo | Valor min | Valor max |
|---|---|---|---|
| Boolean | booleano | Falso | Verdadero |
| Byte | Entero de 8 bits | $-2^{7}$<br>-128 | $2^{7}-1$<br>127 |
| Short | Entero de 16 bits | $-2^{15}$<br>- 32768 | $2^{15}-1$<br>32767 |
| Int | Entero de 32 bits | $-2^{31}$<br>-2147483648 | $2^{31}-1$<br>2147483647 |
| Long | Entero largo de 64 bits | $-2^{63}$<br>-9223372036854775808 | $2^{63}-1$<br>9223372036854775807 |
| Float | Número en coma flotante<br>32 bits | $-2^{-149}$<br>1.4E-45 | $(2-2^{-23}) * 2^{127}$<br>3.4028235 E 38 |
| Double | Número de precisión doble<br>64 bits | $-2^{-1074}$<br>2.2250738585072014 E -308 | $(2-2^{-52}) * 2^{1023}$<br>1.7976931348623157 E 308 |
| Char | Carácter | | |
| String | Matriz de caracteres | | |

**B4R**

Lista de tipos de variables y sus rangos:
Tipos numéricos:

**Byte** 0 - 255
**Int** (2 bytes) -32768 - 32768. Similar al tipo Short de las otras herramientas B4x.
**UInt** (2 bytes) 0 - 65535. Especifico de B4R.
**Long** (4 bytes) -2,147,483,648 - 2,147,483,647. Similar al tipo Int de las otras herramientas B4x.
**ULong** (4 bytes) 0 - 4,294,967,295 Especifico de B4R.
**Double** (4 bytes) Número en coma flotante de 4 bytes. Similar al tipo Float de las otras herramientas B4x.
Float es el mismo que Double. Short es el mismo que Int.

Lo anterior es cierto en todas las placas, incluyendo la Arduino Due.

Otros tipos:
**Boolean** Verdadero o Falso. Prácticamente se guarda como un byte con el valor de 1 o 0.
**String** Las cadenas de caracteres están hechas de una matriz de bytes que terminan con un byte nulo (byte con el valor de 0).
**Object** Los objetos pueden contener otros tipos de valores

Las variables de tipo primitiva, se pasan siempre por valor a otras subs o cuando son asignadas a otras variables.
Por ejemplo:

```
Sub S1
   Private A As Int
   A = 12                    La variable A = 12
   S2(A)                     Se pasa por valor a la rutina S2
   Log(A)  ' Muestra 12      La variable A todavía vale 12, aunque B fue cambiada en la rutina S2.
End Sub

Sub S2(B As Int)             Variable B = 12
   B = 45                    Su valor es cambiado a B = 45
End Sub
```

Todos los otros tipos de variables, incluyendo las matrices de variables primitivas, se clasifican como variables no-primitivas.
Cuando se pasa una variable no-primitiva a una sub o cuando se asigna a una variable distinta, se pasa una copia de la referencia.
Esto significa que los datos en sí no están duplicados.
Es ligeramente diferente pasar una variable por referencia ya que no se puede cambiar la referencia de la variable original.

Todos los tipos pueden ser tratados como objetos.
Colecciones como lists y maps, trabajan con objetos y por lo tanto pueden almacenar cualquier valor.

Aquí vemos un error común donde el desarrollador intenta añadir varias matrices a una lista:

```
Private arr(3) As Int
Private List1 As List
List1.Initialize
For i = 1 To 5
   arr(0) = i * 2
   arr(1) = i * 2
   arr(2) = i * 2
   List1.Add(arr)    'Se agrega toda la matriz como un solo elemento
Next
arr = List1.Get(0)   'Obtenemos el primer elemento de la lista
Log(arr(0))          '¿Qué va a ser mostrado aquí???
```

Podemos esperar que se muestre 2. Sin embargo, se muestra 10.
Hemos creado una matriz y añadido 5 elementos de esta matriz a la lista.
Los valores en la matriz son los valores establecidos en la última iteración.
Para solucionar esto tenemos que crear una nueva matriz con cada iteración.
Esto se hace declarando como Private la matriz en cada iteración:

```
Private arr(3) As Int 'Esta línea es redundante en este caso.
Private List1 As List
List1.Initialize
For i = 1 To 5
   Private arr(3) As Int
   arr(0) = i * 2
   arr(1) = i * 2
   arr(2) = i * 2
   List1.Add(arr) 'Se añade toda la matriz como un solo elemento
Next
arr = List1.Get(0) 'Obtenemos el primer elemento de la lista
Log(arr(0)) 'se muestra 2
```

## 2.2 Nombres de variables

Depende de nosotros darle cualquier nombre a una variable, a excepción de las palabras reservadas. Un nombre de variable debe comenzar con una letra y debe estar compuesto de los siguientes caracteres A-Z, a-z, 0-9, y el guión bajo "_", sin espacios, paréntesis etc.
Los nombres de variables no son sensibles a mayúsculas, lo que significa que Index e index se refieren a la misma variable.

Pero es una buena práctica darles nombres que tengan sentido.
Ejemplo:

| | |
|---|---|
| Interest = Capital * Rate / 100 | Tiene sentido |
| n1 = n2 * n3 / 100 | No tiene sentido |

Para Views (B4A, B4i), Nodes (B4J), es útil añadir al nombre un prefijo de tres caracteres que define su tipo.
Ejemplos:

| | | |
|---|---|---|
| `lblCapital` | lbl > Label | Capital > propósito |
| `edtInterest` | edt > EditText | Interest > propósito |
| `btnNext` | btn > Button | Next > propósito |

## 2.3 Declarando variables

### 2.3.1 Variables simples

Las variables son declaradas con la palabra clave `Private` o `Public` seguido por el nombre de la variable y la palabra clave `As` seguido del tipo de variable. Para más detalles leer el capítulo Ámbito.
También existe la palabra clave `Dim` , esta se mantiene por compatibilidad.

Ejemplos:

```
Private Capital As Double
Private Interest As Double
Private Rate As Double
```

Declara tres variables como Double, números de precisión doble.

```
Private i As Int
Private j As Int
Private k As Int
```

Declara tres variables como Int, números enteros.

```
Private lblCapital As Label
Private lblInterest As Label
Private lblRate As Label
```

Declara tres variables como ‘views’ de tipo Label.

```
Private btnNext As Button
Private btnPrev As Button
```

Declara dos variables como ‘views’ de tipo Button.

Las mismas variables también pueden ser declaradas de una forma más corta.

```
Private Capital, Interest, Rate As Double
Private i, j, k As Int
Private lblCapital, lblInterest, lblRate As Label
Private btnNext, btnPrev As Button
```

Los nombres de las variables separados por comas y seguido por la declaración del tipo de variable.

Las siguientes declaraciones de variables son válidas:

```
Private i = 0, j = 2, k = 5 As Int

Private txt = "test" As String, value = 1.05 As Double, flag = False As Boolean
```

Las vistas deben ser declaradas si queremos acceder a ellas mediante código.
Por ejemplo, si queremos cambiar el texto en una vista Label mediante código, de esta forma:
`lblCapital.Text = "1200"`,
Tenemos que hacer referencia a esta vista Label por el nombre `lblCapital` que le hemos dado mediante la declaración `Private`.
Si no necesitamos acceder a esta vista Label mediante código, no necesitamos declararla.
El uso de una rutina de evento para una vista no necesita que esta sea declarada.

Para asignar un valor a una variable, se debe escribir su nombre seguido por el signo igual y seguido por el valor, de esta forma:

```
Capital = 1200
LastName = "SMITH"
```

Ten en cuenta que para `Capital` escribimos solo `1200` porque `Capital` es un número.
Pero para `LastName` escribimos `"SMITH"` porque `LastName` es una cadena de texto.
Las cadenas de texto siempre deben ser escritas entre comillas dobles.

## 2.3.2 Matriz de variables

Las matrices (Arrays), son colecciones de datos u objetos, cuyos elementos pueden ser seleccionados mediante índices. Las matrices pueden tener varias dimensiones.
La declaración contiene las palabras clave `Private` o `Public` seguida por el nombre de la variable `LastName`, el número de elementos entre paréntesis `(50)`, la palabra clave `As` y el tipo de variable `String`.
Para más detalles consulta el capítulo Ámbito. También existe la palabra clave `Dim` , esta se mantiene por compatibilidad.

**Nota: B4R sólo admite matrices de una única dimensión!**

Ejemplos:

| | |
|---|---|
| `Public LastName(50) As String` | Matriz de tipo String de una dimensión, elementos totales 50. |
| `Public Matrix(3, 3) As Double` | Matriz de tipo Double de dos dimensiones, elementos totales 9. |
| `Public Data(3, 5, 10) As Int` | Matriz de tipo Int de tres dimensiones, elementos totales 150. |

El índice del primer elemento en cada dimensión es 0.
`LastName(0), Matrix(0,0), Data(0,0,0)`

El índice del último elemento en cada dimensión es igual al número de elementos menos 1.
`LastName(49), Matrix(2,2), Data(2,4,9)`

```
Public LastName(10) As String
Public FirstName(10) As String
Public Address(10) As String
Public City(10) As String
```
o
```
Public LastName(10), FirstName(10), Address(10), City(10) As String
```

Este ejemplo muestra cómo obtener acceso a todos los elementos en una matriz de tres dimensiones.

```
Public Data(3, 5, 10) As Int

For i = 0 To 2
   For j = 0 To 4
      For k = 0 To 9
         Data(i, j, k) = ...
      Next
   Next
Next
```

Una forma más versátil de declarar matrices es usando variables.

```
Public NbPers = 10 As Int
Public LastName(NbPers) As String
Public FirstName(NbPers) As String
Public Address(NbPers) As String
Public City(NbPers) As String
```

Declaramos la variable `Public NbPers = 10 As Int` y establecemos su valor en 10.
Luego declaramos las matrices con esta variable en lugar del número 10 como antes.
La gran ventaja de hacerlo de esta forma es que si en algún momento queremos cambiar el número de elementos, debemos cambiar solo UN valor.

Para una matriz de datos podemos utilizar el siguiente código.

```
Public NbX = 2 As Int
Public NbY = 5 As Int
Public NbZ = 10 As Int
Public Data(NbX, NbY, NbZ) As Int
```

Y la rutina de acceso.

```
For i = 0 To NbX - 1
   For j = 0 To NbY - 1
      For k = 0 To NbZ - 1
         Data(i, j, k) = ...
      Next
   Next
Next
```

Llenar una matriz con la palabra clave Array :

```
Public Name() As String
Name = Array As String("Miller", "Smith", "Johnson", "Jordan")
```

### 2.3.3 Matrices de views / nodes (objetos)

Views / nodes u objetos pueden estar en una matriz. El siguiente código muestra un ejemplo:
En B4A y B4i los objetos de la interfaz son llamados *views* y llamados *nodes* en B4J.

En el siguiente ejemplo los Buttons se añaden al view / node padre mediante código.

**B4A**

```
Sub Globals
   Private Buttons(6) As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private i As Int

   For i = 0 To 5
      Buttons(i).Initialize("Buttons")
      Activity.AddView(Buttons(i), 10dip, 10dip + i * 60dip, 150dip, 50dip)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

**B4i**

```
Sub Process_Globals

   Private Buttons(6) As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)

   Private i As Int
   For i = 0 To 5
      Buttons(i).Initialize("Buttons")
      Page1.RootPanel.AddView(Buttons(i), 10dip, 10dip + i * 60dip, 150dip, 50dip)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

**B4J**

```
Sub Process_Globals

   Private Buttons(6) As Button
End Sub

Sub AppStart (Form1 As Form, Args() As String)

   Private i As Int
   For i = 0 To 5
      Buttons(i).Initialize("Buttons")
      MainForm.RootPane.AddNode(Buttons(i), 10, 10 + i * 60, 150, 50)
      Buttons(i).Tag = i + 1
      Buttons(i).Text = "Test " & (i + 1)
   Next
End Sub

Sub Buttons_MouseClicked (EventData As MouseEvent)
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

Los Buttons podrían también ser añadidos mediante un archivo de diseño, en este caso no necesitan se inicializados, ni añadidos al view / node padre y las propiedades Text y Tag también se deben establecer en el diseñador.
En ese caso, el código sería este:

**B4A**

```
Sub Globals
   Private b1, b2, b3, b4, b5, b6, b7 As Button
   Private Buttons() As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)

   Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

### B4i

```
Sub Process_Globals

   Private b1, b2, b3, b4, b5, b6, b7 As Button
   Private Buttons(6) As Button
End Sub

Private Sub Application_Start (Nav As NavigationController)

   Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub

Sub Buttons_Click
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

### B4J

```
Sub Process_Globals

   Private b1, b2, b3, b4, b5, b6, b7 As Button
   Private Buttons(6) As Button
End Sub

Sub AppStart (Form1 As Form, Args() As String)

   Buttons = Array As Button(b1, b2, b3, b4, b5, b6, b7)
End Sub

Sub Buttons_MouseClicked (EventData As MouseEvent)
   Private btn As Button
   btn = Sender
   Log("Button " & btn.Tag & " clicked")
End Sub
```

## 2.3.4 Variables Type, solo B4A, B4i y B4J

**Las variables Type No pueden ser privadas. Una vez declaradas se puede acceder a ellas desde cualquier sitio (similar a los módulos de Clase).**
El mejor lugar para declararlas es en la rutina Process_Globals en el módulo Main.

Vamos a volver a utilizar el ejemplo con los datos de una persona.
En lugar de declarar cada parámetro por separado, podemos definir una variable Type:

```
Public NbUsers = 10 As Int
Type Person(LastName As String, FirstName As String. Address As String, City As String)
Public User(NbUsers) As Person
Public CurrentUser As Person
```

La nueva variable Type es `Person` , a continuación declaramos una variable simple y una matriz como variables de tipo Person.
Para acceder a un elemento en particular, podemos utilizar el siguiente código.

```
CurrentUser.FirstName
CurrentUser.LastName

User(1).LastName
User(1).FirstName
```

El nombre de la variable, seguido de un punto y el parámetro deseado.
Si la variable es una matriz, entonces debemos indicar el índice entre paréntesis después del nombre.

Es posible asignar una variable Type a otra variable del mismo tipo, como se muestra a continuación.

```
CurrentUser = User(1)
```

## 2.4 Conversión de tipos

B4x evalúa los tipos de la variable automáticamente según sea necesario. También evalúa los números como cadenas de texto y viceversa de forma automática.
En muchos casos, se necesita explícitamente evaluar un Objeto como un tipo específico.
Esto se puede hacer asignando el objeto a una variable del tipo requerido.
Por ejemplo, la palabra clave Sender hace referencia al objeto que provocó el evento.
El siguiente código cambia el color del button pulsado.
Ten en cuenta que hay varios buttons que comparten la misma rutina para el evento Click.

```
Sub Globals
   Private Btn1, Btn2, Btn3 As Button
End Sub

Sub Activity_Create(FirstTime As Boolean)
   Btn1.Initialize("Btn")
   Btn2.Initialize("Btn")
   Btn3.Initialize("Btn")
   Activity.AddView(Btn1, 10dip, 10dip, 200dip, 50dip)
   Activity.AddView(Btn2, 10dip, 70dip, 200dip, 50dip)
   Activity.AddView(Btn3, 10dip, 130dip, 200dip, 50dip)
End Sub

Sub Btn_Click
   Private btn As Button
   btn = Sender     ' Definimos el objeto Sender como Button
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

El código anterior también se podría haber escrito de forma más elegante:

```
Sub Globals

End Sub

Sub Activity_Create(FirstTime As Boolean)
   Private i As Int
   For i = 0 To 9   ' creamos 10 Buttons
      Private Btn As Button
      Btn.Initialize("Btn")
      Activity.AddView(Btn, 10dip, 10dip + 60dip * i, 200dip, 50dip)
   Next
End Sub

Sub Btn_Click
   Private btn As Button
   btn = Sender     ' Definimos el objeto Sender como Button
   btn.Color = Colors.RGB(Rnd(0, 255), Rnd(0, 255), Rnd(0, 255))
End Sub
```

## 2.5 Ámbito

### 2.5.1 Variables de proceso

Estas variables son accesibles todo el tiempo de vida del proceso. Deben ser declaradas dentro de la rutina Process_Globals. Esta rutina es llamada una vez cuando se inicia el proceso (esto es cierto para todos los módulos, no solo para el módulo main).
Estas variables son "public". Lo que significa que pueden ser accedidas también desde otros módulos.

Sin embargo, en B4A, no todos los tipos de objetos pueden ser declarados como variables de proceso.
Por ejemplo, views / nodes no pueden ser declarados como variables de proceso.
La razón, es que no queremos mantener una referencia a objetos que deben ser destruidos junto con la Activity. En otras palabras, una vez que la Activity se destruye, todas las views que contenía esa Activity también son destruidas.
Si mantenemos una referencia a una View, el recolector de basura no será capaz de liberar recursos y vamos a tener un problema de memoria El compilador hace cumplir este requisito.

Para acceder a variables de proceso desde otros módulos, debes indicar el nombre del módulo, donde han sido declaradas, como prefijo.
Ejemplo:
Variable declarada en un módulo con el nombre: *MyModule*

```
Sub Process_Globals
   Public MyVar As String
End Sub
```

Accediendo a la variable desde el módulo *MyModule*:

```
   MyVar = "Text"
```

Accediendo a la variable desde cualquier otro módulo:

```
   MyModule.MyVar = "Text"
```

Las Variables pueden ser declaradas con:

```
Dim MyVar As String
```

En este caso la variable es pública igual que usando la palabra clave Public.

Es una buena práctica declarar las variables así:

```
Public MyVar As String
```

Esta variable es pública.

Es posible declarar variables privadas en la rutina Process_Globals de esta forma:

```
Private MyVar As String
```

Esta variable es privada a la Activity o al módulo donde fue declarada.
Para Activities es mejor declararlas en la rutina Globals.

Para variables declaradas en un módulo de Clase en la rutina Class_Globals, las mismas reglas de arriba son válidas.

```
   Public MyVarPublic As String        ' pública
   Private MyVarPublic As String       ' privada
   Dim MyVar As String                 ' pública como Public
```

El uso de Dim en una rutina Class_Globals no es recomendable!

### 2.5.2 Variables de Activity, solo B4A

Estas variables pertenecen a la Activity.
Deben ser declaradas dentro de la rutina Globals.
Estas variables son "private" y solo pueden ser accedidas desde el módulo de Activity donde han sido declaradas.
Todos los tipos de objeto pueden ser declarados como variables de Activity.
Cada vez que la Activity es creada, la rutina Globals es llamada (antes que Activity_Create).
Estas variables existen mientras la Activity exista.

### 2.5.3 Variables locales

Las variables declaradas en una subrutina son locales a esa subrutina.
Son "private" y solo pueden ser accedidas desde la subrutina donde fueron declaradas.
Todos los tipos de objetos pueden ser declarados como variables locales.
En cada llamada a la subrutina, las variables locales son inicializadas a su valor por defecto o a cualquier otro valor que sea definido en el código y son 'destruidas' al salir de la subrutina.

## 2.6 Consejos

Un view / node puede ser asignado a una variable de modo que podemos cambiar fácilmente sus propiedades.

Por ejemplo, el siguiente código desactiva todas las Views contenidas en un Panel / Pane:

```
For i = 0 To MyPanel.NumberOfViews - 1
   Private v As View
   v = MyPanel.GetView(i)
   v.Enabled = False
Next
```

Si solo queremos deshabilitar los buttons:

```
For i = 0 To.NumberOfViews - 1
   Private v As View
   v = MyPanel.GetView(i)
   If v Is Button Then 'comprobar si se trata de un Button
      v.Enabled = False
   End If
Next
```

Nota: `MyPanel` es un *Panel* en B4A y B4i pero es un *Pane* en B4J.

# 3 Lenguaje Basic

## 3.1 Expresiones

Una expresión, en un lenguaje de programación, es una combinación de valores explícitos, constantes, variables, operadores y funciones que se interpretan de acuerdo a las reglas de precedencia y de asociación para un determinado lenguaje de programación, que calcula y luego produce (devuelve) otro valor. Este proceso, como para las expresiones matemáticas, se denomina evaluación. El valor puede ser de diversos tipos, tales como numérico, cadena, y lógica (fuente: Wikipedia).

Por ejemplo, 2 + 3 es una expresión aritmética y de programación que se evalúa a 5. Una variable es una expresión porque es un puntero a un valor en memoria, de modo que y + 6 es una expresión. Un ejemplo de una expresión relacional es 4 = 4 que se evalúa como verdadero (fuente: Wikipedia).

### 3.1.1 Expresiones matemáticas

| Operador | Ejemplo | Nivel de prioridad | Operación |
|---|---|---|---|
| + | `x + y` | 3 | Suma |
| – | `x - y` | 3 | Resta |
| * | `x * y` | 2 | Multiplicación |
| / | `x / y` | 2 | División |
| `Mod` | `x Mod y` | 2 | Modulo |
| `Power` | `Power(x,y)` $x^y$ | 1 | Potencia de |

Nivel de prioridad: En una expresión, operaciones con el nivel 1 son evaluadas antes que las operaciones con el nivel 2, que son evaluadas antes que las operaciones con el nivel 3.

Ejemplos:

`4 + 5 * 3 + 2 = 21` > `4 + 15 + 2`

`(4 + 5) * (3 + 2) = 45` > `9 * 5`

$(4 + 5)^2 * (3 + 2) = 405$ > $9^2 * 5$ > `81 * 5`
`Power(4+5,2)*(3+2)`

`11 Mod 4 = 3` > Mod es el resto de 10 / 4

$23^3$ `Power(23,3)` > 23 elevado a la potencia de 3

$- 2^2 = - 4$
$(-2)^2 = 4$

### 3.1.2 Expresiones relacionales

En ciencias de la computación, las expresiones relacionales, se comprueba algún tipo de relación entre dos entidades. Estos incluyen la igualdad numérica (por ejemplo., 5 = 5) y desigualdades (por ejemplo, 4 >= 3).
En B4x estos operadores devuelven **Verdadero** o **Falso**, dependiendo de si se cumple la relación condicional entre los dos operandos o no.

| Operador | Ejemplo | Se utiliza para comprobar |
|---|---|---|
| = | x = y | La equivalencia de los dos valores |
| <> | x <> y | La no equivalencia de los dos valores |
| > | x > y | Si el valor a la izquierda de la expresión es mayor que el de la derecha |
| < | x < y | Si el valor a la izquierda de la expresión es menor que el de la derecha |
| >= | x >= y | Si el valor a la izquierda de la expresión es mayor o igual que el de la derecha |
| <= | x <= y | Si el valor a la izquierda de la expresión es menor o igual que el de la derecha |

### 3.1.3 Expresiones booleanas

En ciencias de la computación, una expresión Booleana, es una expresión que produce un valor Booleano cuando se evalúa, es decir, **Verdadero** o **Falso**. Una expresión Booleana puede estar compuesta por una combinación de constantes Booleanas **Verdadero** o **Falso**, Variables tipo Booleanas, Operadores de valor Booleano, y funciones de valor Booleano (fuente: Wikipedia).

Los operadores Booleanos se utilizan en instrucciones condicionales como IF-Then y Select-Case.

| Operador | Comentario |
|---|---|
| O | O Booleano Z = X Or Y Z = verdadera si X o Y son verdaderas, o ambas son verdaderas |
| And | Y Booleano Z = X And Y Z = verdadera si X e Y son ambas verdaderas |
| Not ( ) | NO Booleano X = verdadero Y = Not(X) > Y = Falso |

| | | O | And |
|---|---|---|---|
| X | Y | Z | Z |
| Falso | Falso | Falso | Falso |
| Verdadero | Falso | Verdadero | Falso |
| Falso | Verdadero | Verdadero | Falso |
| Verdadero | Verdadero | Verdadero | Verdadero |

## 3.2 Palabras Clave Estándar

No todas las palabras clave están disponibles en B4R.

- **Abs** (Number As Double) As Double
- **ACos** (Value As Double) As Double
- **ACosD** (Value As Double) As Double
- **Array**
- **Asc** (Char As Char) As Int
- **ASin** (Value As Double) As Double
- **ASinD** (Value As Double) As Double
- **ATan** (Value As Double) As Double
- **ATan2** (Y As Double, X As Double) As Double
- **ATan2D** (Y As Double, X As Double) As Double
- **ATanD** (Value As Double) As Double
- **BytesToString** (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String
- **CallSub** (Component As Object, Sub As String) As Object
- **CallSub2** (Component As Object, Sub As String, Argument As Object) As Object
- **CallSub3** (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object
- **Catch**
- **cE** As Double
- **Ceil** (Number As Double) As Double
- **CharsToString** (Chars() As Char, StartOffset As Int, Length As Int) As String
- **Chr** (UnicodeValue As Int) As Char
- **Continue**
- **Cos** (Radians As Double) As Double
- **CosD** (Degrees As Double) As Double
- **cPI** As Double
- **CRLF** As String
- **Dim**
- **Exit**
- **False** As Boolean
- **Floor** (Number As Double) As Double
- **For**
- **GetType** (object As Object) As String
- **If**
- **Is**
- **IsNumber** (Text As String) As Boolean
- **LoadBitmap** (Dir As String, FileName As String) As Bitmap
- **LoadBitmapSample** (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap
- **Log** (Message As String)
- **Logarithm** (Number As Double, Base As Double) As Double
- **LogColor** (Message As String, Color As Int)
- **Max** (Number1 As Double, Number2 As Double) As Double
- **Me** As Object
- **Min** (Number1 As Double, Number2 As Double) As Double
- **Not** (Value As Boolean) As Boolean

- **Null** As Object
- **NumberFormat** (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String
- **NumberFormat2** (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String
- **Power** (Base As Double, Exponent As Double) As Double
- **QUOTE** As String
- **Regex** As Regex
- **Return**
- **Rnd** (Min As Int, Max As Int) As Int
- **RndSeed** (Seed As Long)
- **Round** (Number As Double) As Long
- **Round2** (Number As Double, DecimalPlaces As Int) As Double
- **Select**
- **Sender** As Object
- **Sin** (Radians As Double) As Double
- **SinD** (Degrees As Double) As Double
- **Sleep** (Milliseconds As Int)
- **SmartStringFormatter** (Format As String, Value As Object) As String
- **Sqrt** (Value As Double) As Double
- **Sub**
- **SubExists** (Object As Object, Sub As String) As Boolean
- **TAB** As String
- **Tan** (Radians As Double) As Double
- **TanD** (Degrees As Double) As Double
- **True** As Boolean
- **Try**
- **Type**
- **Until**
- **While**

### Abs (Number As Double) As Double

Devuelve el valor absoluto.

### ACos (Value As Double) As Double

Devuelve el ángulo medido en radianes.

### ACosD (Value As Double) As Double

Devuelve el ángulo medido en grados.

### Array

Crea una matriz unidimensional del tipo especificado.
La sintaxis es la siguiente: Array [As tipo] (lista de valores).
Si el tipo es omitido, se creará una matriz de objetos.
Ejemplo:

```
Dim Days() As String
Days = Array As String("Sunday", "Monday", ...)
```

### Asc (Char As Char) As Int

Devuelve el código Unicode del carácter o el primer carácter en la cadena.

### ASin (Value As Double) As Double

Devuelve el ángulo medido en radianes.

### ASinD (Value As Double) As Double

Devuelve el ángulo medido en grados.

### ATan (Value As Double) As Double

Devuelve el ángulo medido en radianes.

### ATan2 (Y As Double, X As Double) As Double

Devuelve el ángulo medido en radianes.

### ATan2D (Y As Double, X As Double) As Double

Devuelve el ángulo medido en grados.

### ATanD (Value As Double) As Double

Devuelve el ángulo medido en grados.

### BytesToString (Data() As Byte, StartOffset As Int, Length As Int, CharSet As String) As String

Decodifica la matriz de bytes como una cadena de texto.
Data - matriz de bytes.
StartOffset - Primer byte a leer.
Length - Número de bytes a leer.
CharSet - El nombre del conjunto de caracteres.
Ejemplo:

```
Dim s As String
s = BytesToString(Buffer, 0, Buffer.Length, "UTF-8")
```

### CallSub (Component As Object, Sub As String) As Object

Llama a una rutina. CallSub puede ser usado para llamar a una rutina que pertenece a un módulo diferente.
Sin embargo, la rutina solo será llamada si el módulo no está en pausa. En este caso, una cadena vacía será devuelta.
Puedes utilizar IsPaused para comprobar si un módulo está en pausa
Esto significa que un Activity no puede llamar a una rutina de una Activity diferente. Ya que la otra Activity seguro que está en pausa.
CallSub permite a una Activity llamar a una rutina de servicio y permite a un servicio llamar a una rutina de Activity.
Ten en cuenta que no es posible llamar a rutinas de módulos de código.
CallSub también puede ser usado para llamar a rutinas del módulo actual. Usar 'Me' como 'component' en ese caso.
Ejemplo:

```
CallSub(Main, "RefreshData")
```

### CallSub2 (Component As Object, Sub As String, Argument As Object) As Object

Similar a CallSub. Llama a una rutina con un único argumento

### CallSub3 (Component As Object, Sub As String, Argument1 As Object, Argument2 As Object) As Object

### Catch

Cualquier excepción dentro de un bloque Try se verá atrapada en el bloque Catch.
Llamar a LastException para obtener la excepción.
Sintaxis:

```
Try
  ...
Catch
  ...
End Try
```

### cE As Double

e (número de Euler, base de los logaritmos naturales) constante.

### Ceil (Number As Double) As Double

Devuelve el menor Double que es mayor o igual que el número especificado y es un número entero.

### CharsToString (Chars() As Char, StartOffset As Int, Length As Int) As String

Crea una nueva cadena de texto desde una matriz de caracteres.
Comenzará la copia en el elemento que indica StartOffset y el número de caracteres copiados será igual a Length.

### Chr (UnicodeValue As Int) As Char

Devuelve el carácter que está representado por el valor Unicode indicado.

### Continue

Detiene la ejecución de la iteración actual y continúa con la siguiente.

### Cos (Radians As Double) As Double

Calcula la función trigonométrica coseno. Ángulo medido en radianes

### CosD (Degrees As Double) As Double

Calcula la función trigonométrica coseno. Ángulo medido en grados

### cPI As Double

Constante PI.

### CRLF As String

Carácter de nueva línea. El valor de Chr(10).

### Dim

Declara una variable.
Sintaxis:
Declarar una única variable:
Dim nombre de variable [As type] [= expression]
El tipo predeterminado es String.

Declarar varias variables. Todas las variables serán del tipo especificado.
Dim [Const] variable1 [= expression], variable2 [= expression], ..., [As type]
Ten en cuenta que la sintaxis abreviada sólo se aplica a la palabra clave Dim.
Ejemplo: `Dim a = 1, b = 2, c = 3 As Int`

Declarar una matriz:
Dim variable(Rank1, Rank2, ...) [As type]
Ejemplo: `Dim Days(7) As String`
El rango real puede ser omitido para matrices de longitud cero.

### Exit

Sale del bucle más interno.
Ten en cuenta que 'Exit' dentro de un bloque 'Select' saldrá del bloque 'Select'.

### False As Boolean

### Floor (Number As Double) As Double

Devuelve el menor Double que es menor o igual que el número especificado y es un número entero..

### For

Sintaxis:
```
For variable = value1 To value2 [intervalo]
  ...
Next
```
Si la variable de iteración no se ha declarado antes, será del tipo Int.

O:
```
For Each variable As type In collection
  ...
Next
```
Ejemplos:
```
For i = 1 To 10
  Log(i) 'mostrará de 1 a 10 (incluido).
Next
For Each n As Int In Numbers 'una matriz
  Sum = Sum + n
Next
```

Ten en cuenta que los límites del bucle se comprueban una vez antes de la primera iteración.

### GetType (object As Object) As String

Devuelve una cadena que representa el tipo de objeto java.

### If

En una línea:
```
If condición Then instrucción si es verdadero [Else instrucción si es falso]
```
Multilínea:
```
If condición Then
  instrucción
Else If condición Then
  instrucción
  ...
Else
  instrucción
End If
```

### IsNumber (Text As String) As Boolean

Comprueba si la cadena de texto especificada puede ser analizada de forma segura como un número.

### LoadBitmap (Dir As String, FileName As String) As Bitmap

Carga un mapa de bits.
Ten en cuenta que el sistema de archivos de Android es sensible a mayúsculas y minúsculas.
Debes considerar el uso de LoadBitmapSample si el tamaño de la imagen es grande.
El tamaño real del archivo no es relevante ya que por lo general se almacenan comprimidas.
Ejemplo:
Activity.SetBackgroundImage(LoadBitmap(File.DirAssets, "SomeFile.jpg"))

### LoadBitmapSample (Dir As String, FileName As String, MaxWidth As Int, MaxHeight As Int) As Bitmap

Carga un mapa de bits.
El decodificador cambiará las proporciones si MaxWidth o MaxHeight son más pequeñas que las dimensiones de la imagen.
Esto puede ahorrar una gran cantidad de memoria cuando se cargan imágenes de gran tamaño.
Ejemplo:

```
Panel1.SetBackgroundImage(LoadBitmapSample(File.DirAssets, "SomeFile.jpg",
Panel1.Width, Panel1.Height))
```

### Log (Message As String)

Registra un mensaje. Los mensajes registrados se pueden ver en la pestaña Logs.

### Logarithm (Number As Double, Base As Double) As Double

### LogColor (Message As String, Color As Int)

Registra un mensaje. El mensaje se mostrará en la pestaña Logs con el color especificado.

### Max (Number1 As Double, Number2 As Double) As Double

Devuelve el número mayor de los dos especificados.

### Me As Object

Para Clases: Devuelve una referencia a la instancia actual.
Para Activities y servicios: Devuelve una referencia a un objeto que puede ser utilizado con CallSub, CallSubDelayed y SubExists.
No se puede utilizar en los módulos de código.

### Min (Number1 As Double, Number2 As Double) As Double

Devuelve el número menor de los dos especificados.

### Not (Value As Boolean) As Boolean

Invierte el valor de la variable booleana.

### Null As Object

### NumberFormat (Number As Double, MinimumIntegers As Int, MaximumFractions As Int) As String

Convierte un número en una cadena.
La cadena incluirá al menos la cantidad de dígitos de la parte entera como indique MinimumIntegers y como máximo la cantidad de dígitos de la parte decimal como indique MaximumFractions.
Ejemplo:

```
Log(NumberFormat(12345.6789, 0, 2)) '"12,345.68"
Log(NumberFormat(1, 3 ,0)) '"001"
```

### NumberFormat2 (Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean) As String

Convierte un número en una cadena.
La cadena incluirá al menos la cantidad de dígitos de la parte entera como indique MinimumIntegers, como máximo la cantidad de dígitos de la parte decimal como indique MaximumFractions y como mínimo la cantidad de dígitos de la parte decimal como indique MinimumFractions.
GroupingUsed - Determina si se incluirá un separador cada 3 dígitos.
Ejemplo:

```
Log(NumberFormat2(12345.67, 0, 3, 3, false)) '"12345.670"
```

### Power (Base As Double, Exponent As Double) As Double

Devuelve el valor de 'Base' elevado a la potencia de 'Exponent'.

### QUOTE As String

Dobles comillas. El valor de Chr(34).

### Regex As Regex

Métodos relacionados con expresiones regulares.

### Return

Regresa de la actual rutina y, opcionalmente, devuelve un valor.
Sintaxis: `Return [valor]`

### Rnd (Min As Int, Max As Int) As Int

Devuelve un número entero aleatorio entre 'Min' (incluido) y 'Max' (excluido).

### RndSeed (Seed As Long)

Establece el valor del número devuelto por la función Rnd.
Este método puede ser utilizado para depuración, ya que permite obtener los mismos resultados cada vez.

### Round (Número As Double) As Long

Devuelve el número 'Long' más cercano al número dado.

### Round2 (Número As Double, DigitosDecimales As Int) As Double

Redondea el número dado a los decimales indicados.

### Select

Compara un valor único con varios valores.
Ejemplo:

```
Dim value As Int
value = 7
Select value
  Case 1
    Log("Uno")
  Case 2, 4, 6, 8
    Log("Par")
  Case 3, 5, 7, 9
    Log("Impar mayor que uno")
  Case Else
    Log("mayor que 9")
End Select
```

### Sender As Object

Devuelve el objeto que provocó el evento.
Solo es válido dentro de la rutina del evento.
Ejemplo:

```
Sub Button_Click
  Dim b As Button
  b = Sender
  b.Text = "He sido pulsado"
End Sub
```

### Sin (Radians As Double) As Double

Calcula la función trigonométrica Seno. Ángulo medido en radianes

### SinD (Degrees As Double) As Double

Calcula la función trigonométrica Seno Ángulo medido en grados

### Sleep (Value As Double) As Double

Pausa la ejecución de la actual rutina y se reanuda después del tiempo especificado..

### SmartStringFormatter (Format As String, Value As Object) As String

Palabra clave interna utilizada por 'Smart String literal'.

### Sqrt (Value As Double) As Double

Devuelve la raíz cuadrada positiva.

### Sub

Declara una rutina con los parámetros y el tipo de variable que devuelve.
Sintaxis: Sub nombre [(lista de parámetros)] [As tipo-de-variable-devuelta]
En los parámetros se deben incluir el nombre y el tipo.
La longitud de las dimensiones de las matrices no debe ser incluida.
Ejemplo:

```
Sub MySub (FirstName As String, LastName As String, Age As Int, OtherValues() As
Double) As Boolean
  ...
End Sub
```

En este ejemplo OtherValues es una matriz de una sola dimensión.

### SubExists (Object As Object, Sub As String) As Boolean

Comprueba si el objeto incluye el método especificado.
Devuelve Falso si el objeto no se ha inicializado o no es una estancia de la clase de usuario.

### TAB As String

Carácter Tabulador.

### Tan (Radians As Double) As Double

Calcula la función trigonométrica Tangente. Ángulo medido en radianes

### TanD (Degrees As Double) As Double

Calcula la función trigonométrica Tangente. Ángulo medido en grados

### True As Boolean

### Try

Cualquier excepción dentro de un bloque Try se verá atrapada en el bloque Catch.
Llamar a LastException para obtener la excepción.
Sintaxis:

```
Try
...
Catch
...
End Try
```

## Type

Declara una estructura.
Solo puede ser usada dentro de la rutina Globals o Process_Globals.
Sintaxis:
Type nombre (campo1, campo2, ...)
Los campos deben incluir el nombre y el tipo.
Ejemplo:

```
Type MyType (Name As String, Items(10) As Int)
Dim a, b As MyType
a.Initialize
a.Items(2) = 123
```

## Until

Repite hasta que la condición sea verdadera.
Sintaxis:

```
Do Until condición
  ...
Loop
```

## While

Repite mientras la condición sea verdadera.
Sintaxis:
Do While condición
...
Loop

## 3.3 Instrucciones condicionales

Diferentes sentencias condicionales están disponibles en Basic.

### 3.3.1 If – Then – End If

La estructura **If-Then-Else** permite comparar condiciones y ejecutar diferentes secciones del código, según el resultado de la comparación.
Caso general:

```
If test1 Then
   ' código1
Else If test2 Then
   'código2
Else
   ' código3
End If
```

La estructura **If-Then-Else** funciona de la siguiente manera:

1. Al llegar a la línea con la palabra clave **`If`**, **`test1`** se comprueba.
2. Si el resultado de la prueba es **True**, entonces **`código1`** se ejecuta hasta la línea con la palabra clave **`Else If`**. Y continúa en la siguiente línea de la palabra clave **`End If`**.
3. Si el resultado es **False**, entonces **`test2`** se comprueba.
4. Si el resultado es **True**, entonces **`código2`** se ejecuta hasta la línea con la palabra clave **`Else`**.
   Y continúa en la siguiente línea de la palabra clave **`End If`**.
5. Si el resultado es **False**, entonces **`código3`** se ejecuta y continúa en la siguiente línea de la palabra clave **`End If`**.

Las comprobaciones pueden ser cualquier tipo de prueba condicional que devuelva **True** o **False**.
Algunos ejemplos:

```
If b = 0 Then
   a = 0
End If
```

La estructura **If-Then** más simple.

```
If b = 0 Then a = 0
```

Los mismo, pero en una línea.

```
If b = 0 Then
   a = 0
Else
   a = 1
End If
```

La estructura **If-Then-Else** más simple.

```
If b = 0 Then a = 0 Else a = 1
```

Los mismo, pero en una línea.

Personalmente, prefiero la estructura en varias líneas, se lee mejor.
Un viejo hábito de HP Basic hace algunas décadas, este Basic solo aceptaba una instrucción por línea.

Nota. Diferencia entre:

| B4x | VB |
|---|---|
| `Else If` | `ElseIf` |

En B4x hay un espacio entre **`Else`** e **`If`**.

Algunos usuarios intentan utilizar esta notación:

```
If b = 0 Then a = 0 : c = 1
```

Hay una gran diferencia entre B4x y VB que da errores :
El código anterior es equivalente a :

B4x

```
If b = 0 Then
   a = 0
End If
c = 1
```

VB

```
If b = 0 Then
   a = 0
   c = 1
End If
```

El carácter de dos puntos ' : ' en la línea anterior se trata en B4x como un carácter CR de Retorno de Carro.

Esta estructura dará un error.

```
Sub Plus1 : x = x + 1 : End Sub
```

No puedes tener una declaración Sub y End Sub en la misma línea.

## 3.3.2 Select – Case

La estructura **Select - Case** permite comparar una**TestExpression** con otras **Expressions** y ejecutar diferentes secciones de código de acuerdo a la coincidencia entre **TestExpression** y **Expressions**.

Caso general:

```
Select TestExpression
Case ExpressionList1
   ' código1
Case ExpressionList2
   'código2
Case Else
   ' código3
End Select
```

**TestExpression** es la expresión a comprobar.

**ExpressionList1** es una lista de expresiones para comparar con **TestExpression**

**ExpressionList2** es otra lista de expresiones para comparar con **TestExpression**

La estructura **Select - Case** funciona de la siguiente manera:

1. **TestExpression** es evaluada.
2. Si un elemento de **ExpressionList1** coincide con **TestExpression** entonces se ejecuta **código1** y continúa en la siguiente línea de la palabra clave **End Select**.
3. Si un elemento de **ExpressionList2** coincide con **TestExpression** entonces se ejecuta **código2** y continúa en la siguiente línea de la palabra clave **End Select**.
4. Si no hay ninguna coincidencia con **TestExpression** se ejecuta **código3** y continúa en la siguiente línea de la palabra clave **End Select**.

**TestExpression** puede ser cualquier expresión o valor.
**ExpressionList1** es una lista de cualquier expresión o valores.

Ejemplos:

```
Select Value
Case 1, 2, 3, 4
```

La variable Value es un valor numérico.

```
Select a + b
Case 12, 24
```

**TestExpression** es la suma de a + b

```
Select Txt.CharAt
Case "A", "B", "C"
```

**TestExpression** es un carácter

```
Sub Activity_Touch (Action As Int, X As Float, Y As Float)
   Select Action
   Case Activity.ACTION_DOWN

   Case Activity.ACTION_MOVE

   Case Activity.ACTION_UP

   End Select
End Sub
```

Nota. Diferencias entre:

| B4x | VB |
|---|---|
| `Select Value` | `Select Case Value` |
| `Case 1,2,3,4,8,9,10` | `Case 1 To 4 , 8 To 9` |

En VB la palabra clave `Case` se añade después de la palabra clave `Select`.
VB acepta `Case 1 To 4` , esto no está implementado en B4x.

## 3.4 Estructuras de bucle

Diferentes estructuras de bucle están disponibles en Basic.

### 3.4.1 For – Next

En un bucle **For–Next** el mismo código se ejecuta un número determinado de veces.
Ejemplo:

```
For i = n1 To n2 Step n3
   'código especifico
Next
```

| | |
|---|---|
| `i` | variable a incrementar |
| `n1` | valor inicial |
| `n2` | valor final |
| `n3` | Valor de incremento |

El bucle **For–Next** funciona de la siguiente manera:

1. Al principio la variable a incrementar **i** es igual al valor inicial **n1**.
   i = n1
2. El código especifico entre las palabras claves `For` y `Next` se ejecuta.
3. Al llegar a **Next**, Se incrementa **i** con el valor de incremento de **n3**.
   i = i + n3.
4. El programa salta a `For`, compara si la variable a incrementar **i** es menor o igual que el valor final **n2**.
   comprueba si i <= n2
5. Si **es igual o menor**, el programa continua en el paso 2, la línea siguiente a la palabra clave `For`.
6. Si **No es igual o menor**, el programa continua en la línea siguiente a la palabra clave `Next`.

Si el valor de incremento es igual a '+1' la palabra clave 'step' no es necesaria.

```
For i = 0 To 10
Next
```

es lo mismo que

```
For i = 0 To 10 Step 1
Next
```

El valor incremental puede ser negativo.

```
For i = n3 To 0 Step -1

Next
```

Es posible salir de un bucle For – Next con la palabra clave `Exit`.

```
For i = 0 To 10
   ' código
   If A = 0 Then Exit
   ' código
Next
```

En este ejemplo, si la variable A es igual a 0

Entonces sale del bucle.

**Nota :** Diferencias entre:

| B4 | VB |
|---|---|
| `Next` | `Next i` |
| `Exit` | `Exit For` |

En VB :

- La variable a incrementar se añade después de la palabra clave **`Next`**.
- El tipo de bucle se especifica después de la palabra clave **`Exit`**.

## 3.4.2 For - Each

Es una variante del bucle For - Next.

Ejemplo:

```
For Each n As Type In Array
   'código especifico
Next
```

| | |
|---|---|
| n | variable de cualquier tipo de objeto |
| Type | tipo de la variable n |
| Array | Matriz de valores u objetos |

El bucle **For–Each** funciona de la siguiente manera:

1. Al principio, **n** toma el valor del primer elemento de la matriz.
   n = Array(0)
2. El código especifico entre las palabras claves **`For`** y **`Next`** se ejecuta.
3. Al llegar a **Next**, el programa si **n** es el último elemento de la matriz.
4. Si **No es el último**, la variable **n** toma el siguiente valor de la matriz y continúa en el paso 2, la línea siguiente a la palabra clave **`For`**.
   n = Array(next)
5. Si **es el último**, el programa continúa en la línea siguiente a la palabra clave **`Next`**.

Ejemplo For - Each :

```
Private Numbers() As Int
Private Sum As Int

Numbers = Array As Int(1, 3, 5 , 2, 9)

Sum = 0
For Each n As Int In Numbers
   Sum = Sum + n
Next
```

El mismo ejemplo pero con un bucle For - Next:

```
Private Numbers() As Int
Private Sum As Int
Private i As Int

Numbers = Array As Int(1, 3, 5 , 2, 9)

Sum = 0
For i = 0 To Numbers.Length - 1
   Sum = Sum + Numbers(i)
Next
```

Este ejemplo muestra el poder del bucle For - Each:

```
For Each lbl As Label In Activity
   lbl.TextSize = 20
Next
```

El mismo ejemplo con un bucle For - Next:

```
For i = 0 To Activity.NumberOfViews - 1
   Private v As View
   v = Activity.GetView(i)
   If v Is Label Then
      Private lbl As Label
      lbl = v
      lbl.TextSize = 20
   End If
Next
```

### 3.4.3 Bucle Do - Loop

Existen varias configuraciones:

```
Do While test
   ' código
Loop
```

**test** es cualquier expresión
Ejecuta el **código** mientras **test** es **Verdadero**

```
Do Until test
   ' código
Loop
```

**test** es cualquier expresión
Ejecuta el **código** hasta que **test** es **Verdadero**

El bucle **Do While -Loop** funciona de la siguiente manera :

1. Al principio, **test** es evaluada.
2. si es **Verdadera**, entonces ejecuta **código**
3. Si es **Falsa** continua en la línea siguiente a la palabra clave **Loop**.

El bucle **Do Until -Loop** funciona de la siguiente manera:

1. Al principio, **test** es evaluada.
2. Si es **Falsa**, entonces ejecuta **código**
3. Si es **Verdadera** continúa en la línea siguiente a la palabra clave **Loop**.

Es posible salir de un bucle Do-Loop con la palabra clave Exit.

```
Do While test
   ' código
   If a = 0 Then Exit
   ' código
Loop
```

Si **a = 0** entonces sale del bucle

Ejemplos:

Bucle Do Until:

```
Private i, n As Int

i = 0
Do Until i = 10
   ' código
   i = i + 1
Loop
```

Bucle Do While :

```
Private i, n As Int

i = 0
Do While i < 10
   ' código
   i = i + 1
Loop
```

Leer un archivo de texto y llenar una lista :

```
Private lstText As List
Private line As String
Private tr As TextReader

tr.Initialize(File.OpenInput(File.DirInternal, "test.txt"))
lstText.Initialize
line = tr.ReadLine
Do While line <> Null
   lstText.Add(line)
   line = tr.ReadLine
Loop

tr.Close
```

**Nota :** Diferencia entre:

| B4x | VB |
|---|---|
| `Exit` | `Exit Loop` |

En VB el tipo de bucle se especifica después de la palabra clave **`Exit`**.

VB también acepta los siguientes bucles, que no son soportados en B4x.

```
Do                          Do
  ' código                    ' código
Loop While test             Loop Until test
```

## 3.5 Subs

Una Subrutina ("Sub") es una pieza de código. Puede ser de cualquier longitud, tiene un nombre distintivo y un alcance definido (referido al ámbito de las variables explicado anteriormente). En código B4x , una subrutina es se llama "Sub", y es equivalente a los procedimientos, funciones, métodos y subs en otros lenguajes de programación. Las líneas de código dentro de una Sub son ejecutadas desde la primera a la última, como se describe en el capítulo flujo de código.
No es recomendable tener Subs con una gran cantidad de código, son más difíciles de entender.

### 3.5.1 Declaración

Una Sub es declarada de la siguiente manera:

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

Se inicia con la palabra clave **`Sub`**, seguido por el nombre de la subrutina, seguido por una lista de parámetros, seguido del tipo de variable que devuelve y termina con las palabras claves **`End Sub`**.
Las Subs son siempre declaradas en el nivel superior del módulo, no se puede anidar una Sub dentro de otra.

### 3.5.2 Llamadas

Cuando se desea ejecutar las líneas de código dentro de una Sub, simplemente se escribe el nombre de la Sub.

Por ejemplo:

```
   Interest = CalcInterest(1234, 5.2)
```

| | |
|---|---|
| Interest | Toma el valor retornado por la Sub. |
| `CalcInterest` | Nombre de la Sub. |
| `1235` | Valor del parámetro 'Rate' pasado a la Sub. |
| `5.25` | Valor del parámetro 'Rate' pasado a la Sub. |

### 3.5.3 Llamar a una Sub desde otro módulo

A una subrutina declarada en un módulo de código se puede acceder desde cualquier otro módulo, pero en este caso, se debe añadir el nombre del módulo donde se declaró la Sub como prefijo.

Ejemplo: Si la rutina `CalcInterest` se declara en el módulo `MyModule` entonces la llamada a la rutina debería ser:

```
   Interest = MyModule.CalcInterest(1234, 5.2)
```

En lugar de:

```
   Interest = CalcInterest(1234, 5.2)
```

### 3.5.4 Nombres

Básicamente, para poner un nombre a una Sub puedes usar las mismas reglas que para poner nombres a las variables. Se recomienda poner un nombre significativo, como **`CalcInterest`** en el ejemplo, así puedes saber qué hace el código viendo el nombre de la Sub.
No hay límite en el número de Subs que puedes añadir al programa, pero no se permite tener dos Subs con el mismo nombre en el mismo módulo.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

### 3.5.5 Parámetros

A una Sub se le pueden pasar parámetros. La lista de parámetros debe ir entre paréntesis  después del nombre de la Sub.
El tipo de parámetro debe ser declarado directamente en la lista.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

en B4x, los parámetros se pasan por valor y no por referencia.

### 3.5.6 Valor devuelto

Una sub puede devolver un valor, este puede ser cualquier objeto.
La devolución de ese valor se hace mediante la palabra clave Return.
El tipo de valor devuelto se añade en la declaración de la Sub después de la lista de parámetros.

```
Sub CalcInterest(Capital As Double, Rate As Double) As Double
   Return Capital * Rate / 100
End Sub
```

## 3.6 Subs resumibles

La reanudación de Subs es una nueva característica añadida en B4A v7.00 / B4i v4.00 / B4J v5.50.
Simplifica enormemente el manejo de las tareas asíncronas.
(Esta característica es una variante de coroutines sin pila.)

Encontrarás más ejemplos en el foro.

La característica especial de la reanudación de Subs es que pueden ser pausadas sin necesidad de detener la ejecución del hilo y más tarde ser reanudada.
El programa no espera a la reanudación de la Sub. Otros eventos se activarán como de costumbre.

Cualquier Sub con una o más llamadas a 'Sleep' o 'Wait For' es una Sub resumible.
El IDE muestra este indicador junto a la declaración de la Sub:

```
Private Sub CountDown(Start As Int)
   For i = Start To 0 Step -1
      Label1.Text = i
      Sleep(1000)
   Next
End Sub
```

### 3.6.1 Sleep

Pausa la ejecución de la Sub actual y se reanuda después del tiempo especificado.

**Sleep** (Milliseconds As Int) Milliseconds, tiempo de espera en milisegundos.
Ejemplo:
Sleep(1000)

Usar Sleep es simple:

```
Log(1)
Sleep(1000)
Log(2)
```

La Sub se pausa durante 1000 milisegundos y luego se reanuda.

Puedes usar Sleep(0) para la más breve pausa. Esto se puede usar para permitir que la interfaz de usuario se actualice. Es una buena alternativa para DoEvents (que no existe en B4J ni B4i y debe ser evitado en B4A).

```
Sub VeryBusySub
   For i = 1 To 10000000
      'hacer algo
      If i Mod 1000 = 0 Then Sleep(0) 'Permite a la interfaz de usuario actualizarse
cada 1000 iteraciones.
   Next
   Log("Terminado!")
End Sub
```

### 3.6.2 Wait For

B4X es un lenguaje de programación por eventos. Tareas asíncronas se ejecutan de fondo y activan un evento cuando se completan.
Con la nueva palabra clave Wait For puedes manejar ese evento desde la actual Sub.

Por ejemplo, este código esperará por el evento Ready de GoogleMap (ejemplo B4J):

```
Sub AppStart (Form1 As Form, Args() As String)
   MainForm = Form1
   MainForm.RootPane.LoadLayout("1") 'Carga el archivo de diseño de la interfaz.

   gmap.Initialize("gmap")
   Pane1.AddNode(gmap.AsPane, 0, 0, Pane1.Width, Pane1.Height)
   MainForm.Show
   Wait For gmap_Ready '<----------------
   gmap.AddMarker(10, 10, "Marker")
End Sub
```

Un ejemplo un poco más complicado con FTP:
Lista todos los archivos de un directorio remoto y después descarga todos los archivos:

```
Sub DownloadFolder (ServerFolder As String)
  FTP.List(ServerFolder)
  Wait For FTP_ListCompleted (ServerPath As String, Success As Boolean, Folders() As
    FTPEntry, Files() As FTPEntry) '<----
  If Success Then
    For Each f As FTPEntry In Files
      FTP.DownloadFile(ServerPath & f.Name, False, File.DirApp, f.Name)
      Wait For FTP_DownloadCompleted (ServerPath2 As String, Success As Boolean) '<----
      Log($"File ${ServerPath2} downloaded. Success = ${Success}"$)
     Next
  End If
  Log("Terminado")
End Sub
```

Cuando la palabra clave Wait For es llamada, la Sub se pausa y el manejador interno de eventos cuida de reanudarla cuando se produce el evento. Si el evento nunca se produce, la Sub nunca se reanudará. El programa todavía seguirá ejecutándose y respondiendo a otros eventos.
Si más tarde se vuelve a usar Wait For con el mismo evento, entonces la nueva instancia de la Sub reemplazará a la anterior.

Supongamos que queremos crear una Sub que descarga una imagen y la carga en una vista ImageView:

```
'Mal ejemplo. No usar.
Sub DownloadImage(Link As String, iv As ImageView)
   Dim job As HttpJob
   job.Initialize("", Me) 'Nota que el parámetro de nombre, ya no es necesario.
   job.Download(Link)
   Wait For JobDone(job As HttpJob)
   If job.Success Then
      iv.SetImage (job.GetBitmap) 'reemplazar con iv.Bitmap = job.GetBitmap en B4A / B4i
   End If
   job.Release
End Sub
```

Funcionará correctamente si la llamamos una sola vez (más correctamente, si no la llamamos de nuevo antes de que la llamada anterior se complete).
Si usamos este código:

```
   DownloadImage("https://www.b4x.com/images3/android.png", ImageView1)
   DownloadImage("https://www.b4x.com/images3/apple.png", ImageView2)
```

Solo cargaremos la segunda imagen, porque la segunda llamada a Wait For JobDone sobrescribirá a la anterior.
Esto nos lleva a la segunda variante de Wait For.
Para resolver este problema, Wait For puede distinguir entre eventos basado en el remitente del evento.
Esto se hace con un parámetro opcional:

*Wait For* (<sender>) <event signature>

Ejemplo:

```
'Buen ejemplo. Usar.
Sub DownloadImage(Link As String, iv As ImageView)
   Dim job As HttpJob
   job.Initialize("", Me) 'Nota que el parámetro de nombre, ya no es necesario.
   job.Download(Link)
   Wait For (job) JobDone(job As HttpJob)
   If job.Success Then
      iv.SetImage (job.GetBitmap) 'reemplazar con iv.Bitmap = job.GetBitmap en B4A / B4i
   End If
   job.Release
End Sub
```

Con el código anterior, cada instancia de la Sub resumible va a esperar por un evento diferente y no se verá afectada por las otras instancias.

### 3.6.3 Flujo de código

```
Sub S1
   Log("S1: A")
   S2
   Log("S1: B")
End Sub

Sub S2
   Log("S2: A")
   Sleep(0)
   Log("S2: B")
End Sub
```

El resultado es:
S1: A
S2: A
S1: B
S2: B

Siempre que las palabras claves Sleep o Wait For son usadas, la actual Sub se pausa. Esto es equivalente a llamar a Return.

### 3.6.4 Esperar la terminación de una sub para reanudar otra

Cuando una Sub resumible llama a una segunda Sub resumible, el código en la primera Sub continuara después de la primera llamada a Sleep o Wait For (en la segunda Sub).

Si quieres esperar a que la segunda sub se complete, puedes provocar un evento en la segunda Sub y esperar por ese evento en la primera Sub:

```
Sub FirstSub
   Log("FirstSub comenzada")
   SecondSub
   Wait For SecondSub_Complete
   Log("FirstSub completada")
End Sub

Sub SecondSub
   Log("SecondSub comenzada")
   Sleep(1000)
   Log("SecondSub completada")
   CallSubDelayed(Me, "SecondSub_Complete")
End Sub
```

Registros:
FirstSub comenzada
SecondSub comenzada
SecondSub completada
FirstSub completada

Notas:
- Es más seguro utilizar CallSubDelayed que CallSub. CallSub fallará si la segunda Sub nunca entra en pausa (por ejemplo si Sleep solo es llamada usando alguna condición).
- Aquí se asume que FirstSub nunca va a ser llamada de nuevo hasta que se haya completado.

### 3.6.5 DoEvents

A partir de B4A v7.0, aparecerá el siguiente aviso para las llamadas a DoEvents:
*DoEvents está en desuso. Esto puede llevar a problemas de estabilidad. Usa Sleep(0) en su lugar (si es realmente necesario).*

El propósito de Doevents era permitir que la interfaz de usuario se actualizara mientras el hilo principal estaba ocupado. DoEvents, que comparte la misma implementación que los cuadros de diálogo modales, es una implementación de bajo nivel. Accede a la cola de mensajes y ejecuta algunos de los mensajes en espera.

Según ha ido evolucionando Android, el manejo de la cola de mensajes se ha vuelto más sofisticado y frágil.
Las razones para dejar de usar DoEvents son:

1. Es una fuente importante de problemas de inestabilidad. Puede conducir a una difícil depuración o a mensajes ANR (aplicación no responde). Ten en cuenta que esto también es cierto para los cuadros de diálogo modales (como los Msgbox y los InputList).
2. Hay mejores maneras de mantener el hilo principal libre. Por ejemplo usando métodos asíncronos SQL en lugar de los métodos síncronos.
3. No hace lo que muchos de los desarrolladores esperan que haga Ya que solo se encarga de los mensajes relacionados con la interfaz de usuario, la mayoría de los eventos no pueden ser manejados desde una llamada a DoEvents.
4. Ahora es posible usar Sleep para pausar la actual Sub y reanudarla después de que se procesen los mensajes en espera. La implementación de Sleep es completamente diferente a DoEvents. No bloquea el hilo. En lugar de eso, lo libera preservando el estado de la Sub.
A diferencia de DoEvents que solo procesa los mensajes relacionados con el interfaz de usuario, con Sleep se procesan todos los mensajes y otros eventos serán activados.
(Nota que usar Wait For para esperar por un evento es mejor que usar Sleep en un bucle.)

Con todo esto, DoEvents todavía se puede usar y las aplicaciones existentes funcionarán exactamente igual que antes.

### 3.6.6 Cuadros de Diálogo

Diálogos modales = diálogos que bloquean el hilo principal hasta que el diálogo se cierra.

Como se ha indicado anteriormente, los cuadros de diálogo modales comparten la misma implementación que DoEvents. Por lo tanto, es recomendable cambiar a los nuevos cuadros de diálogo asíncronos. El uso de Wait For es un cambio realmente sencillo:

En lugar de:

```
Dim res As Int = Msgbox2("Borrar?", "Título", "Si", "Cancelar", "No", Null)
If res = DialogResponse.POSITIVE Then
   '...
End If
```

Debes utilizar :

```
Msgbox2Async("Borrar?", "Título", "Si", "Cancelar", "No", Null, False)
Wait For Msgbox_Result (Result As Int)
If Result = DialogResponse.POSITIVE Then
       '...
End If
```

*Wait For* no bloquea el hilo principal. En su lugar, guarda el estado actual de la Sub y lo libera. El código se reanudará cuando el usuario haga clic en alguno de los botones del cuadro de diálogo. Los otros nuevos métodos similares son: MsgboxAsync, InputListAsync and InputMapAsync.

Con la excepción de MsgboxAsync, los nuevos métodos también añaden un nuevo parámetro *cancelable*. Si es verdadero, entonces el cuadro de diálogo puede ser cerrado al hacer clic en la tecla volver o fuera del cuadro de diálogo. Este es el comportamiento por defecto de los antiguos métodos.

Como otros códigos se pueden ejecutar mientras un cuadro de diálogo es visible, es posible que varios cuadros de diálogo aparezcan al mismo tiempo.
Si este es el caso de tu aplicación, debes configurar el filtro del remitente en el parámetro de la llamada a Wait For:

```
Dim sf As Object = Msgbox2Async("Borrar?", "Título", "Si", "Cancelar", "No", Null,
False)
Wait For (sf) Msgbox_Result (Result As Int)
If Result = DialogResponse.POSITIVE Then
   '...
End If
```

Esto permite manejar correctamente los eventos activados por cada resultado devuelto por los distintos cuadros de diálogo.

## 3.6.7 SQL con Wait For

La nueva característica de las Sub resumibles, hace que sea más sencillo trabajar con grandes conjuntos de datos con un mínimo efecto en la capacidad de respuesta del programa.

El nuevo estándar en la forma de insertar los datos es:

```
For i = 1 To 1000
   SQL1.AddNonQueryToBatch("INSERT INTO table1 VALUES (?)", Array(Rnd(0, 100000)))
Next
Dim SenderFilter As Object = SQL1.ExecNonQueryBatch("SQL")
Wait For (SenderFilter) SQL_NonQueryComplete (Success As Boolean)
Log("NonQuery: " & Success)
```

Los pasos son:
- Llamar a AddNonQueryToBatch por cada comando que debe ser emitido.
- Ejecutar los comandos con ExecNonQueryBatch. Este es un método asíncrono. Los comandos se ejecutarán en segundo plano y el evento NonQueryComplete se disparará cuando se haya completado.
- Esta llamada devuelve un objeto que se puede utilizar como parámetro del remitente . Esto es importante ya que podría haber varios comandos por lote ejecutándose en segundo plano. Con el parámetro del remitente el evento será capturado por la correcta llamada de Wait For en todos los casos.
- Ten en cuenta que SQL1.ExecNonQueryBatch comienza y termina una transacción internamente.

### 3.6.7.1 Consultas

En la mayoría de los casos las consultas serán rápidas y por lo tanto deben ser hechas mediante la forma síncrona con SQL1.ExecQuery2. Sin embargo, si hay una consulta lenta, entonces debes cambiar a SQL1.ExecQueryAsync:

```
Dim SenderFilter As Object = SQL1.ExecQueryAsync("SQL", "SELECT * FROM table1", Null)
Wait For (SenderFilter) SQL_QueryComplete (Success As Boolean, rs As ResultSet)
If Success Then
   Do While rs.NextRow
      Log(rs.GetInt2(0))
   Loop
   rs.Close
Else
   Log(LastException)
End If
```

Como en el caso anterior, el método ExecQueryAsync devuelve un objeto que es usado como parámetro para filtrar el remitente.

Consejos:
1. El tipo ResultSet en B4A extiende el tipo Cursor. Puedes cambiar el tipo Cursor si lo prefieres. La ventaja de usar ResultSet es que es compatible con B4J y B4i.
2. Si el número de filas devueltas por la consulta es grande, entonces el bucle Do While será lento en modo 'debug'. puedes hacer que sea más rápido poniéndolo en otra Sub y limpiando el proyecto (Ctrl + P):

```
   Wait For (SenderFilter) SQL_QueryComplete (Success As Boolean, rs As ResultSet)
   If Success Then
      WorkWithResultSet(rs)
   Else
      Log(LastException)
   End If
End Sub

Private Sub WorkWithResultSet(rs As ResultSet)
   Do While rs.NextRow
      Log(rs.GetInt2(0))
   Loop
   rs.Close
End Sub
```

Esto está relacionado con una optimización del depurador que se encuentra deshabilitado en las Sub resumibles.
El rendimiento de ambas soluciones, será el mismo en modo 'release'.

### 3.6.7.2 B4J

- Requiere jSQL v1.50+ (https://www.b4x.com/android/forum/threads/updates-to-internal-libaries.48274/#post-503552).
- Se recomienda establecer el modo de diario a WAL: https://www.b4x.com/android/forum/t...ent-access-to-sqlite-databases.39904/#content

### 3.6.8 Notas y consejos

- Las Subs resumibles no pueden devolver un valor.
- La sobrecarga de rendimiento de las Subs resumibles en modo 'release' debería ser insignificante en la mayoría de los casos. La sobrecarga puede ser mayor en modo 'debug' (Si esto se convierte en un problema mover las partes lentas de código a otra Sub y llamarla desde la Sub resumible.)
- Los manipuladores de eventos Wait For preceden a los manipuladores de eventos regulares.
- Las Subs resumibles no crean hilos adicionales. El código es ejecutado en el hilo principal o en el manipulador de hilo en soluciones de servidor.

## 3.7 Eventos

En la programación orientada a objetos, tenemos objetos que pueden reaccionar a diferentes acciones del usuario llamados eventos.
El número y el tipo de eventos de un objeto pueden aumentar dependiendo del tipo de objeto.

### 3.7.1 B4A

Los objetos de interfaz de usuario son llamados 'Views' en Android.

Resumen de los eventos para las diferentes views:

| | **Eventos** | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Views** | Click | LongClick | Touch | Down | Up | KeyPress | KeyUp | ItemClick | ItemLongClick | CheckedChange | EnterPressed | FocusChanged | TextChanged | ScrollChanged | ValueChanged | TabChanged | OverrideUrl | PageFinished |
| Activity | ■ | ■ | ■ | | | ■ | ■ | | | | | | | | | | | |
| Button | ■ | ■ | | ■ | ■ | | | | | | | | | | | | | |
| CheckBox | | | | | | | | | | ■ | | | | | | | | |
| EditText | | | | | | | | | | | ■ | ■ | ■ | | | | | |
| HorizontalScrollView | | | | | | | | | | | | | | ■ | | | | |
| ImageView | ■ | ■ | | | | | | | | | | | | | | | | |
| Label | ■ | ■ | | | | | | | | | | | | | | | | |
| ListView | | | | | | | | ■ | ■ | | | | | | | | | |
| Panel | ■ | ■ | ■ | | | | | | | | | | | | | | | |
| RadioButton | | | | | | | | | | ■ | | | | | | | | |
| ScrollView | | | | | | | | | | | | | | ■ | | | | |
| SeekBar | | | | | | | | | | | | | | | ■ | | | |
| Spinner | | | | | | | | ■ | | | | | | | | | | |
| TabHost | ■ | ■ | | | | | | | | | | | | | | ■ | | |
| ToggleButton | | | | | | | | | | ■ | | | | | | | | |
| WebView | | | | | | | | | | | | | | | | | ■ | ■ |

Los eventos más comunes son:

- **Click** El evento se desencadena cuando el usuario hace clic en la 'view'.
  Ejemplo:
  ```
  Sub Button1_Click
    'Tu código
  End Sub
  ```

- **LongClick** El evento se desencadena cuando el usuario hace clic en la 'view' y mantiene pulsado durante un momento.
  Ejemplo:
  ```
  Sub Button1_LongClick
    'Tu código
  End Sub
  ```

- **Touch** `(Action As Int, X As Float, Y As Float)`
  El evento se desencadena cuando el usuario toca la pantalla.

  Se manejan tres acciones diferentes:
  - Activity.Action_DOWN, El usuario toca la pantalla.
  - Activity.Action_MOVE, El usuario mueve el dedo sin dejar de tocar la pantalla.
  - Activity.Action_UP, El usuario deja de tocar la pantalla.

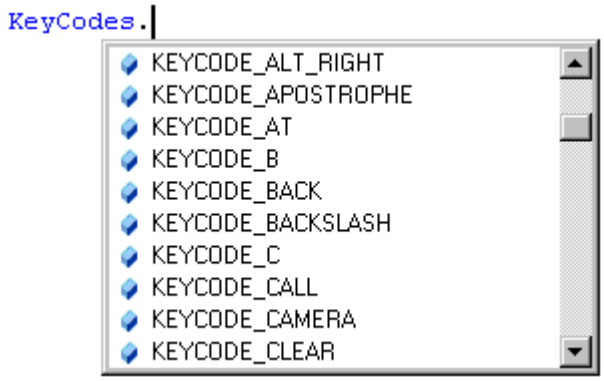
  Las coordenadas X e Y, de la posición del dedo en la pantalla, son pasadas.

  Ejemplo:
  ```
  Sub Activity_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case Activity.ACTION_DOWN
      'Tu código para la acción DOWN
    Case Activity.ACTION_MOVE
      'Tu código para la acción MOVE
    Case Activity.ACTION_UP
      'tu código para la acción UP
    End Select
  End Sub
  ```

- **CheckChanged** `(Checked As Boolean)`
  El evento se desencadena cuando el usuario hace clic en un CheckBox o un RadioButton
  Checked es igual a Verdadero si la view esta activada o Falso si no está activada.

  Ejemplo:
  ```
  Sub CheckBox1_CheckedChange(Checked As Boolean)
    If Checked = True Then
      'Tu código si activado
    Else
      'Tu código si no activado
    End If
  End Sub
  ```

- **KeyPress** (KeyCode As Int) As Boolean
  El evento se desencadena cuando el usuario presiona una tecla física o virtual.
  KeyCode es el código de la tecla que se presiona, se puede conseguir con la palabra clave 'KeyCodes'.




El evento puede devolver:
- Verdadero, el evento es 'consumido', considerado por el sistema operativo como ya ejecutado y no se realizan más acciones.
- Falso, el evento no es consumido y se transmite al sistema para futuras acciones.

Ejemplo:

```
Sub Activity_KeyPress(KeyCode As Int) As Boolean
  Private Answ As Int
  Private Txt As String

  If KeyCode = KeyCodes.KEYCODE_BACK Then    'Comprueba si el código es el de la
tecla 'Volver'
    Txt = "¿Realmente deseas salir del programa?"
    Answ = Msgbox2(Txt,"A T E N C I Ó N","Si","","No",Null)' MessageBox
    If Answ = DialogResponse.POSITIVE Then  ' Si el valor devuelto es 'Si'
entonces
      Return False     ' Return = False  el evento no será consumido
    Else               '                   salimos del programa
      Return True      ' Return = True   el evento será consumido para evitar
    End If             '                  salir del programa
  End If
End Sub
```

### 3.7.2 B4i

Los objetos del interfaz de usuario son llamados 'Views' en iOS.

Resumen de los eventos para las diferentes 'Views':

| | **Eventos** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Views** | Click | LongClick | BeginEdit | EndEdit | EnterPressed | TextChanged | Touch | Resize | ScrollChanged | ValueChanged | ItemSelected | IndexChanged | OverrideUrl | PageFinished |
| Button | ■ | ■ | | | | | | | | | | | | |
| TextField | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | |
| TextView | ■ | ■ | | | | | | | | | | | | |
| ImageView | ■ | ■ | | | | | | | | | | | | |
| Label | ■ | ■ | | | | | | | | | | | | |
| Panel | ■ | ■ | | | | | ■ | ■ | | | | | | |
| ScrollView | ■ | ■ | | | | | | | ■ | | | | | |
| Slider | ■ | ■ | | | | | | | | ■ | | | | |
| Picker | ■ | ■ | | | | | | | | | ■ | | | |
| Stepper | ■ | ■ | | | | | | | | ■ | | | | |
| Switch | ■ | ■ | | | | | | | | ■ | | | | |
| SegmentedControl | ■ | ■ | | | | | | | | | | ■ | | |
| Slider | ■ | ■ | | | | | | | | ■ | | | | |
| Stepper | ■ | ■ | | | | | | | | ■ | | | | |
| WebView | ■ | ■ | | | | | | | | | | | ■ | ■ |

Los eventos más comunes son:

- **Click** El evento se desencadena cuando el usuario hace clic en la 'view'.
  Ejemplo:
  ```
  Private Sub Button1_Click
    'Tu código
  End Sub
  ```

- **LongClick** El evento se desencadena cuando el usuario hace clic en la 'view' y mantiene pulsado durante un momento.
  Ejemplo:
  ```
  Private Sub Button1_LongClick
    'Tu código
  End Sub
  ```

- **Touch** `(Action As Int, X As Float, Y As Float)`
  El evento se desencadena cuando el usuario toca un Panel en la pantalla.

  Se manejan tres acciones diferentes:
  - Panel.ACTION_DOWN, El usuario toca la pantalla.
  - Panel.ACTION_MOVE, El usuario mueve el dedo sin dejar de tocar la pantalla.
  - Panel.ACTION_UP, El usuario deja de tocar la pantalla.

  Las coordenadas X e Y de la posición del dedo son pasados en Puntos no en Píxeles.

  Ejemplo:
  ```
  Private Sub Panel_Touch (Action As Int, X As Float, Y As Float)
    Select Action
    Case Panel.ACTION_DOWN
      'Tu código para la acción DOWN
    Case Panel.ACTION_MOVE
      'Tu código para la acción MOVE
    Case Panel.ACTION_UP
      'tu código para la acción UP
    End Select
  End Sub
  ```

## 3.7.3 B4J

Los objetos de la interfaz de usuario se llaman 'Nodes' en Java.

Resumen de los eventos para los diferentes 'Nodes':

| | | | **Eventos** | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Nodes** | Action | FocusChanged | MouseClicked | MouseDragged | MouseMoved | MousePressed | MouseReleased | Resize | CheckedChange | SelectedIndexChanged | ValueChange | SelectedChange | H / VScrollChanged | TabChanged | TextChanged | PageFinished |
| Button | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | |
| Canvas | | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | | |
| CheckBox | | | | | | | | | ■ | | | | | | | |
| ChoiceBox | | | | | | | | | | ■ | | | | | | |
| ComboBox | | | | | | | | | | ■ | ■ | | | | | |
| ImageView | | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | | |
| Label | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | |
| ListView | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | ■ | | | | | | |
| Pane | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | |
| RadioButton | | | | | | | | | | | | ■ | | | | |
| ScrollPane | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | | | ■ | | | |
| Slider | | | | | | | | | | | ■ | | | | | |
| Spinner | | | | | | | | | | | ■ | | | | | |
| TabPane | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | ■ | | |
| TextArea | | ■ | ■ | | | | | | | | | | | | ■ | |
| TextField | ■ | ■ | ■ | | | | | | | | | | | | ■ | |
| ToggleButton | | | | | | | | | | | | ■ | | | | |
| WebView | | ■ | ■ | ■ | ■ | ■ | ■ | | | | | | | | | ■ |

Los eventos más comunes son:

- **Action** El evento se desencadena cuando el usuario hace clic en un 'node' (Button o TextField).
  Ejemplo:
  ```
  Private Sub Button1_Action
    'Tu código
  End Sub
  ```

- **FocusChanged** (`HasFocus As Boolean`) El evento se desencadena cuando el 'node' obtiene o pierde el foco.
  Ejemplo:
  ```
  Private Sub Button1_FocusChanged (HasFocus As Boolean)
    'Tu código
  End Sub
  ```

- **MouseClicked** (`EventData As MouseEvent`)
  El evento se desencadena cuando el usuario hace clic en un 'node'.
  Ejemplo:
  ```
  Private Sub Pane1_MouseClicked (EventData As MouseEvent)
    'Tu código
  End Sub
  ```

- **MouseDragged** (`EventData As MouseEvent`)
  El evento se desencadena cuando el usuario arrastra el cursor sobre el 'node' (mueve el ratón con un botón presionado).
  Similar a ACTION_MOVE del evento Touch en B4A.
- Ejemplo:
  ```
  Private Sub Pane1_MouseDragged (EventData As MouseEvent)
    'Tu código
  End Sub
  ```

- **MouseMoved** (`EventData As MouseEvent`)
  El evento se desencadena cuando el usuario muevo el cursor sobre el 'node' (sin presionar ningún botón del ratón).
  Ejemplo:
  ```
  Private Sub Pane1_MouseMoved (EventData As MouseEvent)
    'Tu código
  End Sub
  ```

- **MousePressed** (`EventData As MouseEvent`)
  El evento se desencadena cuando el usuario pulsa sobre un 'node'.
  Similar a ACTION_DOWN del evento Touch en B4A.
  Ejemplo:
  ```
  Private Sub Pane1_MousePressed (EventData As MouseEvent)
    'Tu código
  End Sub
  ```

- **MouseReleased** (`EventData As MouseEvent`)
  El evento se desencadena cuando el usuario libera el 'node'.
  Similar a ACTION_UP del evento Touch en B4A.
  Ejemplo:
  ```
  Private Sub Pane1_MouseReleased (EventData As MouseEvent)
    'Tu código
  End Sub
  ```

- **MouseEvent**
  Datos devueltos por los eventos de Ratón:

  - **ClickCount** Devuelve el número de clics asociados con este evento.
  - **Consume** Consume el evento actual y evita que sea manipulado por los 'nodes' padres.
  - **MiddleButtonDown** Devuelve Verdadero si el botón central está actualmente pulsado.
  - **MiddleButtonPressed** Devuelve Verdadero si el botón central fue el responsable de desencadenar el actual evento clic.
  - **PrimaryButtonDown** Devuelve Verdadero si el botón primario está actualmente pulsado.
  - **PrimaryButtonPressed** Devuelve Verdadero si el botón principal fue el responsable de desencadenar el actual evento clic.
  - **SecondaryButtonDown** Devuelve Verdadero si el botón secundario está actualmente pulsado.
  - **SecondaryButtonPressed** Devuelve Verdadero si el botón secundario fue el responsable de desencadenar el actual evento clic.
  - **X** Devuelve la coordenada X en relación a los límites del 'node'.
  - **Y** Devuelve la coordenada Y en relación a los límites del 'node'.

### 3.7.4 B4R

En B4R, los objetos 'Pin' y 'Timer' son los únicos que generan eventos:

- Pin
  **StateChanged** (State As Boolean) El evento se desencadena cuando el pin cambia su estado.

  Ejemplo:
  ```
  Sub Pin1_StateChanged(State As Boolean)
    'Tu código
  End Sub
  ```

- Timer
  **Tick** El evento se desencadena cada intervalo de tiempo dado

  Ejemplo:
  ```
  Sub Timer1_Tick
    'Tu código
  End Sub
  ```

## 3.7.5 Resumen del interfaz de usuario

Los objetos ‘estándar’ de la interfaz de usuario.
Esto muestra la diferencia entre los tres sistemas operativos.
Algunas 'views' / 'nodes' que no existen como objetos estándar, pueden existir como 'CustomViews' en otros sistemas operativos. Debes buscar en los foros.

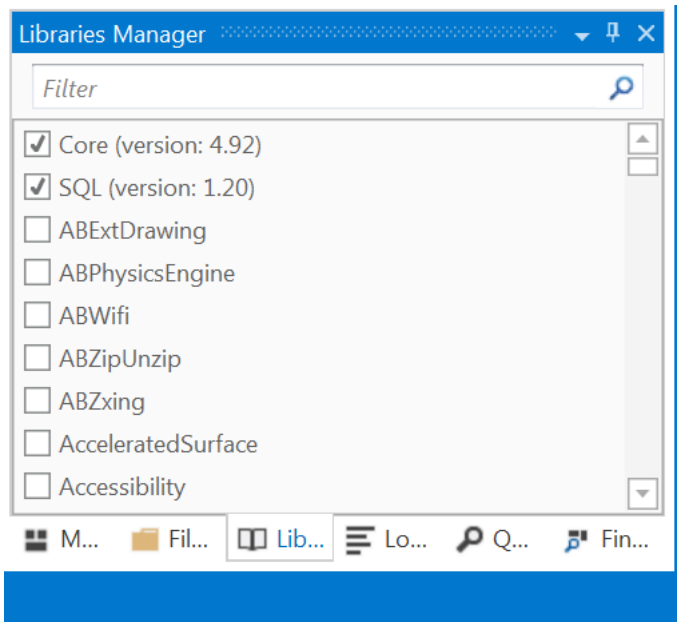
| View / node | B4A | B4i | B4J |
|---|---|---|---|
| Activity | ■ | | |
| Button | ■ | ■ | ■ |
| CheckBox | ■ | | ■ |
| EditText | ■ | | |
| HorizontalScrollView | ■ | | |
| ImageView | ■ | ■ | ■ |
| Label | ■ | ■ | ■ |
| ListView | ■ | | ■ |
| Panel | ■ | ■ | |
| RadioButton | ■ | | ■ |
| ScrollView | ■ | | |
| SeekBar | ■ | | |
| Spinner | ■ | | ■ |
| TabHost | ■ | | |
| ToggleButton | ■ | | ■ |
| WebView | ■ | ■ | ■ |
| TextField | | ■ | ■ |
| TextView | | ■ | |
| ScrollView diferente del de B4A 2D | | ■ | |
| Slider | | ■ | ■ |
| Picker | | ■ | |
| Stepper | | ■ | |
| Switch | | ■ | |
| SegmentedControl | | ■ | |
| Canvas un ‘node’ en si mismo | | | ■ |
| ChoiceBox | | | ■ |
| ComboBox | | | ■ |
| Pane similar a ‘Panel’ en B4A y B4i | | | ■ |
| ScrollPane similar a ‘ScrollView’ | | | ■ |
| TabPane | | | ■ |
| TextArea | | | ■ |

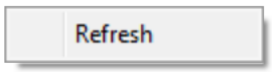
## 3.8 Librerías

Las librerías añaden más objetos y funcionalidades a B4x.
Algunas de estas librerías se distribuyen con los productosB4x y son parte del sistema de desarrollo estándar.
Otras, a menudo desarrolladas por los usuarios, se pueden descargar (sólo usuarios registrados) para añadir funcionalidades complementarias a los entornos de desarrollo B4x.

Cuando necesites una librería, tienes que:

- Comprobar en la pestaña 'Libraries Manager', si ya tienes la librería.
- Para bibliotecas adicionales, comprobar si es la versión más reciente.
  Puedes comprobar la versión en la página de la documentación
  B4A, B4i, B4J, B4R
  Para encontrar una librería haz una consulta como
  http://www.b4x.com/search?query=betterdialogs+library
  en tu navegador.
- Si **ya la tienes**, entonces selecciona la librería en la lista para activarla.




- Si **no la tienes**, descárgala, descomprímela y copia los archivos <NombreLibrería>.jar y <NombreLibrería>.xml en la carpeta de 'additional libraries'.
- Botón secundario en la lista de la pestaña de 'Libraries Manager', clic en
  
  
  y selecciona la librería en la lista para activarla.

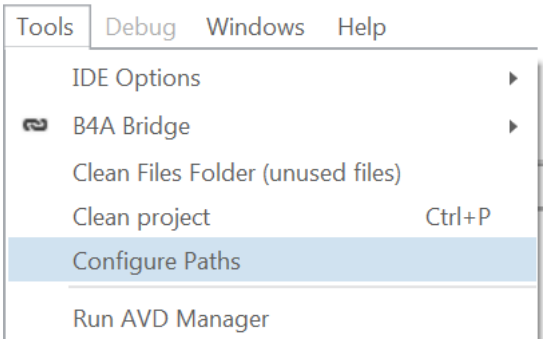
### 3.8.1 Librerías estándar

Las librerías estándar de B4x se guardan en la subcarpeta 'libraries' dentro de la carpeta de instalación del programa B4x.
Normalmente en:
C:\Program Files\Anywhere Software\B4A\Libraries
C:\Program Files\Anywhere Software\B4i\Libraries
C:\Program Files\Anywhere Software\B4J\Libraries
C:\Program Files\Anywhere Software\B4R\Libraries

## 3.8.2 Carpeta de librerías adicionales

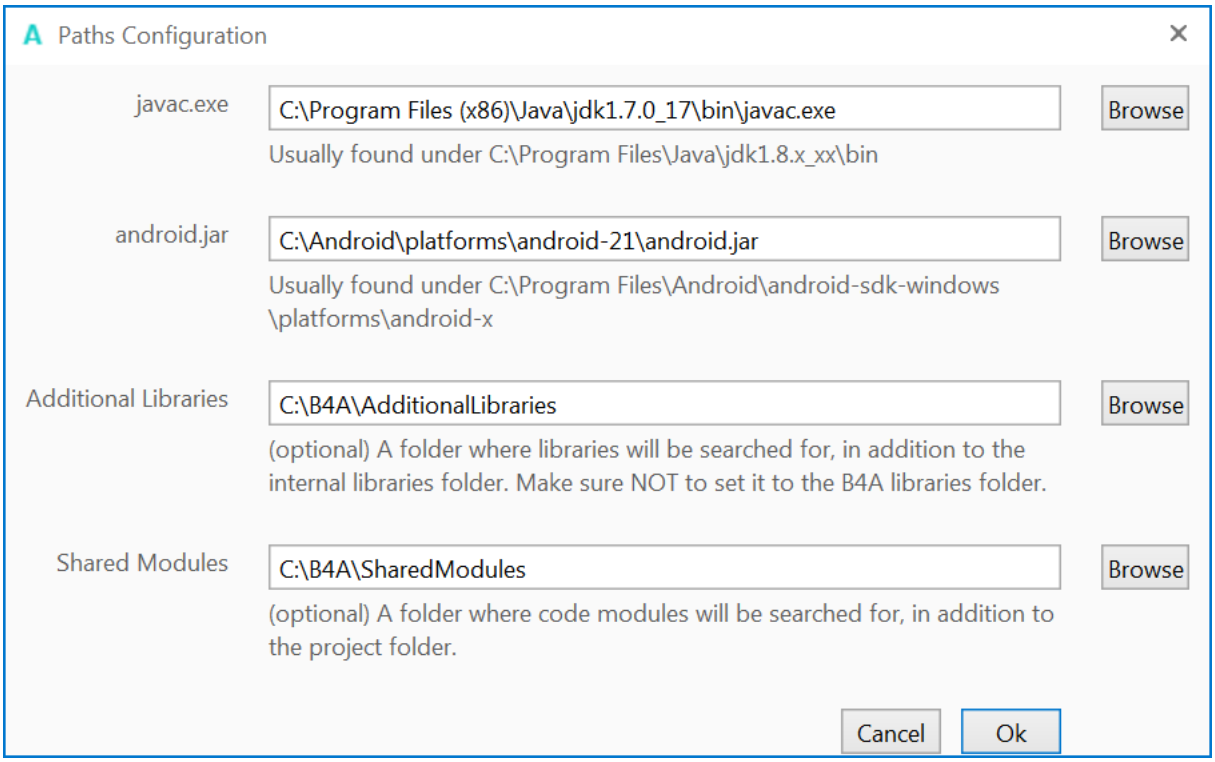
Para las librerías adicionales es útil crear una carpeta especial para guardarlas en otra parte.
Por ejemplo:
C:\B4A\AddLibraries, C:\B4i\AddLibraries, C:\B4J\AddLibraries, C:\B4R\AddLibraries

Cuando instalas una nueva versión de B4x, todas las librerías estándar se actualizan automáticamente, pero las librerías adicionales no están incluidas. La ventaja de usar una carpeta especial es que no tienes que preocuparte por estas librerías, porque esta carpeta no se ve afectada cuando se instala una nueva versión de B4x.
Las librerías adicionales no son sistemáticamente actualizadas con la nueva versión de B4x.




Cuando el IDE se inicia, mira primero las librerías disponibles en la carpeta Libraries de B4x y después en la carpeta de librerías adicionales.

Para configurar la carpeta especial de librerías adicionales clic en menú del IDE 'Tools / Configure Paths'.




Ejemplo para B4A. Es similar para los otros productos B4x.

Introduce el nombre de la carpeta y pulsa en Ok.

### 3.8.3 Cargar y actualizar una librería

Una lista de las librerías oficiales y el resto de librerías con enlaces a sus correspondientes páginas de ayuda, se puede encontrar en el sitio de B4x en:
Página de documentación de B4A: Lista de Librerías.
Página de documentación de B4i: Lista de Librerías.
Página de documentación de B4J: Lista de Librerías.
Página de documentación de B4R: Lista de Librerías.

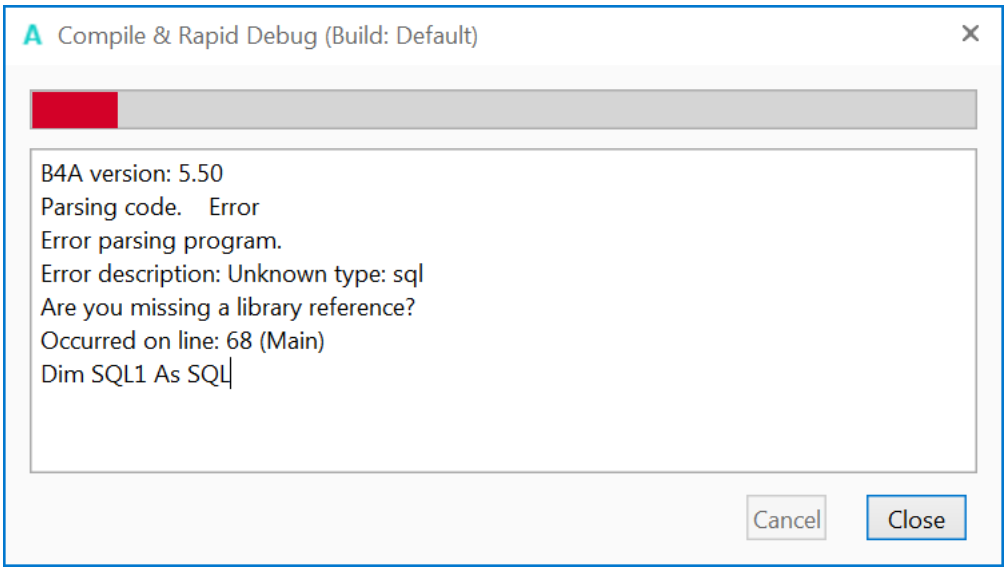
Para encontrar una librería haz una consulta como
http://www.b4x.com/search?query=betterdialogs+library
en tu navegador.

Para cargar o actualizar una librería sigue los pasos a continuación:

- Descarga el archivo zip de la librería de algún lugar.
- Descomprímelo.
- Copia los archivos xxx.jar y xxx.xml en
  - La carpeta 'libreries' de B4x si es una librería estándar de B4x
  - La carpeta configurada en 'Additional libraries' si es una librería adicional.
- Botón secundario en la lista de librerías de la pestaña 'Libraries Manager' , clic en Refresh y selecciona la librería.

### 3.8.4 Mensaje de error "¿falta la referencia de una librería?"

Si obtienes un mensaje similar a este, significa que te has olvidado de activar la librería especificada en la lista de librerías de la pestaña 'Libreries Manager' !



## 3.9 Manipulación de cadenas

### 3.9.1 B4A, B4i, B4J

B4A, B4i y B4J permiten la manipulación de cadenas de texto como otros lenguajes Basic pero con algunas diferencias.

Estas manipulaciones se pueden hacer directamente en una cadena.
Ejemplo:

```
txt = "123,234,45,23"
txt = txt.Replace(",", ";")
```

Resultado: `123;234;45;23`

Las diferentes funciones son:

- **CharAt(Index)** Devuelve el carácter en el índice dado.
- **CompareTo(Other)** Compara lexicográficamente la cadena con otra cadena.
- **Contains(SearchFor)** Comprueba si la cadena contiene el texto indicado.
- **EndsWith(Suffix)** devuelve Verdadero si la cadena termina con el 'Suffix' dado.
- **EqualsIgnoreCase(Other)** Devuelve Verdadero si ambas cadenas son iguales, ignorando mayúsculas y minúsculas.
- **GetBytes(Charset)** Codifica la cadena con un nuevo código de caracteres en una matriz de bytes.
- **IndexOf(SearchFor)** Devuelve el índice de la primera ocurrencia del texto dado en la cadena. El primer índice es 0. Devuelve -1 si no se encuentra ninguna ocurrencia.
- **IndexOf2(SearchFor, Index)** Devuelve el índice de la primera ocurrencia del texto dado en la cadena. Comienza a buscar desde el índice dado.
  El primer índice es 0. Devuelve -1 si no se encuentra ninguna ocurrencia.
- **LastIndexOf(SearchFor)** Devuelve el índice de la primera ocurrencia del texto dado en la cadena. La búsqueda comienza desde el final de la cadena y avanza hacia el principio de la cadena.
  El primer índice es 0. Devuelve -1 si no se encuentra ninguna ocurrencia.
- **LastIndexOf2(SearchFor, Index)** Devuelve el índice de la primera ocurrencia del texto dado en la cadena. La búsqueda comienza desde el índice dado y avanza hacia el principio de la cadena.
  El primer índice es 0. Devuelve -1 si no se encuentra ninguna ocurrencia.
- **Length** Devuelve la longitud, el número de caracteres de la cadena.
- **Replace(Target, Replacement)** Devuelve una nueva cadena como resultado de la sustitución de todas las ocurrencias de 'Target' con 'Replacement'.
- **StartsWith(Prefix)** Devuelve Verdadero si la cadena comienza con el 'Prefix' dado.
- **Substring(BeginIndex)** Devuelve una cadena que es una subcadena de la cadena original.
  La nueva cadena incluye el carácter en el índice indicado en 'BeginIndex' y se extiende hasta el final de la cadena.
- **Substring2(BeginIndex, EndIndex)** Devuelve una cadena que es una subcadena de la cadena original. La nueva cadena incluye el carácter en el índice indicado en 'BeginIndex y se extiende hasta el carácter anterior al índice indicado en 'EndIndex'.
  Ten en cuenta que 'EndIndex' es el índice final y no la longitud como en otros lenguajes.
- **ToLowerCase** Devuelve una nueva cadena que es el resultado de convertir a minúsculas la cadena.

- **ToUpperCase** Devuelve una nueva cadena que es el resultado de convertir a mayúsculas la cadena.
- **Trim** Devuelve una copia de la cadena original borrando los espacios al principio y al final de la cadena.

**Nota:** Las funciones de cadena son sensibles a mayúsculas.
si quieres utilizar funciones que no sean sensibles a mayúsculas, debes utilizar los métodos ToLowerCase o ToUpperCase.

Ejemplo: `NuevaCadena = CadenaOriginal.ToLowerCase.StartsWith("pre")`

### 3.9.2 B4R

B4R no soporta la manipulación de cadenas como otros lenguajes Basic.

Este tipo de manipulaciones se pueden hacer con el objeto 'ByteConverter' de la librería 'rRandomAccesFile'.

Las cadenas en B4R son diferentes que en otras herramientas B4x. Las razones de estas diferencias son:

- Memoria muy limitada.
- Falta de codificadores Unicode.

Un objeto 'String' en B4R es el mismo que 'char* String' en lenguaje C. Es una matriz de bytes con un byte 0 adicional al final.
El requisito del byte 0 al final hace imposible crear una subcadena sin copiar la memoria a una nueva dirección.
**Por esta razón, las matrices de bytes son preferibles sobre las cadenas**.
Los distintos métodos relacionados con el trabajo con cadenas funcionan con matrices de bytes.

Convertir una cadena a una matriz de bytes es muy simple y no requiere ninguna copia en memoria.
El compilador lo hará automáticamente cuando sea necesario:

```
Private b() As Byte = "abc" 'equivalente a Private b() As Byte = "abc".GetBytes
```

Solo se soportan dos funciones:

Estas funciones son:

- **GetBytes(Charset)** Devuelve la cadena como una matriz de bytes.
  Ten en cuenta que la matriz y la cadena comparten la misma memoria
- **Length** Devuelve la longitud, el número de caracteres de la cadena.

**Métodos de cadena**

Los métodos de cadena estándar están disponibles en el tipo 'ByteConverter type' (librería 'rRandomAccessFile').

Son similares a los métodos de cadena de otras herramientas B4x:

```
Private Sub AppStart
   Serial1.Initialize(115200)
   Log("AppStart")
   Dim bc As ByteConverter
   Log("IndexOf: ", bc.IndexOf("0123456", "3")) 'IndexOf: 3
   Dim b() As Byte = " abc,def,ghijkl "
   Log("Substring: ", bc.SubString(b, 3)) 'Substring: c,def,ghijkl
   Log("Trim: ", bc.Trim(b)) 'Trim: abc,def,ghijkl
   For Each s() As Byte In bc.Split(b, ",")
     Log("Split: ", s)
     'Split: abc
     'Split: def
     'Split: ghijkl
   Next
   Dim c As String = JoinStrings(Array As String("Number of millis: ", Millis, CRLF, "N
umber of micros: ", Micros))
   Log("c = ", c)
   Dim b() As Byte = bc.SubString2(c, 0, 5)
   b(0) = Asc("X")
   Log("b = ", b)
   Log("c = ", c) 'el primer carácter será X
End Sub
```

Nota como cadenas y matrices pueden ser utilizadas ya que el compilador convierte las cadenas a matrices de bytes automáticamente.

Con la excepción de 'JoinStrings', ninguno de los métodos anteriores hace una copia de la cadena o los bytes originales.
Esto significa que la modificación de la matriz devuelta como en las últimas tres líneas también va a modificar la matriz original.

Esto también sucede con cadenas literales ya que todas comparten el mismo bloque de memoria:

```
Private Sub AppStart
   Serial1.Initialize(115200)
   Log("AppStart")
   Dim bc As ByteConverter
   Dim b() As Byte = bc.Trim("abcdef ")
   b(0) = Asc("M") 'Esta línea cambiará el valor de la cadena literal
   Dim s as String = "abcdef "
   Log(s) 'Mbcdef
End Sub
```

Manipulación de cadenas con el objeto 'ByteConverte':

- **EndsWith(Source As Byte(), Suffix As Byte())**
  devuelve Verdadero si la cadena termina con el 'Suffix' dado.
- **IndexOf(Source As Byte(), SearchFor As Byte())**
  Devuelve el índice de la primera ocurrencia de 'SearchFor' en la cadena.
- **IndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**
  Devuelve el índice de la primera ocurrencia del texto dado en la cadena. Comienza a buscar desde el índice dado.
- **LastIndexOf(Source As Byte(), SearchFor As Byte())**
  Devuelve el índice de la primera ocurrencia del texto dado en la cadena. Comienza a buscar desde el final de la cadena y avanza hasta el principio.
- **LastIndexOf2(Source As Byte(), SearchFor As Byte(), Index As UInt)**
  Devuelve el índice de la primera ocurrencia del texto dado en la cadena. Comienza a buscar desde el índice dado y avanza hasta el principio.
- **StartsWith(Source As Byte(), Prefix As Byte())**
  Devuelve Verdadero si la cadena comienza con el 'Prefix' dado.
- **Substring(Source As Byte(), BeginIndex As UInt)**
  Devuelve una cadena que es una subcadena de la cadena original.
  La nueva cadena incluye el carácter en el índice indicado en 'BeginIndex' y se extiende hasta el final de la cadena.
- **Substring2(Source As Byte(), BeginIndex As UInt, EndIndex As UInt)**
  Devuelve una cadena que es una subcadena de la cadena original. La nueva cadena incluye el carácter en el índice indicado en 'BeginIndex y se extiende hasta el carácter anterior al índice indicado en 'EndIndex'.
- **Trim(Source As Byte())**
  Devuelve una copia de la cadena original borrando los espacios al principio y al final de la cadena.

**Formato de números**, muestra los números como cadenas con diferentes formatos:

- **NumberFormat**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)
  `NumberFormat(12345.6789, 0, 2)` = 12,345.68
  `NumberFormat(1, 3 ,0)` = 001
  `NumberFormat(Value, 3 ,0)` pueden utilizarse variables.
  `NumberFormat(Value + 10, 3 ,0)` pueden utilizarse operaciones aritméticas.
  `NumberFormat((lblscore.Text + 10), 0, 0)` si una variable es una cadena, añade paréntesis.

## 3.10 Formato de números

### 3.10.1 B4A, B4i, B4J

Formato de números, muestra los números como cadenas con diferentes formatos, hay dos palabras claves:

- **NumberFormat**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)
  `NumberFormat(12345.6789, 0, 2)` = 12,345.68
  `NumberFormat(1, 3 ,0)` = 001
  `NumberFormat(Value, 3 ,0)` pueden utilizarse variables.
  `NumberFormat(Value + 10, 3 ,0)` pueden utilizarse operaciones aritméticas.
  `NumberFormat((lblscore.Text + 10), 0, 0)` si una variables es una cadena, añade paréntesis.

- **NumberFormat2**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int, MinimumFractions As Int, GroupingUsed As Boolean)
  `NumberFormat2(12345.67, 0, 3, 3, True)` = 12,345.670

### 3.10.2 B4R

**Formato de números**, muestra los números como cadenas con diferentes formatos:

- **NumberFormat**(Number As Double, MinimumIntegers As Int, MaximumFractions As Int)
  `NumberFormat(12345.6789, 0, 2)` = 12,345.68
  `NumberFormat(1, 3 ,0)` = 001
  `NumberFormat(Value, 3 ,0)` pueden utilizarse variables.
  `NumberFormat(Value + 10, 3 ,0)` pueden utilizarse operaciones aritméticas.
  `NumberFormat((lblscore.Text + 10), 0, 0)` si una variable es una cadena, añade paréntesis.

## 3.11 Temporizadores

Un objeto temporizador genera Eventos cada cierto intervalo de tiempo. El uso de un temporizador es una buena alternativa a un bucle largo, ya que permite a la interfaz de usuario manejar otros eventos y mensajes.
Ten en cuenta que los eventos del temporizador no se activarán mientras la interfaz de usuario está ocupada ejecutando el resto del código(a menos que uses 'DoEvents').
Los eventos de temporizador no se activarán cuando la 'Activity' esté en pausa o un cuadro de dialogo (como 'Msgbox') es visible.
También es importante desactivar el temporizador cuando la 'Activity' está en pausa y volver a habilitarlo cuando se reanuda. Esto ahorrará uso de CPU y batería.

Un temporizador tiene:

- Tres parámetros.
  - **Initialize** Inicializa el temporizador con dos parámetros, el nombre del evento y el intervalo.
    Timer1.Initialize(EventName As String, Interval As Long)
    Ejemplo: `Timer1.Initialize("Timer1", 1000)`

  - **Interval** Establece el intervalo de tiempo en milisegundos.
    Timer1. Interval = Interval
    Ejemplo: `Timer1.Interval = 1000`, 1 segundo

  - **Enabled** Activa o desactiva el temporizador. **Es Falso por defecto.**
    Ejemplo: `Timer1.Enabled = True`

- Un Evento
  - **Tick** La rutina Tick es llamada cada intervalo de tiempo configurado con el parámetro 'Interval'.
    Ejemplo: `Sub Timer1_Tick`

**El temporizador debe ser declarado en la rutina 'Process_Global'.**

```
Sub Process_Globals
   Public Timer1 As Timer
```

**Pero debe ser inicializado en una de las siguientes rutinas en el módulo donde la rutina del evento 'Tick' es usado.**

**B4A:** Rutina 'Activity_Create'

```
Sub Activity_Create(FirstTime As Boolean)
   If FirstTime = True Then
      Timer1.Initialize("Timer1", 1000)
   End If
```

**B4i:** Rutina 'Application_Start'

```
Private Sub Application_Start (Nav As NavigationController)
   Timer1.Initialize("Timer1", 1000)
```

**B4J:** Rutina 'Activity_Create'

```
Sub AppStart (Form1 As Form, Args() As String)
   Timer1.Initialize("Timer1", 1000)
```

**B4R:** Rutina 'Activity_Create'

```
Private Sub AppStart
   Timer1.Initialize("Timer1", 1000)
```

Y la rutina del evento 'Tick' del temporizador.
Esta rutina será llamada cada segundo (1000 milisegundos) por el sistema operativo.

```
Sub Timer1_Tick
   ' Hacer algo
End Sub
```

## 3.12 Archivos B4A, B4i, B4J

Muchas aplicaciones requieren acceso a un almacenamiento persistente. Los dos tipos más comunes de almacenamiento son los archivos y las bases de datos.

Android e iOS tienen sus propios sistemas de archivos. Ni B4A ni B4i tienen acceso al sistema de archivos del sistema Windows,

Para agregar archivos a tu proyecto debes agregarlos a la pestaña 'Files' del IDE. Estos archivos se agregarán a la carpeta 'Files' del proyecto.

### 3.12.1 El objeto 'File'

El objeto predefinido `'File'` tiene una serie de funciones para trabajar con archivos.

Ubicación de los archivos - Hay varias importantes ubicaciones donde puedes escribir y leer archivos.

**File.DirAssets**
La carpeta 'assets' incluye los archivos añadidos desde la pestaña 'File Manager' en el IDE.
Es la carpeta 'Files' en la carpeta del proyecto.
**Estos archivos son de solo lectura**.
No se pueden crear nuevos archivos en esta carpeta (que se encuentra realmente dentro del archivo apk)).
Si tienes un archivo de base de datos en la carpeta 'Assets', necesitas copiarla a otra ubicación antes de usarla.

**File.DirInternal / File.DirInternalCache**
Estas dos carpetas se ubican en la memoria principal del dispositivo y son privadas de la aplicación.
Otras aplicaciones no pueden acceder a estos archivos.
La carpeta de caché puede ser borrada por el sistema operativo si necesita más espacio.

**File.DirRootExternal**
La carpeta raíz de la tarjeta de almacenamiento.

**File.DirDefaultExternal**
La carpeta predeterminada para la aplicación en la tarjeta SD.
La carpeta es: <tarjeta de almacenamiento>/Android/data/<nombre del paquete>/files/
Será creada si es necesario.

Ten en cuenta que usar cualquiera de las dos propiedades anteriores, agregará el permiso 'EXTERNAL_STORAGE' a la aplicación.

Sugerencia: También puedes comprobar si hay una tarjeta de almacenamiento y si está disponible con **'File.ExternalReadable'** y **'File.ExternalWritable'**.

Para comprobar si ya existe un archivo usa:
**File.Exists** ( Dir As String, FileName As String)
Devuelve Verdadero si el archivo existe y Falso si no existe.

El objeto 'File' incluye varios métodos de escritura y lectura de archivos.
Para ser capaz de escribir o leer un archivo, éste debe ser abierto.

**File.OpenOutput** (Dir As String, FileName As String, Append As Boolean)
- Abre el archivo para escritura, el parámetro 'Append' indica si el texto se añade al final del texto ya existente o no. Si el archivo no existe se crea.

**File.OpenInput** (Dir As String, FileName As String)
- Abre el archivo para lectura.

**File.WriteString** (Dir As String, FileName As String, Text As String)
- Escribe el texto en un archivo nuevo.

**File.ReadString** (Dir As String, FileName As String) As String
- Lee un archivo y devuelve su contenido como una cadena.

**File.WriteList** (Dir As String, FileName As String, List As List)
- Escribe todos los valores almacenados en una lista en un archivo. Todos los valores son convertidos al tipo 'String' si es necesario. Cada valor se almacenará en una línea.
Ten en cuenta que si el valor contiene un carácter de nueva línea se guardara en más de una línea y cuando vuelvas a leer el archivo lo harás como varios elementos.

**File.ReadList** (Dir As String, FileName As String) As List
- Lee un archivo y almacena cada línea como un elemento en una lista.

**File.WriteMap** (Dir As String, FileName As String, Map As Map)
- Toma un objeto 'map' que contiene pares de clave y valor y los almacena en un archivo de texto.
El formato de archivo es conocido como archivo Java de propiedades: .properties - Wikipedia, the free encyclopedia
El formato de archivo no es demasiado importante, a menos que se suponga que el archivo sea para ser editado manualmente. Este formato hace que sea fácil editarlo manualmente.
Un uso común de 'File.WriteMap' es para guardar un mapa de 'configuraciones' a un archivo.

**File.ReadMap** (Dir As String, FileName As String) As Map
- Lee un archivo de propiedades y devuelve sus pares clave/valor como un objeto 'Map'. Ten en cuenta que el orden de los elementos devueltos podría ser diferente del orden original.

Algunas otras funciones útiles:

**File.Copy** (DirSource As String, FileSource As String, DirTarget As String, FileTarget As String)
- Copia el archivo origen ('FileSource') desde el directorio origen ('DirSource') al archivo destino ('FileTarget') en el directorio destino ('DirTarget').
Ten en cuenta que no es posible copiar archivos a la carpeta 'Assets'.

**File.Delete** (Dir As String, FileName As String)
- Elimina el archivo en el directorio indicado.

**File.ListFiles** (Dir As String) As List
- Lista los archivos y subdirectorios en el directorio indicado.
Ejemplo:
```
Private List1 As List
List1 = File.ListFiles(File.DirRootExternal)
```
List1 puede ser declarada en 'Sub Globals'

**File.Size** (Dir As String, FileName As String)
- Devuelve el tamaño en bytes del archivo especificado.
Este método no es compatible con los archivos en la carpeta 'assets'.

### 3.12.2 Nombres de archivo

En B4x los nombres de archivos permiten los siguientes caracteres:
**de a** a **z**, de **A** a **Z**, de **0** a **9,** punto **.** guión bajo **_** y los siguientes caracteres **+ - % &**
Espacios y estos caracteres *** ?** no están permitidos.

Ejemplo : MyFile.txt

Ten en cuenta que en B4x los nombres de archivos son sensibles a mayúsculas !
`MyFile.txt` es diferente que `myfile.txt`

### 3.12.3 Subcarpetas

Puedes crear subcarpetas en B4x con:

```
File.MakeDir(File.DirInternal, "Pictures")
```

Para acceder a la subcarpeta se debe añadir "/" entre el nombre de la carpeta y la subcarpeta.

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal & "/Pictures", "test1.png")
```

O añadir el nombre de la subcarpeta más "/" antes del nombre del archivo.

```
ImageView1.Bitmap = LoadBitmap(File.DirInternal, "Pictures/test1.png")
```

Ambas posibilidades funcionan.

## 3.12.4 TextWriter

Hay otras dos útiles funciones para los archivos de texto: '**TextWriter**' y 'TextReader':

**TextWriter.Initialize** (OutputStream As OutputStream)
- Inicializa el objeto 'TextWriter' como un flujo de escritura.

Ejemplo:
```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirRootExternal, "Test.txt" , False))
```

La variable 'Writer' puede ser declarada en 'Sub Globals'.

**TextWriter.Initialize2** (OutputStream As OutputStream , Encoding As String)
- Inicializa el objeto 'TextWriter' como un flujo de escritura.
- 'Encoding' indica la página de código (también llamado conjunto de caracteres), la codificación del texto (ver el capítulo siguiente).

Ejemplo:
```
Private Writer As TextWriter
Writer.Initialize2(File.OpenOutput(File.DirRootExternal,"Test.txt" ,False)," ISO-8859-1")
```

La variable 'Writer' puede ser declarada en 'Sub Globals'.
Ver : Codificación del texto

**TextWriter.Write** (Text As String)
- Escribe el texto indicado en el flujo.

**TextWriter.WriteLine** (Text As String)
- Escribe el texto indicado en el flujo seguido por un carácter de nueva línea LF Chr(10).

**TextWriter.WriteList** (List As List)
- Escribe cada elemento de la lista como una sola línea.
Ten en cuenta que un valor que contenga CRLF se guardará como dos líneas (que se devolverá como dos elementos cuando se lea con 'ReadList').
Todos los valores se convierten en cadenas.

**TextWriter.Close**
- Cierra el flujo.

Ejemplo:

```
Private Writer As TextWriter
Writer.Initialize(File.OpenOutput(File.DirDefaultExternal, "Text.txt", False))
Writer.WriteLine("Esta es la primera línea")
Writer.WriteLine("Esta es la segunda línea")
Writer.Close
```

## 3.12.5 TextReader

Hay otras dos útiles funciones para los archivos de texto: 'TextWriter' y **'TextReader'**:

**TextReader.Initialize** (InputStream As InputStream)
- Inicializa 'TextReader' como flujo de lectura.

Ejemplo:
```
Private Reader  TextReader
Reader.Initialize(File.InputOutput(File.DirRootExternal, "Test.txt"))
```

La variable 'Reader' puede ser declarada en 'Sub Globals'.

**TextReader.Initialize2** (InputStream As InputStream, Encoding As String)
- Inicializa 'TextReader' como flujo de lectura.
- 'Encoding' indica la página de código (también llamado conjunto de caracteres), la codificación del texto.

Ejemplo:
```
Private Reader  TextReader
Reader.Initialize2(File.OpenInput(File.DirRootExternal, "Test.txt", "ISO-8859-1")
```

La variable 'Reader' puede ser declarada en 'Sub Globals'.
Ver : Codificación del texto

**TextReader.ReadAll** As String
- Lee el resto del texto y cierra el flujo.

Ejemplo:
```
txt = Reader.ReadAll
```

**TextReader.ReadLine** As String
- Lee la siguiente línea del flujo.
Los caracteres de nueva línea no son devueltos.
Devuelve 'Null' si no hay más caracteres a leer.

Ejemplo:

```
Private Reader As TextReader
Reader.Initialize(File.OpenInput(File.DirDefaultExternal, "Text.txt"))
Private line As String
line = Reader.ReadLine
Do While line <> Null
   Log(line)
   line = Reader.ReadLine
Loop
Reader.Close
```

**TextReader.ReadList** As List
- Lee el resto del texto y devuelve un objeto 'List' rellenando cada elemento de la lista con una línea. Cierra el flujo al terminar.

Ejemplo:
```
List1 = Reader.ReadList
```

### 3.12.6 Codificación del texto

La codificación de texto o codificación de caracteres consiste en un código que empareja cada carácter de un determinado repertorio con algo más. Otros términos como conjunto de caracteres, y a veces mapa de caracteres o página de código se utilizan casi indistintamente (fuente: Wikipedia).

El conjunto de caracteres por defecto en Android es Unicode UTF-8.

En Windows los conjuntos de caracteres más comunes son ASCII y ANSI.

- ASCII incluye definiciones para 128 caracteres, 33 son caracteres de control no imprimibles (ya casi en desuso), que afectan a como el texto y el espacio son procesados.
- ANSI, Windows-1252 o CP-1252 es una codificación de caracteres del alfabeto latino, utilizado de forma predeterminada en los componentes heredados de Microsoft Windows en inglés y otros idiomas occidentales con 256 definiciones (de un byte). Los primeros 128 caracteres son los mismos que en la codificación ASCII.

Muchos de los archivos generados por los programas de Windows son codificados con el conjunto de caracteres ANSI en los países occidentales. Por ejemplo: Archivos csv de Excel, archivos del Bloc de notas por defecto.
Pero con el Bloc de notas, los archivos se pueden guardar codificados con *UTF-8*.

Android puede utilizar los siguientes conjuntos de caracteres:

- UTF-8 conjunto de caracteres por defecto
- UTF -16
- UTF - 16 BE
- UTF - LE
- US-ASCII Conjunto de caracteres ASCII
- ISO-8859-1 Casi equivalente al conjunto de caracteres ANSI
- Windows-1251 Caracteres cirílicos
- Windows-1252 Alfabeto latino

Para leer los archivos de Windows codificados con la norma ANSI, debes utilizar el conjunto de caracteres *Windows-1252*.
Si necesitas escribir archivos para su uso con Windows también se debe utilizar el conjunto de caracteres *Windows-1252*.

Otra diferencia entre Windows y Android es el carácter de final de línea:

- Android, solo añade el carácter LF (avance de línea) Chr(10) al final de una línea.
- Windows, añade dos caracteres CR (retorno de carro Chr(13)) y LF Chr(10) al final de una línea. Si necesitas escribir archivos para Windows debes añadir CR por ti mismo.

El símbolo para el final de la línea es :

- B4x CRLF Chr(10)
- Basic4PPC CRLF Chr(13) & Chr(10)

Para leer o escribir archivos con una codificación diferente, debes utilizar los objetos 'TextReader' o 'TextWriter' con el método 'Initialize2'. Incluso para leer archivos csv.

Sugerencia para leer archivos csv de Excel:
Puedes:

- En el escritorio, cargar el archivo csv en un editor de texto como *el Bloc de notas* o *Notepad++*
- Guardar el archivo codificado con *UTF-8*
  Con *Notepad++* usa Codificar en UTF-8 sin BOM, ver más abajo.

O

- Leer todo el archivo con 'TextReader.Initializa2' y codificación "Windows-1252".
- volver a guardarlo con 'TextWriter.Initialize' con la codificación estándar de Android.
- Leer el archivo con 'LoadCSV' o 'LoadCSV2' de la librería 'StringUtils'.

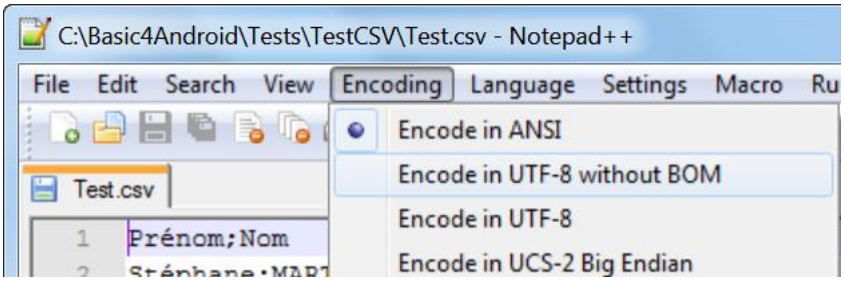
```
Private txt As String
Private tr As TextReader
tr.Initialize2(File.OpenInput(File.DirAssets, "TestCSV1_W.csv"), "Windows-1252")
txt = tr.ReadAll
tr.Close

Private tw As TextWriter
tw.Initialize(File.OpenOutput(File.DirInternal, "TestCSV1_W.csv", False))
tw.Write(txt)
tw.Close

lstTest = StrUtil.LoadCSV2(File.DirInternal, "TestCSV1_W.csv", ";", lstHead)
```

Cuando se guarda un archivo con el Bloc de notas se agregan tres bytes adicionales.
Estos bytes son llamados caracteres BOM (Byte Order Mark).
En *UTF-8* son representados por esta secuencia de bytes: `0xEF,0xBB,0xBF`.
Un editor de textos o un navegador web que interprete el texto como *Windows-1252* mostrará en pantalla los caracteres `ï»¿`.

Para evitar esto se puede utilizar *Notepad++* en lugar del *Bloc de notas* y usar codificar en *UTF-8* sin BOM.




Otra posibilidad para cambiar un texto de *Windows-1252* a *UTF-8* es utilizar el código de abajo.

```
Private var, result As String
var = "Gestión"
Private arrByte() As Byte
arrByte = var.GetBytes("Windows-1252")
result = BytesToString(arrByte, 0, arrByte.Length, "UTF8")
```

## 3.13 Listas, sólo B4A, B4i and B4J

Las listas son similares a las matrices dinámicas.

Una lista debe ser inicializada antes de ser usada.

- Initialize Inicializa una lista vacía.
  ```
  Private List1 As List
  List1.Initialize
  Lista1.AddAll(Array As Int(1, 2, 3, 4, 5))
  ```

- Initialize2 (AlgunaMatriz)
  Inicializa una lista con los valores dados. Este método debe ser utilizado para convertir matrices en listas. Ten en cuenta que si pasas una lista a este método, a continuación ambos objetos comparten la misma lista, y si pasas una matriz la lista será de un tamaño fijo.

  Lo que significa que más tarde no podrás agregar o quitar elementos.
  Ejemplo 1:
  ```
  Private List1 As List
  List1.Initialize2(Array As Int(1, 2, 3, 4, 5))
  ```
  Ejemplo 2:
  ```
  Private List1 As List
  Private AlgunaMatriz(10) As String
  ' Llenar la matriz
  List1.Initialize2(AlgunaMatriz)
  ```

Puedes añadir y eliminar elementos de una lista y en consecuencia su tamaño va a cambiar.
Con:

- Add (item As Object)
  Agrega un valor al final de la lista.
  ```
  List1.Add(Valor)
  ```

- AddAll (Array As String("value1", "value2"))
  Añade todos los elementos de una matriz al final de la lista.
  ```
  Lista1.AddAll(Lista2)
  Lista1.AddAll(Array As Int(1, 2, 3, 4, 5))
  ```

- AddAllAt (Index As Int, List As List)
  Inserta todos los elementos de una matriz en la lista, empezando en la posición indicada.
  ```
  Lista1.AddAll(12, Lista2)
  Lista1.AddAllAt(12, Array As Int(1, 2, 3, 4, 5))
  ```

- InsertAt (Index As Int, Item As Object)
  Inserta el elemento especificado en el índice especificado.
  Como resultado, todos los elementos con mayor o igual índice que el especificado, se desplazan.
  ```
  Lista1.InsertAt(12, Valor)
  ```

- RemoveAt (Index As Int)
  Elimina el elemento en la posición dada de la lista.
  ```
  Lista1.RemoveAt(12)
  ```

Una lista puede contener cualquier tipo de objeto. Sin embargo, si una lista es declarada como un objeto 'Process Global' no puede contener objetos de 'Activity' (como las 'Views').
B4x convierte automáticamente las matrices normales en listas. Así que cuando se espera una lista, se puede pasar una matriz en su lugar.

Obtener el tamaño de una lista:

- `Lista1.Size`

Utiliza el método 'Get' para obtener un elemento de la lista con (Índices de lista basados en 0):
Para obtener el primer elemento utiliza Get(0).
Para obtener el último elemento utiliza Get(Lista1.Size - 1).

- Get(Index As Int)
  ```
  elemento = Lista1.Get(i)
  ```

  Puedes utilizar un bucle 'For' para iterar a través de todos los valores:
  ```
  For i = 0 To List1.Size - 1
     Private elemento As Int
     elemento = Lista1.Get(i)
     ...
  Next
  ```

Las listas se pueden guardar y cargar desde archivos con:

- File.WriteList(Dir As String, FileName As String, List As List)
  ```
  File.WriteList(File.DirRootExternal, "Test.txt", Lista1)
  ```
- File.ReadList (Dir As String, FileName As String)
  ```
  Lista1 = File.ReadList(File.DirRootExternal, "Test.txt")
  ```

Un simple elemento se puede cambiar con :

- Lista1. Set(Index As Int, Item As Object)
  ```
  Lista1.Set(12, Valor)
  ```

Una lista puede ser ordenada (los elementos deben ser números o cadenas de texto) con :

- Sort(Ascending As Boolean)
  `Lista1.Sort(True)` orden ascendente
  `Lista1.Sort(False)` orden descendente
- SortCaseInsensitive(Ascending As Boolean)

Borra una lista con :

- `Lista1.Clear`

## 3.14 Maps, solo B4A, B4i and B4J

Un 'Map' es una colección que contiene pares de claves y valores.

Las claves son únicas. Lo que significa que si se agrega un par de clave/valor (entrada) y la colección ya tiene una entrada con la misma clave, el de la entrada anterior será eliminado del 'map'.

La clave debe ser una cadena de texto o un número. El valor puede ser cualquier tipo de objeto.

Al igual que listas, un 'map' puede contener cualquier objeto, sin embargo, si se trata de una variable de 'Process_Global', no puede contener objetos de 'Activity' (como las 'Views').

Los 'Maps' son muy útiles para almacenar configuraciones.

Los 'Maps' se utilizan en estos ejemplos:

- Módulo DBUtils
  Usados para las entradas de base de datos, las claves son la columna nombres y los valores la columna valores.

Un 'Map' debe ser inicializado antes de ser utilizado.

- Initialize Inicializa un 'Map' vacío.
```
Private Map1 As Map
Map1.Initialize
```

Agregar una nueva entrada :

- Put(Key As Object, Value As Object)
```
Map1.Put("Idioma", "Español")
```

Obtener una entrada :

- Get(Key As Object)
```
Idioma = Map1.Get("Idioma")
```

Obtener una clave o un valor en un determinado índice (solo B4A y B4J):

Devuelve el valor del elemento en el índice dado.
'GetKeyAt' y 'GetValueAt' debe ser utilizado para iterar entre todos los elementos.
Estos métodos están optimizados para iterar entre los elementos en orden ascendente.

- GetKeyAt(Index As Int)
```
Clave = Map1.GetKeyAt(12)
```

Obtener un valor en un determinado índice (solo B4A y B4J):

- GetValueAt(Index As Int)
```
Valor = Map1.GetValueAt(12)
```

Comprobar si el 'Map' contiene una entrada, comprueba si hay una entrada con la clave dada :

- ContainsKey(Key As Object)
```
If Map1.ContainsKey("Idioma") Then
   Msgbox("Ya existe una entrada con esta clave!", "ATENCIÓN")
   Return
End If
```

Eliminar una entrada :

- Remove(Key As Object)
  ```
  Map1.Remove("Idioma")
  ```

Borrar una entrada, borra todos los elementos de un 'Map' :

- Clear
  ```
  Map1.Clear
  ```

Los 'Maps' se pueden guardar y cargar con :

- File.WriteMap(Dir As String, FileName As String, Map As Map)
  ```
  File.WriteMap(File.DirInternal, "configuraciones.txt", mapConfiguraciones)
  ```

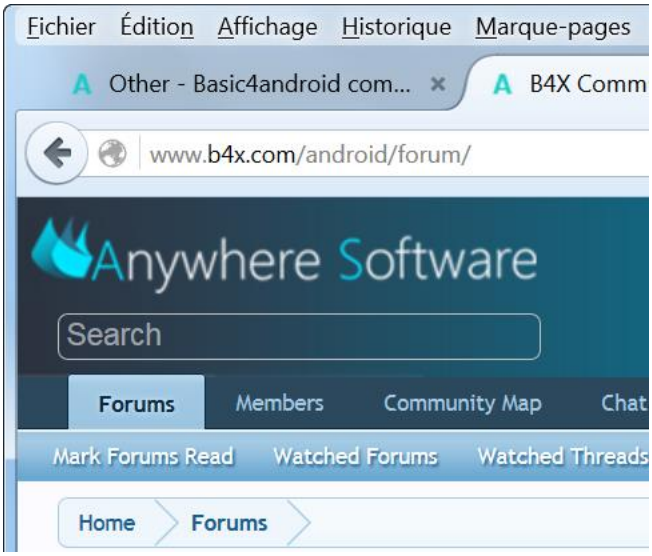
- ReadMap(Dir As String, FileName As String)
  Lee el archivo y analiza cada línea como un par clave-valor (de cadenas de texto).
  Ten en cuenta que el orden de los elementos en el 'Map' puede no ser el mismo que el orden en el archivo.
  ```
  mapConfiguraciones = File.ReadMap(File.DirInternal, "configuraciones.txt")
  ```

- File.ReadMap2(Dir As String, FileName As String, Map As Map)
  Similar a 'ReadMap'. ReadMap2 agrega los elementos al 'Map' indicado.
  Mediante el uso de 'ReadMap2' con un 'map' con elementos, se puede forzar el orden de los elementos según sea necesario.
  ```
  mapConfiguraciones = File.ReadMap2(File.DirInternal, "configuraciones1.txt",
  mapconfiguraciones)
  ```

# 4 Herramientas de ayuda

Para encontrar respuestas a muchas preguntas acerca de B4x las siguientes herramientas son muy útiles.

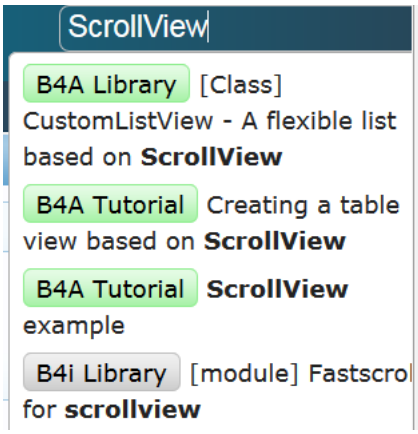
## 4.1 Función de búsqueda en el foro



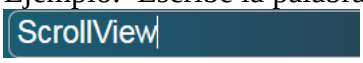

En la esquina superior izquierda se encuentra el cuadro de búsqueda para el foro. Dependiendo del tamaño de la ventana puede ser en la parte superior derecha.

Escribe una pregunta o una palabra clave y pulsa 'Intro".

La función muestra los mensajes que coinciden con la solicitud.



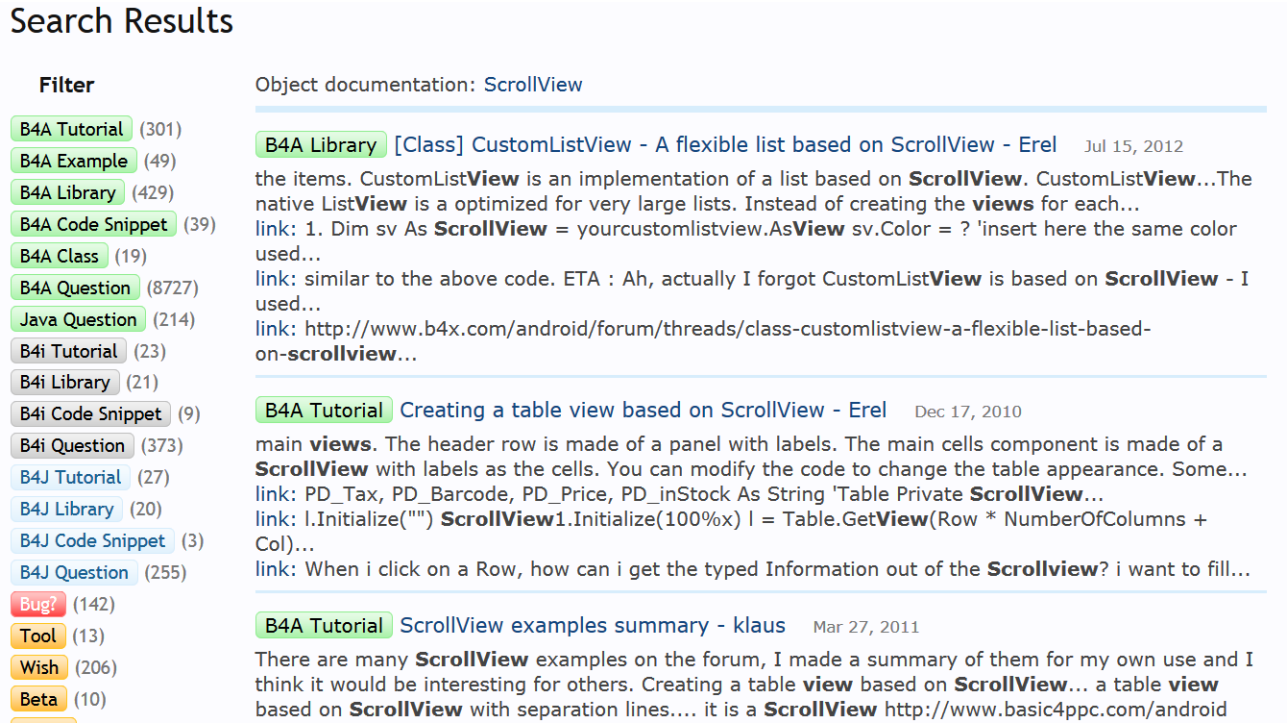

Ejemplo: Escribe la palabra clave 'ScrollView' :




Una lista de los resultados se muestra debajo del cuadro de búsqueda.

Clic en un elemento para mostrar el hilo.

Y el resultado:



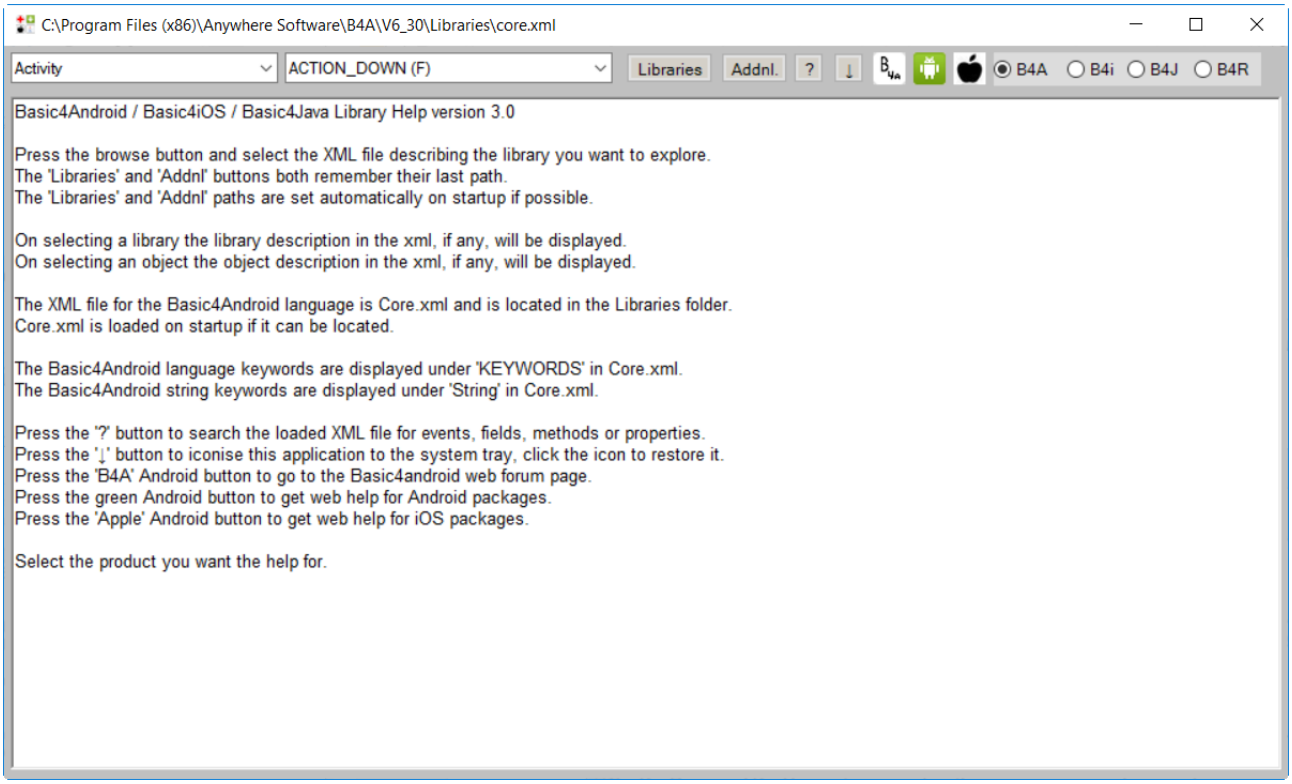

A la izquierda tienes una lista de foros, los cuales se pueden filtrar.

Clic en el título para mostrar la entrada seleccionada.

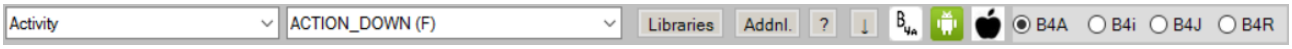
## 4.2 Visor de ayuda de B4x (B4xHelp)

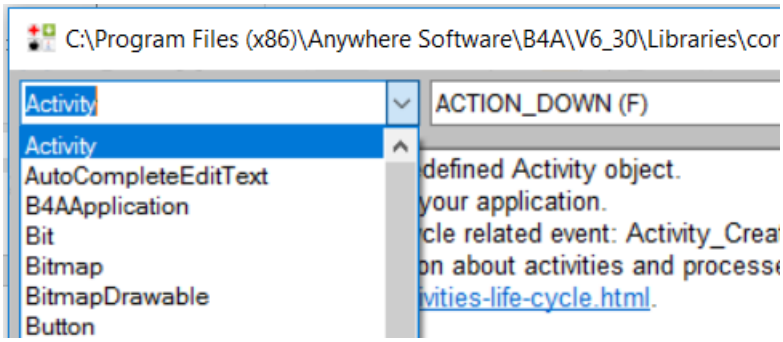
Este programa muestra los archivos xml de ayuda. Fue escrito originalmente por Andrew Graham (agrham) para B4A. Lo he modificado, con el consentimiento de Andrew, para mostrar los archivos xml de ayuda de B4A, B4J, B4i y B4R.

El programa puede ser descargado en el foro.

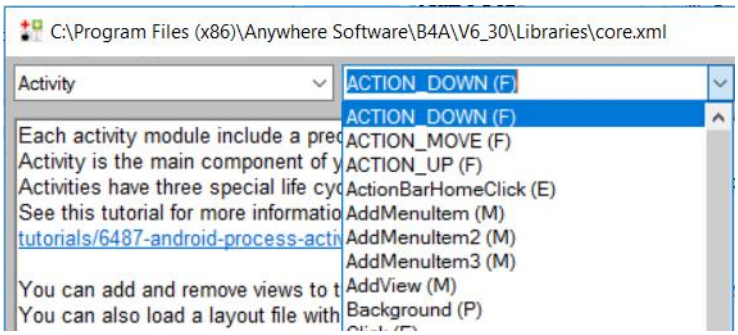


En la parte superior encontramos:




En la esquina superior izquierda una lista desplegable muestra los diferentes objetos que se incluyen en la librería seleccionada.

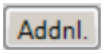



Además de la lista de objetos puedes encontrar una lista desplegable con los

- métodos(M)
- eventos(E)
- propiedades(P)
- campos(F) constantes

del objeto seleccionado.

Selecciona para mostrar las librerías estándar.




Selecciona para mostrar las librerías adicionales.

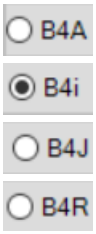
Motor de búsqueda para encontrar el objeto por una palabra clave determinada.

Cierra B4AHelp

Inicia el foro de la comunidad en línea.

Inicia el sitio de desarrolladores del Android.

Inicia el sitio de desarrolladores de iOS.



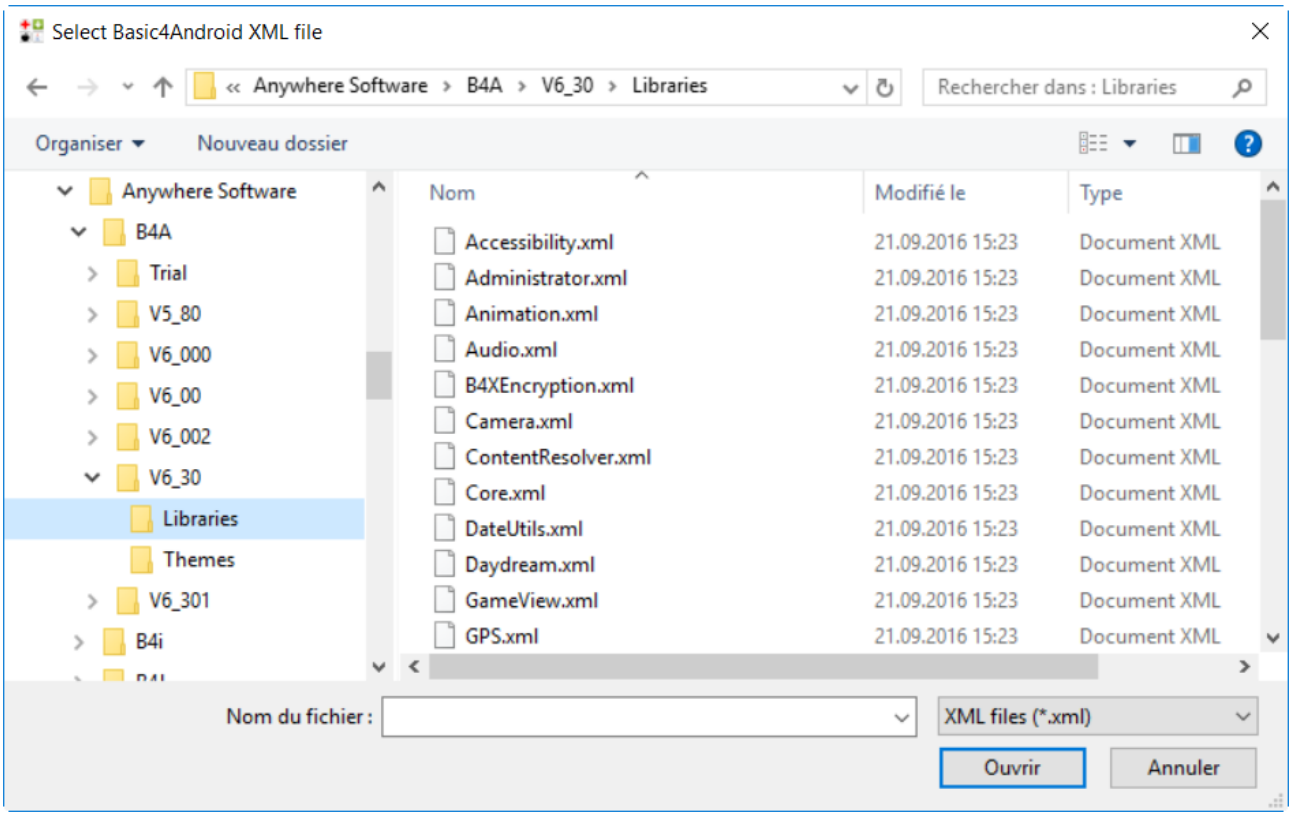

Archivos de ayuda de B4A.

Archivos de ayuda de B4i.

Archivos de ayuda de B4J.

Archivos de ayuda de B4R.

Libraries Librerías estándar



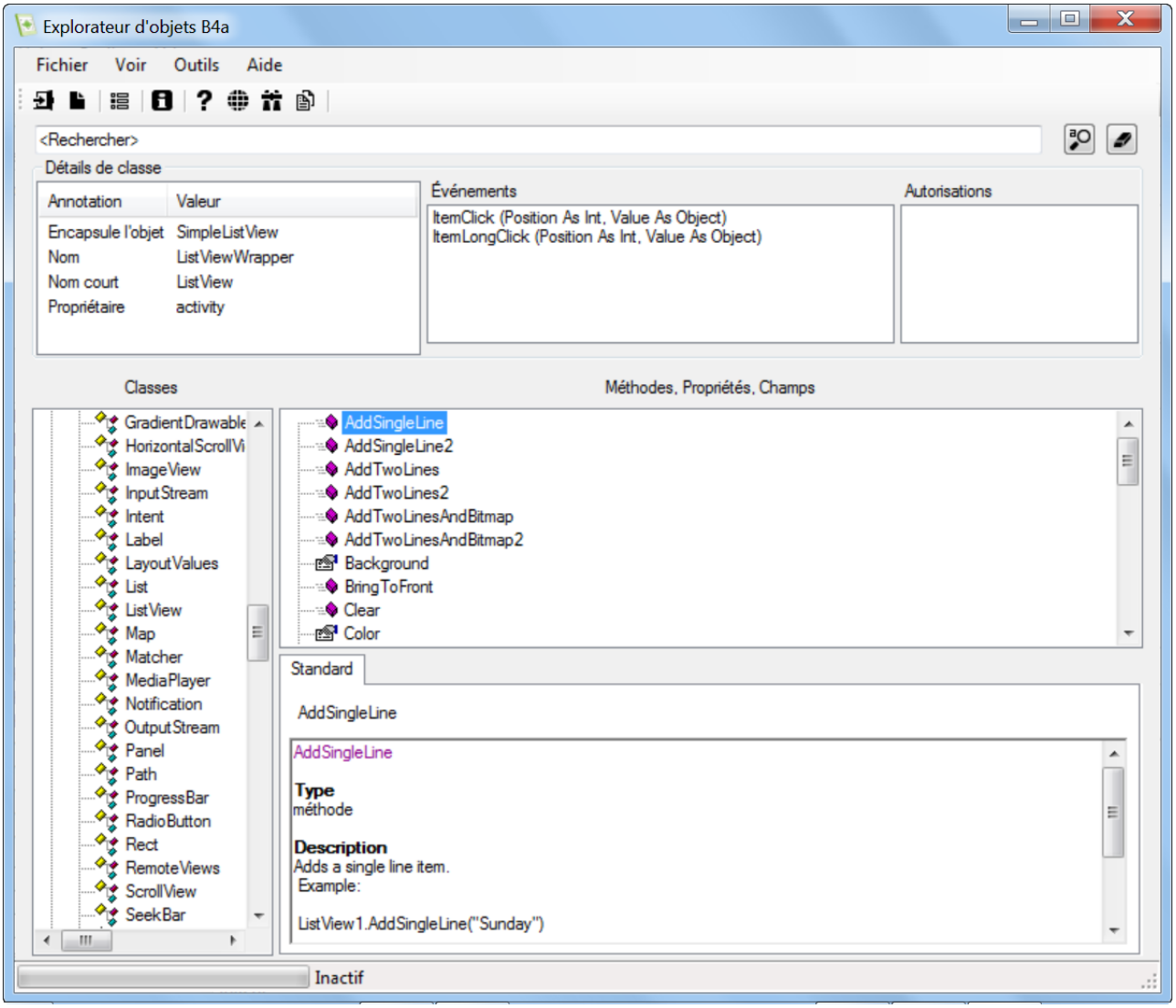

Selecciona el directorio de las librerías estándar y clic en Ouvrir (Abrir).

Aquí « Anywhere Software › B4A › V6_30 › Libraries se puede seleccionar el directorio donde son guardadas las librerías estándar.

Una vez seleccionado el directorio se guarda para la próxima vez que se inicie el programa.

## 4.3 Documentación de ayuda - B4A Navegador de objetos

Este también es un programa independiente de Windows que muestra los archivos de ayuda de las librerías.

Ha sido escrito por Vader y se puede descargar aquí.

Un pdf con la documentación sobre cómo utilizar el programa es parte de la descarga.