

Programación con B4X

Breve teoría

Version 1.0, marzo 2021

Autor original: [Prokopis Leon](#)

Traducido al español por [José Miguel López](#)



Anywhere Software

Tabla de Contenidos

Tema 1 – Por qué B4X	9
Instalación de B4X	9
Tema 2 – El significado de Problema	10
El concepto de problema	10
Comprender el problema	10
Buscando una solución o un conjunto de soluciones.	10
Elección de la solución correcta.....	11
Implementación de la solución.....	11
Comprobar si esta solución obtuvo los resultados deseados.....	11
Tema 3 – Mi primer Programa.....	12
Hola Mundo.....	12
Ejercicios	13
Tema 4 - Variables y Rango.....	14
Cómo averiguar cuántas variables necesitas	15
Cómo dar nombre a las Variables.....	16
Declaración de Variables	16
Mi Primera Variable.....	16
Comentarios.....	17
El área de log y la función log	18
Operadores matemáticos.....	19
Cadenas	19
Ejercicios	21
Tema 5 – Diseñador	23
Primeros pasos en el diseño	24
Diseñador visual	25
El Árbol de Vistas	25
Propiedades.....	26
Diseñador Abstracto.....	26
Ejemplo 1	26
Decidir el tamaño de la ventana de tu aplicación.....	26
Establecer una variante adecuada	27
Diseña un esquema de tu pantalla.....	27
Crear las Vistas	27
Ejercicios	30
Tema 6 – Del Diseñador al Código	32



Class_Globals	32
Un estudio más profundo del uso de variables.....	33
Paso de Valores al Código	33
Eventos	34
Escribiendo código en el Evento.....	35
Propiedades.....	36
Ejercicios	37
Tema 7 – Sentencias Condicionales.....	38
Variables Lógicas o Booleanas	38
Operadores relacionales o de comparación	38
Operadores Lógicos	39
Operadores Lógicos en programación.....	40
Ejemplos de evaluación de sentencias lógicas.....	40
Sentencia If	41
If – Else.....	42
If – else - else if.....	42
Algoritmos con If.....	45
Ejercicios	45
Tema 8 – Subrutinas	48
.....	48
Crear una subrutina en B4J.....	48
Ejemplo 1.....	48
La memoria de una subrutina en B4X	50
Devolución de una valor desde una subrutina	51
Ejemplo 2	51
Ejercicios	52
Tema 9 – Clases	53
Clases	53
Ejemplo de clase en B4J	54
Metodología de implementación	54
Insertar un libro	55
Mostrar libro	55
Cambiar libro	55
La rutina “Initialize”	55
Cómo usar la Clase	56
Uso de los métodos.....	56

Ejercicios	57
Tema 10 – B4XPages.....	58
La estructura de las carpetas de una aplicación	58
Iniciar una aplicación con B4XPage	59
Qué es Root.....	59
Crear una nueva B4XPage.....	60
Invocar a una nueva B4XPage	62
Cerrar una B4XPage	63
Transferir información entre páginas	64
La Vida de las B4XPages.....	65
Ejercicios	66
Tema 11 – Primer Proyecto	67
Aplicación de tienda de móviles	67
Planificación	69
Borrador de las pantallas.....	70
Clases	71
B4XPages	72
Tema 12 – Bucles	73
La sentencia Do-While	73
Ejemplos de la sentencia Do While.	75
Bucles con un número de repeticiones desconocidas	77
La sentencia Do Until	78
Bucle For	81
Ejemplos de bucle For	82
Ejercicios	82
Tema 13 – XUI Views	84
Bibliotecas en in B4X	84
La Biblioteca “XUI Views”.....	85
Uso de XUI Views.....	86
Etiqueta desplazable	86
B4XFloatTextField.....	87
RoundSlider.....	87
AnotherProgressBar	88
B4XImageView.....	89
B4XDialogs	90
MsgBoxAsync.....	90

MsgBox2Async	90
Wait For.....	91
CustomDialog	91
Templates - Plantillas.....	93
B4XDateTemplate.....	93
B4XColorTemplate	94
B4XLongTextTemplate.....	95
Ejercicios	96
Tema 14 – Matrices - Arrays	97
Declaración de Arrays	97
Funciones sobre Arrays	98
Insertar elementos en un array.....	98
Mostrar los elementos de un array	99
Mostrar elementos en orden inverso	99
Calcular la suma y la media de los elementos de un array	99
Encontrar el máximo y el mínimo	99
Algoritmos de búsqueda	100
Búsqueda secuencial	100
Búsqueda Binaria	100
Ordenación.....	101
Ejercicios	104
Tema 15 – Listas	105
Creación de una lista	105
Insertar un ítem en una posición determinada	106
Eliminar un ítem de una posición determinada	106
Cambiar el valor en una posición concreta.....	106
Más métodos para trabajar con listas	107
Ejercicios	108
Tema 16 – Mapas	110
Un mapa es una colección que contiene pares clave-valor. Por ejemplo, las palabras de un diccionario o bien los registros de llamadas telefónicas. Algunas veces se les llama directamente diccionarios....	110
Crear un mapa.....	111
Insertar ítems en un Mapa	111
Inglés	111
Griego	111

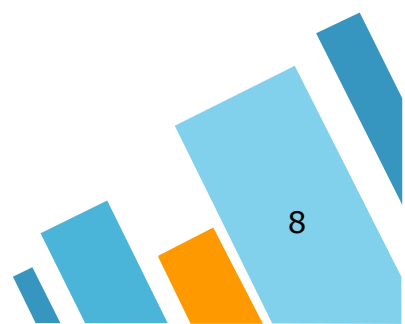


Inglés	111
Español	111
Memory	111
μνήμη	111
Memory	111
Memoria	111
Screen	111
οθόνη.....	111
Screen	111
Pantalla	111
Printer.....	111
εκτυπωτής	111
Printer.....	111
Impresora	111
Programming Language	111
Προγραμματισμός.....	111
Programming Language	111
Lenguaje de programación.....	111
Mapa 1 InglesGriego	111
Mapa 2 InglesEspañol	111
Usar el valor de un mapa.....	112
Índices en Mapas	112
La sentencia "for each"	113
Comprobar si existe una clave.....	113
Borrar una Clave y vaciar un Mapa.....	113
Ejercicios	114
Tema 17 – Ficheros.....	115
Carpetas o Directorios de almacenamiento	115
File.DirAssets	115
xui.DefaultFolder	115
File.DirData	116
File.DirApp	116
File.DirTemp	116
Crear carpetas/directorios	116
Comprobar la existencia de un fichero	116
Crear y escribir en un fichero	117



Lectura y escritura de datos	117
Variables tipo cadena de texto	117
Listas	117
Mapas	118
Ejercicios	119
Tema 18 – Tipos de datos complejos y Vistas	120
La sentencia “type”	120
Mostrar Ítems	121
Array de un Tipo	122
Listas de un Tipo	122
Mapas de un Tipo	123
Guardar en ficheros KVS	123
Declaración de un fichero KVS	123
Inicializar el fichero KVS	124
Insertar ítems en un fichero KVS.....	124
Recuperar ítems de un fichero KVS.....	124
Comprobar la existencia de un registro	125
Borrar un registro de un fichero KVS.....	125
Más Views.....	126
CustomListView.....	126
B4XComboBox	128
Ejercicios	130
Tema 19 – Proyecto Final	131
Crear una Aplicación de Biblioteca Escolar	131
Ficheros.....	132
Datos	133
Tipos	133
Diseño de la pantalla y Funciones	134
Hoja de pistas	135
Tema 20 – De B4J a B4A.....	137
¡Felicidades!	137
Descripción de la aplicación en B4J	137
Diseñador	137
Transferir la app a B4A y Android.....	140
Transferir el diseño.....	140
Instalar la app en tu móvil	141





Tema 1 – Por qué B4X

🕒 1h

Muchos programadores nuevos se preguntan en qué lenguaje de programación invertir su tiempo y esfuerzo. Cada lenguaje de programación tiene ventajas pero también desventajas que deben tenerse en cuenta. A menudo, la selección de un idioma también determina la evolución posterior del desarrollador. Más aún, cuando se trata de utilizar el lenguaje con fines educativos, hay elementos individuales que deben tenerse en cuenta.

Así pues, el lenguaje de programación elegido debe ser:

- Moderno y estructurado.
- Ser fácil de aprender para niños y nuevos programadores.
- Proporcionar un entorno de desarrollo integrado sin confusión.
- Permitir al alumno desarrollar aplicaciones en diferentes plataformas como Windows, Android, IOS, Linux, Raspberry Pi, Arduino etc.
- Proporcionar todas las estructuras de datos modernas como listas, mapas, etc.
- Provide all modern data structures as lists, maps etc.
- Mantener el interés de los estudiantes brindándoles la posibilidad de desarrollar diferentes tipos de aplicaciones, por ejemplo, juegos, etc.

El lenguaje de programación **B4X** proporciona todas las características anteriores y también es un lenguaje para desarrollar aplicaciones de alto nivel que pueden ser un trampolín para el desarrollo profesional futuro. Además, cuenta con un público muy entusiasta que con mucho gusto ayuda en cualquier tipo de duda.

Para mas información, se puede visitar el sitio web: <https://www.b4x.com/learn.html>

Instalación de B4X

La información más actualizada sobre cómo instalar B4X está siempre aquí: <https://www.b4x.com/b4j.html>



Tema 2 – El significado de Problema

🕒 1h

Lo que los estudiantes aprenderán

- ¿Qué es un problema?
- ¿Qué son los datos?
- ¿Qué es información?
- Qué pasos deben tomarse para resolver un problema.
- Qué es el algoritmo.

El concepto de problema

Todos los días en nuestras vidas tenemos problemas. Problemas simples de la vida cotidiana y, a menudo, incluso mayores. Como problema solemos definir una situación que estamos viviendo y que nos dificulta afrontarla o solucionarla (Cambridge, 2021).

Cuando pensamos en un problema, estos son los pasos que se vuelven inconscientes en nuestra mente:

- Comprender el problema
- Buscar una solución o un conjunto de soluciones.
- Elegir la solución adecuada.
- Implementar la solución.
- Verificar si esta solución tuvo los resultados deseados.

Comprender el problema

Comprender el problema es el primer paso para diseñar y resolver un problema. Es una etapa particularmente crítica porque esto también afectará el desarrollo de la solución. Incluye los siguientes pasos:

1. Descripción del problema

La descripción del problema generalmente la hacemos nosotros mismos o el que lo tiene. En este paso es importante aclarar todos sus puntos 'oscuros' y no tener dudas sobre su redacción.

2. Encontrar los datos

Encontrar los datos significa en qué se basan estos elementos para resolver un problema. Por ejemplo, en una ecuación $ax + b = 0$, los datos son los factores a y b .

3. Encontrar lo que se pide

La información que necesitamos encontrar para solucionar el problema. Continuando con el ejemplo anterior, la información es la x de la ecuación $ax + b = 0$.

Buscando una solución o un conjunto de soluciones.

Una vez que hemos reconocido los datos y lo que se solicita, se debe buscar una solución para solucionar el problema. A menudo eso no es

fácil. Por lo tanto, es necesario buscar un método o **un conjunto lógico de pasos que conduzcan a la solución**. Estos pasos deben conducir a una solución **siempre** que surja el mismo problema y, además, deben realizarse en **un período relativamente corto** de tiempo.

En matemáticas y ciencias de la computación, un algoritmo es una secuencia finita de instrucciones bien definidas que se pueden implementar por computadora, típicamente para resolver una clase de problemas o realizar un cálculo.

Elección de la solución correcta

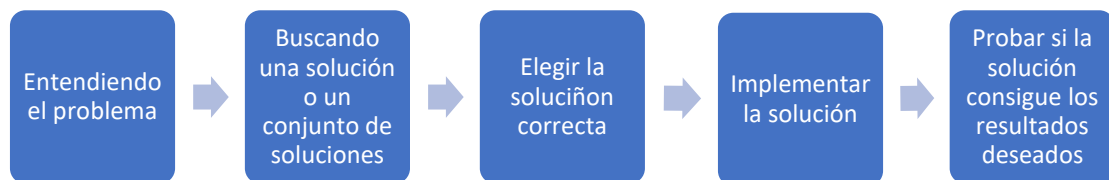
A menudo, un problema puede tener más de una solución o algunas de ellas pueden ser más eficientes que otras. Elegir la solución adecuadamente conduce a un resultado más corto y confiable.

Implementación de la solución

Una vez seleccionado el método de resolución, se deben tomar una serie de acciones para implementarlo. En otras palabras, aplicar **el Algoritmo**.

Comprobar si esta solución obtuvo los resultados deseados

Finalmente, tras aplicar la solución es necesario probar con varios datos, para examinar si conduce a resultados correctos cada vez que se realiza sin situaciones problemáticas.



Dibujo 1. Los pasos

Tema 3 – Mi primer Programa

🕒 2h

Lo que los estudiantes aprenderán

- Cómo iniciar B4J
- Cómo crear y guardar un nuevo proyecto
- Cómo ejecutar un proyecto
- Qué es una pantalla de error
- Cómo usar los comandos Turtle
- Cómo escribir un nuevo proyecto que use Turtle

Hola Mundo



Haz un programa que dibuje una línea recta con la ayuda de la tortuga en la pantalla de la computadora.

Instrucciones recomendadas para el profesor

El objetivo del ejercicio anterior es familiarizar a los estudiantes en la creación de un proyecto con B4J y tener el primer contacto con un entorno de programación. Las siguientes instrucciones no limitan en modo alguno al profesor a adaptar su curso a las condiciones educativas particulares.



Consejo para el profesor

Este es el primer tema, así que ¡haz que sea sencillo!

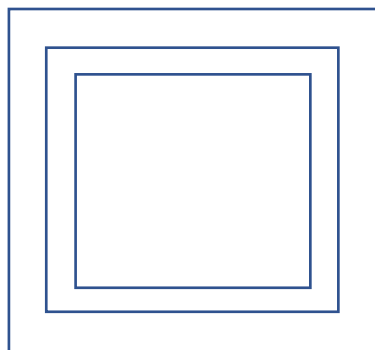
Función	Descripción
1 Cómo crear tu primer proyecto con B4J	<i>Menú Archivo -> Nuevo -> B4Xturtle</i> <i>Carpeta del proyecto</i> <i>Nombre del proyecto</i> Explique a los estudiantes la importancia de los nombres correctos en cada proyecto que crean y el valor del almacenamiento correcto en carpetas de una manera estructurada
2 Ejecutar un Proyecto	<i>Menú Proyecto -> Compilar y Ejecutar</i> Explique qué significa compilar y cómo reconocer errores de sintaxis en el registro. No es necesario que proporcione mucha información sobre la compilación. Solo lo básico para ejecutar un proyecto.
3 #Region Project Attributes	Cambie los valores en <i>MainFormWidth</i> y <i>MainFormHeight</i> para hacer una aplicación de diferente tamaño.



Función	Descripción
4 Sub Turtle_Start	¿Qué significa Sub? Una pequeña cantidad de código que está realizando una determinada actividad.
5 Métodos Turtle	¿Qué es Turtle? ¿Qué hace .MoveForward? ¿Cómo podemos encontrar más comandos? (Dícales a los estudiantes que escriban "Turtle". Para ver la lista de métodos.
6 Errores	Cómo identificar un error en la pantalla de registro (Log)

Ejercicios

1. Usa la tortuga y los métodos **MoveForward**, **TurnLeft**, **TurnRight** para dibujar un cuadrado del tamaño que quieras.
2. Usando los comandos anteriores y **PenUp**, **PenDown** y **Move** dibuja 3 cuadrados como la imagen de abajo.



3. Usando los comandos anteriores, haz el dibujo que quieras. Dale un nombre a tu dibujo y explica cómo lo hiciste.

Tema 4 - Variables y Rango

🕒 3h

Lo que los estudiantes aprenderán

- Explicar cómo un ordenador almacena datos en la RAM
- Explicar qué es una variable
- Cómo nombrar una variable
- Asignar un valor a una variable
- Usar operadores matemáticos
- Usar el comando Log para mostrar una variable

Imagínate que vives en una calle con varios millones de casas seguidas; cada casa tiene su dirección que comienza en el número 1 y termina con el número de la última casa; para poder localizar a un amigo que vive en esa calle es necesario conocer el número de la casa; así que tenemos por un lado un número de casa y por otro el amigo que vive en esa casa.



Figure 1 Computer Memory (<https://www.freepik.com>)

La memoria central de la computadora funciona de la misma manera. Hay muchas casas cada una con su dirección y un "residente" dentro de cada casa. Esta dirección se denomina **dirección de memoria** y al "residente" se le llama contenido. En el ordenador muchas veces el "residente" (que llamaremos **variable**) necesita varias casas para poder tener hueco.

Para que un programador utilice la memoria, debe conocer los datos que necesita y el tipo de datos. Estos pueden ser números enteros o reales, palabras o letras o valores lógicos (verdadero o falso). También necesita un "hogar" en la memoria de la computadora para almacenarlos representados por la dirección.

En B4X los datos se pueden almacenar en diferentes tipos como:

B4X	Tipo de dato
Boolean	Booleano o lógico (verdadero/falso)
Byte	Entero de 8 bits (de 0 a 255)
Short	Entero de 16 bits (0 a 65535)
Int	Entero de 32 bits (0 a 4.294.967.296)
Long	Entero largo de 64 bits (0 a 2^{64})
Float	Número decimal de 32 bits
Double	Número decimal de doble precisión de 64 bits



Char	Carácter ('a', '1', '%', etc.)
String	Cadena de caracteres ("hola", "adiós", "juan",...)

Tabla 1- Tipos básicos de las variables

Cada tipo necesita un espacio diferente en la memoria para almacenar su contenido.

Debido a que es difícil para el desarrollador recordar todas las direcciones de sus datos, a cada dirección se le pone un nombre. Por suerte, esto lo hace el propio lenguaje de programación y todo lo que se necesita es pensar en un buen nombre para tus datos. Por ejemplo, un dato que sea un número entero para la edad podría llamarse "edad". Ahora, hay un "hogar" llamado edad en la memoria de la computadora.



Recuerda

Las variables se utilizan para almacenar información para ser referenciada y manipulada en un programa. También proporcionan una forma de etiquetar los datos con un nombre descriptivo, para que el lector y nosotros mismos podamos entender nuestros programas con mayor claridad. Es útil pensar en las variables como contenedores que contienen información.

Cómo averiguar cuántas variables necesitas

En cualquier problema de programación que encuentre un desarrollador, debería poder ubicar los datos y la información del problema.

En programación debemos dar un nombre a todos aquellos elementos que necesitamos conocer para resolver un problema. Normalmente en un problema de programación los podemos extraer del enunciado del ejercicio con la ayuda de verbos clave como:

- Leer
- Registrar
- Preguntar
- Aceptar
- Teclear

Ejemplo 1: Escribe un programa que convierta los euros que escribimos en dólares.

Ejemplo 2: Haga un programa que acepte un número entero positivo y calcule su cuadrado, cubo y raíz cuadrada.

Información

La Información en un programa es la transformación que tenemos que aplicar a los datos para conseguir un resultado deseado. Normalmente la manera de obtener la información la podemos obtener del enunciado del problema fijándonos en palabras clave como:

- Calcular
- Mostrar



- Escribir
- Contar
- Convertir

¿Qué debemos calcular en los ejemplos anteriores?

Cómo dar nombre a las Variables

Los nombres de las variables en B4X deben seguir estas reglas:

- Deben comenzar con mayúscula o minúscula.
- Después, pueden tener dígitos o el carácter de subrayado (_).
- B4X no distingue entre mayúsculas y minúsculas.

Además, es una buena práctica poner 3 letras delante del nombre de la variable indicando el tipo de variable que es y continuar con 1 letra mayúscula y una palabra significativa. Por ejemplo:

- Dim **intEdad** as Int
- Dim **fltCantidad** as Float
- Dim **strNombre** as String

Esto te ayudará mucho cuando encuentre una variable en el código pudiendo así reconocer el tipo y el valor que almacena.

Declaración de Variables

Mi Primera Variable



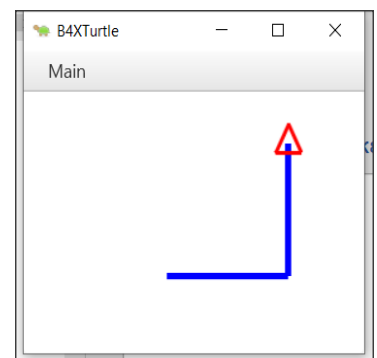
Escribe un programa que asigne un valor a un número entero y luego dibuje con la ayuda de la tortuga una línea de longitud tan larga como el valor de la variable.

En B4X, para usar una variable, primero debemos informar al lenguaje de su existencia para que le asigne espacio en la memoria del ordenador para almacenar su valor.

Por ejemplo, en el siguiente código, la declaración es la siguiente:

```
'Programa: Mi Primera Variable
Sub Turtle_Start
  Private intDistancia As Int
  Public intGiro As Int
  intDistancia = 100
  intGiro = 90

  Turtle.SetPenColor(xui.Color_Blue).SetPenSize(5)
  Turtle.MoveForward(intDistancia)
  Turtle.TurnLeft(intGiro)
  Turtle.MoveForward(intDistancia)
End Sub
```



La declaración de variables comienza con la palabra clave **Private** o **Public**.

Private significa que la variable se conoce solo en el espacio específico declarado y ningún otro programa o subprograma conoce su existencia y, por lo tanto, el valor que contiene.

En cambio, una declaración de variable que comienza con la palabra clave **Public** puede ser conocida por otros programas, subprogramas o clases, etc.

Después de la palabra clave **Private** o **Public** sigue el nombre de la variable. Aquí es donde se aplican las reglas discutidas anteriormente. Finalmente, sigue el tipo de variable. Para variables simples, los tipos son todos los descritos en el *Tabla 1- Tipos básicos de las variables*.



Consejo para el profesor

No es necesario que explique todas las variables ni su uso. Para que sus estudiantes comiencen a programar, los conceptos básicos de integer, float, string son suficientes. A medida que avanza en los cursos, puede incluir otros tipos según sus necesidades.

Comentarios

En programación un comentario es una explicación o anotación que se escribe en el código fuente de un programa. Se agregan con el propósito de hacer que el código fuente sea más fácil de entender para los humanos y generalmente son ignorados por compiladores e intérpretes. La sintaxis de los comentarios en varios lenguajes de programación varía considerablemente. (Wikipedia, 2021)

En B4X, los comentarios se insertan poniendo el carácter ' (comilla simple) antes del comentario. Ese carácter hace que se ignore todo lo que viene a continuación. Generalmente, los comentarios se suelen poner en lugares donde es importante recordar lo que se está haciendo. Los comentarios se distinguen fácilmente en el código por el color verde que les da el entorno de programación (IDE).

Ejemplo

```
'Programa: Mi Primera Variable
'Este programa dibuja un ángulo recto cuyos lados son tan largos
'como el valor de la variable intDistancia
Sub Turtle_Start
  Private intDistancia As Int
  Public intGiro As Int
  intDistancia = 100      'Longitud de los lados del ángulo recto
  intGiro = 90           '90 grados del ángulo recto

  Turtle.SetPenColor(xui.Color_Blue).SetPenSize(5)
  Turtle.MoveForward(intDistancia)
  Turtle.TurnLeft(intGiro)
  Turtle.MoveForward(intDistancia)
```

El área de log y la función log

Cuando se programa se suelen producir errores. Generalmente, los errores en la programación se dividen en dos categorías: **sintácticos** y **lógicos**. Por ahora trataremos los errores de sintaxis que son reconocidos por el lenguaje de programación y los indicaremos en la pantalla de logs. Para acceder a la pantalla de Logs, debemos hacer clic en la **pestaña de Log** en la parte inferior derecha.

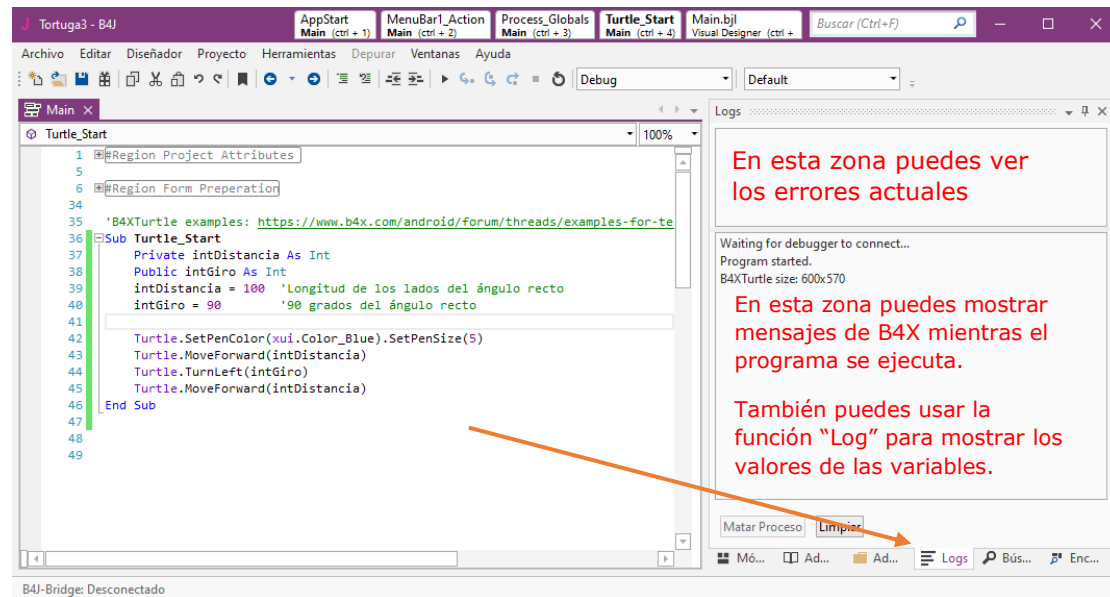


Figura 2. Pantalla de Log

La pantalla de Logs está dividida en dos zonas; en la superior se muestran los errores y en la inferior los mensajes generados por el propio lenguaje B4J o bien aquella información que nosotros hemos dicho que queremos mostrar con la función **Log()**. El uso de la función **Log ()** te ayuda a mostrar mensajes mientras se ejecuta un programa, así como valores de variables para ayudar a controlar el correcto funcionamiento del programa.

Para mostrar cualquier información en la pantalla es suficiente usar la función **log ()** como en la siguiente imagen.

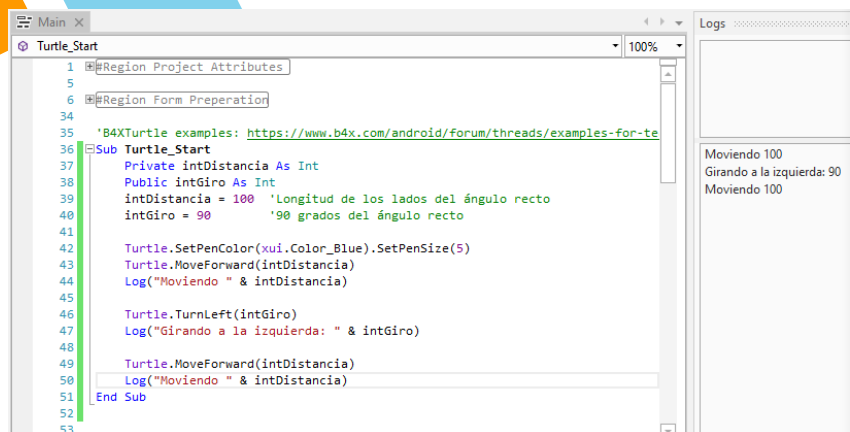


Figura 3. Uso de la función Log

Operadores matemáticos

B4X admite todas las **operaciones** matemáticas más comunes:

Operador	Ejemplo	Operación
+	$x + y$	Suma
-	$x - y$	Resta
*	$x * y$	Multipliación
/	x / y	División
Mod	$x \text{ Mod } y$	Resto de la división
Power	$\text{Power}(x,y) \quad x^y$	Potencia

Ejemplos:

```

Private intA, intB, intC, intS As Int
Private fltD, fltM As Float
intA = 40
intB = 20
intC = 30

intS = intA + intB + intC
Log(intS)           'Muestra 90

fltD = intS / 3
Log(fltD)           'Muestra 30

intA = intA + 1
Log(intA)           'Incrementar intA en 1
                    'Muestra 41

intS = Power(intA - 11, 2)
Log(intS)           ' 30²
                    'Muestra 900 (30*30)

fltM = intA mod 2
Log(fltM)           'Resto de dividir 41 entre 2
                    'Muestra 1

```

Cadenas

En programación de ordenadores, una cadena es tradicionalmente una secuencia de caracteres, ya sea una constante literal o una variable. En este último caso se puede cambiar su contenido y también su longitud, o puede ser de longitud fija (después de crearla) (Wikipedia, Wikipedia - Strings, 2021).

Una cadena se declara como las otras variables usando la declaración **String**

```
Private strNombre as String
```

La asignación de valor a una cadena se puede hacer con el símbolo = o leyendo un valor que escribe el usuario (algo que veremos más adelante).

```
Private strNombre, strApellidos as String  
strNombre = "Juan"  
strApellidos = "García Gómez"
```

También podemos unir dos cadena usando el carácter &

```
Private strNombre, strApellidos as String  
strNombre = "Juan"  
strApellidos = "García Gómez"  
Private strPersona as String  
  
strPersona = strNombre & " " & strApellidos  
log(strPersona) ` Muestra Juan García Gómez en la pantalla de Log  
  
Private strNombre2 as String  
strNombre2 = "Antonio"  
strNombre2 = strNombre2 & " López"
```

También hay muchas funciones para trabajar con cadenas que son muy útiles:

CharAt (Índice)	Devuelve el carácter en el índice dado.
CompareTo (Otra)	Compara lexicográficamente la cadena con la Otra cadena.
Contains (SearchFor)	Comprueba si la cadena contiene la cadena SearchFor dada.
EndsWith (Sufijo)	Devuelve True si la cadena termina con la subcadena de sufijo dada.
EqualsIgnoreCase (Otra)	Devuelve True si ambas cadenas son iguales ignorando si están en mayúsculas o minúsculas.
Length	Devuelve la longitud, el número de caracteres, de la cadena.
Replace (Destino, Reemplazo)	Devuelve una nueva cadena resultante de la sustitución de todas las apariciones de Destino con Reemplazo.
StartsWith (Prefijo)	Devuelve True si esta cadena comienza con el prefijo dado.
ToLowerCase	Devuelve una nueva cadena que es el resultado de ponerla en minúsculas.
ToUpperCase	Devuelve una nueva cadena que es el resultado de ponerla en mayúsculas.
Trim	Devuelve una copia de la cadena original sin espacios en blanco iniciales o finales.

Tabla 2. Funciones de cadena (<https://www.b4x.com/android/documentation.html>)



Consejo para el profesor

Puedes encontrar más información sobre manipulación de cadenas en los folletos sobre el lenguaje en:

<https://www.b4x.com/android/documentation.html>

Ejercicios

1. En los siguientes ejercicios, identifica las variables que necesita declarar. Para cada uno de ellas, escriba la declaración y asígnele un nombre apropiado.
 - Calcula el volumen de un cilindro con un radio de un metro y una altura de dos metros.
 - Haga un programa que acepte un número entero positivo y calcule su cuadrado, cubo y raíz cuadrada.
 - Haga un programa que lea una suma de dinero en € y calcule y muestre la cantidad correspondiente en \$.
 - Escriba un programa que lea la longitud de los lados de un rectángulo desde el teclado y calcule y muestre su área.
 - La resistencia total R de dos resistencias $R1$ y $R2$ conectadas en serie es $R1 + R2$ y paralelo $(R1 * R2) / (R1 + R2)$ respectivamente. Crea un programa que lea dos valores de resistencia $R1$ y $R2$ y calcule la resistencia total en serie y en paralelo y la muestre con la función "Log".

2. En los siguientes nombres de variables, seleccione cuáles son correctas y cuáles no:

Nombre	Correcto	Incorrecto
int Edad	<input type="checkbox"/>	<input type="checkbox"/>
_fltCantidad	<input type="checkbox"/>	<input type="checkbox"/>
strNombre	<input type="checkbox"/>	<input type="checkbox"/>
1miEdad	<input type="checkbox"/>	<input type="checkbox"/>
int_valor	<input type="checkbox"/>	<input type="checkbox"/>

3. Es el final del trimestre y obtuviste tus calificaciones de tres clases: Geometría, Álgebra y Física. Crea un programa que:
 - Guarde en 3 variables las calificaciones de estas 3 clases (las calificaciones van de 0 a 10)
 - Calcule la nota media de sus calificaciones y la muestre con la función "Log".



4. ¡Has comprado un Bitcoin y ahora está subiendo! Crea un programa que:

- Asigne el valor del bitcoin en el momento de la compra.
- Asigne el porcentaje de aumento (o disminución)
- Emplee la función "Log" para ver el valor total de tu bitcoin.
- Emplee la función "Log" para ver el valor de aumento o disminución.

5. Tienes una casa y deseas calcular su área total. Crea un programa que:

1. Lea el ancho y el alto en dos variables.
2. Calcule y muestre con la función "Log" el área total.

6. Quieres comprar un nuevo portátil. Miras el precio y ves que el precio es de 300 euros sin incluir el IVA del 21%. Crea un programa que:

1. Asigne el precio del portátil en una variable.
2. Asigne el porcentaje de impuestos en una segunda variable.
3. Calcule y muestre con la función "Log" el precio final con IVA.

7. En una empresa, el salario mensual de un empleado se calcula partiendo del salario mínimo de 400 € al mes, más 20€ multiplicado por el número de años empleados, más 30€ por cada hijo que tenga. Cree un programa que:

1. Asigne el número de años del empleado en una variable.
2. Asigne el número de hijos que tiene el empleado en la segunda variable.
3. Calcule y muestre con la función "Log" el salario del empleado.

8. Crea un programa que use la función "Log" para mostrar el último dígito de un número entero dado.

9. Crea dos variables 'a' y 'b' y ponles un valor inicial diferente a cada una. Escribe un programa que intercambie ambos valores.

Ejemplo: a = 10, b = 20

Salida: a = 20, b = 10

10. Crea dos variables 'a' y 'b' y ponles un valor inicial diferente a cada una. Escribe un programa que doble el valor de la variable 'a' e incremente el valor de la variable 'b' en 1.

Ejemplo: a = 10, b = 20

Salida: a = 20, b = 21

Tema 5 – Diseñador

🕒 2h

Lo que los estudiantes aprenderán

- Hablar sobre el diseñador
- Diseñar la primera pantalla.
- Insertar y personalizar vistas: etiquetas, campos de texto, botones, paneles
- Guardar formularios
- Diseñar tu propia pantalla principal utilizando esquemas

Hasta ahora, hemos usado la tortuga para moverla por la pantalla y la función “Log” para mostrar información en la pantalla de Log de B4X. ¿Qué pasa si le pide al usuario que introduzca un valor? ¿O qué sucede cuando se desea mostrar información al usuario? B4X tiene un entorno de diseño de interfaces de usuario especial. A través de él puedes diseñar el aspecto de las pantallas y comunicarte con los usuarios de tu aplicación.

Cada vez que debas diseñar una aplicación, debes tener en cuenta que el aspecto visual de tu aplicación es lo que atraerá a los usuarios. Es decir, no basta con que haga lo que tiene que hacer, sino que tiene que ser fácil de usar y ofrecer información de forma organizada sin confundir.

Antes de diseñar una aplicación, ten en cuenta estos principios de diseño (usability.org, 2021):

Manten la interfaz simple. Las mejores interfaces son casi invisibles para el usuario. Evitan elementos innecesarios y son claros en el lenguaje que utilizan en las etiquetas y en los mensajes.

Sea coherente y utilice elementos concidos en la interfaz de usuario. Al utilizar elementos comunes en su interfaz de usuario, los usuarios se sienten más cómodos y pueden hacer las cosas más rápidamente.

Usa color y textura correctamente. Puedes dirigir o desviar la atención sobre los elementos usando el color, la luz, el contraste y la textura.

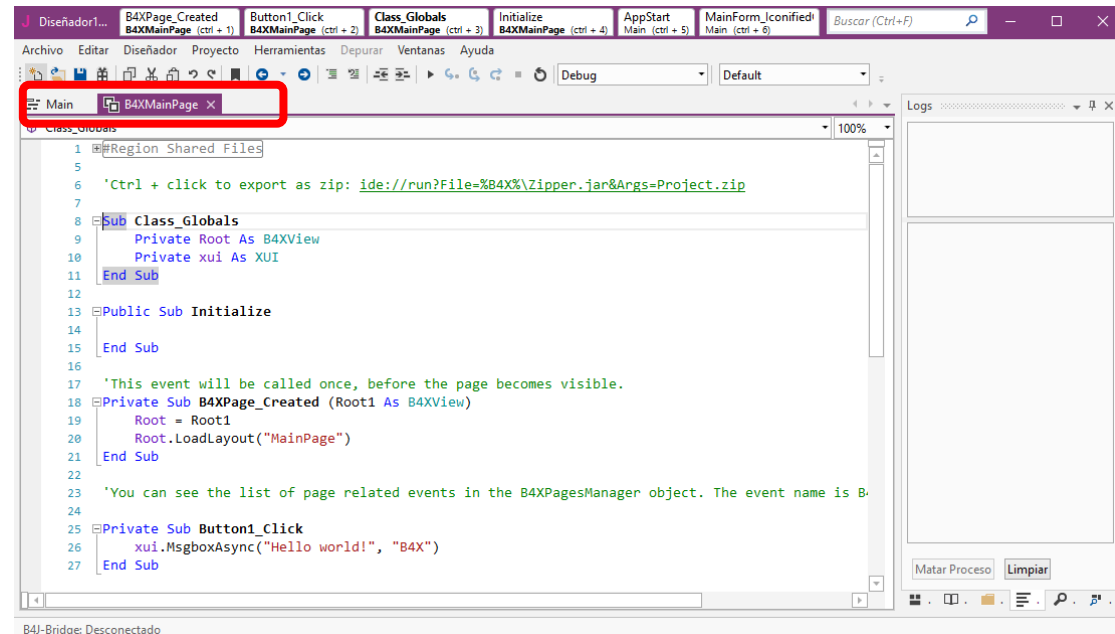
Utilice la tipografía para crear jerarquía y claridad. Considere cuidadosamente cómo usa la tipografía. Diferentes tamaños, fuentes y disposición del texto para ayudar a aumentar la escalabilidad y la legibilidad.

Asegúrese de que el programa comunique lo que está sucediendo. Informe siempre a sus usuarios sobre la ubicación, las acciones, los cambios de estado o los errores.

Piensa en los valores por defecto. Al pensar detenidamente y anticiparse a lo que hacen tus usuarios, puedes crear valores por defecto representativos. Esto es muy importante cuando se trata del diseño de formularios, ya que los campos pueden estar rellenos previamente.

Primeros pasos en el diseño

En primer lugar, arranca B4J y desde el menú de **Archivo**, elija **Nuevo** y **B4XPages**. Elija un directorio y escriba un nombre para su proyecto. Verá el código a continuación. Verás dos pestañas de código, la primera llamada **Main** y la segunda **B4XMainPage**.



No te preocupes por estas pestañas ahora. Hablaremos más adelante sobre ellas. ¡Ahora todo lo que necesitas saber es que dentro de B4XMainPage suceden todas las cosas interesantes de nuestro código!

Ahora desde el menú **Diseñador** elige **Abrir Diseñador Interno**.

Aquí comienza el proceso de diseño. Se abrirán dos ventanas, la primera es el diseñador y la segunda es la vista previa de la pantalla que se está diseñando.

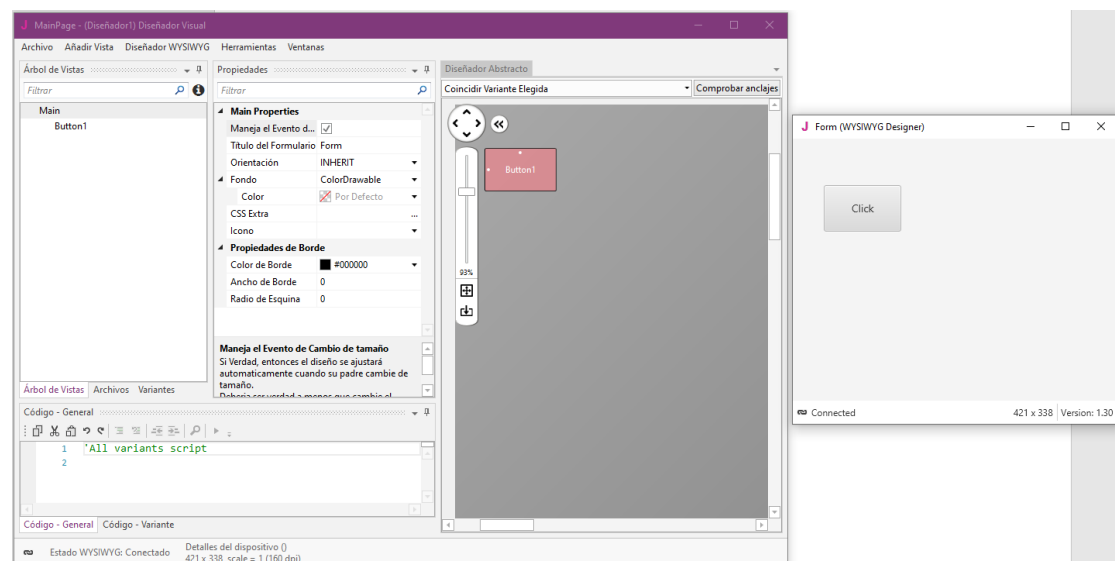


Figura 2. La pantalla del Diseñador

Diseñador visual

El menú **Añadir Vista** incluye todos los objetos necesarios para crear nuestra pantalla.

Elige la opción **Label** desde el menú **Añadir Vista** y muévelo dentro de la ventana **Diseñador Abstracto** donde quieras:



Recuerda

Puedes mover todos los objetos seleccionándolos y arrastrándolos con el ratón.

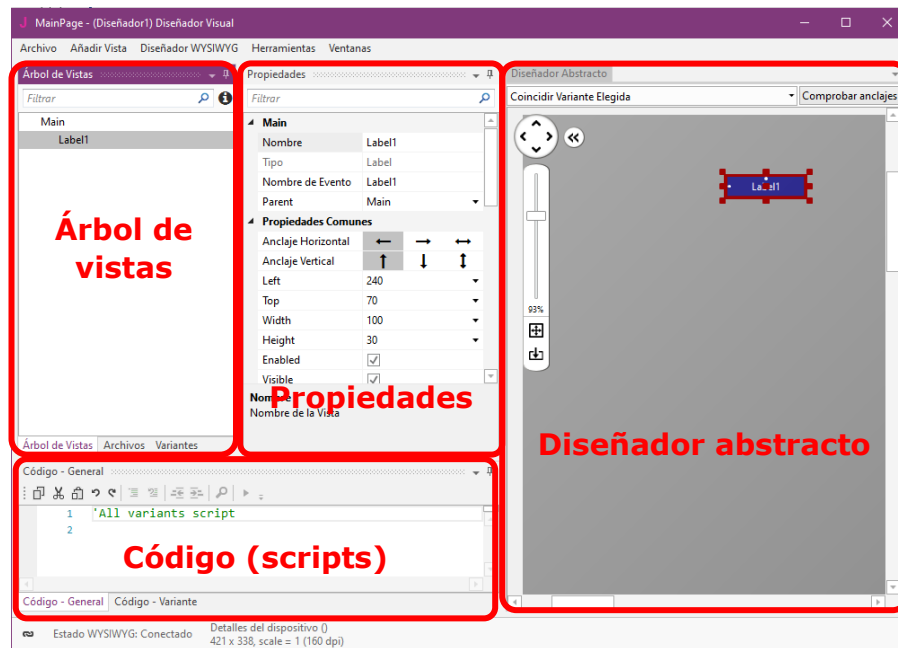


Figura 3. Zonas del Diseñador

El Árbol de Vistas

Aquí puedes ver todos los objetos de tu diseño. Ten en cuenta que los objetos se dibujan en pantalla empezando por el primero, con lo que los que aparecen en primer lugar quedan ocultos por los de más abajo:

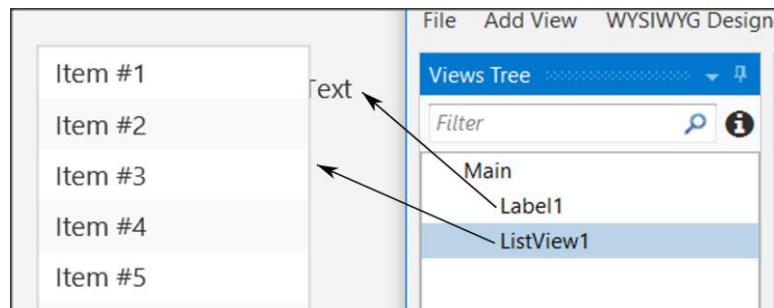


Figura 4. Árbol de vistas. Fíjate que "Label1" queda por debajo de "ListView1"

Propiedades

Cada objeto tiene sus propiedades como tamaño, posición en pantalla, colores, tipo de letra, etc. Cada propiedad se puede cambiar dentro de la ventana de propiedades o posteriormente a través del código del programa.

Una de las propiedades más importantes es el **nombre del objeto**. Al igual que con las variables, debes seguir unas reglas para indicar su tipo. Por ejemplo, en la *Tabla 3. Nombrar objetos* mostramos algunos casos:

Tipo	Prefijo	Ejemplo
Label	lbl	lblNombre
Button	btn	btnGuardar
TextField	txt	txtEdad
Spinner	spn	spnAños
Pane	pn	pnLínea1

Tabla 3. Nombrar objetos

Diseñador Abstracto

El Diseñador Abstracto permite seleccionar la posición y cambiar el tamaño de las Views (vistas). Es una función muy útil para colocar rápidamente objetos en la posición correcta (sin embargo, para una ubicación más precisa es mejor usar la pestaña Propiedades).

Ejemplo 1



Imagina que quieres hacer un programa que lea en pantalla dos enteros y que calcule y muestre su suma.

Decidir el tamaño de la ventana de tu aplicación

Dependerá de la cantidad de información que debemos mostrar, así como de los elementos individuales como menús, gráficos, etc.

El tamaño de la aplicación debes cambiarlo **antes de iniciar el Diseñador**. Para hacerlo, vete a la pestaña **Main** y cambia las primeras líneas del código Width y Height:

```
#Region Project Attributes
    #MainFormWidth: 600
    #MainFormHeight: 400
#End Region
```

Guarda tu proyecto y abre el **Diseñador Interno**.



Establecer una variante adecuada

Una variante es una versión de tu aplicación con una resolución diferente a la que tienes por defecto.

Por lo general, debes establecer las variantes con los parámetros **MainFormWidth** y **MainFormHeight**. Esto te ayudará a diseñar sin arriesgarte a salirte de la pantalla.

Para crear una variante, elige la pestaña **Variantes** y luego **Nueva variante** y escribe el ancho y el alto.

Puedes tener tantas variantes como quieras para diferentes tamaños de pantalla, pero por ahora, sólo usaremos una. Puedes eliminar cualquier variante seleccionándola y eligiendo **Eliminar Seleccionado**.

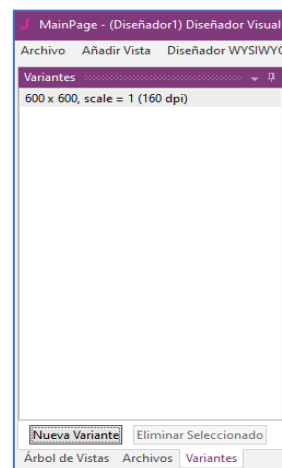


Figura 5. Pantalla de Variantes

Diseña un esquema de tu pantalla

Para aplicaciones pequeñas, este paso es opcional, pero es una buena costumbre haber decidido desde el principio dónde quieres mostrar tus datos. Puede utilizar una simple hoja de papel o varios programas para ayudar a crear esquemas.

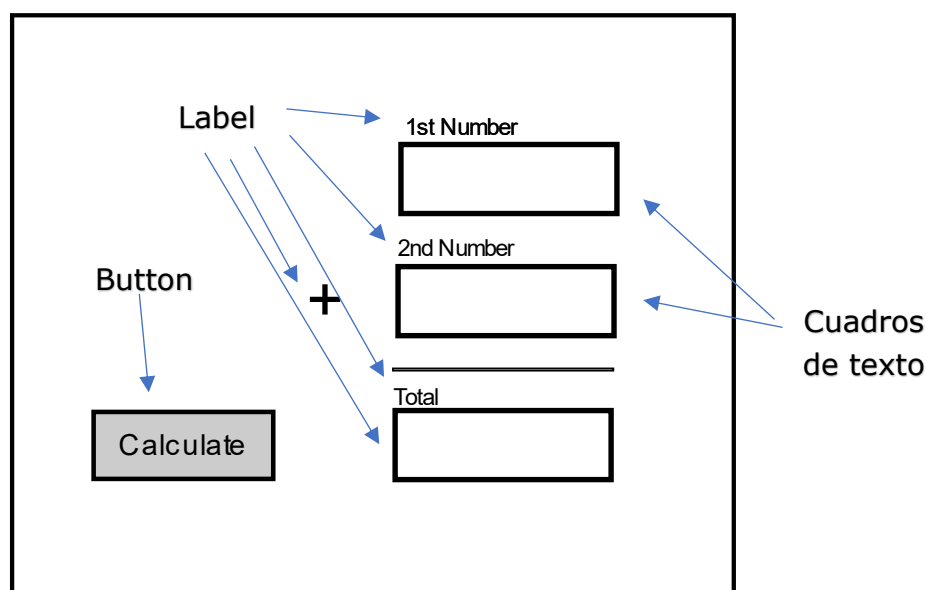


Figura 6. Esquema

Crear las Vistas

Ahora que ya sabes lo que necesitas y dónde colocarlo, vamos a usar las herramientas del Diseñador para completar el diseño de la interfaz.

Insertar una etiqueta (label)

En el menú **Añadir Vista**, seleccionamos **Label** y aparecerá un objeto de etiqueta en tu **Árbol de vistas** y en el **Diseñador Abstracto**. Muévelo en el lugar que quieras según tu esquema de la pantalla y elige un nombre adecuado dentro de Propiedades.

Ahora desplázate hacia abajo en Propiedades y fija estos valores:

- **Width:** 180
- **Height:** 30
- **Text:** Primer Número
- **Alineación:** CENTER_LEFT
- **Font:** SansSerif
- **Tamaño:** 13

Experimenta las propiedades y observa cómo queda en el panel de previsualización.

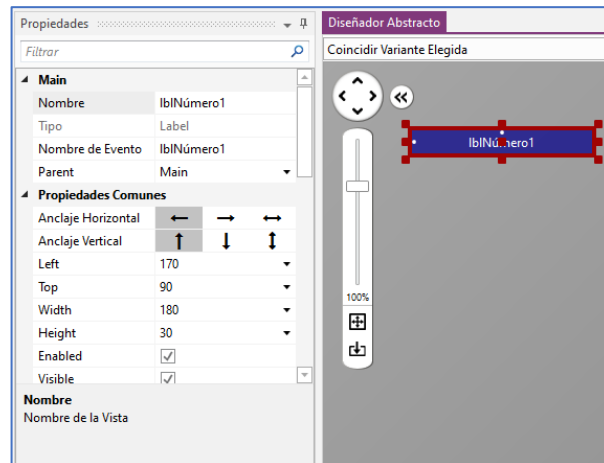


Figura 7. Etiquetas

Inserta una segunda etiqueta o duplica la primera. Selecciónala y presiona Ctrl-D. El segundo método genera una segunda etiqueta exactamente igual a la primera, excepto en la propiedad de nombre. Fija el nombre en "lblNúmero2" y "Segundo número" como **Text**. Crea una tercera etiqueta con el nombre "lblTotal" y **Text**: "Total".

Insertar un campo de texto (TextField)

Los campos de texto se utilizan para introducir datos al programa. No hay restricciones sobre el tipo de datos que puede leer. Desde el menú **Añadir Vista**, elige **TextField** y configúralo así:

- **Nombre:** txtNúmero1
- **Width:** 180
- **Height:** 40
- **Font:** SansSerif
- **Bold:** marcado

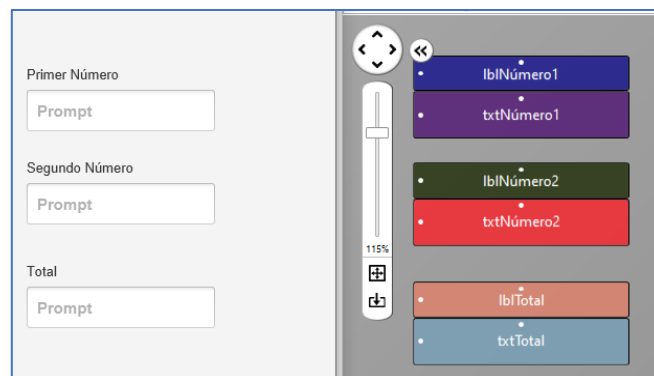


Figura 8. Campos de texto

Coloca el TextField debajo de la etiqueta "Primer número". Crea otro TextField con el nombre "txtNúmero2" y colócalo debajo de la etiqueta "Segundo número". Al final, crea un tercer TextField con el nombre "txtTotal" y colócalo debajo de la etiqueta "Total". Debe quedarte algo similar a la **Figura 8. Campos de texto**

Insertar un botón (button)

Los botones se utilizan para realizar acciones. El programa detecta el clic y luego ejecuta los comandos apropiados según el botón presionado.

Para cada botón, puedes configurar diferentes propiedades como tamaño, color, forma, etc. para que se destaque en pantalla y los usuarios lo vean fácilmente.

Para añadir un botón, haz clic en **Añadir Vista**, elige **Button** y déjalo así:

- **Nombre:** btnCalcular
- **Width:** 150
- **Height:** 40
- **Color de borde:** #3C0000
- **Ancho de borde:** 2
- **Radio de esquina:** 20
- **Text:** Calcular
- **Color de texto:** #FF3C0000
- **Font:** SansSerif
- **Tamaño:** 15
- **Bold:** marcado

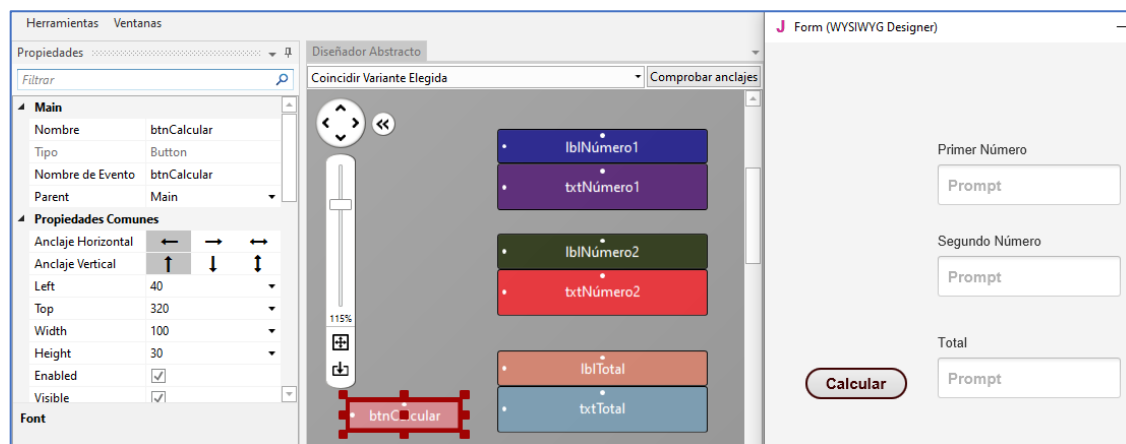


Figura 9. Botones



Recuerda

Archivo -> Salvar (o Ctrl – S) cada vez que hagas algo importante.

Insertar un panel

Puedes usar un panel para agrupar visualmente objetos en la pantalla. El panel está rodeado de un marco y tiene propiedades como el color, borde, relleno, etc. También puedes ponerle una altura muy pequeña (1 o 2) para que sea una línea en pantalla.

En este ejemplo usaremos un panel para dibujar una línea antes del total. Haz clic en **Añadir Vista**, elige **Pane** y configúralo así:

- **Name:** pnLínea
- **Width:** 180
- **Height:** 1
- **Color de borde:** #000000
- **Ancho de borde:** 2

Ahora, elige la opción **Archivo→Salvar** para guardar el formulario. Normalmente se le asigna el nombre **"MainPage"** indicando que es la "página principal" de tu programa. No hace falta que le pongas otro nombre.

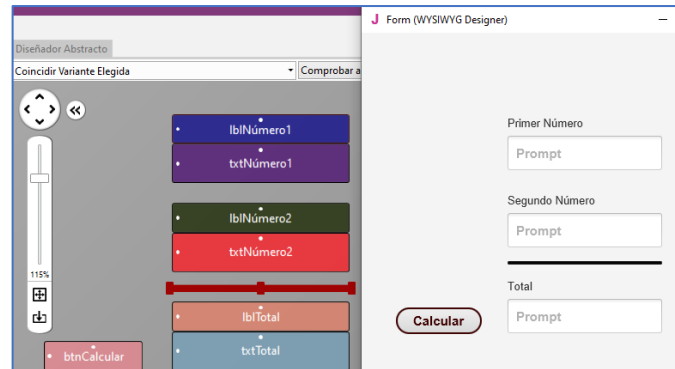


Figura 10. Panel

Ejercicios


1. Usa el diseñador para crear los siguientes esquemas.



Consejo para el profesor

Esta es la parte divertida. Deja que los pupilos experimenten libremente con las vistas.

Nombre	Clase	
<input type="text"/>	<input type="text"/>	
Apellidos	Edad	
<input type="text"/>	<input type="text"/>	
<input type="button" value="Salvar"/>	<input type="button" value="Borrar"/>	<input type="button" value="Cancelar"/>



Calculadora de nota media			
Matemáticas	<input type="text"/>	Música	<input type="text"/>
Física y Quím.	<input type="text"/>	Informática	<input type="text"/>
Lengua	<input type="text"/>	Ed. Física	<input type="text"/>
Inglés	<input type="text"/>	Geo. e Historia	<input type="text"/>
Nota media		<input type="text"/>	<input type="button" value="Calcular"/>

11. Piensa y diseña tu “Aplicación Soñada”. Dale un nombre, crea un esquema de su interfaz de usuario en tu ordenador y crea la Vista de Diseño.

Tema 6 – Del Diseñador al Código

🕒 2h

Lo que los estudiantes aprenderán

- Class_Globals
- Variables y subrutinas (Subs)
- Paso de Valores al Código
- Eventos
- Atributos

Diseñar la apariencia de la aplicación en el Diseñador es el primer paso al crear una aplicación. A menudo, los nuevos programadores recurren a él para rediseñar, corregir o agregar información individual.

Cuando la pantalla está lista, el desarrollador pasa a la siguiente etapa que consiste en programar las funciones. Es decir, hay que hacer que todos los elementos que se incluyeron en el diseño realicen sus funciones correctamente. Por ejemplo, los cuadros de texto deben poder leer datos, los botones deben lanzar funciones, las listas deben mostrar datos, etc.

Class_Globals

Al comienzo del código en la pestaña **B4XMainPage** hay un conjunto de declaraciones de variables entre **Sub Class_Globals** y **End Sub**.

```
8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11 End Sub
```

Imagen 2. Sub Class_Globals

Como se vio en el tema 3, una Sub (subrutina) es un trozo de código que realiza una operación específica. Dentro de **Class_Globals** se recogen las declaraciones de variables que queremos que se

conozcan a lo largo del código de la pestaña **B4XMainPage**, es decir, en cada subrutina.

Además, si una sentencia de declaración de variables empieza con la palabra **Public**, entonces estará disponible para cualquier otra "pestaña".



Un estudio más profundo del uso de variables

En el siguiente código puedes ver 3 subrutinas:

B4XMainPage

```
10 Sub Class_Globals
11   Private Root As B4XView
12   Private xui As XUI
13   Private intNumber As Int = 32
14   Private intNewTotal As Int
15 End Sub
16
17 Public Sub Initialize
18 End Sub
19
20 Private Sub B4XPage_Created (Root1 As B4XView)
21   Root = Root1
22   Root.LoadLayout("MainPage")
23   Private intN1, intN2, intTotal As Int
24   intN1 = 45
25   intN2 = 32
26   intTotal = intN1 + intN2
27   Log("Total = " & intTotal)
28
29   intNewTotal = intTotal + intNumber
30   Log("New Total = " & intNewTotal)
31 End Sub
32
33 Sub Button1_Click
34   xui.MsgboxAsync("Hello world!", "B4X")
35   Private intN1 As Int = 5
36   intNewTotal = intN1 + intNumber
37   Log("New Total = " & intNewTotal)
38 End Sub
```

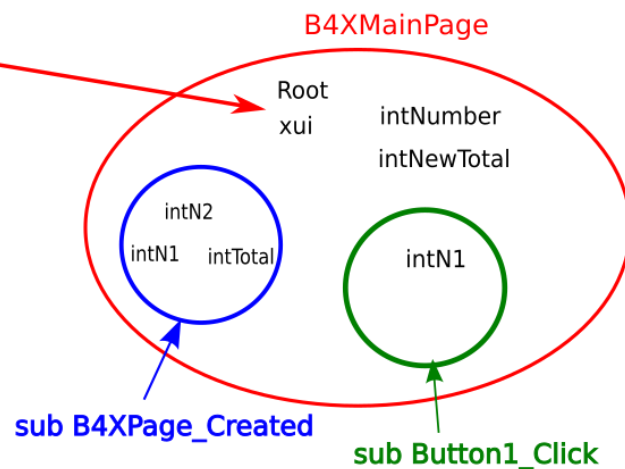


Imagen 3. Variables y ámbitos

Las variables **intNumber**, **intNewTotal**, **Root** y **xui** declaradas dentro de **Class_Globals** "viven" dentro del módulo **B4XMainPage**. Se trata de variables de ámbito global que pueden ser accedidas desde cualquier subrutina dentro de la pestaña **B4XmainPage**.

Por el contrario, las variables **intN1**, **intN2**, **intTotal** (color azul) "viven" dentro de la subrutina **B4XPage_Created** y ninguna otra subrutina podrá usarlas. Al mismo tiempo, la variable **intN1** declarada dentro del botón **Button1_Click** **no es la misma** que la que hay dentro de **B4XPage_Created**. Ambas tienen su propia zona de memoria y pueden tener el mismo nombre por estar en subrutinas distintas.



Consejo para el profesor

El uso y el ámbito de las variables son conceptos que confunden a los nuevos programadores. En función de tu alumnado, sería útil usar ejemplos que les resulten familiares.

Paso de Valores al Código

En la pantalla que has diseñado en el tema 5 hay objetos (etiquetas, textos, botones, etc.) que hay que declarar dentro de **Class_Globals** antes de usarlos en tu programa.



Algunos de los objetos de la pantalla no hay que incluirlos en el código porque no se van a usar. Por ejemplo, ninguna de las “etiquetas” va a ser gestionada en el código. Sin embargo, los botones y los campos de texto (TextFields) sí que van a ser modificados en el código.

Hay dos formas de insertar objetos en el código. La primera se hace fácilmente a través del **Diseñador** siguiendo estos pasos:

- Vamos a **Diseñador** → **Abrir Diseñador interno**
- Elegimos dentro del menú **Herramientas** la opción **Generar Miembros**.

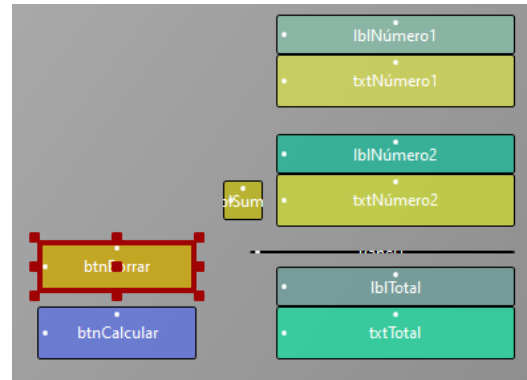


Imagen 4. Ejemplo 1. Diseñador Abstracto

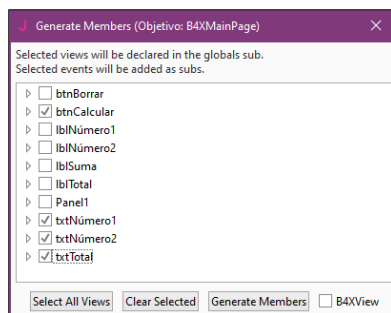


Imagen 5. Generate Members

Dentro de la pantalla de **Generate Members** hacemos clic en los objetos:

- btnCalcular
- txtNúmero1
- txtNúmero2
- txtTotal

y pulsamos en **Generate Members**.

Tu código en la subrutina **Class_Globals** se actualizará automáticamente con las nuevas variables.

```

8 Sub Class_Globals
9     Private Root As B4XView
10    Private xui As XUI
11    Private txtTotal As TextField
12    Private btnCalcular As Button
13    Private txtNúmero1 As TextField
14    Private txtNúmero2 As TextField
15 End Sub

```

Imagen 6. Class_Globals



Recuerda

Cada objeto que importamos es de un tipo concreto que coincide con el tipo de las variables que se han creado

La segunda opción consiste en escribir las variables uno mismo, prestando atención a que los nombres de los objetos de la pantalla sean los mismos que los que escribes como variables (y también su tipo: Button, TextField, etc).

Eventos

Tras declarar las variables para los objetos, el último paso es activar las funciones del formulario.

Esto dependerá de lo que haga tu aplicación. En el ejemplo que estamos viendo hay un botón llamado **Calcular** que lo que hará es sumar los dos números de la pantalla y mostrará el resultado en pantalla.

La operación de suma es disparada por un proceso llamado **Evento**. El programador debe detectar el evento de pulsar en el botón **Calcular** para realizar las operaciones necesarias para mostrar el resultado.

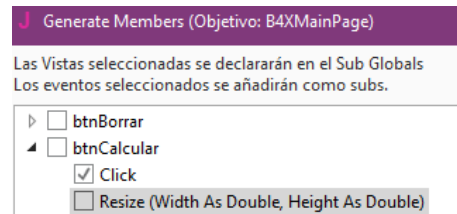


Recuerda

Hay miles de eventos que pueden dispararse en una aplicación; es tu responsabilidad como programador decidir cómo responder a ellos.

La detección de un evento es fácil y sólo hay que crear una nueva subrutina con el nombre del evento. Esto se puede hacer al mismo tiempo que se realizó la declaración de las variables a través del **Diseñador**.

- Vamos a **Diseñador** → **Abrir Diseñador interno**
- Elegimos dentro del menú **Herramientas** la opción **Generar Miembros**.
- Dentro de la pantalla de **Generate Members** desplegamos la lista que hay en el botón **btnCalcular**.
- Marcamos el evento **Click** y de nuevo pulsamos en **Generate Members**.



Ahora aparecerá una **nueva subrutina** para el evento **btnCalcular_Click** donde podrás escribir el código necesario:

```
29 Private Sub btnCalcular_Click
30
31 End Sub
```

Imagen 7. El evento Click

Escribiendo código en el Evento

Dentro de la subrutina de un Evento se escriben todas las acciones que deben lanzarse cuando se produce ese evento.

```
29 Private Sub btnCalcular_Click
30     txtTotal.Text = txtNúmero1.Text + txtNúmero2.Text
31 End Sub
```

Imagen 8. Realizar cálculos con los TextFields

En la Imagen 9 estamos haciendo lo siguiente:

El contenido del TextField llamado txtTotal será igual a la suma de los contenidos de los TextFields txtNúmero1 y txtNúmero2

Propiedades

Cada objeto que insertar en tu código tiene varias propiedades. Por ejemplo, puede tener un color, un tamaño, una posición en pantalla, un contenido, etc. Estas propiedades pueden leerse o modificarse dentro de tu código. Por ejemplo, la propiedad **txtNúmero1.Text** nos dice qué contiene el campo de texto **txtNúmero1**; pero también nos permite modificar su contenido dependiendo de cómo la usemos.

Las propiedades de un objeto podemos averiguarlas simplemente escribiendo el nombre del objeto y poniendo un punto (.) a continuación del nombre. Entonces, nos aparecerá una lista de todas las propiedades (ver Imagen 10).

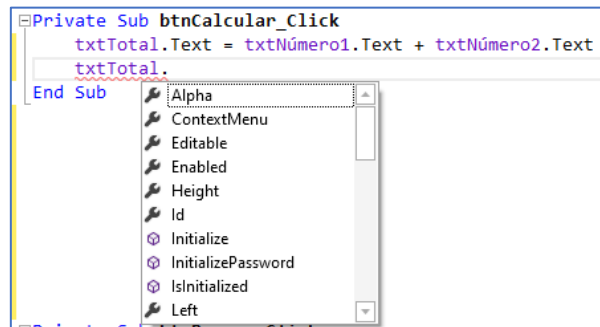


Imagen 9. Lista de propiedades de un objeto

Imaginemos que queremos crear una nueva función en la calculadora que borre el formulario para escribir nuevos números. Las acciones que tienes que hacer son las siguientes:

- Abrir el **Diseñador** y añadir un nuevo botón llamado **btnBorrar**.
- Crear una variable dentro de **Class_Global** para acceder a ella.
- Crear el evento **btnBorrar_Click**.



Recuerda

Puedes hacer operaciones matemáticas con cadenas de texto **sólo si contienen números**.

- Dentro del evento, fija la propiedad **Text** de **txtNúmero1**, **txtNúmero2** y **txtTotal** al valor "" (cadena vacía).

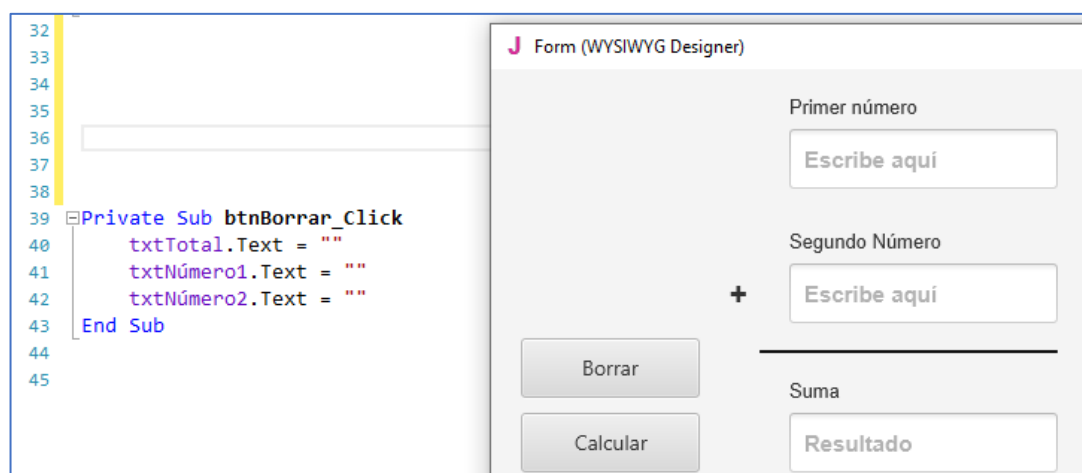


Imagen 10. Botón de Borrar y código asociado

Cada vez que pulses el botón **Borrar**, se borrará el formulario.

Ejercicios

1. Amplía el ejemplo visto en clase para que realice las 4 operaciones: suma, resta, multiplicación y división cuando se pulse el botón adecuado. Añade los objetos necesarios en el diseñador y completa el código.
2. En el ejercicio 2 del tema anterior, continúa y completa la aplicación de calcular notas medias para que el botón "Calcular" calcule la nota media de todas las materias.

Calculadora de nota media			
Matemáticas	<input type="text"/>	Música	<input type="text"/>
Física y Quím.	<input type="text"/>	Informática	<input type="text"/>
Lengua	<input type="text"/>	Ed. Física	<input type="text"/>
Inglés	<input type="text"/>	Geo. e Historia	<input type="text"/>
Nota media		<input type="text"/>	<input type="button" value="Calcular"/>

Debes comprobar que las notas de cada materia están entre 0 y 10. Si no, deberías mostrar un mensaje de error.

Tema 7 – Sentencias Condicionales

🕒 4h

Lo que los estudiantes aprenderán

- Variables Booleanas (o lógicas)
- Operadores relacionales
- Operadores lógicos
- Sentencia If
- Sentencia If-Else
- Sentencia If-Else-Else If
- Algoritmos para calcular el Máximo

Variables Lógicas o Booleanas

Anteriormente hemos visto tres tipos de variables: entero, decimal y cadena. Las variables lógicas son un tipo de dato muy sencillo, ya que sólo admiten dos valores posibles: True o False (Verdadero o Falso). Internamente el ordenador las representa como un 1 o un 0 (si hay corriente eléctrica o no).

La declaración de una variable lógica (Booleana) se hace igual que con los otros tipos de datos ya vistos:

```
Private intDistancia = 100, intTotalViaje = 0 As Int
Private blnIndicador As Boolean = False
Private blnHecho As Boolean
```

Operadores relacionales o de comparación

Los operadores relacionales se usan para realizar comparaciones entre valores. Son los mismos que en matemáticas, aunque en informática se escriben de una forma un poco diferente:

Símbolo matemático	B4X	Significado
=	=	Igual a
≤	<=	Menor o igual a
≥	>=	Mayor o igual a
≠	<>	Distinto
<	<	Menor que
>	>	Mayor que



En general, para hacer una comparación debes **comparar variables del mismo tipo**. Por ejemplo, enteros con enteros, decimales con decimales, cadenas con cadenas, etc. Sin embargo, en B4X puedes comparar también enteros con decimales y cadenas con números porque internamente se realiza la conversión de cadenas a números.

```

38 Private intDistancia = 100 As Int ' Fijate en la diferente forma de declarar
39 Private intTotalViaje As Int = 0 ' dos enteros
40 Private fltD As Float = 100.45
41 Private strN As String = "100"
42 Private s As String = "Colegio"
43
44 Log( fltD > intDistancia) 'Muestra True
45 Log( strN = intDistancia) 'Muestra True
46 Log( strN = intTotalViaje) 'Muestra False
47 Log( s = intTotalViaje) 'Muestra False
48 Log( intTotalViaje <> strN ) 'Muestra True

```

Imagen 11. Comparación de Variables



Recuerda

El resultado de una comparación es siempre un valor lógico (True o False)

Operadores Lógicos

Piensa en una afirmación como "Voy al colegio ahora". ¿Es cierta o falsa?

El matemático George Boole creó un álgebra basada en sentencias lógicas.

En el álgebra booleana los valores de las variables son verdadero o falso. Normalmente se representan por un 1 o un 0, respectivamente. Al contrario que con el álgebra a la que estás acostumbrado, en el álgebra booleana hay 3 operaciones posibles: Y, O y Negación (en inglés, **AND**, **OR** y **NOT**).



Recuerda

AND (conjunción), indicado como **x AND y**. El resultado será 1 si **x** e **y** valen los dos 1. En otro caso, el resultado será 0.

OR (disyunción), indicado como **x OR y**. El resultado será 0 si **x** e **y** valen 0. En cualquier otro caso, el resultado es 1.

NOT (negación), indicado como **NOT x**. El resultado será 0 si **x** valía 1, o bien 1 si **x** valía 0.

Ejemplo:

1ª Afirmación: *Está diluviando,*

2ª Afirmación: *Voy al colegio ahora*

Está diluviando (P1)	Voy al colegio ahora (P2)	P1 AND P2	P1 OR P2	NOT P1
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True



En la tabla podemos ver:

- Dos afirmaciones que al unirse con el operador **AND** dan como resultado un valor cierto sólo si las dos eran ciertas.
- Dos afirmaciones que al unirse con el operador **OR** dan como resultado un valor cierto si alguna de las dos eran ciertas.
- El operador NOT que invierte la veracidad o falsedad de una afirmación.

Operadores Lógicos en programación

Los operadores lógicos se usan en programación para crear expresiones de comparación complejas. Esto permite al programador optimizar su código con menos líneas y con un código más sencillo.

En B4X las variables lógicas se usan así:

```
50 'Operadores lógicos
51 Private blnL1, blnL2 As Boolean
52 blnL1 = True
53 blnL2 = False
54
55 Log (blnL1 And blnL2) ' Muestra False
56 Log (blnL1 Or blnL2) ' Muestra True
57 Log (Not(bl nL1)) ' Muestra False
58 Log (Not(bl nL2)) ' Muestra True
```

Imagen 12. Ejemplos de uso de variables lógicas

Fíjate que los operadores lógicos emplean variables lógicas o expresiones lógicas como veremos más adelante.

Ejemplos de evaluación de sentencias lógicas

Supongamos que las siguientes variables tienen estos valores:

```
Private intA = 10, intB = 20, intC = 30 As int
Private strNombre1 = "Jorge", strNombre2 = "Málaga" As String
Private blnA = True, blnB = False, blnC = False As Boolean
```

Calcula el valor de las siguientes expresiones lógicas:

1.

blnA	AND	blnB
True		False
False		

2.

Inta	>	intC	AND	blnA
10		30		True
False		False		

3.

intA + intB	>=	intC	AND	(bInA	OR	bInB)
10 + 20		30		True		False
	True		True		True	

4.

intA + intB	>=	intC	OR	(bInA	AND	bInB)
10 + 20		30		True		False
	True		True		False	

5.

strNombre1	=	"Jorge"	OR	strName2	=	"Juan"
Jorge		Jorge		Málaga		Juan
	True		True		False	

Sentencia If

Al igual que la vida real nos hacemos preguntas, en programación también hace falta comprar valores o cambiar el orden de ejecución del programa para que haga diferentes cosas.

La **sentencia If** se usa para realizar esas preguntas.

Su forma básica es al siguiente:

```
If ( condición ) Then
    Instrucciones
End If
```

Donde **condición** debe contener una comparación o una expresión lógica de las vistas antes.

El significado es: **SI** la condición es **TRUE**, ejecuta las instrucciones que hay entre **Then** y **End If**.

Ejemplos:

```
Private intA = 10, intB = 20 As Int
Private fltA As Float

If intA > 0 Then
    Log(intA & " es un número positivo")
End If

If intA > 10 Or intB > 10 Then
    Log("Uno o los dos números son mayores que 10")
End If

If intA Mod 2 = 0 Then
    Log(intA & " es un número par ")
End If
```



If – Else

El comando **Else** permite ejecutar instrucciones cuando la condición del **If** es falsa:

Su forma básica es la siguiente:

```
If ( condición ) Then
    Instrucciones_Verdadero
Else
    Instrucciones_Falso
End If
```

El significado es: Si la **condición** es **True**, entonces ejecutar las **Instrucciones_Verdadero**. Si la condición es **False**, ejecutar las **Instrucciones_Falso**.

Ejemplos:

```
Private intA = 10, intB = 20 As Int
Private fltA As Float

If intA > 0 Then
    Log(intA & " es un número positivo ")
Else
    Log(intA & " no es un número positivo ")
End If

If intA > 10 Or intB > 10 Then
    Log("Uno o los dos números son mayores que 10")
Else
    Log("Ninguno de los números es mayor que 10")
End If

If intA Mod 2 = 0 Then
    Log(intA & " es un número par")
Else
    Log(intA & " es un número impar")
End If
```

If – else - else if

Podemos usar varios **If** con varias condiciones para extender la funcionalidad de un único comando **if** (se llama "anidar" comandos **If**).

Cómo lo construimos:

```
If ( condición1 ) Then
    Instrucciones1
Else If ( condición2 ) Then
    Instrucciones2
Else If ( condición3 ) Then
    Instrucciones3
Else If ( condición4 ) Then
    Instrucciones4
...
Else
    Instrucciones_Todo_Falso
End If
```

El funcionamiento de los múltiples **If** anidados es el siguiente:

1. Se comprueba la primera condición (**condición1**). Si es **True**, ejecutamos el código **Instrucciones1** y el **If** finaliza (no se ejecuta nada más).
2. Si la primera condición del primer **If** (**condición1**) es **False**, entonces se comprueba la condición del segundo **If** (**condición2**) y, si es cierta, se ejecutan las **Instrucciones2** (y no se ejecuta nada más).
3. Para el resto de **If**, se comprueban sus condiciones si las anteriores son falsas.
4. Si **ninguna** de las condiciones de los **If** es cierta, se ejecutan las **Instrucciones_Todo_Falso**. El **Else** no es obligatorio usarlo.

Ejemplo 3

Un restaurante de comida rápida ofrece estas comidas:

Comida	Precio
Hamburguesa	5 €
Pizza	3 €
Salchicha	1,5 €

Crea un programa que:

Lea el tipo de comida que el cliente quiere. Imprimir el coste de la comida.
Por ejemplo: Entrada: "Salchicha". Salida: "Salchicha 1,50 €"

Solución

Paso 1

Crea un nuevo proyecto de tamaño 300x300.

Paso 2

En el diseñador, crea la pantalla de la aplicación

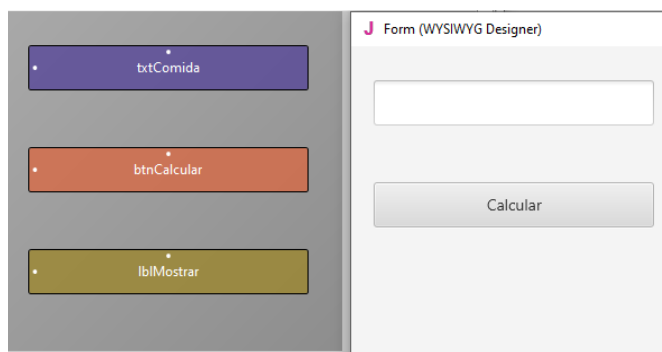
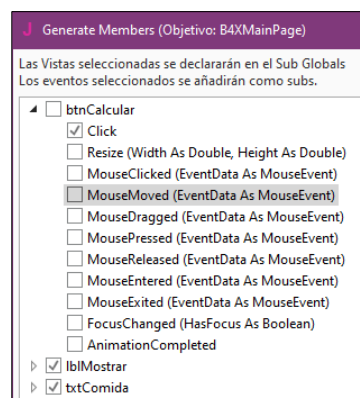


Imagen 13. Pantalla de la aplicación

Paso 3

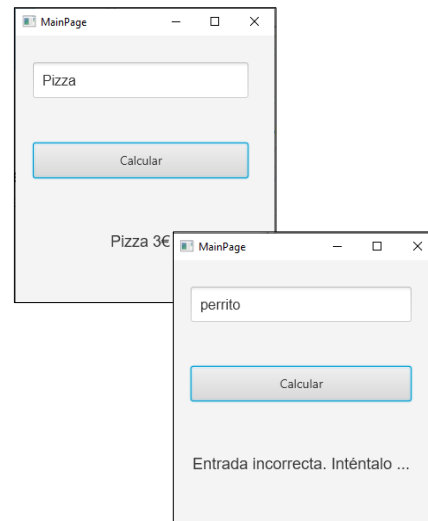
Genera miembros para **txtComida**, **btnCalcular**, **lblMostrar** y **btnCalcular_Click**.



Paso 4

El código que tienes que escribir en el **btnCalcular_Click** es:

```
7 Sub Class_Globals
8     Private Root As B4XView
9     Private xui As XUI
10    Private btnCalcular As Button
11    Private lblMostrar As Label
12    Private txtComida As TextField
13 End Sub
14
15 Public Sub Initialize
16
17 End Sub
18
19 'This event will be called once, before the page becomes visible.
20 Private Sub B4XPage_Created (Root1 As B4XView)
21     Root = Root1
22     Root.LoadLayout("MainPage")
23 End Sub
24
25 Private Sub btnCalcular_Click
26     If txtComida.Text = "Hamburguesa" Then
27         lblMostrar.Text = "Hamburguesa 5€"
28     else if txtComida.Text = "Pizza" Then
29         lblMostrar.Text = "Pizza 3€"
30     else if txtComida.Text = "Salchicha" Then
31         lblMostrar.Text = "Salchicha 1.5€"
32     Else
33         lblMostrar.Text = "Entrada incorrecta. Inténtalo de nuevo"
34     End If
35 End Sub
```



Fíjate que el texto que introducimos distingue entre mayúsculas y minúsculas, con lo que si escribimos "perrito" con la letra "p" en minúscula, no lo reconoce como válido porque en el código hemos dicho que tiene ponerse "Perrito", con la "P" en mayúscula.

Algoritmos con If

Vamos a ver cómo encontrar el máximo de un grupo de números.

Supongamos que leemos 3 números enteros y queremos encontrar el mayor de los 3. Veamos 3 formas diferentes de hacerlo:

Método 1 – Sentencia If simple

```
If Inta > intB AND Inta >
intC then
    Log(Inta)
End If
If intB > Inta AND intB >
intC then
    Log(intB)
End If
If intC > Inta AND IntC >
```

Método 2 –If anidado

```
If Inta > intB then
    If Inta > intC then
        Log(Inta)
    End If
End If
If IntB > IntA then
    If IntB > intC then
        Log(IntB)
    End If
End
If IntC > IntA then
    If IntC > IntA then
        Log(IntC)
    End If
End If
```

Método 3 – Algoritmo Máx

```
Máx = inta
If intB > Máx then
    Máx = intB
End If
If intC > Máx then
    Máx = intC
End If
Log(Máx)
```

Ejercicios

1. Escribe las siguientes condiciones como expresiones lógicas:

- i. **A** pertenece al intervalo [-5, 6)
- ii. **A** es menor que 3 o mayor que 15
- iii. **A** es igual a **B** y **C**
- iv. **A** no tiene el valor de 3
- v. **A** es menor que 2 o **B** es mayor de 78
- vi. **A** y **B** son verdad y **C** es falsa
- vii. **A** es verdad y o bien **B** o **C** son verdad

2. Calcula el resultado (True o False) de las siguientes expresiones lógicas suponiendo los siguientes valores de las variables:

A = 10, B = 2, C = -4, D = 9 and E = 1

- i. (A>B) or (D=10)
- ii. (D >= B) and (E <> C)
- iii. not (E<=C) or (D<=C)
- iv. not ((B<=C) and (D<2))
- v. not (not (B<=E) or not (C<=B))
- vi. ((E<=A) and (E>=C)) and not (C>=A)
- vii. not (not (A >= 2) and (C <>9))

3. Un restaurante de comida rápida ofrece estas comidas:

Comida	Precio
Hamburguesa	5 €
Pizza	3 €
Salchicha	1,5 €

Crea un programa que:

Primero lea la comida que el cliente quiere y después cuántos productos quiere. Muestre el coste total de la comida.

Ejemplo de entrada: "Salchicha", 2

Salida: " 2 x Salchicha 3 €"

4. Has gastado una cantidad de X megabits en la Wikipedia y una cantidad Y de megabits en memes. El coste de visitar la Wikipedia es de 0'1€ por megabit y el de ver memes es de 0'05 € por megabit. Si el consumo total es mayor de 100 €, debes mostrar el mensaje "Consumo excesivo". Si gastas más en ver memes que en visitar la Wikipedia, debes imprimir el mensaje "Demasiados memes". Crea un programa que:
- Lea el valor de X (consumo de megabits de Wikipedia) y el de Y (consumo de megabits por ver memes).
 - Calcule el consumo total.
 - Si el consumo total es mayor de 100€, debe imprimir el mensaje adecuado. Si el consumo en ver memes es mayor que el de visitar la Wikipedia, se imprime el mensaje correcto.
5. Un cibercafé tiene 2 formas de cobrar. Si el usuario es miembro, pagaría 2€ por hora, si no, paga 5€ por hora. Averigua si alguien es miembro o no y calcula el precio que pagaría en función de las horas que emplee. Si el usuario es miembro, el impuesto es del 10%, si no, es del 20%. Crea un programa que:
- Lea cuántas horas ha usado el cliente.
 - Pregunte si es miembro o no.
 - Añada el impuesto adecuado.
 - Imprima lo que tiene que pagar el usuario mostrando el texto: "El usuario es miembro y ha estado 2 horas por 2€/hora más el 10% de impuesto, que hace un total de 4'4€".
6. Quiere comprar algo en Amazon. El vendedor cobra diferentes gastos de envío según tu ubicación. Para EEUU son 5€, para Europa 7€, por Canadá 3€ y para el resto 9€. Crea un programa que:
- Lea el coste del producto.
 - Lea tu ubicación.
 - Imprima los gastos de envío.
 - La salida debe ser: "Tienes que pagar 23€, 20€ por el producto y 3€ por los gastos de envío".

7. Una empresa vende un producto por 0'70€ la unidad si se piden hasta 200 unidades, o bien 0'50€ si se piden más de 200 unidades. Lee el número de piezas que se piden y calcula su valor.

8. Una empresa de telefonía tiene las siguientes tarifas:

Coste fijo 25€	
Duración de la llamada (en segundos)	Coste (€/por segundo)
1-500	0,01
501-800	0,008
801+	0,005

Crea un programa que:

- Lea el número de segundos que todas las llamadas.
 - Calcule la factura mensual para el cliente.
 - Imprima la cantidad total.
 - Salida: "Importe total: 48€".
 - Fíjate que el cargo de los primeros 500 segundos es 0'01€ por segundo, para los segundos 501 a 800 es de 0'008€ y a partir de ahí 0'005€.
9. En la fase de clasificación en salto de longitud de los juegos olímpicos, un atleta realiza 3 intentos iniciales y, si logra una distancia mayor a 7'50 metros, entonces pasa de ronda y puede realizar otros 3 intentos más. Escribe un programa que lea los primeros 3 intentos de un atleta e imprima un mensaje diciendo si puede pasar de ronda o no y que también muestre la mayor distancia que ha saltado.

Puedes usar el método Visible = True or False para ocultar o mostrar Labels, TextFields y Buttons.

10. En una ciudad hay aparcamientos de pago. El coste de aparcar se calcula de la siguiente forma:

Duración aparcamiento	Coste por hora
Hasta 1 hora	3.50 €
Las siguientes 2 horas	8.00 €
Las siguientes 2 horas	12.00€
Más de 5 horas	15.00 €

Escribe un programa que lea el tiempo que alguien dejó su coche en el aparcamiento y que calcule el coste total.

Tema 8 – Subrutinas

🕒 3h

Lo que los estudiantes aprenderán

- Qué es una subrutina (Sub)
- Declaración de una Subrutina
- Paso de Valores
- Devolución de Valores de una Subrutina

En programación, una subrutina es una secuencia de instrucciones que realizan una tarea concreta y que forman una unidad. Esta unidad puede usarse después en los programas dondequiera que esa tarea sea realizada.

Las subrutinas pueden definirse dentro de nuestro programa o dentro de bibliotecas que agrupan múltiples subrutinas para que puedan ser usadas por otros programas. Según el lenguaje de programación, a las subrutinas se les puede llamar funciones, subprogramas, métodos o procedimiento. Para crear una subrutina el programador debe tener en cuenta esto:

- Debe realizar una única tarea.
- Debe ser relativamente pequeña e, idealmente, no mayor que lo que cabría en una pantalla para que pueda leerse fácilmente.
- Debe tener un nombre que haga referencia a lo que hace.

Crear una subrutina en B4J

Ya hemos visto la subrutina **Button1_Click** en B4J cuando se crea un nuevo programa. Fíjate que los eventos de "Click" se gestionan mediante subrutinas.

Ejemplo 1

Imagínate una función que deba sumar dos números introducidos por el usuario. Se trata de un ejemplo muy sencillo, pero lo usaremos para entender el concepto de subrutina y ver cómo funciona.

Los dos enteros intA e intB se declaran en el programa, se les asigna un valor y después invocamos la subrutina **MostrarSuma1**.

```
8 Sub class_globals
9   Private Root As B4XView
10  Private xui As XUI
11 End Sub
12
13 Public Sub Initialize
14
15 End Sub
16
17 'This event will be called once, before the page becomes visible.
18 Private Sub B4XPage_Created (Root1 As B4XView)
19   Root = Root1
20   Root.LoadLayout("MainPage")
21 End Sub
22
23 'You can see the list of page related events in the B4XPagesManager class
24
25 Sub Button1_Click
26   xui.MsgboxAsync("¡Hola mundo!", "B4X")
27 End Sub
```

Imagen 14. Subrutinas



Invocar a una subrutina se hace escribiendo su nombre (el que queramos) y entre paréntesis ponemos las variables cuyos valores debe conocer para funcionar.

La subrutina se escribe antes o después del programa actual:

```
18 Private Sub B4XPage_Created (Root1 As B4XView)
19     Root = Root1
20     Root.LoadLayout("MainPage")
21     Private intA, intB As Int
22     intA = 10
23     intB = 30
24
25     MuestraSuma1(intA, intB)
26     Log(intSum)
27
28     Log(Suma(intA, intB))
29 End Sub
30
31 Private Sub MuestraSuma1(a As Int, b As Int)
32     intSum = a + b
33     Log(intSum)
34 End Sub
```

Imagen 15. Invocando una subrutina

Si delante ponemos la palabra reservada **Private**, significará que esta subrutina sólo se conocerá dentro del código B4XmainPage; si ponemos **Public**, la subrutina se podrá usar en otras partes de nuestra aplicación.

La palabra "Sub" es una abreviatura de subrutina. Entre paréntesis escribimos el nombre de las variables cuyos valores se enviarán a la función.

Fíjate en la **Imagen 2** en que los datos se envían a la función en el orden en que están escritos al invocar a la subrutina. Por ej. el valor de "intA" se introduce en la variable "a" y el valor de "intB" en la variable "b".



Recuerda

Las variables usadas en las subrutinas cuando son invocadas se llaman **parámetros**.

La subrutina funcionará con los datos contenidos en los parámetros "a" y "b" y NO con las variables "intA" e "intB".



La memoria de una subrutina en B4X

Cada subrutina tiene su propio espacio de memoria donde almacenar variables. La única excepción es la subrutina **Class_Globals** cuyas variables pueden ser consultados por todas las rutinas de B4XMainPage por sus nombres.

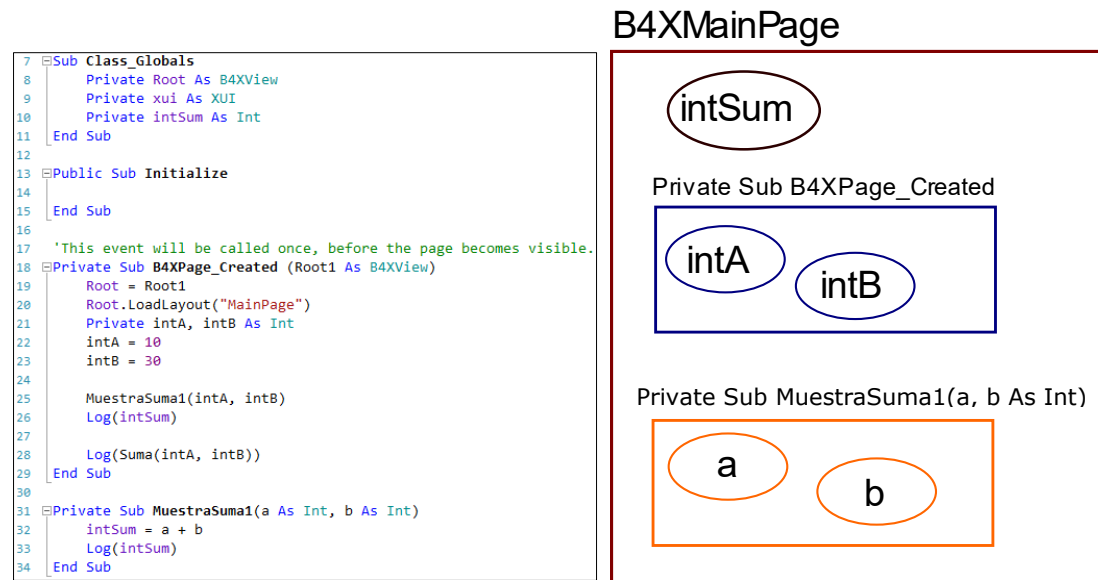


Imagen 16. La memoria

En la **Imagen 3** puedes ver que la variable **intSum** es accesible desde cualquier subrutina y se puede consultar escribiendo su nombre. En B4X estas variables se representan en un color diferente para que el programador pueda distinguirlas fácilmente. Por el contrario, las variables declaradas dentro de una subrutina sólo son accesible desde esa subrutina.

Recuerda



Las variables declaradas dentro de **Class_Globals** se pueden ver desde cualquier subrutina y se llaman **Globales**.

Las variables declaradas dentro de una subrutina sólo se pueden ver dentro de esa subrutina y se llaman **Locales**.

Devolución de un valor desde una subrutina

Una subrutina puede devolver un valor al código que la invocó. Esto se hace de la siguiente forma:

```
18 Private Sub B4XPage_Created (Root1 As B4XView)
19     Root = Root1
20     Root.LoadLayout("MainPage")
21     Private intA, intB As Int
22     intA = 10
23     intB = 30
24
25     MuestraSuma1(intA, intB)
26     Log(intSum)
27
28     Log(Suma(intA, intB))
29 End Sub
30
31 Private Sub MuestraSuma1(a As Int, b As Int)
32     intSum = a + b
33     Log(intSum)
34 End Sub
35
36 Private Sub Suma(a As Int, b As Int) As Int
37     Return(a+b)
38 End Sub
```

Imagen 17. Devolver valores al programa

1. En la declaración de la subrutina tienes que incluir el tipo de variable que devolverá (**as Int** en la imagen).
2. Debes usar la sentencia **Return** dentro de la subrutina para devolver el valor que quieras.
3. El código que ha invocado la subrutina usará el valor devuelto por la subrutina como si fuera una variable.



Recuerda

Normalmente, a las subrutinas que devuelven valores al código se les llama **Funciones**.

Ejemplo 2

Escribe un programa que lea 3 enteros y devuelva el mayor.

```
5 Sub Class_Globals
6     Private Root As B4XView
7     Private xui As XUI
8 End Sub
9
10 Public Sub Initialize
11 End Sub
12
13 Private Sub B4XPage_Created (Root1 As B4XView)
14     Root = Root1
15     Root.LoadLayout("MainPage")
16     Private intA, intB, intC As Int
17     intA = 20
18     intB = 10
19     intC = 300
20
21     Log(CalcularMax(intA, intB, intC)) ' Muestra 300
22 End Sub
23
24 Private Sub CalcularMax(a As Int, b As Int, c As Int) As Int
25     Private intM As Int
26     intM = a
27     If b > intM Then
28         intM = b
29     End If
30     If c > intM Then
31         intM = c
32     End If
33
34     Return(intM)
35 End Sub
```

Imagen 18. Ejemplo 2

1. Declaramos 3 variables enteras (Inta, intB, intC) en la subrutina **B4XPage_Created**.
2. Asignamos valores a las 3 variables.
3. Invocamos a la subrutina CalcularMax con las 3 variables como parámetros.
4. La subrutina aplica el algoritmo Máximo a las variables **a, b, c** y devuelve el resultado intM calculado
5. El valor mayor es mostrado en la ventana de Log.



Recuerda

Nos encantan las subrutinas porque:

- Es aburrido escribir siempre el mismo código.
- Es más rápido usarlas que escribir de nuevo el código.
- Es útil no repetir los mismos fallos. ¡Ya los has corregido una vez!
- Los buenos programadores usan subrutinas.



Ejercicios



Consejo para el profesor

Anima a los estudiantes a solucionar y discutir sus soluciones en clase.
Dedícale al menos una hora a explicarlos.

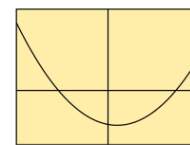
1. Escribe un programa que calcule el área de un círculo. El usuario debe introducir el radio del círculo en un textField y después usa una subrutina para calcular y devolver el área.
2. Escribe un programa que calcule la solución a la ecuación de segundo grado $ax^2 + bx + c = 0$.
 - a. El usuario introducirá los coeficientes **a**, **b** y **c** en TextFields.
 - b. El resultado aparecerá en uno o dos textField según el valor del discriminante.
 - c. El discriminante debe calcularse con una subrutina que devolverá el valor.
 - d. No se permitirá calcular el discriminante hasta que todos los coeficientes a, b y c se introduzcan.

Para mostrar mensajes de error puedes usar `xui.MsgboxAsync("Message","Title")`

Imagen 20. Esquema

Ecuación cuadrática

$$ax^2 + bx + c = 0$$



$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Imagen 19. Origen:
Wikipedia.org

3. Construye un programa que use la tortuga y que dibuje cuadrados de lado igual a un valor introducido por el usuario. Crea una subrutina que reciba la longitud del lado y que dibuje el cuadrado empezando por el lugar donde esté actualmente la tortuga y que se mueva en el sentido de las agujas del reloj.
4. Un teatro tiene 3 tipos de entradas: anfiteatro, galería y proscenio. Cada entrada vale 20, 30 y 40€ respectivamente. Construye un programa que:
 - a. Lea los números 1, 2 o 3 que representan las 3 categorías (1: anfiteatro; 2: galería y 3: proscenio).
 - b. Le el número de entradas que se quieren.
 - c. Calcule el valor total de las entradas con una subrutina que devuelva la cantidad y que se muestre el resultado en un textField.



Tema 9 – Clases



Lo que los estudiantes aprenderán

- ¿Qué es una clase?
- ¿Qué es un objeto?
- ¿Qué son los atributos y los métodos?
- Crear y usar una clase sencilla en B4J.

A menudo en programación necesitamos referirnos a objetos similares de una única forma. Por ejemplo, los estudiantes en el colegio. Cada estudiante posee un nombre, una dirección, unas clases a las que asistir, títulos, etc. Para gestionar esta información un programador debe organizar estos datos sistemáticamente.



Consejo para el profesor

Las Clases son un tema complejo en programación. Evita entrar en detalles profundos como herencia, encapsulación, etc.

Clases

En el ejemplo de los estudiantes podríamos decir que cada estudiantes posee la siguiente información:

Estudiante	
1	Identificador Escolar
2	Nombre
3	Apellidos
4	Dirección
5	Teléfono
6	Correo electrónico

También necesitamos las siguientes funciones:

- Registrar un nuevo estudiante
- Cambiar la información del estudiante
- Traspasar un estudiante
- Mostrar la información de un estudiante
- Borrar un estudiante

Así pues, para 3 estudiantes, tendríamos los siguientes elementos:

Estudiante 1	
1	125310
2	Augusta
3	Ada Byron
4	Londres
5	37535795
6	ada@lon.uk

Estudiante 2	
1	125311
2	Maria
3	Curie
4	Varsovia
5	678433
6	maria@var.pol

Estudiante 3	
1	125312
2	Muhammad
3	al-Khwarizmi
4	Jiva
5	646456456
6	algor@jiva.uz



Del ejemplo anterior, podemos concluir que los estudiantes poseen datos y les podemos aplicar funciones similares. El hecho de agrupar los datos de los estudiantes y las funciones que podemos hacer con ellos en un único trozo de código se llama "**clase**". Cada estudiante es un **Objeto** o **Instancia** de la clase. Las variables que caracterizan a un estudiante (nombre, teléfono, dirección, etc.) se llaman **propiedades** y las funciones que podemos aplicarles se llaman **Métodos**.



Recuerda

Llamamos **clase** al agrupamiento de datos y de funciones en un único trozo de código independiente.

Objeto o **Instancia** de la clase son todos los elementos independientes que resultan de usar la clase.

Las variables de un objeto se llaman **propiedades**.

Las funciones que aplicamos a un objeto se llaman **Métodos**.

Las ventajas de usar clases son la flexibilidad en el uso del código, mayor velocidad y facilidad de desarrollo de aplicaciones y la posibilidad de reutilizar el código en otros programas.

Ejemplo de clase en B4J

Una biblioteca posee un conjunto de libros que presta a los lectores registrados. Cada libro tiene propiedades como el título, el autor, la editorial o el año de publicación. Los libros se puede insertar, mostrar o cambiar.

Crea un aplicación en B4J que implemente la clase Libro con las propiedades y métodos descritos.

Metodología de implementación

1. Crea un aplicación **B4XPages** y ponle de nombre "biblioteca".
2. Elige la opción **Proyecto – Añadir nuevo módulo – Módulo de Clase - Standard Class**.
3. En la ventana, ponle de nombre **clsLibro** (significa: clase Libro)
4. Aparecerá una nueva pestaña llamada **clsLibro**.

```
1 Sub Class_Globals
2     Private fx As JFX
3     Public strEscritor, strT
4 End Sub
5
6 'Inicializa el objeto. Puede
7 Public Sub Initialize
8
9 End Sub
```

5. Dentro de la rutina **Class_Globals**, añade las variables que representarán las propiedades de la clase:
 - a. Título del libro
 - b. Nombre del autor
 - c. Editorial
 - d. Año de publicación



```

Sub Class_Globals
    Private fx As JFX
    Public strEscritor, strTítulo, strAño, strEditorial As String
End Sub

```

6. Finalmente, implementa las subrutinas que ejecutarán los métodos:
- Insertar un libro
 - Mostrar un libro
 - Cambiar un libro



Consejo para el profesor

Ten cuidado de explicar que todas las variables y métodos que hemos creado son públicos, con lo que los objetos podrán usarlos.

Insertar un libro

Esta subrutina aceptará 4 cadenas de texto como parámetros y las asignará en el orden en que aparecen a las variables **strTítulo**, **strEscritor**, **strAño** y **strEditorial**.

```

14 Public Sub insertarLibro(str1, str2, str3, str4 As String)
15     strTítulo = str1
16     strEscritor = str2
17     strAño = str3
18     strEditorial = str4
19 End Sub

```

Mostrar libro

Esta subrutina mostrará con el comando "Log" las propiedades del libro indicado.

```

21 Public Sub mostrarLibro
22     Log("Título : " & strTítulo)
23     Log("Escritor : " & strEscritor)
24     Log("Año : " & strAño)
25     Log("Editorial : " & strEditorial)
26 End Sub

```

Cambiar libro

Este método aceptará 4 parámetros para cambiar las propiedades del libro en el mismo orden que en el método "insertarLibro":

```

28 Public Sub cambiarLibro(str1, str2, str3, str4 As String)
29     strTítulo = str1
30     strEscritor = str2
31     strAño = str3
32     strEditorial = str4
33 End Sub

```

La rutina "Initialize"

Esta rutina se usa para dar valores iniciales a las variables o bien para realizar alguna otra acción al crear un objeto de la clase:



```

7 Public Sub Initialize
8     strTitulo = ""
9     strEscriitor = ""
10    strAño = ""
11    strEditorial = ""
12 End Sub

```

Cómo usar la Clase

Volvemos a la pestaña **B4XMainPage** para usar la clase **clsLibro**.

1. Primero creamos los objetos de la clase clsLibro.

```

7 Sub Class_Globals
8     Private Root As B4XView
9     Private xui As XUI
10    Private libro1 As clsLibro
11    Private libro2 As clsLibro
12 End Sub

```

donde libro1 y libro2 son dos objetos de la clase clsLibro con todas las propiedades y métodos discutidos anteriormente.

Uso de los métodos

```

19 Private Sub B4XPage_Created (Root1 As B4XView)
20     Root = Root1
21     Root.LoadLayout("MainPage")
22
23     libro1.Initialize
24     libro2.Initialize
25
26     libro1.insertarLibro("Neuromante", "William Gibson", "1984", "Ace")
27     libro2.insertarLibro("2001: Una odisea del espacio", "Arthur C. Clarke", "1968", "Ace")
28
29     libro1.mostrarLibro
30     libro2.mostrarLibro
31 End Sub

```

El primer método que se usará en los objetos es el método **Initialize**.



Recuerda

Initialize **no es un método opcional**. Es el primer método que hay que invocar antes de usar un objeto.

El método **insertarLibro** introduce los valores de los dos libros en las propiedades de los objetos.

El método **mostrarLibro** muestra las propiedades de cada libro.

```

Titulo : Neuromante
Escriitor : William Gibson
Año : 1984
Editorial : Ace
Titulo : 2001: Una odisea del espacio
Escriitor : Arthur C. Clarke
Año : 1968
Editorial : Ace

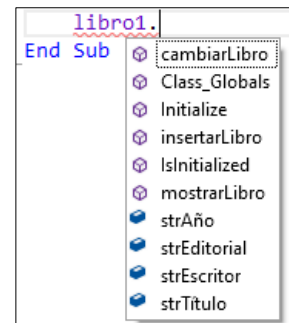
```



Recuerda

Cada propiedad se puede consultar escribiendo el nombre del objeto, seguido de un punto y después el nombre de la propiedad o método. Algunos métodos necesitan parámetros y otros no.

Cada vez que escribas el nombre de un objeto y pulses la tecla con el punto, B4X mostrará una ventana con todas las propiedades y métodos de la clase que hay disponibles. El método **IsInitialized** comprueba si el objeto está inicializado y existe en todas las clase que has creado.



Ejercicios

1. Siguiendo el primer ejemplo del tema, implementa la clase estudiante en B4J con las siguientes propiedades:

- Identificador escolar
- Nombre
- Apellidos
- Clase
- Teléfono
- Correo electrónico

Y los métodos:

- Nuevo estudiante
- Mostrar estudiante
- Cambiar estudiante
- Cambiar teléfono

2. Supongamos que un profesor sólo imparte una materia en un colegio. Implementa la clase Curso con las siguientes propiedades:

- a. Tema
- b. Clase
- c. Horas
- d. Profesor

Y los métodos:

- a. Nuevo Curso
- b. Cambiar Horas
- c. Cambiar Profesor
- d. Mostrar Curso

3. Una tienda vende ordenadores. De cada uno guarda lo siguiente:

- a. El tipo (sobremesa, portátil)
- b. El modelo
- c. El precio
- d. Su CPU (i3, i5, i7, i9)

Crea una clase que implemente un ordenador con las propiedades anteriores y los métodos nuevoOrdenador, mostrarOrdenador, cambiarCPU y cambiarPrecio. Comprueba que los valores introducidos en CPU y en Tipo son los que aparecen entre paréntesis.

Tema 10 – B4XPages

🕒 3h

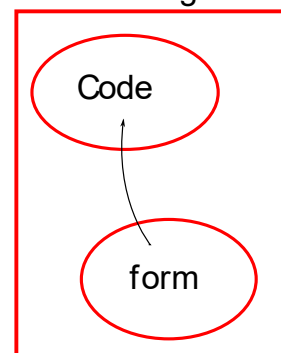
Lo que los estudiantes aprenderán

- Qué es una B4XPage
- Cómo crear y borrar una B4XPage
- Paso de valores entre páginas

B4XPages es una biblioteca de software. Incluye clases y métodos para crear múltiples formularios de comunicación con el usuario. Además, ayuda a portar aplicaciones a diferentes plataformas usando las herramientas de B4A, B4i y B4J.

Cada aplicación que has creado con B4J ya incluye una B4XPage. Se trata de la B4XMainPage que siempre es el primer formulario que se muestra al usuario. De forma más general, podemos decir que cada B4XPage gestiona todo el código necesario para que la interfaz de usuario (GUI) funcione.

B4XMainPage



La estructura de las carpetas de una aplicación

Cuando se crea un nuevo programa con B4XPage, se crea la siguiente estructura de carpetas:

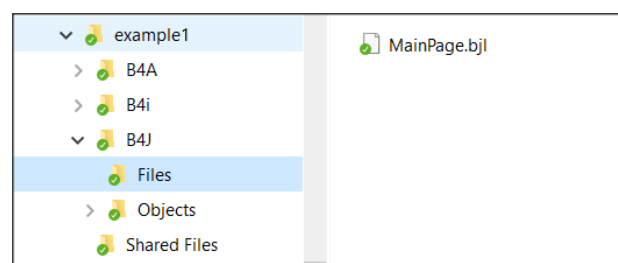


Imagen 1. Carpetas del Ejemplo 1

Cada una de las carpetas B4A, B4i y B4J incluye el código necesario para crear aplicaciones para Android, iOS y PCs (Windows, Linux, etc.) respectivamente.

En concreto, en la carpeta **B4J** está la carpeta **Files** que

contiene todos los ficheros creados con el **Diseñador** y otros ficheros que se usan al ejecutar el programa como, por ejemplo, las imágenes. El fichero **MainPage.bjl** se crea automáticamente al crear la aplicación y es la pantalla de inicio del programa. La carpeta **Shared Files** incluye los ficheros que los 3 tipos de aplicaciones pueden compartir si el programador **desea** crear una aplicación para Android iOS y PC.



La carpeta raíz de la aplicación contiene todos los ficheros que crean las diferentes B4XPages de nuestra aplicación:

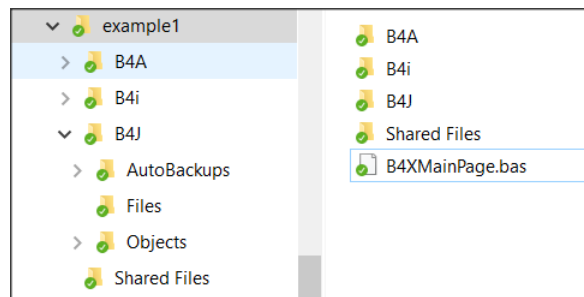


Imagen 2. Ficheros de la B4XPage

La primera página que se crea debe tener el nombre **"B4XMainPage.bas"** y no puede cambiarse; a todas las demás páginas se les puede cambiar el nombre.

Iniciar una aplicación con B4XPage

Al crear una nueva aplicación con B4Xpage, se crea automáticamente la primera página cuyo nombre es **B4XMainPage.bas**. Además, se crea un formulario (o pantalla de interfaz de usuario) para comunicarse con el usuario (que se llama **MainPage.bjl**). Para gestionar las páginas se crea también un mecanismo llamado B4XPagesManager.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
End Sub

Public Sub Initialize

End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
End Sub
```

Qué es Root

La variable **Root** es un objeto de la clase **B4XView**. Se encarga de gestionar la pantalla en los diferentes formularios que crea el programador (también está relacionado con la compartición de código en B4J, B4A, B4i). Así, con la sentencia **Root.LoadLayout("MainPage")**, el objeto Root carga en pantalla el formulario **MainPage**.

Crear una nueva B4XPage

Paso 1.

Creemos un formulario desde el menú **Proyecto – Añadir nuevo módulo – Módulo de Clase – B4XPage** y lo nombramos **B4XPage1**.

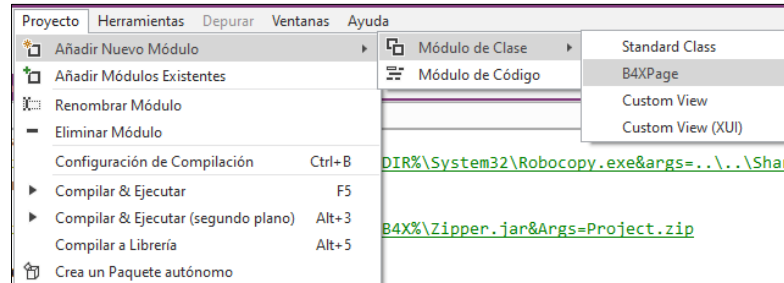


Imagen 21. Crear una B4XPage

Se creará una clase con el nombre "B4XPage1" más algún código básico para iniciarla. La interfaz de usuario (GUI) no se ha creado aún. Ello se hará después mediante el Diseñador.

```
Sub Class_Globals
    Private Root As B4XView 'ignore
    Private xui As XUI 'ignore
End Sub

'You can add more parameters here.
Public Sub Initialize As Object
    Return Me
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    'load the layout to Root
End Sub
```

Imagen 22. La nueva B4XPage

Paso 2.

Abre el **Diseñador** y elige en el menú **Archivo** la opción **Nuevo**.

En la pestaña **Variantes** indica las dimensiones del formulario que quieras diseñar y añade una etiqueta y un botón al formulario como se ve en la Imagen 3.

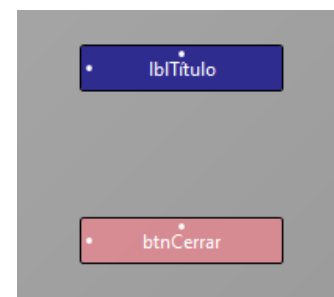


Imagen 23. Formulario

Paso 3.

Elige la opción **Herramientas – Generar** Miembros para insertar los dos objetos en tu código junto con el evento Clic del botón. Recuerda que esta acción debe hacerse cuando estés en el código de la **B4XPage1**.

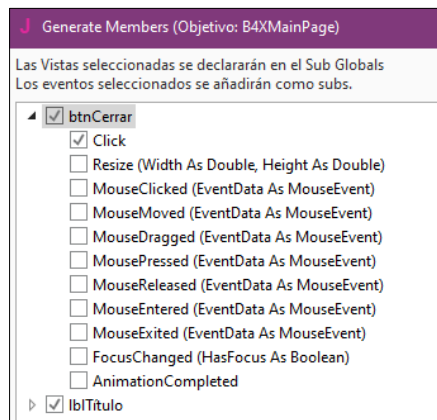


Imagen 24. Generar Miembros

Desde el menú **Archivo – Salvar** guarda el formulario con el nombre **frmPage1** (puedes elegir un nombre más representativo en otra aplicación).

Se creará el siguiente código (Imagen 5) y el fichero **frmPage1.bjl** aparecerá en la carpeta **“Files”**.

```
1 Sub Class_Globals
2     Private Root As B4XView 'ignore
3     Private xui As XUI 'ignore
4     Private btnCerrar As Button
5     Private lblTitulo As Label
6 End Sub
7
8 'You can add more parameters here.
9 Public Sub Initialize As Object
10     Return Me
11 End Sub
12
13 Private Sub B4XPage_Created (Root1 As B4XView)
14     Root = Root1
15     'load the layout to Root
16
17 End Sub
18
19
20 Private Sub btnCerrar_Click
21
22 End Sub
```

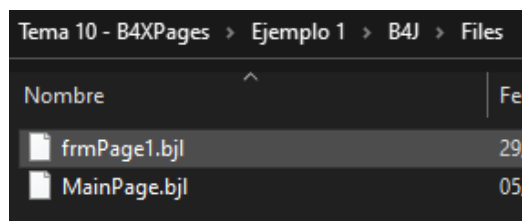


Imagen 25. frmPage1



Paso 4.

Para enlazar el formulario frmPage1 con B4XPage1 hay que invocar al método LoadLayout con la sentencia Root.LoadLayout("frmPage1") dentro del evento B4XPage_Created:

```
Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    'load the layout to Root
    Root.LoadLayout("frmPage1")
End Sub
```

Los siguientes pasos consistirán en programar el resto de botones y acciones en función de lo que el programa quiera hacer.

En nuestro ejemplo, hemos hecho que se cambie a la pantalla B4XPage1 al hacer clic en un botón de la B4XMainPage y que se vuelva a la B4XMainPage al hacer clic en el botón que hemos creado en la B4XPage1.

Invocar a una nueva B4XPage

Cada B4XPage que creas es una clase. Así, antes de usarla hay que crear un objeto basado en ella. Esto se suele hacer en la B4XPage que invocará a la nueva página. Por lo tanto, en el ejemplo anterior escribimos lo siguiente en B4XMainPage (**iError! No se encuentra el origen de la referencia.**):

```
7 Sub Class_Globals
8     Private Root As B4XView
9     Private xui As XUI
10    Private página1 As B4XPage1
11    Private Boton1 As Button
12 End Sub
13
14 Public Sub Initialize
15
16 End Sub
17
18 Private Sub B4XPage_Created (Root1 As B4XView)
19     Root = Root1
20     Root.LoadLayout("MainPage")
21     página1.Initialize
22     B4XPAGES.AddPage("Mi primera página", página1)
23 End Sub
24
25
26 Private Sub Boton1_Click
27
28     B4XPAGES.ShowPage("Mi primera página")
29
30     B4XPAGES.ShowPageAndRemovePreviousPages("mi primera página")
31 End Sub
```

1. Creamos un objeto de la clase B4XPage1.

2. Invocamos el método **Initialize** para iniciar el objeto que acabamos de crear.

3. Creamos un identificador para la página (en el ejemplo se llama "Mi primera página").

4. Mostramos la nueva página mientras la principal sigue abierta.

5. Entre comentarios, indicamos cómo mostrar la nueva página cerrando la ventana anterior (puedes ver el resultado en la Imagen 7).

Imagen 26. Crear una B4XPage



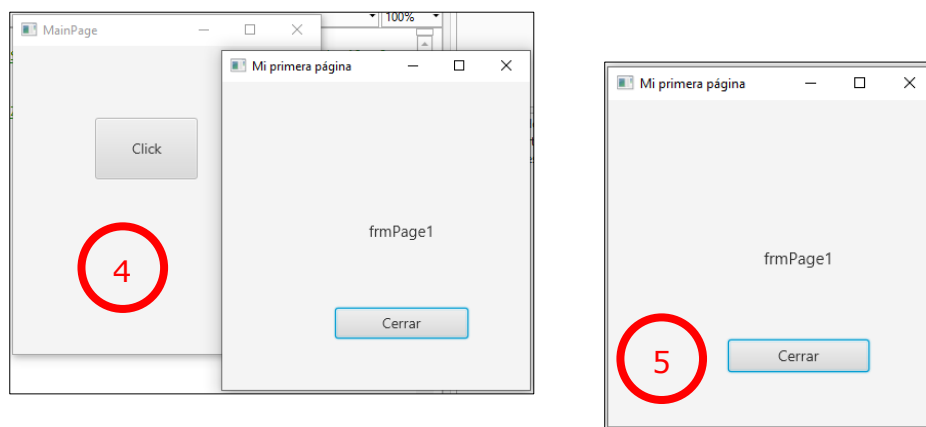


Imagen 27. Las dos formas de abrir una B4Xpage

Cerrar una B4XPage

Cuando se invoca a una B4XPage, el control del programa se pasa a esa página. Así, para cerrar una página tiene que ocurrir algún evento como, por ejemplo, pulsar un botón o pulsar el botón de cerrar ventana en la esquina superior derecha de la ventana. Normalmente, dependerá de cómo se haya abierto la página:

Cuando se abre con:	Normalmente se cierra con:
<code>B4XPages.ShowPage("Mi primera Página")</code>	<code>B4XPages.ClosePage(Me)</code>
<code>B4XPages.ShowPageAndRemovePreviousPages("Mi primera página")</code>	<code>B4XPages.ShowPageAndRemovePreviousPages("MainPage")</code>

Con la primera forma se cierra la ventana actual, mientras que con la segunda se abre la página principal (MainPage) cuando se cierra la ventana actual.

Transferir información entre páginas

Para que una página acceda a los datos de otra, deben tener sus variables declaradas con la palabra reservada **Public**. Las propias variables que contienen las páginas deben ser también públicas, con independencia de si se declaran en el **MainPage** o en otro lugar.



Ejemplo 2

Para este ejemplo usaremos la aplicación del ejemplo 2 que incluye 3 páginas: MainPage, B4XPage1 y B4XPage2. En el diseñador se han creado formularios también. Abre el ejemplo 2, ejecútalo y observa su comportamiento.

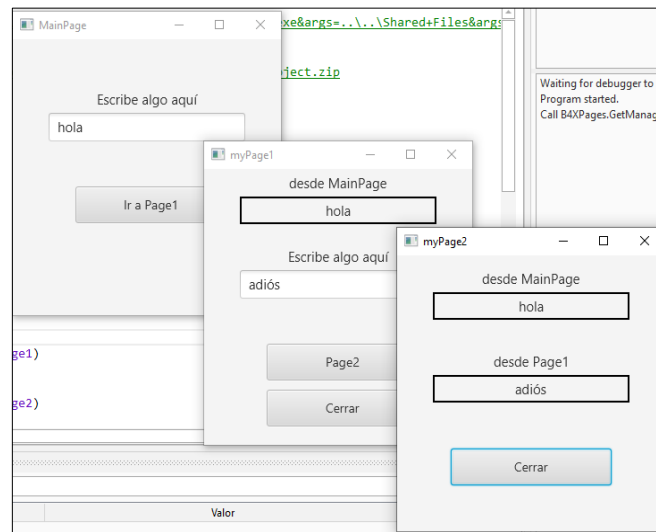


Imagen 28. Ejemplo 2

Al ejecutar la aplicación **Ejemplo 2**, fíjate que el texto de los TextFields se transfiere a las demás páginas. Esto es así porque la **page1**, la **page2** y el **TextFiled** se declararon como públicos.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Public page1 As B4XPage1
    Public page2 As B4XPage2
    Public txtGlobal As TextField
End Sub
```

```
Sub Class_Globals
    Private Root As B4XView 'ignore
    Private xui As XUI 'ignore
    Private lblGlobal1 As Label
    Private lblGlobal2 As Label
    Public txtGlobal2 As TextField
End Sub
```

Imagen 29. Declaraciones Public en MainPage y en B4XPage1

Para que **page1** tenga acceso a la variable **txtGlobal1**, debe indicarse el nombre de la página donde fue creada:

```
lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
```

donde **lblGlobal1** es una etiqueta que muestra el contenido leído en la pantalla de la **page1**.

Del mismo modo, **Page2** tiene acceso a la variable **txtGlobal1** de la **MainPage** y a la variable **txtGlobal2** de la **Page1** haciendo lo siguiente:

```
lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
lblGlobal2.Text = B4XPages.MainPage.page1.txtGlobal2.Text
```

donde **lblGlobal1** y **lblGlobal2** son dos etiquetas que muestran los contenidos de dos variables públicas en la pantalla de la **page2**.



La Vida de las B4XPages

En el anterior ejemplo, prueba a cerrar todas las ventanas excepto **MainPage**, escribe un nuevo texto y pulsa el botón **Ir a Page1**. Verás que el valor mostrado no es el nuevo, sino el primero que escribiste.

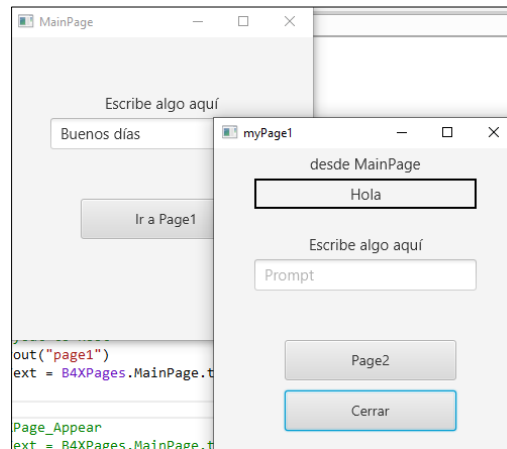


Imagen 30. Vida de una B4XPage

Esto sucede porque las B4XPages permanecen en la memoria del ordenador, así que el evento **B4XPage_Created** no se ejecuta de nuevo cuando volvemos a abrir la página. Para evitar esto, podemos usar el evento **B4XPage_Appear** para volver a leer las variables desde MainPage:

```
Private Sub B4XPage_Appear
    lblGlobal1.Text = B4XPages.MainPage.txtGlobal.Text
End Sub
```

Al contrario que el evento **B4XPage_Created** que se ejecuta una única vez al crear por primera vez la página, el evento **B4XPage_Appear** se ejecuta cada vez que la página aparece en primer plano, con lo que puedes usarlo para transferir variables de unos formularios a otros.

Ejercicios

1. La pequeña enciclopedia de los perros. Crea una aplicación donde 3 razas de perros diferentes se muestren en una pantalla de inicio y, tras hacer clic en el nombre correspondiente, se muestre información acerca de la raza junto con dos fotos.

Puedes usar un TextArea en el diseñador para hacer más grandes los TextAreas con una barra de desplazamiento.

2. Construye una aplicación que resuelva:
 - a. La ecuación de primer grado $ax+b=0$,
 - b. La ecuación de segundo grado $ax^2+bx+c=0$
 - c. Calcule la hipotenusa de un triángulo dadas las longitudes de los dos lados verticales.

La raíz cuadrada de un número x se calcula con \sqrt{x} .

3. Construye una aplicación que cree una tienda virtual como la siguiente: la pantalla de inicio mostrará 4 imágenes de diferentes objetos (por ejemplo, portátiles) y un TextField por cada elemento donde el cliente escribe la cantidad. Después, pulsando el botón **Comprar** el programa mostrará en una nueva página el Valor Total y la cantidad de elementos elegidos. Sin contar con MainPage, sólo debes usar una única página más.



Tema 11 – Primer Proyecto

🕒 5h

Lo que los estudiantes aprenderán

- Usar los conocimientos previos para crear su primera aplicación.

Aplicación de tienda de móviles

Una tienda vende móviles. Para cada móvil se guardan las siguientes características:

- Nombre del modelo
- Tamaño de la pantalla
- Capacidad de almacenamiento
- RAM
- Procesador
- Cantidad en el almacén
- Sistema Operativo (Android, iOS)
- Precio

Construye una aplicación que haga lo siguiente:

1.

1.1. Crea una clase llamada **Phone** con las propiedades anteriores.

1.2. Crea los métodos siguientes:

1.2.1. Initialize (Inicializa todas las cadenas con el valor "" y todos los números con el valor 0).

1.2.2. nuevoMóvil (con los valores adecuados del nuevo móvil).

1.2.3. venderMóvil (acepta un número y resta esa cantidad a la cantidad total disponible de ese modelo).

2. Añade 8 modelos diferentes de móviles con las siguiente propiedades (sus características no son reales, pero se ponen como ejemplo):

	Nombre modelo	Pantalla	Capacidad	Ram	Cpu	Os	Cantidad	Precio
1	Yallomi	6.53"	64	4	Mediatek	Android	12	150\$
2	Smith	6.67"	64	4	Qualcomm	Android	4	220\$
3	Taurus	6.1	128	4	Bionic	iOS	7	780\$
4	Talisman	5.8"	64	4	Mediatek	Android	12	150\$
5	Cranberry	5.8"	32	3	Mediatek	Android	16	130\$
6	OzOn	5.8"	32	2	Mediatek	Android	16	90\$
7	H2O	5.8"	64	3	Qualcomm	Android	2	170\$
8	Zeus	6.67"	128	6	Qualcomm	Android	4	650\$

Para cada una de las páginas que vienen a continuación, es recomendable que dibujes un borrador en papel o con Inkscape. Puedes descargarlo gratis de: <https://inkscape.org/>.

3. Crea la pantalla de inicio con las siguientes opciones:
 - Vender móvil
 - Stock del almacén.
 - Ingresos totales de la tienda
4. Si se elige la opción "Vender móvil", crea una nueva B4XPage y muestra los nombres de los 8 modelos y sus precios. No olvides hacer un borrador primero.



Consejo para el profesor

Quizás tengas que ayudar a tus estudiantes a invocar objetos desde la página principal

- 4.1. Al hacer clic en cada teléfono se mostrará una ventana con las propiedades de cada móvil y 2 fotos tuyas. Además, se mostrará un botón para vender el móvil, otro para volver atrás y un cuadro de texto donde introducir la cantidad que se quiere vender.
- 4.1.1. Si se pulsa el botón "Volver", se volverá a la pantalla anterior.
- 4.1.2. Si se pulsa el botón "Vender", se comprueba si hay suficientes móviles en stock. Si no hay, se mostrará un mensaje de error. Si hay suficientes, se abrirá una ventana donde se pedirán los siguientes datos del cliente:

Nombre:
Apellidos:
Dirección:
Móvil:

- 4.1.3. En la misma ventana se mostrará el número de móviles que va a comprar junto con el importe total de la compra.
- 4.1.4. Cuando se pulse el botón "Vender" se restará la cantidad comprada del stock del almacén y se volverá a la pantalla de inicio.
5. Si se elige la opción "Stock de Almacén" se mostrarán los modelos de móviles en una nueva pantalla (B4XPage) junto con el stock de cada modelo que hay en el almacén.
6. Si se elige la opción "Ingresos totales de la tienda", se mostrarán los beneficios totales de la tienda en una nueva pantalla (B4XPage).



Consejo para el profesor

En las siguientes páginas se muestran varios documentos para ayudar a los estudiantes a realizar este proyecto. En la presentación de Power Point están todos los pasos de la solución. La solución no es "óptima", pero es la que pueden entender los estudiantes por ahora.

Planificación

Proyecto

Fecha

Estudiante

Organiza tu tiempo – Lo que harás y cuándo

Horas

	1	2	3	4	5	6	7	8	9



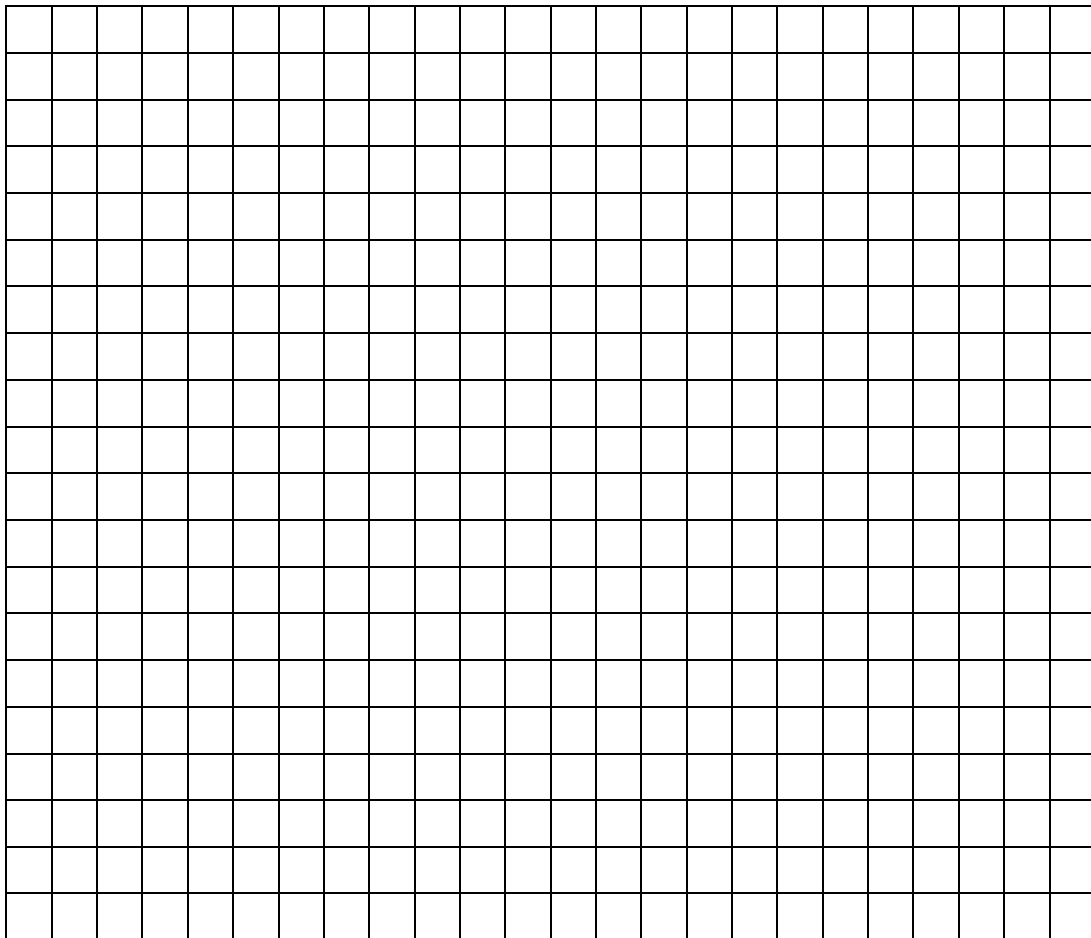


Borrador de las pantallas

Nombre del
Formulario

Fecha

Estudiante



Nombre Clase:

Fecha:

Estudiante:

Atributos:

1.

2.

3.

4.

5.

6.

7.

8.

Método

Descripción:

Método

Descripción:

Método

Descripción:

Método

Descripción:



Nombre B4XPage:

Fecha:

Estudiante:

Variables Públicas:	1.
	2.
	3.
	4.
	5.
	6.
	7.
	8.
	9.
	10.

Evento

Descripción:

Evento

Descripción:

Evento

Descripción:

Tema 12 – Bucles

🕒 4h

Lo que los estudiantes aprenderán

- ¿Qué son los bucles?
- Do While
- Do Until
- For – Next
- Algoritmos con bucles

Las estructuras de repetición son las más importantes de un lenguaje de programación. Un bucle o lazo repite un conjunto de sentencias hasta que una condición se alcanza. En una estructura de repetición se establece una pregunta. Si la respuesta es correcta, se ejecuta el bucle. Esta misma pregunta se pregunta una y otra vez hasta que la respuesta sea falsa y entonces no se ejecutan las acciones de dentro del bucle. Cada vez que se realiza la pregunta se habla de “repetición” o “iteración”.

Un programador que utiliza las mismas líneas de código múltiples veces en un programa puede usar un bucle para ahorrar tiempo, espacio y para facilitar la programación.



Recuerda

Los bucles deben terminar en algún momento, si no, se trata de un error de programación. Un bucle que nunca acaba se llama “bucle infinito”. En ese caso el ordenador se queda atrapado en una repetición perpetua sin posibilidad de salir de ella y sin que sea consciente de lo que sucede.

En B4X hay 4 **sentencias de repetición**: For, Do-While, Do Until, For each.

La sentencia Do-While

Imagina que quieres mostrar en pantalla con la sentencia “Log” los números del 1 al 5. El código que deberías escribir sería el siguiente:

```
Log (1)  
Log (2)  
Log (3)  
Log (4)  
Log (5)
```

Como ves, si quisieras escribir más números tendrías que escribir tantas sentencias Log como números quieras mostrar.



La sentencia Do While se escribe de la siguiente forma:

Do While Condición

Sentencias

...

Loop

Mientras la condición sea cierta

Ejecutar las sentencias

...

Volver a la condición.

Imagen 31. Sentencia Do While

El ejemplo anterior para escribir 5 números sería así:

```
Private i as Int = 1  
  
Do While i <= 5  
    Log(i)  
    i = i + 1  
Loop
```

La variable **i** cuenta cuántas veces se ha ejecutado el bucle.

La condición comprueba si el contenido del contador excede el límite fijado por el programador (p. ej. 5) y, si sucede, el bucle finaliza.

Antes de que la iteración del bucle acabe, el contador se incrementa en 1. A este valor también se le llama "paso".

Este paso puede ser negativo o positivo, pero nunca puede ser 0, porque entonces el bucle nunca acabaría.



Recuerda

Cuando el contador comienza con un valor menor que el final, el **paso** debe ser **positivo**.

Cuando el contador empieza con un valor mayor que el final, el **paso** debe ser **negativo**.



Ejemplos de la sentencia Do While.



Ejemplo 1

Mostrar todos los enteros del 100 al 1

```
Private i as Int = 100

Do While i >= 1
    Log(i)
    i = i - 1
Loop
```

En este ejemplo, el contador comienza con el valor 100 y el bucle acaba cuando alcance el valor 0 (va de un valor mayor a uno menor). Fíjate que en este caso el **Paso** es **Negativo** (-1).



Ejemplo 2

Mostrar los números pares del 1 al 100

Aquí mostramos dos soluciones; la primera usa una sentencia "if" dentro del bucle para comprobar si el número es par y ejecutar así la sentencia "log" con el número. La otra solución usa como **valor inicial** en el **contador** "i" el número 2 y en cada **paso** suma 2; es un algoritmo más rápido.

```
Private i as Int = 1

Do While i <= 100
    If i mod 2 = 0
    then
        Log(i)
    End if
    i = i + 1
Loop
```

```
Private i as Int = 2

Do While i <= 100
    Log(i)
    i = i + 2
Loop
```





Ejemplo 3 – Algoritmo de suma

Construye un programa que genere 10 números y calcule su suma. Los números deben estar en el intervalo -100 a 100

En este ejemplo usamos la función "Rnd" que se emplea así:



Recuerda

La función **Rnd (Primer Valor, Último Valor)** devuelve un número entre el primer y el último valor.

Ej.: A = Rnd(1, 10) guarda en la variable "A" un número entre 1 y 10

```
Private I As Int = 1
Private A as Int
Private intSuma as Int = 0

Do While i <= 10
    A = Rnd(-100, 100)
    intSuma = intSuma + A

    i = i + 1
Loop
```



Ejemplo 4 – Algoritmo para contar

Construye un programa que genere 10 números entre -1000 y 1000 y cuente cuántos negativos hay.

```
Private I As Int = 1
Private A as Int
Private intContador as Int = 0

Do While i <= 10
    A = Rnd(-100, 100)
    If A < 0 then
        intContador = intContador + 1
    End If
    i = i + 1
Loop
```





Ejemplo 5 – Algoritmo Máximo-Mínimo

Construye un programa que genere 10 número entre -1000 y 1000 y que calcule el mayor y el menor.

```
Private I As Int = 1
Private A As Int
Private intMax, intMin As Int

A = Rnd(-100, 100) 1
intMax = A 2
intMin = A
Do While i <= 9
    A = Rnd(-100, 100)

    If intMax < A then
        intMax = A
    End If 3

    If intMin > A then
        intMin = A
    End If

    i = i + 1
Loop
```

1. Primero creamos un número fuera del bucle while.
2. Fijamos un valor inicial para intMax e intMin igual al primer número generado, ya que no hay ninguno con el que comparar de momento.
3. Para cada nuevo número, comprobamos si es menor que intMin (si es así, reemplazamos intMin por el contenido de "A"). Si es mayor que intMax, reemplazamos intMax por A.

Bucles con un número de repeticiones desconocidas

A menudo en programación no se sabe inicialmente cuántas veces se repetirá un bucle. Esto suele ser así cuando leemos un valor introducido por el usuario o bien cuando realizamos algún cálculo dentro del bucle.



Ejemplo 6 – Número de repeticiones indeterminado

Crea un programa que continuamente lea números y cuente cuántos hay pares. El programa acabará cuando se introduzca un número menor que 0.



```

Private A as Int
Private intContador as Int = 0

A = Rnd(-100, 100) 1
Do While A > 0 2
    If A mod 2 = 0 then
        intContador = intContador + 1
    End If

    A = Rnd(-100, 100) 3
Loop

```

1. Generamos un número fuera del bucle.
2. La condición del Do-While comprueba si el número A está en el rango correcto.
3. Se lee un nuevo número antes de finalizar la iteración del bucle.



Ejemplo 7 – Número de repeticiones indeterminado

Crea un programa que lea número y calcule su suma. El programa acabará cuando la suma supere 200.

```

Private A as Int
Private intSuma as Int = 0

Do While intSuma <= 200 1
    A = Rnd(-100, 100)
    intSuma = intSuma + A 2
Loop

```

1. La condición comprueba que la suma no supere 200.
2. Se lee un número y el programa lo añade a la suma. La condición se comprueba de nuevo en el bucle Do While.

La sentencia Do Until

El funcionamiento del bucle Do Until es similar al Do While. La única diferencia es que la condición para finalizar el bucle es la contraria que en el bucle Do-While. La comprobación se realiza en ambos casos al principio del bucle. Las sentencias de dentro del bucle no se ejecutarán si la condición es Falsa.



Recuerda

La condición de la sentencia Do Until es la negación de Do While.



Ejemplo 1

Muestra todos los números del 100 al 1

```

Private i as Int = 100

Do Until i < 1
    Log(i)
    i = i - 1
Loop

```





Ejemplo 2

Mostrar todos los números pares del 1 al 100.

```
Private i as Int = 1

Do Until I > 100
    If I mod 2 = 0 then
        Log(i)
    End if
    i = i + 1
Loop
```

```
Private i as Int = 2

Do Until i > 100
    Log(i)
    i = i + 2
Loop
```



Ejemplo 3 – Algoritmo para sumar

Construye un programa que genere 10 números y calcule su suma. Los números deben estar en el intervalo -100 a 100

```
Private I As Int = 1
Private A as Int
Private intSuma as Int = 0

Do Until I > 10
    A = Rnd(-100, 100)
    intSuma = intSuma + A

    i = i + 1
Loop
```



Ejemplo 4 – Algoritmo para contar

Construye un programa que genere 10 números entre -1000 y 1000 y cuente cuántos negativos hay.

```
Private i as Int = 1
Private A as Int
Private intCounter as Int = 0

Do Until i > 10
    A = Rnd(-100, 100)
    If A < 0 then
        intCounter = intCounter + 1
    End If
    i = i + 1
Loop
```





Ejemplo 5 – Algoritmo Máximo-Mínimo

Construye un programa que genere 10 número entre -1000 y 1000 y que calcule el mayor y el menor.

```
Private I As Int = 1
Private A as Int
Private intMax, intMin as Int

A = Rnd(-100, 100)
intMax = A
intMin = A
Do Until I > 9
    A = Rnd(-100, 100)

    If intMax < A then
        intMax = A
    End If

    If intMin > A then
        intMin = A
    End If

    i = i + 1
Loop
```



Ejemplo 6 – Número de repeticiones indeterminado

Crea un programa que continuamente lea números y cuente cuántos hay pares. El programa acabará cuando se introduzca un número menor que 0.




```

Private A as Int
Private intContador as Int = 0

A = Rnd(-100, 100)
Do Until A < 0

    If A mod 2 = 0 then
        intContador = intContador + 1
    End If

    A = Rnd(-100, 100)
Loop

```



Ejemplo 7 – Número de repeticiones indeterminado

Crea un programa que lea número y calcule su suma. El programa acabará cuando la suma supere 200.

```

Private A as Int
Private intSuma as Int = 0

Do Until intSuma > 200
    A = Rnd(-100, 100)
    intSuma = intSuma + A
Loop

```

Bucle For

El bucle For es seguramente la sentencia de repetición más simple:

```

for i = n1 to n2 Step n3
    Sentencias
...
Next

```

Donde:

i es la variable contador
n1 el valor inicial del contador
n2 el valor final del contador
n3 el paso tras cada iteración

- Al inicio del bucle, la variable "i" contendrá el valor "n1".
- Se ejecutan las sentencias dentro del bucle.
- Al final de la iteración, el valor de "i" se incrementa o decrementa en "n3"
- Se comprueba si "i" ha superado el valor "n2":

- Si el paso es positivo y "i" es menor o igual al valor final, entonces el bucle se ejecuta de nuevo
- Si el paso es negativo y "i" es mayor o igual al valor final, entonces el bucle se ejecuta de nuevo.



Recuerda

Comparado con los bucles **Do While** y **Do Until**, el bucle **For**:

- No necesita inicializar el contador.
- No necesita efectuar ninguna operación con el paso.
- Siempre sabe el número de repeticiones que hará (aunque se puede usar la sentencia "exit" para acabar el bucle cuando quieras).

Ejemplos de bucle For



Ejemplo 1

Mostrar los números del 100 al 1

```
Private I As Int

For i = 100 to 1 step -1
    Log(i)
Next
```



Ejemplo 2

Mostrar los números pares del 1 al 100

```
Private I As Int

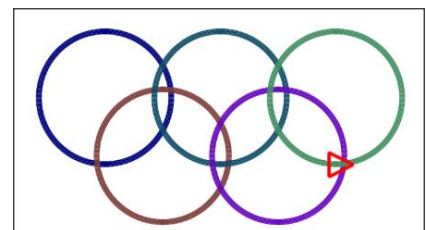
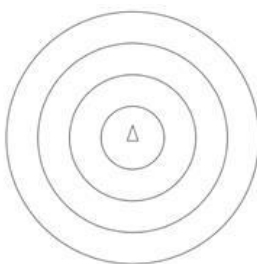
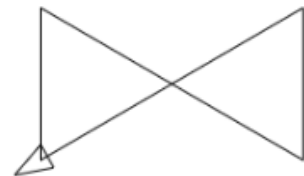
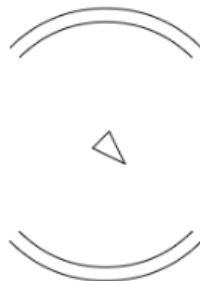
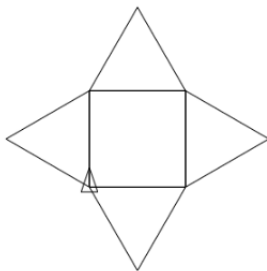
For i = 1 to 100
    If i mod 2 = 0 then
        Log(i)
    End If
Next
```

Ejercicios

1. Escribe un programa que lea un número entero K entre 1 y 200 y calcule la suma de $1+2+3+4+\dots+K$.
2. Escribe un programa donde el usuario introduzca números y calcule la media de los números. El programa finaliza cuando se introduce el 0.
3. Un cliente quiere comprar juguetes para regalar a 10 niños. Desea que el precio total no supere 200€. Escribe un programa que:



- a. Genere un precio para un juego (número aleatorio entre 10 y 50, usando la función `Rnd(10, 50)`).
 - b. Calcule el precio total de los juegos comprados.
 - c. Compruebe si el dinero se ha acabado.
 - d. Muestre al final el precio total de los juegos y el número de juegos que se han comprado.
4. Un año bisiesto es aquel al que se le añade un día adicional (en un calendario lunar, se añade un mes) para mantener el año sincronizado con el año astronómico o año solar. Para comprobar si un año es bisiesto:
 - Si el año no es divisible por 4, NO es bisiesto
 - Si no, si el año no es divisible por 100, entonces el año es bisiesto.
 - Si no, si el año no es divisible por 400, entonces es un año normal.
 - Si no, es un año bisiesto.Escribe un programa que muestre los años bisiestos desde 1900 a 2100. (https://es.wikipedia.org/wiki/Año_bisiesto)
5. Construye un programa que con la ayuda de la Tortuga genere polígonos con un número de ángulos introducido por el usuario en el campo adecuado. El programa incluirá una interfaz donde haya un botón de inicio y un botón para borrar la pantalla que borrará la pantalla cuando se pulse y colocará a la tortuga en el centro de la pantalla.
6. Dibuja las siguientes formas con la Tortuga:



Tema 13 – XUI Views

🕒 **3h**

Lo que los estudiantes aprenderán

- Qué es una biblioteca
- Biblioteca XUI
- Dialogs – Diálogos
- Plantillas

Los lenguajes de programación poseen bibliotecas (es frecuente que se traduzca incorrectamente al español como “librerías”). En general, una biblioteca incluye bloques de código, estructuras, clases y métodos que pueden usar los programadores para facilitar el proceso de desarrollo de programas.

Bibliotecas en in B4X

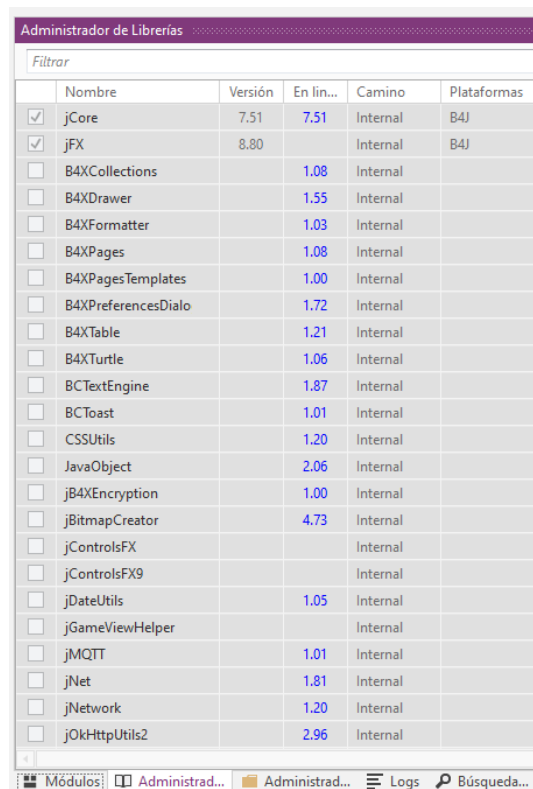
Las bibliotecas se suelen dividir en internas (se instalan junto con el propio lenguaje) y externas (creadas por el propio programados u obtenidas de otras fuentes como GitHub). Para usar una biblioteca, únicamente hay que elegirla de la lista de Bibliotecas de B4X.

La lista de bibliotecas ofrece información útil como:

- La versión actual de la biblioteca que está instalada,
- La última versión publicada para poder actualizar,
- Si es una biblioteca interna o externa,
- En qué plataforma funciona.

Para usar una biblioteca debes saber qué hace y cuáles son sus métodos y datos. Cada biblioteca hay información importante sobre su uso en la web del lenguaje en

<https://www.b4x.com/android/documentation.html>.



Administrador de Librerías					
Filtrar					
	Nombre	Versión	En lin...	Camino	Plataformas
<input checked="" type="checkbox"/>	jCore	7.51	7.51	Internal	B4J
<input checked="" type="checkbox"/>	jFX	8.80		Internal	B4J
<input type="checkbox"/>	B4XCollections		1.08	Internal	
<input type="checkbox"/>	B4XDrawer		1.55	Internal	
<input type="checkbox"/>	B4XFormatter		1.03	Internal	
<input type="checkbox"/>	B4XPages		1.08	Internal	
<input type="checkbox"/>	B4XPagesTemplates		1.00	Internal	
<input type="checkbox"/>	B4XPreferencesDialog		1.72	Internal	
<input type="checkbox"/>	B4XTable		1.21	Internal	
<input type="checkbox"/>	B4XTurtle		1.06	Internal	
<input type="checkbox"/>	BCTextEngine		1.87	Internal	
<input type="checkbox"/>	BCToast		1.01	Internal	
<input type="checkbox"/>	CSSUtils		1.20	Internal	
<input type="checkbox"/>	JavaObject		2.06	Internal	
<input type="checkbox"/>	jB4XEncryption		1.00	Internal	
<input type="checkbox"/>	jBitmapCreator		4.73	Internal	
<input type="checkbox"/>	jControlsFX			Internal	
<input type="checkbox"/>	jControlsFX9			Internal	
<input type="checkbox"/>	jDateUtils		1.05	Internal	
<input type="checkbox"/>	jGameViewHelper			Internal	
<input type="checkbox"/>	jMQTT		1.01	Internal	
<input type="checkbox"/>	jNet		1.81	Internal	
<input type="checkbox"/>	jNetwork		1.20	Internal	
<input type="checkbox"/>	jOkHttpUtils2		2.96	Internal	

Imagen 32. Administrador de Bibliotecas

Las bibliotecas externas deben copiarse en una carpeta específica de tu ordenador. Desde el menú "Herramientas", "Configurar Rutas", "Librerías Adicionales" puedes elegir la carpeta donde se guardarán.



Consejo para el profesor

Más abajo hay algunas "views" importantes. Hay algunas que no están, como el ComboBox, aunque las veremos con más detalle en próximos temas.

La Biblioteca "XUI Views"

El objetivo de la biblioteca XUI Views es permitir crear aplicaciones para B4J, B4A and B4i de forma similar.

Esta biblioteca permite crear formularios y vistas (views), a saber:

- **Views** (objetos)
 - **B4XComboBox** - ComboBox / Spinner / ActionSheet multiplataforma.
 - **ScrollingLabel** - Una etiqueta que permite desplazar el texto cuando es más ancho que la etiqueta.
 - **AnotherProgressBar** - Barra de progreso Vertical u horizontal animada.
 - **B4XLoadingIndicator** - 6 distintos indicadores de carga animados.
 - **RoundSlider** - Un deslizador redondeado.
 - **SwiftButton** - Botón 3D
 - **AnimatedCounter**
 - **B4XFloatTextField** - Un TextField / EditText con un consejo flotante.
 - **B4XSwitch** - Un bonito interruptor.
 - **B4XPlusMinus** - Permite al usuario elegir un número o un ítem de una lista previamente fijada.
 - **B4XBreadCrumb** - Control de navegación.
 - **B4XSeekBar** - Barra de búsqueda / deslizador horizontal o vertical.
 - **MadeWithLove** - Muestra tu amor por B4X :)
 - **B4XImageView** - ImageView con útiles modos de redimensionado.
 - **XUIViewsUtils** - Módulo con código estático de métodos útiles.
- **Dialogs** (Cuadros de diálogo)
 - Una clase que ofrece los servicios necesarios para mostrar un diálogo. Hay 3 métodos para mostrar diálogos: Mostrar - Mostrar un diálogo simple con texto, ShowCustom - Permite que se le envíe un diseño personalizado para mostrarlo en el



diálogo, ShowTemplate – Muestra un diálogo basado en una clase plantilla. Puedes ver el código fuente para la estructura de la plantilla. Es bastante sencillo.

- **Templates (Plantillas)**

- B4XDateTemplate – Basado en AnotherDatePicker.
- B4XColorTemplate – Selector de colores.
- B4XLongTextTemplate – Texto desplazable.
- B4XListTemplate – Una lista de ítems. El usuario puede elegir uno de los ítems.
- B4XSignatureTemplate – Captura la firma del usuario y añade una marca temporal a la imagen de mapa de bits.
- B4XInputTemplate – Plantilla para entradas de texto y numéricas.
- B4XSearchTemplate – Una lista con un campo de búsqueda.
- B4XTimedTemplate – Una plantilla que agrupa otras plantillas y crea un diálogo que se cierra automáticamente tras un tiempo prefijado con una barra de progreso bonita y animada.

Uso de XUI Views

Para usar las XUI Views primero hay que comprobar su nombre en la pestaña del Administrador de Librerías. Después vamos al diseñador para crear los objetos que quieras.



Ejemplo

El programa "Ejemplo" muestra el uso de:

ScrollingLabel

B4XFloatTextField

RoundSlider

AnotherProgressBar

B4XSwitch

B4XImageView

Etiqueta desplazable

Una etiqueta desplazable es una etiqueta donde el texto se puede desplazar para que se pueda mostrar completamente.

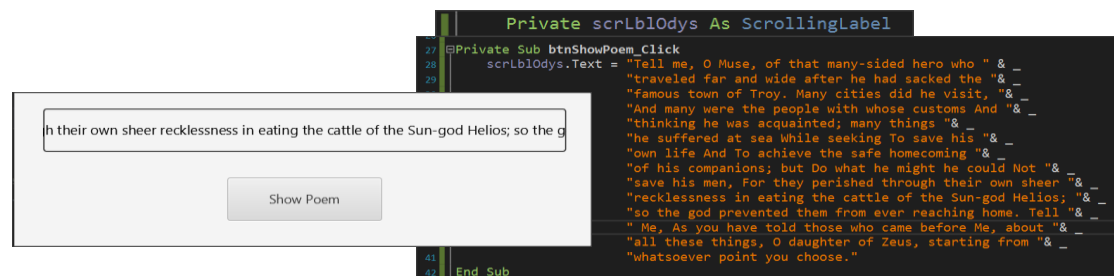


Imagen 33 Etiqueta desplazable

Se usa normalmente para textos muy grandes o para mostrar un mensaje móvil en la pantalla.

B4XFloatTextField

B4XFloatTextField crea un cuadro de texto en pantalla para introducir datos. Se usa de formar parecida a un cuadro de texto simple, pero muestra una

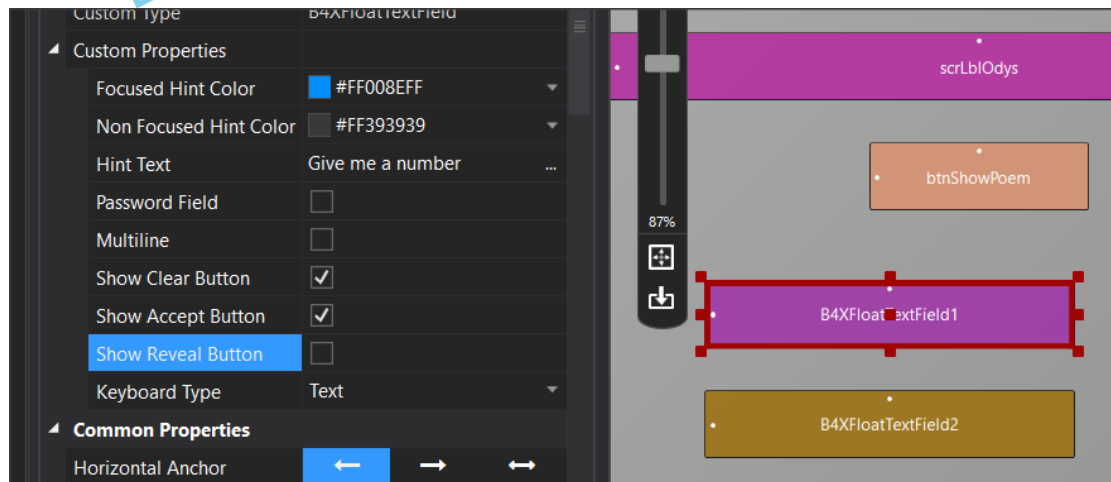


Imagen 34. B4XFloatTextField

etiqueta que ayuda al usuario a identificar el cuadro de texto sin tener que introducir una etiqueta adicional antes del mismo (Hint Text).

Además, permite mostrar los controles necesarios para borrar y activar el cuadro de texto (Show Clear Button, Show Accept button). Finalmente, con la propiedad "Keyboard Type" permite que se introduzca texto, números enteros o decimales.

RoundSlider

RoundSlider es un marcador que se puede mover con el ratón alrededor de un círculo. Para cada punto en el círculo muestra un valor dentro del rango de valores especificado en las propiedades Minimum – Maximum en el Diseñador.

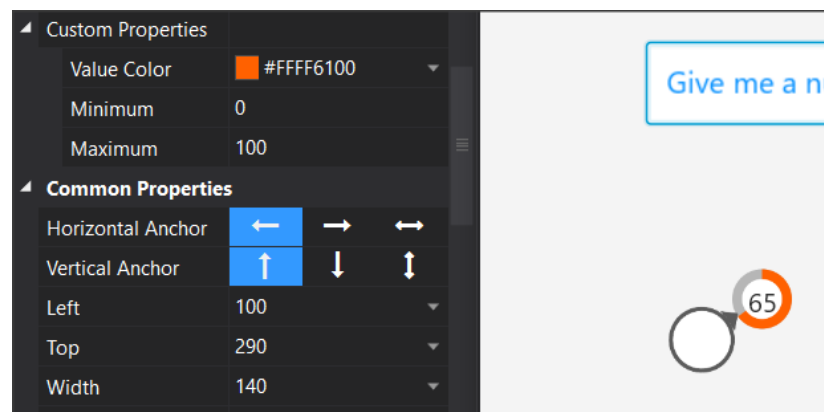


Imagen 35 RoundSlider

Cada vez que el valor del RoundSlider se cambia, se dispara el evento "_ValueChanged" y genera un valor que puede usar el programador.

AnotherProgressBar

AnotherProgressBar muestra una barra con valores entre 0 y 100 y se usa para representar porcentaje u otras proporciones. Su uso es sencillo; tras declarar la variable que representa al objeto, se usa la propiedad ".value" para fijar un valor.

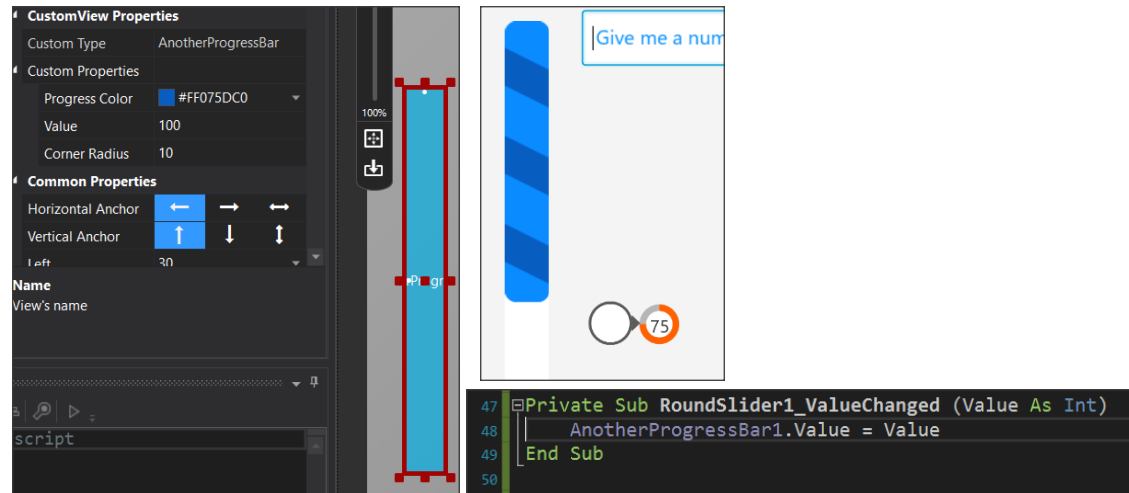


Imagen 36. AnotherProgressBar

En el ejemplo de la imagen, progressBar varía en función del valor recibido de RoundSlider.

B4XSwitch

B4XSwitch crea un interruptor en la pantalla.

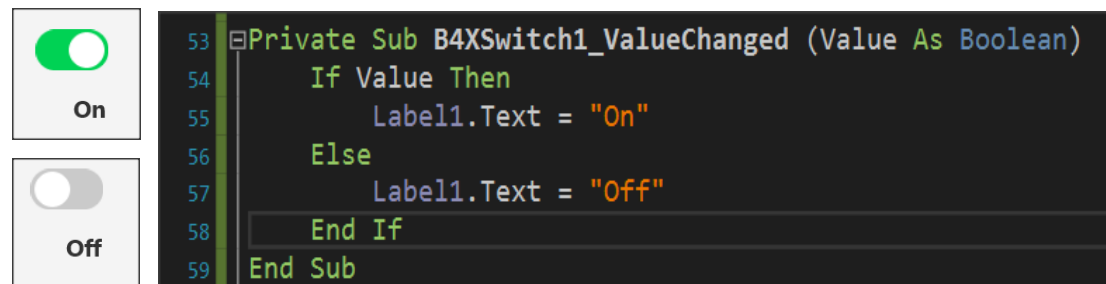


Imagen 37. B4XSwitch

En el ejemplo de la imagen, cuando el valor devuelto es True se muestra el texto "On" en la etiqueta "Label1"; en caso contrario muestra "Off".

B4XImageView

B4XImageView muestra una imagen. Permite elegir la imagen que se mostrará mediante código o mediante las opciones del objeto en el Diseñador.

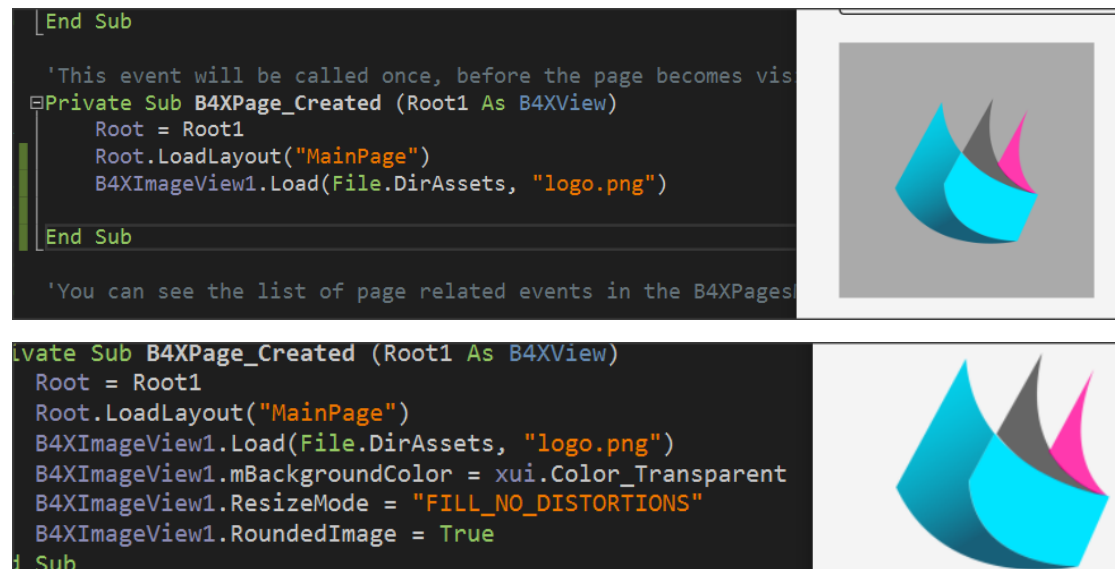
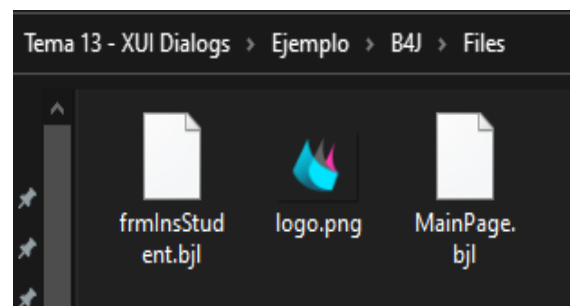


Imagen 38. B4XImageView

En el método "Load", además de la imagen, hay que incluir como primer argumento la carpeta donde está almacenada la imagen. En este caso, la carpeta "Files" (Files.DirAsset) es la que se usa por defecto (ver imagen):



Esta carpeta no es accesible para guardar nuevos ficheros cuando se crea la aplicación. En temas posteriores veremos con más detalle estas carpetas.

Imagen 39. Carpeta "Files" (DirAsset)

Para mostrar una imagen almacenada en la carpeta "Files" debe declararse también en la pestaña "Administrador de Archivos".

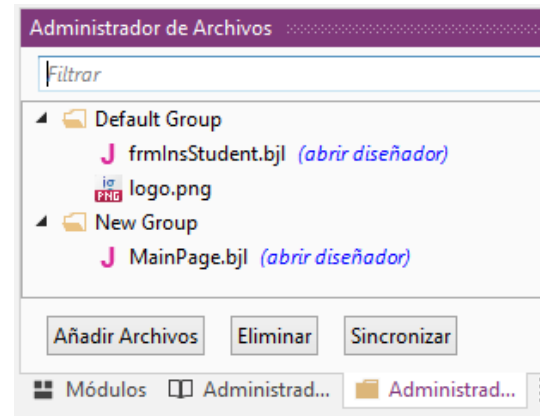


Imagen 40 Files Manager

B4XDialogs

Los cuadros de diálogo son pantallas que se usan para informar al usuario de un evento o para obtener datos de los usuarios sin necesidad de crear otra B4XPage.

MsgBoxAsync

Ya se ha usado anteriormente en otros temas. Es la función **xui.MsgBoxAsync** que muestra un mensaje en pantalla.

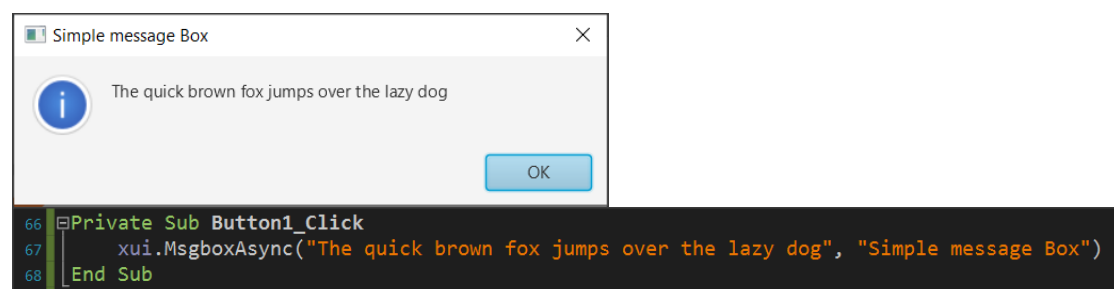
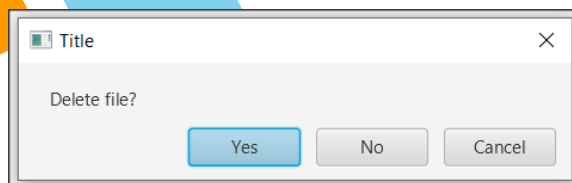


Imagen 41. MsgBoxAsync

Como se ve en la imagen anterior, también se muestra un botón OK para cerrar el mensaje y volver al programa.

MsgBox2Async

El uso de MsgBox2Async es más complejo y permite a los programadores mostrar mensajes y elegir entre 3 diferentes funciones pulsando el botón adecuado.



```
72 Private Sub Button2_Click
73     Dim sf As Object = xui.Msgbox2Async("Delete file?", "Title", "Yes", "Cancel", "No", Null)
74     Wait For (sf) MsgBox_Result (Result As Int)
75     If Result = xui.DialogResult_Positive Then
76         Log("Deleted!!!")
77     End If
78 End Sub
```

Imagen 42, MsgBox2Async

MsgBox2Async devuelve un objeto ("sf" en el ejemplo) que espera a un evento de respuesta por parte del usuario.

- **DialogResponse_Positive** (si se ha pulsado "Yes" en el ejemplo)
- **DialogResponse_Negative** (si se ha pulsado "No")
- **DialogResponse_Cancel** (si se ha pulsado "Cancel")

Los valores anteriores se controlan mediante una sentencia "if" y así se decide qué acción realizar.

Las palabras que aparecen en los botones ("Yes", "No" y "Cancel") se pueden cambiar, pero las respuestas siempre van a seguir el orden de "Positive", "Cancel" y "Negative". Eso es importante tenerlo en cuenta.

Puedes ignorar un botón si pones una cadena vacía "", con lo que el botón no aparecerá en cuadro de diálogo.

El último parámetro permite mostrar un icono a la izquierda del mensaje; con Null no se muestra nada.

Wait For

Wait For detiene la ejecución de una acción hasta que un evento es disparado.

```
Wait For (<remitente>) <event signature>
```

Donde **remitente** es el objeto sobre el que se va a esperar y

Event signature es el evento que se debe generar. En el caso de MsgBox2 Async estos eventos pueden ser DialogResult_Positive, DialogResult_Cancel o DialogResult_Negative.

El tipo de evento se comprueba con una sentencia "if".

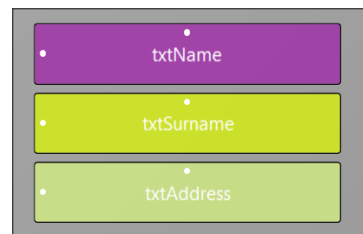
CustomDialog

Crear un diálogo personalizado necesita varios pasos que comienzan en el Diseñador.



Imagen 43. Pasos para crear un diálogo personalizado

Siguiendo con el primer ejemplo, creamos un nuevo formulario en el Diseñador y lo guardamos con el nombre **frmInsStudent**. También creamos las variables para los TextFields desde "Generar Miembros"



```

87 Private Sub Button3_Click
88     dialog.Initialize (Root)
89     dialog.Title = "My Custom Dialog" 1
90
91     Dim p As B4XView = xui.CreatePanel("")
92     p.SetLayoutAnimated(0, 0, 0, 350dip, 250dip) 2
93
94     p.LoadLayout("frmInsStudent")
95     dialog.PutAtTop = True 'put the dialog at the top of the screen 3
96
97     Wait For (dialog.ShowCustom(p, "OK", "", "CANCEL")) Complete (Result As Int) 4
98     If Result = xui.DialogResponse_Positive Then
99         Log(txtName.text & " " & txtSurname.text & " " & txtAddress.text)
100     End If
101 End Sub
  
```

Imagen 44. Diálogo personalizado

1. Dentro de **GlobalSub** creamos un objeto llamado "dialog" del tipo B4XDialog. Después usamos la función "dialog.Initialize(Root)" para inicializarlo. Con "dialog.Title" establecemos el título del cuadro de diálogo que vamos a crear.
2. Creamos un objeto "p" de tipo "B4XView" donde mostraremos los ítems del formulario que hemos diseñado. Fijamos su tamaño en píxeles.
3. Cargamos el formulario con "p.LoadLayout" donde "p" es el Pane que creamos antes y le decimos que aparezca encima de todas las demás ventanas ("p.PutAtTop=True").
4. Esperamos a que se pulse un botón y comprobamos el resultado.



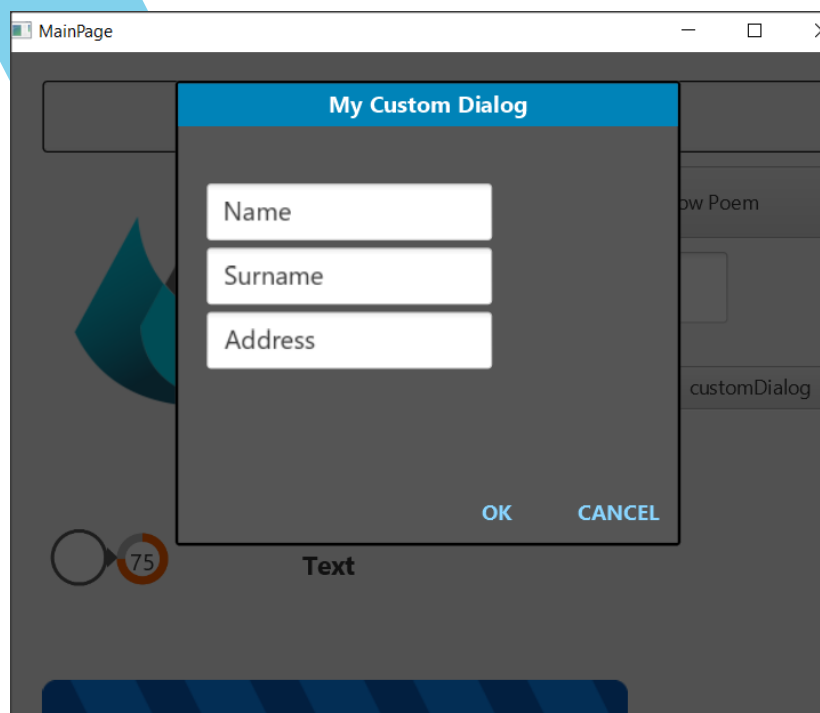


Imagen 45. El cuadro de diálogo

Templates - Plantillas

B4X posee plantillas prediseñadas para crear cuadros de diálogo entre las que destacamos:

- B4XDateTemplate
- B4XColorTemplate
- B4XLongTextTemplate

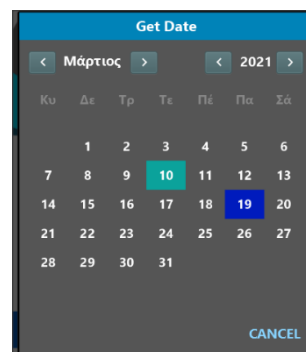


Consejo para el profesor

Más funciones se usarán en los siguientes temas a medida que se expliquen conceptos como los de lista, diccionarios, etc.

B4XDateTemplate

Crea un cuadro de diálogo para elegir la fecha. Este cuadro sólo necesita un botón cancelar porque cuando se elige una fecha se vuelve automáticamente el valor introducido al programa. Cuando el formulario se abre ya tiene prefijada la fecha actual en un color distinto,



B4XDateTemplate se basa en una plantilla de diálogo que se debe declarar e inicializar como con los diálogos personalizados vistos antes.

1. Inicializar y fijar el título del diálogo. En el ejemplo la inicialización se hace en la rutina Class_Globals.

```
108 Private Sub btnDate_Click
109     dialog.Title = "Get Date" 1
110     DateTemplate.Initialize
111     DateTemplate.MinYear = 2010 2
112     DateTemplate.MaxYear = 2030
113     'only CANCEL needed
114     Wait For (dialog.ShowTemplate(DateTemplate, "", "", "CANCEL")) Complete (Result As Int)
115     If Result = xui.DialogResponse_Positive Then
116         btnDate.Text = DateTime.Date(DateTemplate.Date) 3
117     End If
118 End Sub
```

Imagen 46. Crear un cuadro de diálogo para fecha

2. Indicamos el límite de años.
3. Esperar a que se elija la fecha. Cuando se haga, mostramos la fecha en "btnDate" o bien lo podemos usar como queramos.



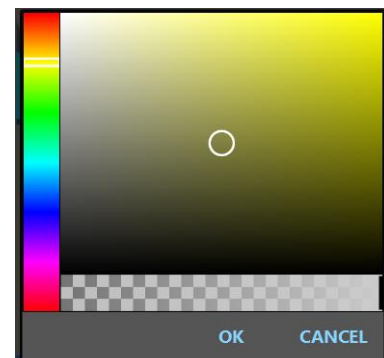
Recuerda

La fecha se devuelve como "tics" que representan los milisegundos pasados desde el 1/1/1970. Los milisegundos se convierten a la fecha con la función "DateTime.Date".

B4XColorTemplate

B4XColorTemplate es similar al de la fecha, pero para elegir un color. De nuevo el código espera con el comando Wait For a que se elija el color y cuando se pulsa "OK" se devuelve el color.

En el ejemplo se usa el color elegido para cambiar el fondo de la pantalla principal con el comando "Root.Color".



```
123 Private Sub btnColor_Click
124     ColorTemplate.Initialize
125     Wait For (dialog.ShowTemplate(ColorTemplate, "OK", "", "CANCEL")) Complete (Result As Int)
126     If Result = xui.DialogResponse_Positive Then
127         Root.Color = ColorTemplate.SelectedColor
128     End If
129 End Sub
```

Imagen 47. Plantilla para crear color

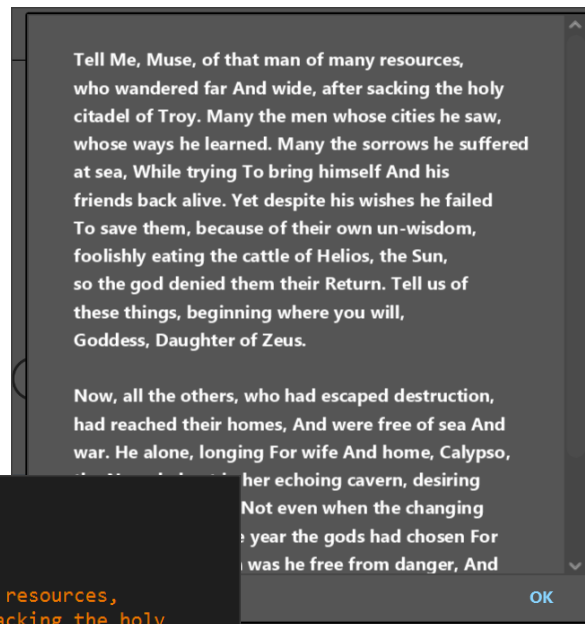
B4XLongTextTemplate

B4XLongTextTemplate crea una ventana en la pantalla para mostrar un texto largo. Añade además un barra de navegación para desplazarse por el texto.

Como en anteriores plantillas, se crea una variable del tipo B4XLongTextTemplate y después se inicializa fijando el tamaño de la ventana que vas a crear con el comando "Resize".

Finalmente, en la propiedad "Text" ponemos el texto que

```
134 Private Sub btnReadPoem_Click
135     LongTextTemplate.Initialize
136     LongTextTemplate.Resize(500, 500)
137     LongTextTemplate.Text = "$"
138     Tell Me, Muse, of that man of many resources,
139     who wandered far And wide, after sacking the holy
140     citadel of Troy. Many the men whose cities he saw,
141     whose ways he learned. Many the sorrows he suffered
142     at sea, While trying To bring himself And his
143     friends back alive. Yet despite his wishes he failed
144     To save them, because of their own un-wisdom,
145     foolishly eating the cattle of Helios, the Sun,
146     so the god denied them their Return. Tell us of
147     these things, beginning where you will,
148     Goddess, Daughter of Zeus.
149     "$
150     dialog.ShowTemplate(LongTextTemplate, "OK", "", "")
151 End Sub
```



queremos mostrar e invocamos al método "ShowTemplate".

Imagen 48. LongTextTemplate

Ejercicios

1. Crea un programa que pida dos fechas entre 2000 y 2021 y muestre cuántos días, meses y años de diferencia hay entre ellas.

Pista:

```
DateUtils.PeriodBetween(fecha1, fecha2).Days  
DateUtils.PeriodBetween(fecha1, fecha2).Months  
DateUtils.PeriodBetween(fecha1, fecha2).Years
```

2. Crea un programa que lea la siguiente información de un cliente:
 - a. Nombre
 - b. Apellidos
 - c. Teléfono
 - d. Saldo del teléfono (en €)

Después crea un botón en el formulario con el texto "pago" y que al pulsarlo muestre el mensaje "¿Desea pagar?". Si se responde "Sí", pone a 0 el contenido de la variable que guarda el saldo del teléfono.

3. Crea un programa que muestre un resumen de 5 libros en una ventana. Los libros se elegirán de un conjunto de 5 botones que tengan una imagen de ellos encima. Los resúmenes y las imágenes de los 5 libros se pueden encontrar en los materiales de apoyo a este ejercicio.
4. Crea un formulario que contenga estos elementos:
 - a. Nombre
 - b. Apellidos
 - c. Teléfono

También incluirá un interruptor (switch) que, cuando el formulario esté abierto, tenga un color de fondo gris claro ("light_gray") con los campos de texto en color blanco. Cuando se cierre, el fondo será gris oscuro (dark_grey) con los campos de texto en gris claro. Añade un etiqueta que diga "día" o "noche" con letras negras o blancas, respectivamente.

*Pista: Usa **JFX library** para establecer los colores del floatTextFields*

```
Private fx As JFX ` dentro de Class Globals  
lblDiaNoche.TextColor = fx.Colors.Black ` donde quieras cambiar el color
```

5. Un depósito de agua mide 5 metros de altura y tiene una base de 3x3 metros. Escribe un programa que:
 - A. Lea continuamente el nivel del agua en metros y muestre una barra que represente la altura de llenado. El campo de texto usado sólo permitirá introducir valores decimales, pero no texto.
 - B. Si la altura del agua está por encima de 5 metros mostrará el mensaje "¡Peligro de desbordamiento!".
 - C. Si la altura es menor de 0'5 metros mostrará el mensaje "Peligro: no hay suficiente agua en el depósito".
 - D. Cada vez que se cambie la altura, se mostrará la cantidad total de agua expresada en metros cúbicos.

Tema 14 – Matrices - Arrays

🕒 4h

Lo que los estudiantes aprenderán

- Matrices unidimensionales (o Vectores)
- Operaciones básicas con matrices
- Elemento Máximo y Mínimo
- Búsqueda lineal
- Búsqueda binaria
- Ordenación con Bubble Sort
- Ordenación con Selection Sort

Un problema común en programación es la gestión de grandes cantidades de datos. Cuando un problema sólo necesita 6 o 7 variables, es fácil declararlas y usarlas. ¿Qué sucede cuando necesitamos usar muchos datos del mismo tipo a la vez? Por ejemplo, 100 estudiantes y 100 calificaciones; ¿cómo declarar 200 variables con nombres y calificaciones y cómo gestionarlas?

Una de las soluciones en ciencias de la computación consiste en usar arrays (o matrices, vectores). En general, un array se define como un conjunto de datos del mismo tipo que puede almacenarse en la memoria del ordenador de forma contigua. Así, el desarrollador puede localizarlas simplemente moviéndose de una ubicación a otra sin tener que emplear nombres diferentes.

En la imagen 1 puedes ver un ejemplo de array con las notas de 14 estudiantes para un tema. Todas las notas se colocan en posiciones contiguas y sólo usamos un nombre (Notas). Para referirse a cada posición dentro de la tabla el programador debe indicar el nombre del array (Notas) y después escribir entre paréntesis el número de la celda donde está el dato que desea consultar.

Notas	98	76	86	45	32	77	56	99	34	71	47	82	69	88
	0													13

Imagen 49. Array llamado "Notas" con 14 huecos para notas

Así, por ejemplo, para mostrar el primer elemento del array tan sólo hay que escribir "Notas(0)", donde 0 es la primera posición del array.

Declaración de Arrays

Un array se declara igual que otras variables:

```
Private Notas(14) As Int
```

Aquí, Notas es el nombre del array y el número entre paréntesis nos dice cuántas **posiciones** (o tamaño) habrá en el array. Una vez que se declara

un array con un tamaño determinado, no se puede cambiar en el código a menos que se vuelva a declarar con un nuevo tamaño.

Funciones sobre Arrays

Insertar elementos en un array

Para insertar un elemento en un array hay que asignar un valor al lugar correspondiente en el array. Por ejemplo:

```
Notas(0) = 89
```

Se podría seguir con todos los demás elementos así, pero es más fácil si usamos un proceso repetitivo para rellenar el array. En el caso del array Notas, en B4X podría ser así:

```
Private Notas(14) As Int
For i = 0 To 13
    Notas(i) = Rnd(1,100)
Next
```

El código anterior rellena el array con números aleatorios de 1 a 100. Fíjate que la primera posición empieza en 0. La variable **i** indica en cada iteración

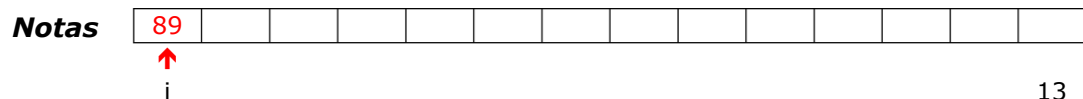


Imagen 50. Insertar elementos en un array

del bucle la posición del array en que insertaremos un valor y se le llama **índice** del array. Moviendo el índice **i** con una sentencia de repetición podemos acceder a cada posición del array.

Una segunda forma de insertar elementos es así:

```
Private Notas() As Int
Notas = Array As Int(19,43,12,65,23,87,45,65,87,23,56)
```

El tamaño del array no se indica al declararlo, pero sí al realizar la inserción de los elementos con la **sentencia Array**.

Es evidente que puedes tener arrays de cualquier tipo como, por ejemplo, cadenas de texto, decimales, etc. Pero **nunca puedes mezclar** los tipos en un array.

Mostrar los elementos de un array

Usando la función "Log" puedes imprimir un elemento de un array o el array al completo mediante una iteración.

```
Log(Notas(0)) ' Muestra el 1er elemento del Array Notas

For i = 0 to 13
    Log(i & ": " & Notas(i))
Next
```

```
0: 26
1: 95
2: 72
3: 17
4: 82
5: 76
6: 72
7: 10
8: 91
9: 79
10: 19
11: 86
12: 28
13: 30
```

El ejemplo anterior usa un bucle For con un índice i para mostrar el valor que hay en la posición i del array.

Intentar acceder a una posición fuera del Array (en el ejemplo, la posición 14 o más alta) generaría un fallo del programa (excepción denominada "out-of-bounds index exception"), así que es muy importante que prestes atención a la posición a la que accedes y a los límites de un array.

Mostrar elementos en orden inverso

El siguiente código muestra los elementos de un array empezando por el final:

```
For i = 13 to 0 Step -1
    Log(i & ": " & Notas(i))
Next
```

```
13: 39
12: 5
11: 88
10: 34
9: 82
8: 99
7: 5
6: 48
5: 63
4: 62
3: 16
2: 49
1: 93
0: 55
```

En general, puedes modificar el índice como quieras en un bucle de repetición o usar los elementos del array que quieras.

Calcular la suma y la media de los elementos de un array

Las reglas aplicables a procesos repetitivos en los algoritmos se pueden aplicar en general a los arrays con pocos cambios. Así, la suma de los elementos de un array necesita del uso de una variable extra que guarde la suma (que llamaremos "suma") y de aplicar una repetición sobre el array.

```
Private intSuma As Int = 0
Private fltMedia as Float
For i = 0 to 13
    intSuma = intSuma + Notas(i)
Next
fltMedia = intSuma / 14
Log(intSuma & " " & fltMedia)
```

Encontrar el máximo y el mínimo

Encontrar el elemento máximo y el mínimo en un array requiere del uso de variables adicionales llamadas, normalmente, **Máx** y **Mín**. Los valores iniciales de estas variables se fijan en el primer elemento del array y después se compara uno a uno cada elemento del array para ver si es mayor que **Máx** (o menor que **Mín**), actualizando sus valores en consecuencia.

```

Private intMáx, intMín As Int
intMáx = Notas(0)
intMín = Notas(0)
For i = 0 To 13
    If intMáx < Notas(i) Then
        intMáx = Notas(i)
    End If
    If intMín > Notas(i) Then
        intMín = Notas(i)
    End If
Next
Log("Máximo = " & intMáx)
Log("Mínimo = " & intMín)

```

Algoritmos de búsqueda

Buscar un elemento en un array consiste en escanearlo para encontrar elemento que cumple una condición concreta.

En este tema veremos la búsqueda secuencial y la búsqueda binaria.

Búsqueda secuencial

Es el más simple, pero el más lento. Consiste en comprobar uno a uno todos los elementos del array para encontrar el elemento buscado. El siguiente código muestra las posiciones del array que contienen el valor clave.

```

'Buscar todas las posiciones que contienen un valor clave
For i = 0 To 999
    If Notas(i) = clave Then
        Log("encontrado en la posición: " & i)
    End If
Next

```

Si sólo hay que buscar la primera aparición de un valor clave, se puede usar una variable lógica llamada "encontrado" que pondremos a "true" cuando encontremos el elemento buscado.

```

Private encontrado As Boolean = False
i = 0
Do While Not (encontrado) And i <= 999
    If Notas(i) = clave Then
        Log("encontrado en la posición : " & i )
        encontrado = True
    End If
    i = i + 1
Loop
If Not(encontrado) then
    Log("No se ha encontrado")
End If

```

Búsqueda Binaria

La búsqueda binaria sólo se aplica a arrays ordenados. La filosofía básica del método consiste en comprobar el valor que hay en la mitad del array.



Si el elemento que buscamos es menor que ese valor central, entonces buscamos en la mitad superior del array. Si es mayor, entonces buscamos en la mitad inferior del array. En el siguiente ejemplo buscamos el valor 80 en un array Notas con 10 valores ordenados de menor a mayor:

Búsqueda Binaria					
Valor clave = 80					
1	47	←Arriba	47	47	47
2	58		58	58	58
3	62		62	62	62
4	69		69	69	69
5	74	←Centro	74	74	74
6	79		79	←Arriba, Centro	79
7	80		80	←Abajo	80
8	83	←Centro	83		83
9	88		88		88
10	95	←Abajo	95		95
	1	2	3	4	

1. Inicialmente, las posiciones primera y última se guardan en las variables "Arriba" y "Abajo". El "centro" del array es el resultado de la división: $(Arriba + Abajo) / 2$.
2. Comparamos el valor de Notas(Centro) con la clave y si es menor, a la variable "Abajo" se le pone el valor de "Centro" más 1 (porque ya hemos comprobado que el valor en "Centro" no es el que buscamos). Si fuera mayor, el valor de "Arriba" sería una posición más abajo de "Centro".
3. Repetimos los pasos anteriores hasta que encontremos el valor buscado o bien la posición de "Arriba" sea mayor que la de "Abajo" (eso implicaría que ya hemos recorrido todo el array y no hemos encontrado la clave buscada).

Ordenación

Existen varios algoritmos para ordenar un array. Veremos el método de la burbuja (Bubble Sort) y la ordenación por selección (Selection Sort).

Ordenación por Burbuja

La ordenación por burbuja es un algoritmo sencillo en el que se recorre todo el array comparando los elementos adyacentes, intercambiándolos si están en el orden incorrecto. El array se recorre varias veces hasta que está ordenado. El nombre de "burbuja" se le puso porque los elementos más pequeños o más grandes suben como una burbuja hacia arriba.

Ejemplo de Ordenación por Burbuja

Ordenaremos un array "Notas" que contiene 5 enteros de menor a mayor.

```
Private Notas() As Int
Notas = Array As Int(65,12,19,43,23)
```

1ª Pasada					
1	65	65	65	65	65
2	12	12	12	12	12
3	19	19	19	19	19
4	43	43	23	23	23
		←k-1	←K	←K	←k-1



Al principio el algoritmo comienza por la última posición del array y compara cada elemento con el anterior secuencialmente:

1. La primera comparación se hace con los valores de las celdas 5 y 4, comprobando que Notas(5) es menor que Notas(4), con lo que se intercambian sus valores.
2. A continuación se comparan las celdas 3 y 4, comprobando que Notas(4) es mayor que Notas(3), con lo que no se hacen cambios en el array.
3. En las posiciones 3 y 2 tampoco se intercambian los valores.
4. En el último paso, las posiciones 1 y 2 se comparan y se intercambian sus valores de nuevo.

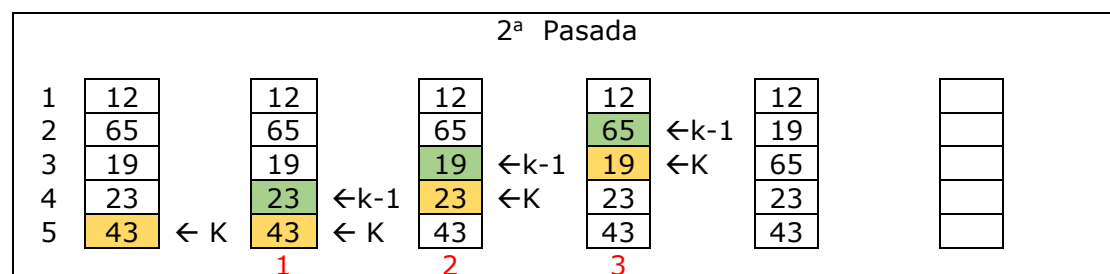
El primer paso se implementa con el siguiente código:

```

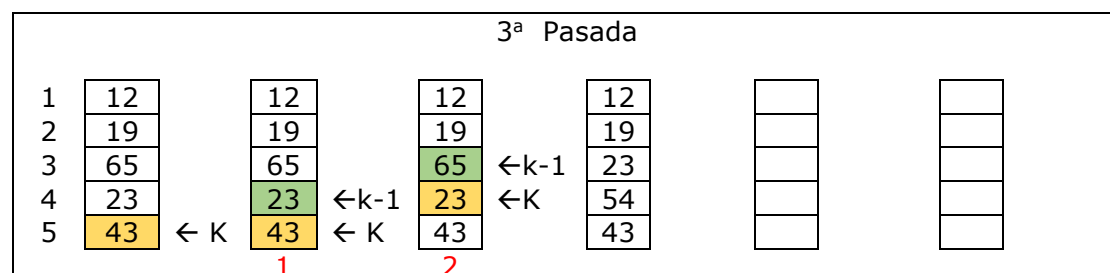
Private Notas() As Int
Private temp As Int
Notas = Array As Int(19,43,12,65,23)
For k = 4 To 1 Step -1
    If Notas(k) < Notas(k-1) Then
        temp = Grades(k)
        Notas(k) = Notas(k-1)
        Notas(k-1) = temp
    End If
Next

```

Ahora el elemento más pequeño está en la parte superior del array.



En la segunda pasada se aplica el mismo procedimiento, excepto que la primera celda no se compara porque el elemento más pequeño ya está en lo más alto del array.



4ª Pasada						
1	12	12	12			
2	19	19	19			
3	23	23	23			
4	54	54	43			
5	43	43	54			
		1				

Las pasadas continúan hasta que se orden por completo el array. Fíjate que en cada pasada se comprueban menos elementos, ya que en cada iteración el elemento más pequeño sube hacia arriba (burbuja).

En general, el número de pasadas que hay que hacer es el tamaño del array -1. En el ejemplo, para un array de 5 elementos, se hacen 4 pasadas. El código completo es el siguiente:

```
Private Notas() As Int
Private temp As Int
Notas = Array As Int(19,43,12,65,23)
For i = 1 to 4 'i cuenta las pasadas
    For k = 4 To i Step -1
        If Notas(k) < Notas(k-1) Then
            temp = Grades(k)
            Notas(k) = Notas(k-1)
            Notas(k-1) = temp
        End If
    Next
Next
```

Ordenación por Selección

El algoritmo divide el array en dos partes: una parte ordenada que se construye de izquierda a derecha (la izquierda es la primera posición del array) y otra parte que es un subarray con los demás elementos desordenados. El algoritmo busca el elemento más pequeño (o el más grande, según el orden que se desee) en el subarray desordenado, intercambiándolo (swapping) con el elemento más a la izquierda de la lista desordenada (dejándolo así en orden).

La implementación según los pasos indicados sería:

1. Elegir el elemento más pequeño
2. Intercambiar el mínimo con el primer elemento
3. Repetir los pasos 1 y 2 para todos los elementos del array

1	65	65	12	12	12	12	12
2	12	12	65	65	19	19	19
3	19	19	19	19	65	23	23
4	43	43	23	23	23	65	65
5	23	23	43	43	43	43	43
	1	2	1	2	1	2	2

```

Private Notas() As Int
Notas = Array As Int(65,12,19,43,23)
Private intMín, intMínPos As Int

For k = 0 To 4
    intMín = Notas(k)
    intMínPos = k
    For i = k To 4 'Buscar el mínimo desde la posición k a la 5ª
        If intMín > Notas(i) Then
            intMín = Notas(i)
            intMínPos = i
        End If
    Next
    Notas(intMínPos) = Notas(k)
    Notas(k) = intMín
Next

```

Ejercicios

1.
 - a. Escribe un programa que rellene un array llamado **A(50)** con 50 números enteros aleatorios entre 1 y 100.
 - b. Calcule y muestre la suma de los elementos del array A(50) en posiciones pares.
 - c. Si la suma de los primeros 25 elementos del array es igual a la suma de los últimos 25 elementos, mostrar el mensaje "Son iguales".
 - d. Si $A(1)=A(50)$, $A(2)=A(49)$, $A(3)=A(48)$... $A(25)=A(26)$, entonces se mostrará el mensaje "Array simétrico".
 - e. Encontrar el elemento máximo y la posición donde está.
 - f. Crea una subrutina que ordene el Array A.
 - g. Crea una subrutina que reciba un array y un entero y aplique la búsqueda binaria a ese número en el array. Deberá mostrar la posición del elemento o -1 si no lo encuentra.
 - h. Usando las dos anteriores subrutinas, ordena la tabla A, busca el número 67 y muestra un mensaje para indicar si se ha encontrado o no.
2.
 - a. Tenemos un array que guarda la temperatura media de cada día de un año que se llama **Temperatura(365)**. Supón que las temperaturas son números enteros entre 1 y 40°C. Encuentra y muestra la frecuencia de cada temperatura.
 - b. En el array anterior **Temperatura**, encuentra la segunda mayor temperatura del año.

Tema 15 – Listas

🕒 2h

Lo que los estudiantes aprenderán

- ¿Qué es una lista?
- Operaciones básicas con listas

Una lista es un conjunto de nodos colocados en línea (uno detrás de otro). Cada nodo contiene datos y un puntero que apunta al siguiente nodo. El puntero del último nodo no apunta a ninguna parte y su valor es NULL.

Creación de una lista

Una lista en B4X se declara así:

```
Private Islas As List  
Islas.Initialize
```

Donde **Islas** es el nombre de la lista creada. Además, una lista tiene que inicializada antes de ser usada con el método **Initialize**.

Para añadir ítems a la lista empleamos el método **Add** que añade un nuevo ítem al final de la lista:

```
Islas.Add("Piraeus")  
Islas.Add("Paros")  
Islas.Add("Thira")  
Islas.Add("Crete")
```



También es posible insertar ítems al final de la lista creando un array y usando el método **AddAll**:

```
Islas.AddAll(Array as String("Piraeus", "Paros", "Naxos", "Crete"))
```



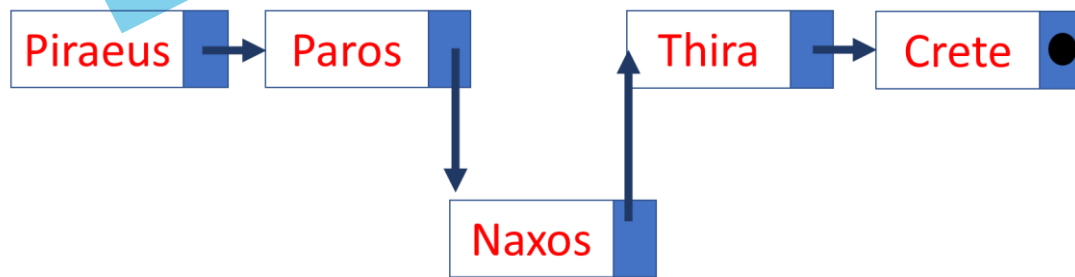
Recuerda

Los elementos de una lista empiezan en 0. Así, en una lista con 4 elementos, el primero es element(0) y el ultimo es element(3).



Insertar un ítem en una posición determinada

Para insertar un ítem entre otros dos se puede usar el método **InsertAt**:



```
Islas.InsertAt(2, "Naxos")
```

El anterior ejemplo inserta la isla de Naxos entre Paros y Thira en la tercera posición.

Eliminar un ítem de una posición determinada

El método **RemoveAt** elimina un ítem en una posición determinada:



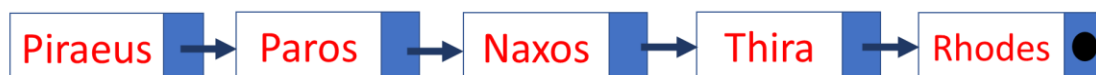
```
Islas.RemoveAt(1)
```

En el ejemplo anterior, la isla de Paros es eliminada de la lista.

Cambiar el valor en una posición concreta

Con el método **Set** se puede cambiar el valor en una posición concreta y reemplazarlo por otro:

```
Islas.Set(4, "Rhodes")
```



El valor de la quinta posición se cambia a "Rhodes".

Más métodos para trabajar con listas

1. Tamaño de la lista.



```
Islas.Size
```

El ejemplo anterior devolvería el valor 5.

2. Ordenar una lista. Se puede ordenar una lista de menor a mayor o de mayor a menor con el método **sort**:

```
Islas.Sort(True) ' Ordenar en orden ascendente  
Islas.Sort(False) ' Ordenar en orden descendente
```



Islas.Sort(True)



Islas.Sort(False)



3. Vaciar una lista

Se puede vaciar una lista con el método **Clear**

```
Islas.Clear
```

4. Recorrer una lista

Se puede recorrer una lista con una sentencia de repetición:

```
For i = 0 to Islas.Size - 1 ' Recuerda que empieza en 0  
    Log(Islas.Get(i))  
Next
```

5. Puedes guardar el contenido de los ítems de una lista en otras variables para usarlos:

```
For i = 0 to Islas.Size - 1  
    Private isla As String  
    isla = Islas.Get(i)  
    Log(isla)  
Next
```

6. Insertar otros tipo de ítems

Una lista puede guardar ítems de cualquier tipo simple o complejo que quieras. Por ejemplo, los ítems de una lista pueden ser objetos o arrays.

En el siguiente ejemplo, cada ítem de la lista es un array de enteros:

```
'Crear una lista de arrays
Private NotasEstudiantes As List<Int()>
NotasEstudiantes.Initialize

For j = 0 To 4
    Private Notas(5) As Int
    For i = 0 To 4
        Notas(i) = Rnd(1, 100) ' Crear 5 notas aleatorias por estudiante
    Next
    NotasEstudiantes.Add(Notas)
Next

'Mostrar las notas de los estudiantes
For j = 0 To NotasEstudiantes.Size - 1
    Log("Estudiante: " & j)
    Private notasEst(5) As Int = NotasEstudiantes.Get(j)
    For I = 0 To 4
        Log(notasEst(i))
    Next
Next
```


1. Declaramos e inicializamos la lista.
2. Tenemos 5 estudiantes y para cada uno de ellos creamos un array con 5 notas aleatorias entre 1 y 100.
3. Cada array es colocado al final de la lista con el método Add.
4. Para todos los ítems de la lista, guardamos en un array cada ítem.
5. Mostramos el array que hay en cada ítem de la lista.

Ejercicios

1.
 - a. Crear una lista de países y el nombre de sus capitales. Por ejemplo, puedes usar los siguientes países:

Cuba	La Habana
Chipre	Nicosia
Chequia	Praga
Egipto	El Cairo
Kenia	Nairobi
México	México DF
Perú	Lima
Vietnam	Hanoi
Portugal	Lisboa

Fuente: <https://www.boldtuesday.com/pages/alphabetical-list-of-all-countries-and-capitals-shown-on-list-of-countries-poster>

- 
- b. Muestra los nombres de los países que empiezan por la letra "P"
 - c. Usando un campo de texto, introduce el nombre de un país y búscalo en la lista de países para ver si existe y muestra su capital.
 - d. Crea un botón que muestre un diálogo preguntando por un nuevo país y su capital para añadirlo a la lista.

Tema 16 – Mapas

🕒 2h

Lo que los estudiantes aprenderán

- ¿Qué es un mapa?
- Operaciones básicas con mapas
- Sentencia For Each

Un mapa es una colección que contiene pares clave-valor. Por ejemplo, las



Imagen 3. Mapa de las páginas amarillas

palabras de un diccionario o bien los registros de llamadas telefónicas. Algunas veces se les llama directamente diccionarios.

Cada clave de un par es única. Eso significa que si añades un par clave-valor (entrada) y la colección ya contiene una entrada con la misma clave, la entrada anterior será sobrescrita por el nuevo valor.

El valor puede ser cualquier tipo desde un tipo simple (entero, cadena, etc.) a un objeto.

Crear un mapa

Un mapa se declara en B4X de la siguiente forma:

```
Private InglesGriego As Map
InglesGriego.Initialize

Private InglesEspañol As Map
InglesEspañol.Initialize
```

Donde **InglesGriego** es el nombre del primer mapa creado y **InglesEspañol** es el nombre del segundo. Los mapas deben ser inicializados antes de usarse.

Insertar ítems en un Mapa

El método **Put** añade pares clave-valor al mapa. Por ejemplo, los dos mapas de más abajo usan el método Put.

<table><tr><th>Inglés</th><th>Griego</th></tr><tr><td>Memory</td><td>μνήμη</td></tr><tr><td>Screen</td><td>οθόνη</td></tr><tr><td>Printer</td><td>εκτυπωτής</td></tr><tr><td>Programming Language</td><td>Προγραμματισμός</td></tr></table> <p>Mapa 1 InglesGriego</p>	Inglés	Griego	Memory	μνήμη	Screen	οθόνη	Printer	εκτυπωτής	Programming Language	Προγραμματισμός	<table><tr><th>Inglés</th><th>Español</th></tr><tr><td>Memory</td><td>Memoria</td></tr><tr><td>Screen</td><td>Pantalla</td></tr><tr><td>Printer</td><td>Impresora</td></tr><tr><td>Programming Language</td><td>Lenguaje de programación</td></tr></table> <p>Mapa 2 InglesEspañol</p>	Inglés	Español	Memory	Memoria	Screen	Pantalla	Printer	Impresora	Programming Language	Lenguaje de programación
Inglés	Griego																				
Memory	μνήμη																				
Screen	οθόνη																				
Printer	εκτυπωτής																				
Programming Language	Προγραμματισμός																				
Inglés	Español																				
Memory	Memoria																				
Screen	Pantalla																				
Printer	Impresora																				
Programming Language	Lenguaje de programación																				

El método Put se usa así:

<nombre mapa>.Put(clave, valor)

El siguiente ejemplo crea dos mapas con las palabras del idioma inglés y los valores en griego y español.

```
Private InglesGriego As Map
InglesGriego.Initialize

Private InglesEspañol As Map
InglesEspañol.Initialize

InglesGriego.Put("Memory", "μνήμη ")
InglesGriego.Put("Screen", "οθόνη")
InglesGriego.Put("Printer ", "εκτυπωτής")
InglesGriego.Put("Programming Language", "Προγραμματισμός")

InglesEspañol.Put("Memory", "Memoria")
InglesEspañol.Put("Screen", "Pantalla")
InglesEspañol.Put("Printer", "Impresora")
InglesEspañol.Put("Programming Language", "Lenguaje de Programación ")
```

Usar el valor de un mapa

Para usar el valor de un mapa, sólo necesitas emplear el método Get.

<nombre mapa>.Get(Clave as Object)

```
PalabraGR = InglesGriego.Get("Screen")
Log(PalabraGR)      ` Muestra οθόνη

PalabraESP = InglesEspañol.Get("Screen")
Log(PalabraESP)     ` Muestra "Pantalla"
```

Se guarda el valor asociado a la clave "Screen" tomado del mapa InglesGriego en la variable **PalabraGR**. Después, se guarda el valor que corresponde a la clave "Screen" del mapa InglesEspañol en la variable



Recuerda

El tipo de la variable para guardar el contenido del valor leído del mapa debe ser del mismo tipo que el tipo del valor del mapa. La clave tiene que ser una cadena de texto o un número.

PalabraESP.

Si la clave no existe en el mapa, se devuelve el valor Null.

```
PalabraESP = InglesEspañol.Get("Keyboard")
Log(PalabraESP) ` Muestra null
```

Índices en Mapas

Se pueden usar índices como en las listas para acceder a la clave o al valor.

<nombre mapa>. GetKeyAt(Índice As Int)

```
Clave = InglesEspañol.GetKeyAt(2)
Log(Clave) ` Muestra "Printer"
```

<nombre mapa>. GetValueAt(Índice As Int)

```
Valor = InglesEspañol.GetValueAt(2)
Log(Valor) ` Muestra "Impresora"
```

Además, los métodos **GetKeyAt** y **GetValueAt** pueden usarse en una sentencia de repetición para obtener todos los valores de un mapa:

```
For i = 0 To InglesGriego.Size - 1
    Log(InglesGriego.GetValueAt(i))
Next
```



La iteración anterior muestra todos los valores del mapa **InglesGriego**.

La sentencia "for each"

Existe una sentencia de iteración que no se ha comentado antes: **for each**. Esta sentencia itera a lo largo de los ítems de una lista de valores como los de un mapa, por ejemplo.

```
For Each palabra As String In InglesEspañol.Values
    Log(palabra)
Next
```

El bucle anterior define una variable ("palabra") que almacenará el ítem asociado a la cada iteración. No es necesario crear un contador o determinar el avance en el bucle, ya que la propia sentencia va recorriendo uno a uno los ítems de la lista hasta que la recorre por completo.

```
For Each clave As String In InglesEspañol.Keys
    Log(clave)
Next
```

Muestra las claves almacenadas en el mapa.

Para obtener las claves y los valores a la vez, se puede usar la siguiente sentencia:

```
For Each clave As String In InglesEspañol.Keys
    Log(clave & InglesEspañol.GetValueAt(clave))
Next
```

Comprobar si existe una clave

Si queremos buscar una clave concreta, podemos recorrer el mapa hasta encontrarla o no, pero es más fácil y más rápido usar el método **ContainsKey**.

<nombre mapa>. ContainsKey(Clave As Object)

```
If InglesGriego.ContainsKey("Keyboard") Then
    Log("¡Ya hay una entrada con esa clave!", "")
Else
    Log("¡No existe esa clave!", "")
End If
```

Borrar una Clave y vaciar un Mapa

El método **Remove** elimina una clave (y, por supuesto, su valor asociado en el Mapa)

<nombre mapa>. Remove(Clave As Object)

```
InglesGriego.Remove("Memory")
```

Para borrar todos los ítems de un mapa usamos el método **Clear**.

Ejercicios

1. Crea un mapa con los nombres de los países como clave y sus capitales como valores:

Cuba	La Habana
Chipre	Nicosia
Chequia	Praga
Egipto	El Cairo
Kenia	Nairobi
México	Ciudad de México
Perú	Lima
Vietnam	Hanoi
Portugal	Lisboa

Fuente: <https://www.boldtuesday.com/pages/alphabetical-list-of-all-countries-and-capitals-shown-on-list-of-countries-poster>

2. Añade 3 países con sus capitales.
3. Muestra los nombres de los países y sus capitales usando la sentencia **for each**.
4. Crea un nuevo mapa que contenga el nombre de las capitales como clave y los nombres de los países como valores.
5. Crea un campo de texto para introducir el nombre de una ciudad y muestra el país que contiene la ciudad.

Tema 17 – Ficheros

🕒 2h

Lo que los estudiantes aprenderán

- ¿Qué es un fichero?
- Ubicación de ficheros en B4J
- Métodos con Ficheros

Un fichero es una colección de datos con contenido similar que normalmente se almacena de forma permanente en el disco duro del ordenador. Es una de las características más importantes de un lenguaje de programación ya que, aunque los datos temporales se van guardando en la memoria del ordenador, cuando finaliza nuestro programa, sus datos deben quedar guardados de forma permanente.

engl_it.txt

Memory	Memoria
Screen	Schermo
Printer	Stampante

Imagen 51. Fichero engl_it.txt

En general, se puede afirmar que los ficheros se pueden dividir en bases de datos y en ficheros simples que serán los que veamos.

Carpetas o Directorios de almacenamiento

Cada sistema operativo posee diferentes carpetas/directorios para guardar los datos de las aplicaciones y otros ficheros. Para que B4J pueda trabajar con ficheros independientemente del sistema operativo usado, utiliza palabras clave para referirse a un tipo concreto de carpeta.

File.DirAssets

Incluye los ficheros contenidos en la carpeta de la aplicación que ha sido añadida al gestor de ficheros de B4J durante la fase de desarrollo de la misma. Estos ficheros son de sólo lectura y no se pueden añadir ficheros mientras se está ejecutando la aplicación. Estos ficheros son creados por el programador para que sean copiados en el proceso de instalación.

xui.DefaultFolder

Devuelve la carpeta donde se guardan los datos de la aplicación, de forma similar a lo que hace **File.DirData**. Es obligatorio invocar una vez al método **SetDataFolder** antes de poder usarlo.

```
xui.SetDataFolder(AppName As String)
```



File.DirData

Devuelve la carpeta donde se guardan los datos de la aplicación y se puede usar para crear ficheros y guardar datos.

En el SO Windows, devuelve la carpeta de datos del usuario ("user data folder") que normalmente está en esta ruta:

C:\Users\[nombre usuario]\AppData\Roaming\[AppName]

En SO no-Windows, devuelve la carpeta donde está instalada la aplicación.

File.DirApp

Devuelve la carpeta donde está instalada la aplicación. En Windows esta carpeta está en "Archivos de Programa" y es de sólo lectura.

File.DirTemp

Devuelve la carpeta para ficheros temporales.

Log(File.DirApp)	C:\Users\usuario\DOCUMENTOS\GitHub\TEACH\1\TEMA17\1\Ejemplo\Ejemplo1\B4\Objects
Log(File.DirAssets)	AssetsDir
Log(File.DirTemp)	C:\Users\usuario\AppData\Local\Temp\
Log(File.DirData("Ejemplo1"))	C:\Users\usuario\AppData\Roaming\Ejemplo1

Imagen 52. Métodos para directorios

Puedes combinar los métodos anteriores para crear una carpeta dentro de las anteriores. Por ejemplo:

```
Private strCarpeta As String = File.DirTemp & "tema17\"  
Log(strCarpeta)
```

Mostraría **C:\Users\usuario\AppData\Local\Temp\tema17**

Crear carpetas/directorios

Puedes crear una nueva carpeta con el método:

File.MakeDir (Padre As String, Dir)

```
File.MakeDir(File.DirTemp, "tema17")
```

Crea una carpeta llamada "tema17" dentro de la carpeta "C:\Users\usuario\AppData\Local\Temp\" del anterior ejemplo.

Comprobar la existencia de un fichero

Antes de usar un fichero es recomendable comprobar si existe. El método para ello es:

File.Exists (Dir As String, NombreFichero As String)

Que devuelve "True" o "False" según el fichero exista o no. Puedes usar una sentencia if para comprobar el resultado:

```
Private nf As String = "misdatos.txt"  
If File.Exists(File.DirTemp, nf) Then  
    Log("El fichero " & nf & " existe")  
Else  
    Log("No existe el fichero: " & nf)  
End If
```



Crear y escribir en un fichero

El método para crear un fichero es:

File.OpenOutput (Dir As String, NombreFi As String, Append As Boolean)

```
Private nf as String = "misdatos.txt"  
File.OpenOutput(File.DirTemp, nf, True)
```

Si el fichero no existe, se crea.

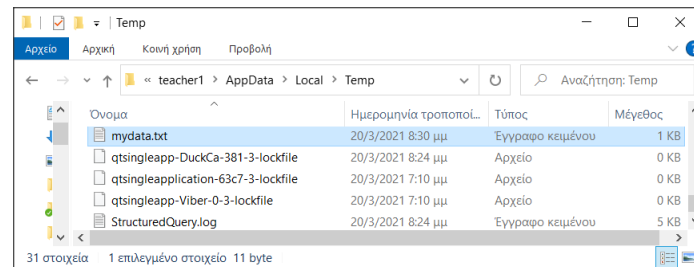


Imagen 53. Crear misdatos.txt dentro de la carpeta temporal

El valor lógico "True" o "False" del tercer parámetro indica si el fichero se abrirá para escritura (borrando todos sus datos anteriores) o si se abrirá para añadir datos al final del mismo.

Lectura y escritura de datos

Puedes leer o escribir desde una palabra a estructuras más complejas como listas o mapas.

Variables tipo cadena de texto

File.WriteString (Dir As String, NombreFichero As String, Texto As String)

```
Private nf as String = "misdatos.txt"  
Private msj As String = "Hola Mundo"  
File.WriteString(File.DirTemp, nf, msj)
```

Escribe una variable de tipo cadena de texto en el fichero. Fíjate en que el fichero se crea al principio, con lo que se borrarían todos sus datos.

Si el fichero ya existe, se puede leer su contenido en una variable de tipo cadena con el método:

File.ReadString (Dir As String, NombreFichero As String) As String

```
Private nf as String = "misdatos.txt"  
Private strContenidoFichero As String  
strContenidoFichero = File.ReadString(strCarpeta, nf)
```


Listas

Para escribir una lista en un fichero se usa el método:

File.WriteList(Dir As String, NombreFich As String, Lista As List)

```
File.WriteList(strCarpeta, " misdatos.txt", Lista)
```

Con el método anterior cada ítem de la lista se convierte en una cadena de texto y se añade una nueva fila al fichero.



Para leer una lista de un fichero se usa el método:

File.ReadList (Dir As String, NombreFichero As String)

```
Lista = File.ReadList(File.DirRootExternal, "misdatos.txt")
```

El método crea un nuevo ítem de la lista para cada línea del fichero

Mapas

Se puede escribir el contenido de un mapa con el método:

File.WriteMap(Dir As String, NombreFich As String, Mapa As Map)

```
File.WriteMap(File.DirInternal, "fichero.txt", mapa)
```

El uso más típico es para crear un fichero de configuración para el programa.

Un mapa se puede leer con el método:

ReadMap(Dir As String, NombreFichero As String)

```
mapa = File.ReadMap(File.DirInternal, " fichero.txt")
```

El orden de los ítems no será necesariamente el mismo que el que aparece en el fichero, pero eso no importa en un mapa.



Ejercicios

Se proporciona el programa “Tema17 – Ej1” que crea una lista de equipos de fútbol y una pequeña historia de ellos. Se pide:

1. Crear un mapa con una clave para el nombre del equipo y como valores su historia.
2. Crear un fichero que guarde el mapa anterior en el fichero “equipos.txt”
3. Crear una cadena de texto que contenga todas las claves y valores del mapa. Emplea **& CRLF** para añadir una nueva línea al final de cada línea.
4. Escribe la cadena generada en un nuevo fichero con el nombre “equipos2.txt”.
5. Abre los ficheros “equipos.txt” y “equipos2.txt” con un editor de textos (por ejemplo, Notepad++) para analizar las diferencias.

Tema 18 – Tipos de datos complejos y Vistas



Lo que los estudiantes aprenderán

- Declaración de un Tipo
- Arrays de un Tipo
- Listas de un Tipo
- Mapas de un Tipo
- Ficheros KVS
- CustomListView
- B4XComboBox

A veces un desarrollador quiere agrupar variables de distinto tipo que están relacionadas. Por ejemplo, un estudiante tiene nombre, apellidos, dirección, teléfono, etc. Sería posible crear variables por separado para cada propiedad, pero es más conveniente crear un nuevo tipo de dato que las agrupe todas.

La sentencia “type”

El tipo de dato que necesitamos se llama **type** y en el ejemplo anterior se declararía así:

Type Estudiante (Apellidos As String, Nombre As String, dirección As String, teléfono As String)

Fíjate que la palabra reservada “type” es la primera que aparece y después se pone el nombre del nuevo tipo de variable que creamos. Seguidamente, entre paréntesis, indicamos todas las variables incluidas en el nuevo tipo de datos. La declaración de tipos se debe poner siempre en la zona **Class_Globals** y es **pública**.

En el ejemplo del estudiante, la sentencia “type” quedaría así:

```
Sub Class_Globals
  Type Estudiante(Apellidos As String, Nombre As String, _
                  Dirección As String, Teléfono As String)
End Sub
```

Ahora ya tenemos un nuevo tipo de dato llamado “Estudiante” que puede usarse para crear nuevas variables basadas en él.

La variable “Estudiante1” es ahora una variable del tipo “Estudiante” y para acceder a los datos que la componen pondremos el nombre de la variable



seguido de un punto y el nombre de la propiedad especificado al crear el nuevo tipo:

```
Estudiante1.Apellidos = "Ioannidis"  
Estudiante1.Nombre = "Alkinoos"  
Estudiante1.Dirección = "Atenas, Grecia"  
Estudiante1.Teléfono = "+303465854234"
```

Puedes crear tantas variables del tipo "Estudiante" que quieras, así como asignar unas a otras:

```
Private Estudiante2 as Estudiante  
Estudiante2 = Estudiante1
```

Aquí, la variable "Estudiante2" contendrá exactamente los mismos datos que la variable "Estudiante1".

Mostrar Ítems

Podemos usar el método "Log" para mostrar los contenidos de "Estudiante1", aunque el resultado sería este:

```
Log(Estudiante1)
```

```
[IsInitialized=false, ID=FS23534X21, Apellidos=Ioannidis  
, Nombre=Alkinoos, Dirección=Atenas, Grecia, Teléfono=+303465854234  
]
```

Para mostrar o usar las propiedades de un tipo se puede usar el nombre junto el nombre del ítem.

Log(Estudiante1.Apellidos & " " & Estudiante1.Nombre)

Es útil también crear una rutina que acepte un tipo y que muestre o procese los elementos de la variable recibida.

```
Private Sub LogEstudiante(est As Estudiante)  
    Log(est.Nombre)  
    Log(est.Apellidos)  
    Log(est.Dirección)  
    Log(est.Teléfono)  
End Sub
```

Array de un Tipo

Se pueden crear arrays de un Tipo para poder gestionar más estudiantes. Por ejemplo:

```
Sub Class_Globals
    Type Estudiante(Apellidos As String, Nombre As String, _
        Dirección As String, Teléfono As String)

    Public Estudiantes(10) As Estudiante
End Sub
```

La sentencia Estudiantes(10) crea un array de 10 estudiantes. Cada ítem puede ser usado en un bucle o referenciándose mediante su índice.

Estudiantes(0). Apellidos = "Paul"

Estudiantes(0). Nombre = "Belmond"

Listas de un Tipo

Una lista puede contener variables de un Tipo creado.

```
Sub Class_Globals
    Type Estudiante(Apellidos As String, Nombre As String, _
        Dirección As String, Teléfono As String)

    Public listaEstudiantes As List
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    ...
    ...
    listaEstudiantes.Initialize
    listaEstudiantes.Add(Estudiante1) 'Añadir estudiante a la lista

    'Obtener la lista de ítems y mostrarlos
    For i = 0 To listaEstudiantes.Size-1
        Private est As Estudiante
        est = listaEstudiantes.Get(i)
        LogEstudiante(est)
    Next
End Sub

Private Sub LogEstudiante(est As Estudiante)
    Log(est.Nombre)
    Log(est.Apellidos)
    Log(est.Dirección)
    Log(est.Teléfono)
End Sub
```



Mapas de un Tipo

También se puede usar un mapa para guardar datos de un tipo creado siempre que como clave se use un valor único.

En el caso de los estudiantes, se podría usar como clave única un número identificativo (ID), un número de registro, un correo electrónico, etc.

```
Sub Class_Globals
  Type Estudiante(id As String, Apellidos As String, _
  Nombre As String, Dirección As String, Teléfono As String)

  Public mapaEstudiantes As Map
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)

  Estudiante1.ID = "FXA47345S3"
  Estudiante1.Apellidos = "Ioannidis"
  Estudiante1.Nombre = "Alkinoos"
  Estudiante1.Dirección = "Atenas, Grecia"
  Estudiante1.Teléfono = "+303465854234"

  mapaEstudiantes.Initialize
  mapaEstudiantes.Put(Estudiante1.ID, Estudiante1)

End Sub
```

De este modo, es muy fácil encontrar un estudiante usando su número identificativo (ID) invocando el método "Get".

Guardar en ficheros KVS

Los ficheros comentados en el tema 17 son ficheros de texto que normalmente no sirve para guardar estructuras complejas como los tipos, listas y mapas. Existe otro tipo de ficheros en B4J llamados ficheros KVS (key value store).

La forma en que funcionan es similar a los mapas. Sólo se necesita un clave y la estructura que quieres guardar. Los ficheros KVS son esencialmente una base de datos, pero lo importante es que para el desarrollador esta gestión es completamente transparente y sólo debe usar los métodos adecuados.

Los métodos en un fichero KVS son los siguientes:

Declaración de un fichero KVS

Para declarar un fichero KVS se debe usar la biblioteca KeyValueStore que debes marcar en la pestaña de bibliotecas (librerías).

Seguidamente, crea un variable de tipo **KeyValueStore**.

```
Private ficheroKVS As KeyValueStore
```



Inicializar el fichero KVS

En el método **Initialize** se indicará la carpeta donde se guardará el fichero y su nombre.

```
File.MakeDir(File.DirTemp, "tema18")

ficheroKVS.Initialize(File.DirTemp & "tema18", "kvsData.dat")

Log(File.DirTemp & "lesson18")
```

La sentencia anterior crea un carpeta llamada "tema18" dentro de la carpeta temporal, crea un fichero llamado "kvsData.dat" y finalmente muestra la ruta al fichero en la pantalla de Log.



Recuerda

El fichero que crees no puede abrirse con otro visor (como Notepad++). Sólo puedes abrirlo con tu programa.

Insertar ítems en un fichero KVS

Se usa el método "Put" para guardar datos en un fichero KVS. Cada inserción se llama "Registro". Es un prerequisite que se use una clave única para poder referirse a cada registro. El siguiente ejemplo inserta la variable "Estudiante1" usando el número identificativo del estudiante como clave.

```
ficheroKVS.Put(Estudiante1.ID, Estudiante1)
```

De este modo puedes guardar cualquier tipo de dato como Listas, Mapas, Cadenas de texto, variable simples (números), tipos y arrays (sólo arrays de bytes u objetos), así como combinaciones (una lista de mapas, por ejemplo).

La declaración de tipos debe hacerse siempre en B4XMainPage.

Los mapas también se pueden guardar usando el método **PutMapAsync**. Es además la mejor forma de guardar un mapa ya que usa la propia clave del mapa como clave para cada registro.

```
ficheroKVS.PutMapAsync(mapaEstudiantes)
```

Recuperar ítems de un fichero KVS

Se usa el método "Get" para obtener ítems de un fichero KVS. El valor devuelto es un objeto. Asegúrate de asignar el valor devuelto a una variable del mismo tipo.

El siguiente ejemplo lee un registro Estudiante de un fichero KVS:

```
Estudiante3 = ficheroKVS.Get("FS23534X21")
LogEstudiante(Estudiante3)
```

El valor "FS23534X21" es la clave del registro que queremos recuperar y el método LogEstudiante recibe un estudiante y lo muestra en pantalla.





Recuerda

Si la clave no existe, entonces tendrás un problema en tu programa. La existencia de un registro debe comprobarse antes de asignar el resultado a una variable. Ello puede hacerse con el método `ContainsKey`.

Se puede leer un mapa completo usando el método **GetMapAsync**. Este método acepta una lista de claves como parámetro y devuelve un mapa con las claves y sus correspondientes valores.

```
Log("Mostrar claves")
Private claves As List = ficheroKVS.ListKeys
For i = 0 To claves.Size-1
    Log(claves.Get(i))
Next
```

Finalmente, a continuación del código anterior, empleamos `GetMapAsync`:

```
Wait For (ficheroEstu.GetMapAsync(keys)) Complete (mapSt As Map)
```

Para guardar un mapa en un fichero KVS usaremos:

```
Wait For (ficheroEstu.PutMapAsync(keys)) Complete (Success As Boolean)
```

Comprobar la existencia de un registro

Para comprobar la existencia de una clave en un fichero KVS se debe usar el método `ContainsKey`:

```
If ficheroKVS.ContainsKey("FS23534X21") Then
    Estudiante3 = ficheroKVS.Get("FS23534X21")
Else
    Log("Clave ID errónea")
End If
```

`ContainsKey` devuelve `True` si encuentra la clave en el fichero.

Borrar un registro de un fichero KVS

El método `"Remove"` borra un registro con una clave indicada y el método `"DeleteAll"` borra todos los registros.

```
ficheroKVS.Remove ("FS23534X21")
ficheroKVS.DeleteAll
```



Recuerda

Es una buena práctica cerrar un fichero que ya no vas a usar más con el método `.close`

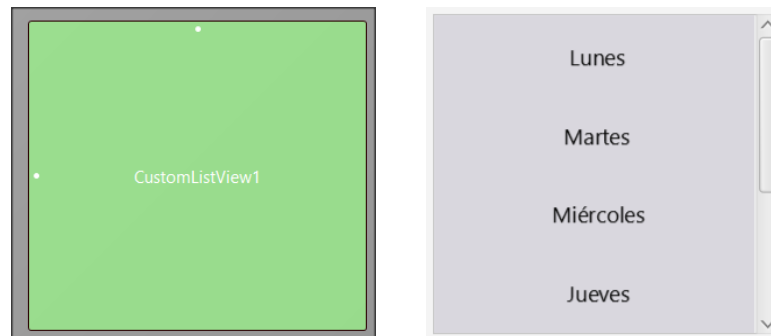


Más Views

Dos vistas importantes que te ayudarán a mostrar o elegir elementos de tipos de datos complejos, listas o mapas son **CustomListView** y **B4XComboBox**.

CustomListView

Crea una lista de ítems que puedes elegir con el ratón:



El valor devuelto es el que se haya indicado al crear el ListView.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private lstItems As List

    Private CustomListView1 As CustomListView 1
    Private lblFecha As Label
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")

    lstItems.Initialize
    lstItems.AddAll(Array As String("Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo")) 2

    For i = 0 To lstItems.Size-1
        CustomListView1.AddTextItem(lstItems.Get(i), i) 3
    Next

End Sub

Private Sub CustomListView1_ItemClick(Índice As Int, Valor As Object) 4
    lblFecha.Text = Valor
End Sub
```

1. Declarar una variable del tipo CustomListView.
2. Crear una lista que contendrá todos los ítems que aparecerán en la CustomListView

3. Recorrer la lista y colocar cada ítem en la CustomListView1 usando el método **AddTextItem**.

AddTextItem recibe un valor que mostrará y un índice que corresponde con el valor que quieres mostrar. El ejemplo anterior corresponden a los valores 0 a 6 para representar los días del lunes al domingo.

4. Cuando se hace clic en un ítem de la lista, se dispara el evento **_ItemClick**. El ejemplo muestra el índice del ítem pulsado en la etiqueta lblFecha.

Otros métodos para CustomListView son:

- **Clear** As String
Borra todos los componentes de la CustomListView.
- **GetValue** (Índice As Int) As Object
Devuelve el valor que hay en el índice indicado como parámetro.
- **Size** As Int [sólo lectura]
Devuelve el número de ítems.

Marcar un ítem de una lista

Como se indicó antes, al hacer clic en un ítem de una lista se dispara el evento **_ItemClick**. Para que el ítem se quede marcado, el programador debe usar alguna herramienta proporcionada por el lenguaje (por ej. La biblioteca CLVSelections) o escribir su propio código. El siguiente algoritmo supone que has creado una variable en **Class_Global** llamada **ítemElegido** de tipo Int.

```
Private Sub CustomListView1_ItemClick (Índice As Int, Valor As Object)
    If ítemElegido = -1 Then
        Private p As B4XView = CustomListView1.GetPanel(Índice)
        p.GetView(0).Color = xui.Color_Blue
        ítemElegido = Índice
    Else
        Private p As B4XView = CustomListView1.GetPanel(selectedItem)
        p.GetView(0).Color = xui.Color_White
        If ítemElegido = Índice Then
            ítemElegido = -1
        Else
            Private p As B4XView = CustomListView1.GetPanel(Índice)
            p.GetView(0).Color = xui.Color_Blue
            ítemElegido = Índice
        End If
    End If
End Sub
```

Inicialmente, **ítemElegido** se pone a -1 para indicar que no se ha elegido nada. El código emplea dos sentencias:

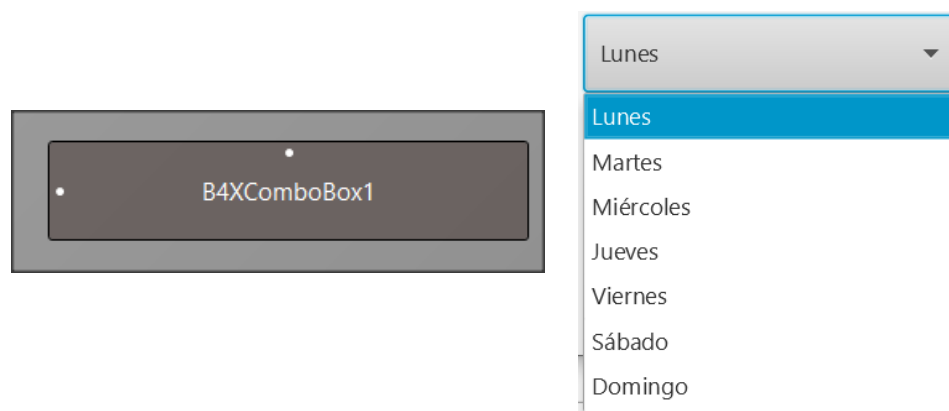
```
Private p As B4XView = CustomListView1.GetPanel(Índice)
p.GetView(0).Color = xui.Color_Blue
```

Cada ítem de una lista se crea dentro de una caja llamada "panel". Puedes fijar su color accediendo a la caja mediante el método **GetPanel(Índice)** donde Índice indica el valor de la línea que ha sido pulsada. La sentencia **p.GetView(0).Color** fija el color que el programador desea usar. Así:

1. Si se hace clic en un ítem de la lista, la rutina comprueba el valor de **ítemElegido** y, si vale -1, entonces pone de color azul el fondo de la línea elegida y guarda en **ítemElegido** el valor del índice que devuelve el evento **_ItemClick**
2. Si **ítemElegido** ya contiene un valor, se pone el color de fondo de la línea de color blanco y después tenemos 2 opciones:
 - a. Si hemos elegido el mismo ítem, deseccionamos el ítem y la variable **ítemElegido** se pone a -1.
 - b. Si hemos elegido otro ítem, el nuevo ítem se pone de color azul y la variable **ítemElegido** se pone con el valor del índice.

B4XComboBox

B4XComboBox muestra una lista de ítems desplegable. El usuario podrá elegir cualquier de ellos. Al contrario que con CustomListView, el valor devuelto siempre es un número que indica la posición que ocupa el ítem dentro del ComboBox, con lo que el programador debe ser capaz de procesar ese correctamente.




```

Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private lstItems As List
    Private B4XComboBox1 As B4XComboBox
    Private lblCmbFecha As Label
End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
    lstItems.Initialize
    lstItems.AddAll(Array As String("Lunes", "Martes",
    "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"))

    B4XComboBox1.SetItems(lstItems)
End Sub

Private Sub B4XComboBox1_SelectedIndexChanged(Índice As Int
    lblCmbFecha.Text = Index
End Sub

```

1. Declarar una variable de tipo **B4XComboBox**.
2. Crear una lista que contenga los ítems que aparecerán en el B4XComboBox
3. Colocar los ítems en la lista B4XComboBox. Cuidado, porque no es necesario un bucle como en CustomListView
4. Si se elige un ítem, se dispara el evento **_SelectedIndexChanged** y devuelve el índice del elemento elegido.

Ejercicios

1. Crea un tipo llamado **Cliente** con las variables siguientes:

ID, Nombre, Apellidos, Teléfono

2. Crea una lista de Clientes con los siguientes ítems:

Id	Nombre	Apellidos	Teléfono
A3473	John	Smith	4563454
B1753	Selim	Al Huarizmi	6532578
C6544	Mateo	Sandor	7345346
C6323	Lucía	Graham	1231345
F1462	Noam	Bell	6978323

3. Guarda la lista en un fichero KVS.
4. Crea un botón que al pulsarse rellene la CustomListView con la información de los clientes anteriores.
5. Crea un botón insertar que, al pulsarse, muestre un DialogBox que pida la información de un nuevo cliente y lo añada a la CustomListView.
6. Guarda el nuevo cliente también en el fichero KVS.
7. Cuando el usuario haga clic en un cliente se mostrará una ventana para confirmar que se desea borrar ese cliente. Si se confirma el borrado, se eliminará de la CustomListView y del fichero KVS.

Tema 19 – Proyecto Final

🕒 **10h**

Lo que los estudiantes aprenderán

- Usar los conocimientos adquiridos para crear un proyecto final



Estimado Profesor

La solución propuesta abarca todo el conocimiento adquirido durante el curso. Puedes ampliar las preguntas para adaptarlas al nivel de tu clase y al tiempo disponible.

Para aprender cómo se crea una biblioteca escolar puedes leer el libro "Setting Up and Running a School Library by Nicola Baird" aquí en [https://files.Eric.Ed. Gov/Fulltext/Ed536911.Pdf](https://files.Eric.Ed.Gov/Fulltext/Ed536911.Pdf)

Los nombres de los libros y los autores han sido tomados del Proyecto Gutenberg https://www.gutenberg.org/ebooks/offline_catalogs.html

Los nombres de las editoriales, las fechas de publicación y el nombre de los estudiantes son aleatorios.

Prokopis Leon, pliroforikos[at]gmail.com

Crear una Aplicación de Biblioteca Escolar

Una biblioteca escolar posee una colección de libros que presta a los estudiantes de tres clases. Los libros se colocan en estantes numerados del 1 al 60. Para cada libro, se guarda la siguiente información:

- Código
- Título
- Escritor
- Año de publicación
- Editorial
- Estante

De los estudiantes, se guarda la siguiente información:

- Número de registro
- Nombre
- Apellidos
- Clase
- Teléfono

Construye la aplicación descrita más abajo:



Ficheros

Se proporcionan dos ficheros llamados `books.txt` y `students.txt` (dentro de la carpeta de materiales).

Algunos de los elementos del fichero son:

`books.txt`

```
30;The Life of Abraham Lincoln;Henry Ketcham;1866;New Public publ.;54
31;Christopher Columbus; Mildred Stapley;1954;Cider publ.;43
32;The Adventures of Ferdinand Count Fathom; Tobias Smollett;1982;Orange punl.;32
33;Tales of the Jazz Age;F. Scott Fitzgerald;1944; Gutenberg publ.;5
34;The Old Stone House;Anne March;1904;Orange punl.;50
```

Cada libro posee los siguientes campos para que se distingan unos de otros y que se separen por el carácter ";":

- Id
- Título
- Escritor
- Año
- Editorial
- Estante

`students.txt`

```
1001;Jude;Segers;A;7900209
1002;Desire;Cid;A;7047635
1003;Madelyn;Pittard;A;9011036
1004;Lorita;Tomczak;A;6677490
1005;Lynwood;Posey;A;9014379
1006;Nella;Felps;A;8423818
```

Cada estudiante posee los siguientes campos para distinguirlos unos de otros y que se separen con el carácter ";":

- Id
- Nombre
- Apellidos
- Clase
- Teléfono

Implementa una función que inserte los anteriores ficheros de texto en dos mapas con nombres `mapBooks` y `mapStudents`. La clave será el primer campo que aparece en cada línea de los ficheros que representan el "id" del libro y del estudiante, respectivamente. Los valores de cada ítem se describen más abajo.



Consejo para el profesor

Antes de leer los ficheros tienes que leer todo su contenido como una cadena de texto (string) y luego ir uno a uno comprobando los caracteres para ir colocando los ítems en el array de ítems que usarás para escribir en el mapa. Para comprobar una cadena usa el método **CharAt**(índice).



Ejemplo de Mapa de Estudiantes

Clave	Tipo				
	Id	Nombre	Apellidos	Clase	Teléfono
1001	1001	Jude	Segers	A	7900209
1002	1002	Desire	Cid	A	7047635
1003	1003	Madelyn	Pittard	A	9011036

Datos

Tipos

1. Crear el tipo Libro

El tipo Libro tendrá estos elementos:

- Id
- Título
- Escritor
- Año
- Editorial
- Estante

Funciones:

- Insertar Libro
- Borrar Libro

2. Crea el tipo Estudiante

Propiedades:

- Id
- Nombre
- Apellidos
- Clase
- Teléfono

Funciones:

- Insertar Estudiante
- Borrar Estudiante

Cada estudiante que haya pedido prestado libros posee su propio fichero KVS con el nombre "ID número.dat". Por ejemplo, el estudiante con ID 21 tiene un fichero llamado "21.dat". En el fichero se guardan los libros prestados al estudiante usando un mapa como estructura. Por ejemplo, si has pedido prestado dos libros, deberías tener un mapa con la siguientes estructura:

("11", "03/27/2021")

("14", "04/01/2021")

Donde el primer número es el ID del libro prestado y también la clave del mapa; la fecha se guarda como valor asociado a la clave.



Diseño de la pantalla y Funciones

El programa debe ofrecer un menú central de botones para invocar a la pantalla de gestión adecuada. En concreto:

1. Libro

Incluirá una lista CLV de libros de la biblioteca que son cargados desde el fichero books.txt, así como botones para insertar y borrar libros. Cuando se pulse el botón insertar, aparecerá un diálogo para indicar los datos del nuevo libro que será guardado en el fichero books.txt. Cuando se pulse el botón Borrar, se borrará el libro elegido de la lista.

2. Estudiante

Incluirá una lista CLV de estudiantes de la escuela que se cargarán del fichero students.txt y botones para borrar e insertar estudiantes. Cuando se pulse el botón insertar, se creará un diálogo para pedir los datos del nuevo estudiante y guardarlo en el fichero students.txt. Al pulsar el botón borrar, se borrará el estudiante de la lista, junto con los libros que se le han prestado suponiendo que los ha devuelto.

3. Préstamo

La pantalla de préstamo tendrá un ComboBox que contendrá los nombres de los estudiantes una lista CLV de los libros de la biblioteca. Habrá un botón "Prestar" que cuando se pulse hará lo siguiente:

Si hay un fichero del estudiante

- Cargará el mapa completo del fichero
- Añadirá la nueva clave con el ID del libro prestado y la fecha del préstamo
- Guardará de nuevo el mapa

Si no hay fichero del estudiante

- Creará el fichero
- Creará un mapa con la información del préstamo (ID y fecha)
- Guardará el fichero

Supón que hay tantas copias de cada libro como necesites.

4. Devolución

La pantalla de devolución incluirá un ComboBox con el nombre de los estudiantes y una lista CLV de los libros prestados. Además, habrá un botón "Devolver".

Si se pulsa el botón "Devolver" se borrará el libro de la lista del estudiante y del fichero del estudiante.

Si no se le ha prestado ningún otro libro, el programa borrará el fichero también.

Hoja de pistas

1. Cómo declarar un tipo

```
Sub Class_Globals
    Type Estudiante(Nombre As String, Apellidos As String, _
        Dirección As String, Teléfono As String)

    Private Estudiante1 As Student
End Sub
```

2. Cómo guardar el contenido de un fichero en una cadena de texto:

```
fichEst = File.ReadString(File.DirAssets, "students.txt")
```

3. Comprobar carácter a carácter una cadena con el contenido de un fichero para crear un array de estudiantes.

```
Private i As Int = 0
For j = 0 To ficheroEst.Length-1
    If fichEst.CharAt(j) <> ";" And fichEst.CharAt(j) <> CRLF Then
        estudiante(i) = estudiante(i) & fichEst.CharAt(j)
    Else if fichEst.CharAt(j) = ";" Then
        i = i + 1
    else if fichEst.CharAt(j) = CRLF Then
        i = 0
        listaDevuelta.Add(estudiante)
        Private estudiante(5) As String
    End If
Next
```

4. Convertir una lista de arrays a un mapa:

```
For i = 0 To lista.Size-1
    Private estud(5) As String
    estud = lista.Get(i) \ Crear un array de un ítem de la lista
    Private est As Student \ Crear una variable del tipo estudiante
    est.Initialize
    est.ID = estud(0) \ Poner cada ítem del array en el tipo
    est.Nombre = estud(1)
    est.Apellidos = estud(2)
    est.Clase = estud(3)
    est.Teléfono = estud(4)
    est.Prestado = 0
    mapaEstudiantes.Put(estud(0), est) \ Insertar tipo en mapa
Next
```

5. Colocar espacios en una cadena para incrementar su tamaño hasta un número indicado:

```
Do While cadenal.Length <= 5
    cadenal = cadenal & " "
Loop
```

6. Marcar – Desmarcar un ítem de una lista CLV

Cada ítem de una lista se crea dentro de una caja llamada "panel". Puedes fijar su color accediendo a la caja con el método `GetPanel(índice)` donde el índice es el valor actual de la lista en la que se ha pulsado. Después:

- Si el ítem de la lista se pulsa, el método comprueba el valor del ítem elegido y, si es -1, entonces establece en azul el color de fondo de la línea elegida y guarda en `ítemElegido` el índice devuelto por el evento `_ItemClick`.
- Si `ítemElegido` ya tiene un valor, se elige el color blanco como color de fondo y entonces habría 2 posibilidades:
 - Se haya pulsado sobre un ítem ya elegido, en cuyo caso se desmarca el ítem y `ítemElegido` se pone a -1
 - Se haya pulsado en otro ítem, con lo que `ítemElegido` tomará el valor del índice.

```
Private Sub clvLibros_ItemClick (Índice As Int, Valor As Object)
    If libroElegido = -1 Then
        Private p As B4XView = clvLibros.GetPanel(Índice)
        p.GetView(0).Color = xui.Color_Blue
        libroElegido = Índice
    Else
        Private p As B4XView = clvLibros.GetPanel(libroElegido)
        p.GetView(0).Color = xui.Color_White
        If libroElegido = Índice Then
            libroElegido = -1
        Else
            Private p As B4XView = clvLibros.GetPanel(Índice)
            p.GetView(0).Color = xui.Color_Blue
            libroElegido = Índice
        End If
    End If
End Sub
```

7. Cargar ítem de un fichero KVS en un mapa

La sentencia se debe ejecutar con `Wait For`.

```
Wait For (fichEstudiante.GetMapAsync(fichEstudiante.ListKeys))
Complete (mapaEstudiante As Map)
```

El método **`GetMapAsync`** devuelve un mapa con los ítems. Primero hay que declarar el mapa **`mapaEstudiante`** e inicializarlo.

8. Guardar los ítems en un fichero KVS de mapas

```
Wait For (fichEstudiante.PutMapAsync(mapaEstudiante)) Complete (Suces As Boolean)
```

9. Borrar un fichero de una carpeta:

```
File.Delete(File.DirTemp, <nombre fichero>)
```


Tema 20 – De B4J a B4A



Lo que los estudiantes aprenderán

- Transferir un proyecto de B4J a B4A

¡Felicidades!

Al haber llegado hasta aquí puedes afirmar que has adquirido un buen nivel de conocimiento del lenguaje B4X. Lo que has aprendido hasta ahora es sólo el principio de tu camino en el “arte” de la programación.

Además, el lenguaje B4X puede usarse para construir aplicaciones móviles en Android e iOS. Con unos pocos cambios y sin necesidad de aprender nuevos comandos puedes transferir una aplicación escrita con B4J en Windows a una aplicación para Android escrita en B4A.

En este tema veremos cómo convertir un programa (ya dado) en un programa para Android.



Imagen 54.
Lenguajes de B4X

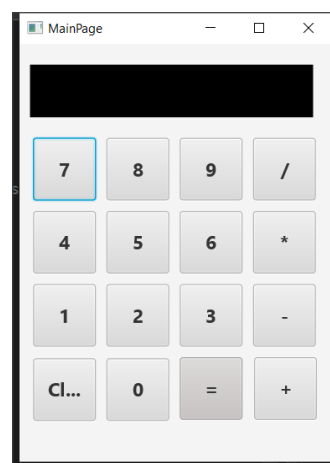
Descripción de la aplicación en B4J

La aplicación es una calculadora simple que realiza operaciones básicas.

Diseñador

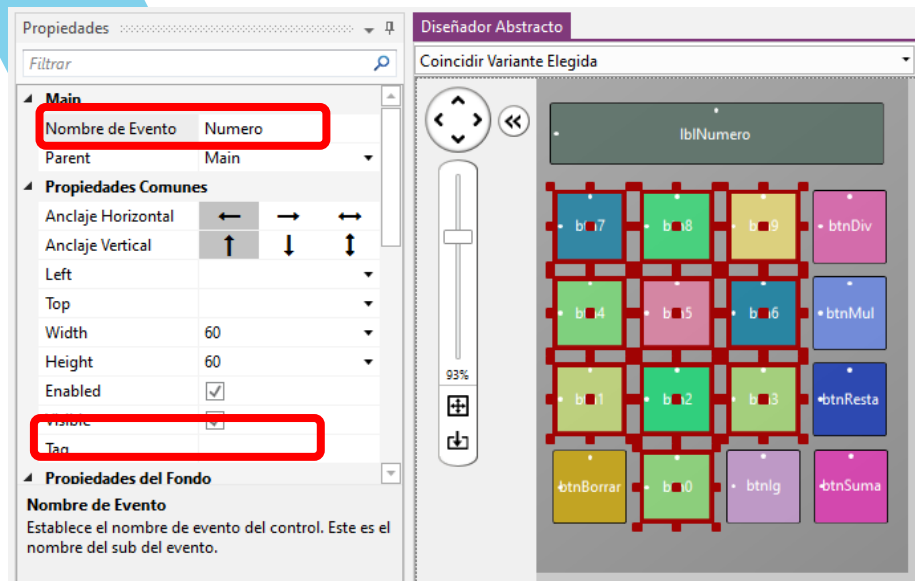
La aplicación posee 16 botones y una etiqueta para mostrar los números.

Hay 10 botones para los números del 0 al 9 y el resto son para las operaciones básicas, el igual y el borrar.



Picture 55 Simple Calculator

En aplicaciones en las que hay que gestionar múltiples botones no hace falta crear eventos `_click` para cada botón, sino que se puede crear un evento `_click` para todos.



Por ejemplo, primero elegimos todos los botones del 0 al 9, después en "Nombre de Evento" escribimos un nombre, por ejemplo "Numero". Ahora todos los botones tendrán un evento _click común llamado "Numero_Click".

```
Private Sub Numero_Click
    Dim b As Button
    b = Sender
    Log(b.Tag)
    If done Then
        lblNumero.Text = 0
        hecho = False
    End If
    lblNumero.Text = lblNumero.Text & b.Tag
End Sub
```

Además, para poder distinguir un botón de otro, tendrás que poner a cada botón un valor en la propiedad "Tag". Por ejemplo, para el botón del 1, podemos poner como "Tag" el número 1, para el botón 2, el valor 2 y así sucesivamente.

Ahora, ya puedes consultar el contenido de la propiedad "tag" dentro del evento "Numero_Click" para ver qué botón se ha pulsado.

"Sender" representa el botón que se ha pulsado. Lo guardamos en "b".

El código del programa es el siguiente. Fíjate que no se han declarado los botones como variables.

```
Sub Class_Globals
    Private Root As B4XView
    Private xui As XUI
    Private lblNumero As Label
    Private fltNumero1, fltNumero2 As Float
    Private operacion As String
    Private hecho As Boolean 'Será True cuando se acabe la operación
End Sub

Public Sub Initialize
```

```

End Sub

Private Sub B4XPage_Created (Root1 As B4XView)
    Root = Root1
    Root.LoadLayout("MainPage")
    lblNumero.Text = ""
    hecho = False
End Sub

' Al pulsar un botón, se añade a la cadena lblNumber el nuevo
' número obtenido de la propiedad "tag"
Private Sub Numero_Click
    Dim b As Button
    b = Sender
    If hecho Then
        lblNumero.Text = 0
        hecho = False
    End If
    lblNumero.Text = lblNumero.Text & b.Tag
End Sub

' Cuando se pulse el botón "Clear", borramos todos los números
' y operaciones
Private Sub btnBorrar_Click
    lblNumero.Text = 0
    fltNumero1 = 0
    fltNumero2 = 0
End Sub

' Cuando se pulse un botón de operación, fijamos la cadena
' "operación" con el contenido de la etiqueta del botón pulsado
Private Sub operacion_Click
    Dim b As Button
    b = Sender
    operacion = b.Tag
    fltNumero1 = lblNumero.Text
    lblNumero.Text = 0
End Sub

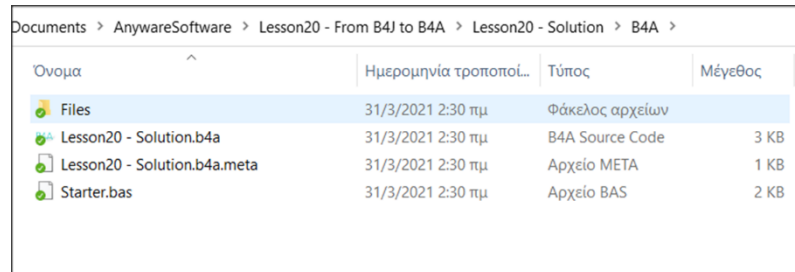
' Cuando se pulse el botón "=", se realiza la operación
Private Sub btnIg_Click
    fltNumero2 = lblNumero.Text
    If operacion = "+" Then
        lblNumero.Text = fltNumero1 + fltNumero2
    else If operacion = "-" Then
        lblNumero.Text = fltNumero1 - fltNumero2
    else If operacion = "*" Then
        lblNumero.Text = fltNumero1 * fltNumero2
    else If operacion = "/" Then
        lblNumero.Text = fltNumero1 / fltNumero2
    End If
    fltNumero1 = lblNumero.Text
    hecho = True
End Sub

```



Transferir la app a B4A y Android

Al iniciar la aplicación ya se crea la carpeta adecuada para B4A y B4i, aunque aún no la hayamos usado:



Όνομα	Ημερομηνία τροποποι...	Τύπος	Μέγεθος
Files	31/3/2021 2:30 πμ	Φάκελος αρχείων	
Lesson20 - Solution.b4a	31/3/2021 2:30 πμ	B4A Source Code	3 KB
Lesson20 - Solution.b4a.meta	31/3/2021 2:30 πμ	Αρχείο META	1 KB
Starter.bas	31/3/2021 2:30 πμ	Αρχείο BAS	2 KB

Para iniciar la transferencia, hay que instalar B4A. Las instrucciones de instalación están en <https://www.b4x.com/b4a.html>.

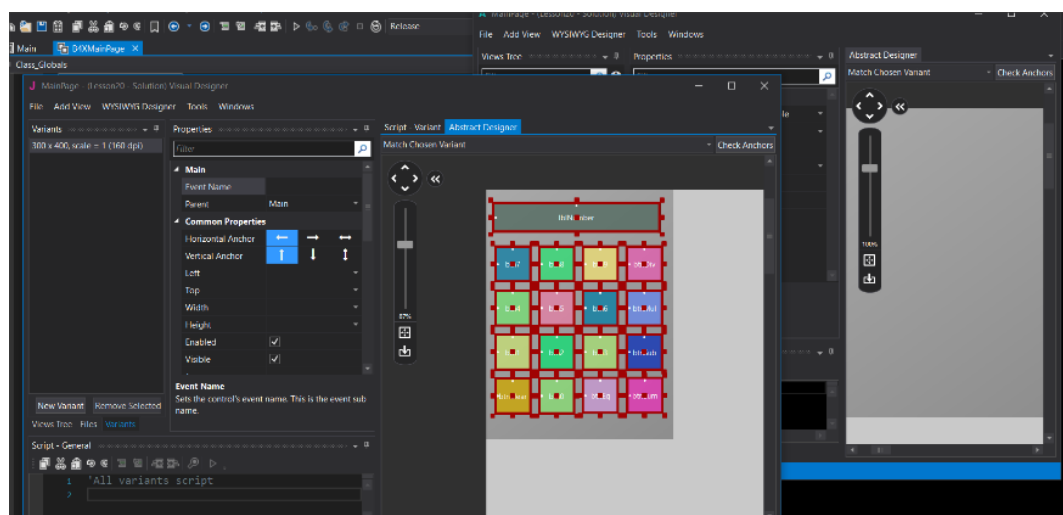
Tras finalizar la instalación, no olvides instalar en tu móvil la aplicación **B4A-Bridge** que te ayudará a transferir la calculadora al móvil. La aplicación es gratuita y se instala desde Google Play Store.

Transferir el diseño

Abre la calculadora con B4J. Abre la calculadora con B4A. Los ficheros importantes están ubicados en la carpeta B4A y tienen extensión "b4a". ¡Todo el código ya está dentro de B4A!

Abre el Diseñador que visualmente es idéntico al de B4J.

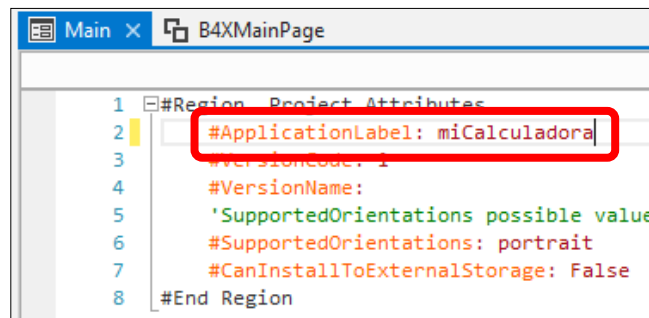
El diseño no se ha transferido aún a B4A, pero se puede hacer fácilmente eligiendo todos los objetos del diseñador de B4J y copiándolos y pegándolos en la pantalla del diseñador de B4A.



Puedes hacer cualquier cambio que quieras al aspecto de tu app. Cuando acabes, guarda los cambios y vuelve a B4A.

Instalar la app en tu móvil

Antes de instalar tu app en un móvil debes indicar el nombre de tu app y el nombre del paquete ("package") que se va a enviar al móvil.



En la pestaña "Main" indica el nombre para tu aplicación en la `#ApplicationLabel` directive:

Ese será el nombre que aparecerá en la pantalla de tu móvil o tableta.

Imagen 56 Etiqueta de la Aplicación

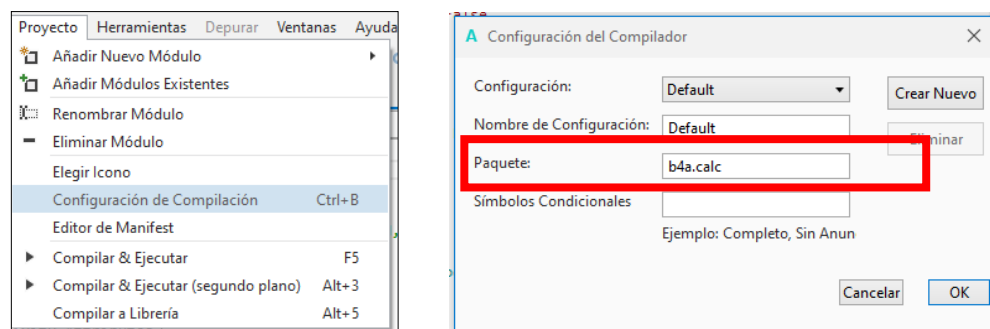
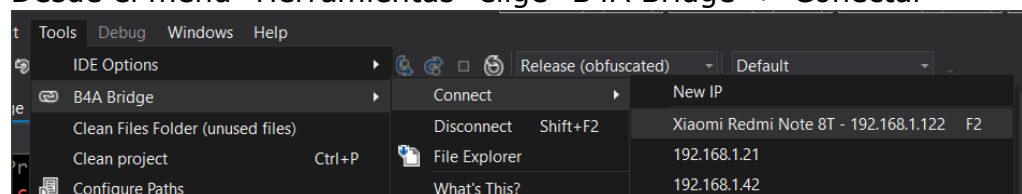


Imagen 57 Configuración de Compilación

Desde el menú "Configuración de Compilación" indica el nombre del paquete: debe ser único para tu dispositivo, con lo que por eso se aconseja que empiece por "b4a". En el ejemplo, el paquete se llama "b4a.calc"

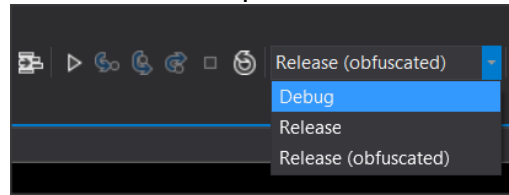
La instalación en el móvil requiere que tengas instalada la aplicación B4A-Bridge previamente:

1. Inicia B4A-Bridge en tu dispositivo Android y asegúrate de que está conectado a tu red local.
2. Desde el menú "Herramientas" elige "B4A Bridge -> Conectar"

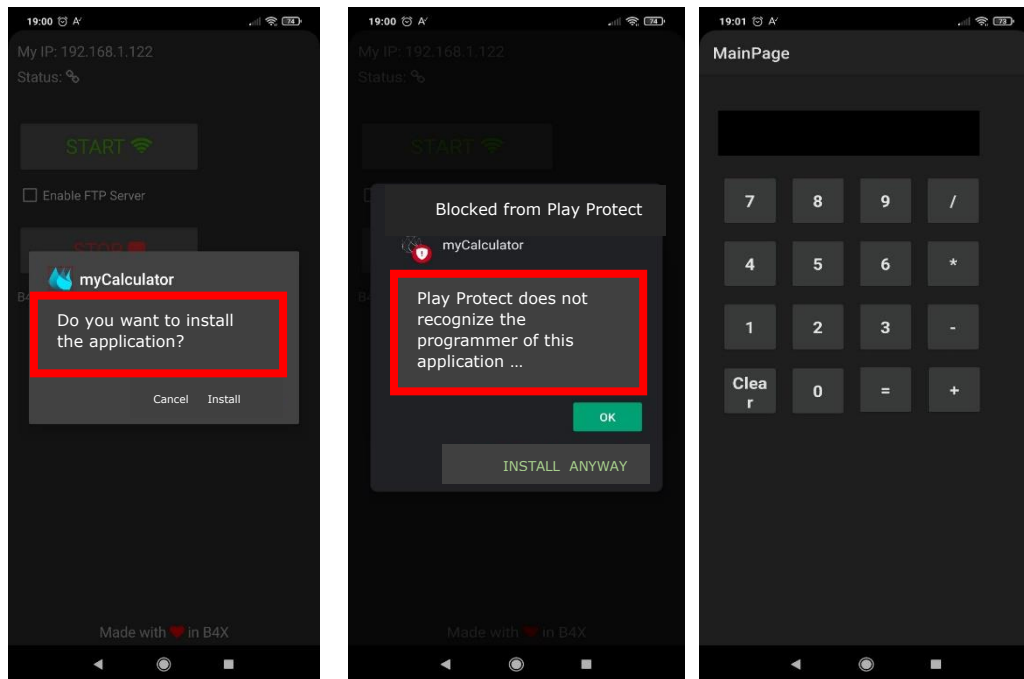


Elige tu dispositivo que debe aparecer en la lista. Si no aparece, asegúrate de comprobar que está conectado a la misma red que tu ordenador.

3. Elige el tipo de instalación que desees. "Debug" si estás depurando la aplicación o "Release" una vez que hayas terminado su desarrollo y quieras ejecutar de forma independiente de B4A.



4. Pulsa el botón "Ejecutar" y comprueba tu móvil en B4A-Bridge.



Durante la instalación, puedes recibir un mensaje de Play Protect, pero puedes ignorarlo y continuar.