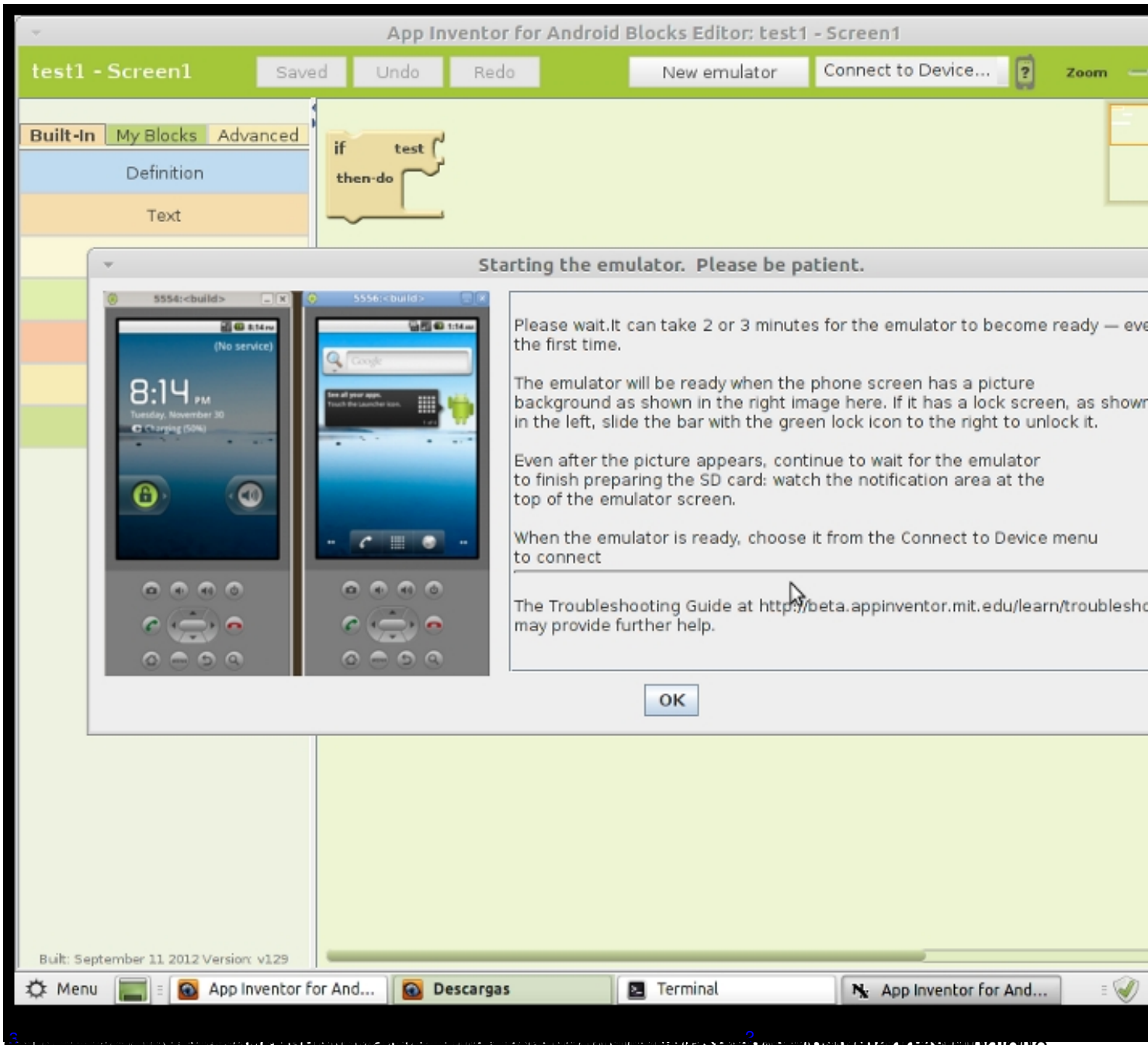
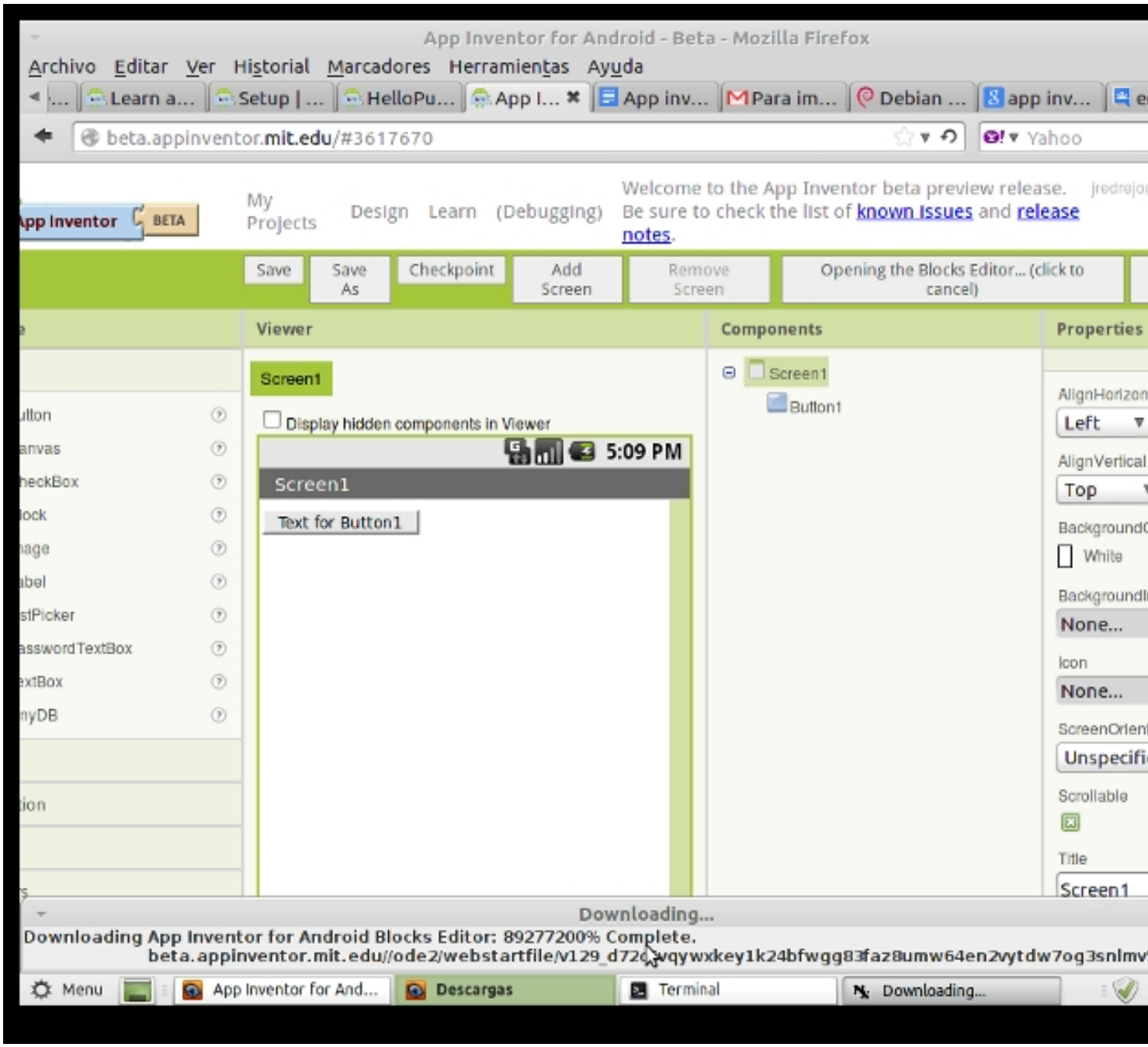
There are no translations available.

En el actual diseño del Bachillerato en España se propone “Tecnologías de la Información y la Comunicación como asignatura optativa”. Entre los objetivos generales de esta asignatura se puede leer “Usar los recursos informáticos como instrumento de resolución de problemas específicos” o “Integrar la información textual, numérica y gráfica obtenida de cualquier fuente para elaborar contenidos propios y publicarlos ... y formatos que faciliten la inclusión de elementos multimedia decidiendo la forma en la que se ponen a disposición del resto de usuarios”. Son muchos los profesores que enseñan algún lenguaje de programación como una de las herramientas para alcanzar estos objetivos.

En este artículo se propone usar App Inventor [1] como parte del curriculum de esta asignatura. App Inventor es al mismo tiempo un lenguaje de programación, una herramienta de diseño y un entorno de desarrollo de aplicaciones para móviles y tablets que funcionen con el sistema operativo Android. App Inventor permite también ejecutar las aplicaciones en un emulador, por lo que no es imprescindible disponer del teléfono para probar los programas que se hagan.


App Inventor for Android Blocks Editor: test1 - Screen1
test1 - Screen1
Saved
Undo
Redo
New emulator
Connect to Device...
Zoom
Built-In
My Blocks
Advanced
Definition
Text
if
test
then-do
Starting the emulator. Please be patient.
5554:<build>
5556:<build>
(No service)
8:14 PM
Please wait.It can take 2 or 3 minutes for the emulator to become ready — eve
the first time.
The emulator will be ready when the phone screen has a picture
background as shown in the right image here. If it has a lock screen, as shown
in the left, slide the bar with the green lock icon to the right to unlock it.
Even after the picture appears, continue to wait for the emulator
to finish preparing the SD card: watch the notification area at the
top of the emulator screen.
When the emulator is ready, choose it from the Connect to Device menu
to connect
The Troubleshooting Guide at http://beta.appinventor.mit.edu/learn/troublesho
may provide further help.
OK
Built: September 11 2012 Version: v129
Menu
App Inventor for And...
Descargas
Terminal
App Inventor for And...




## ¿Por qué App Inventor?

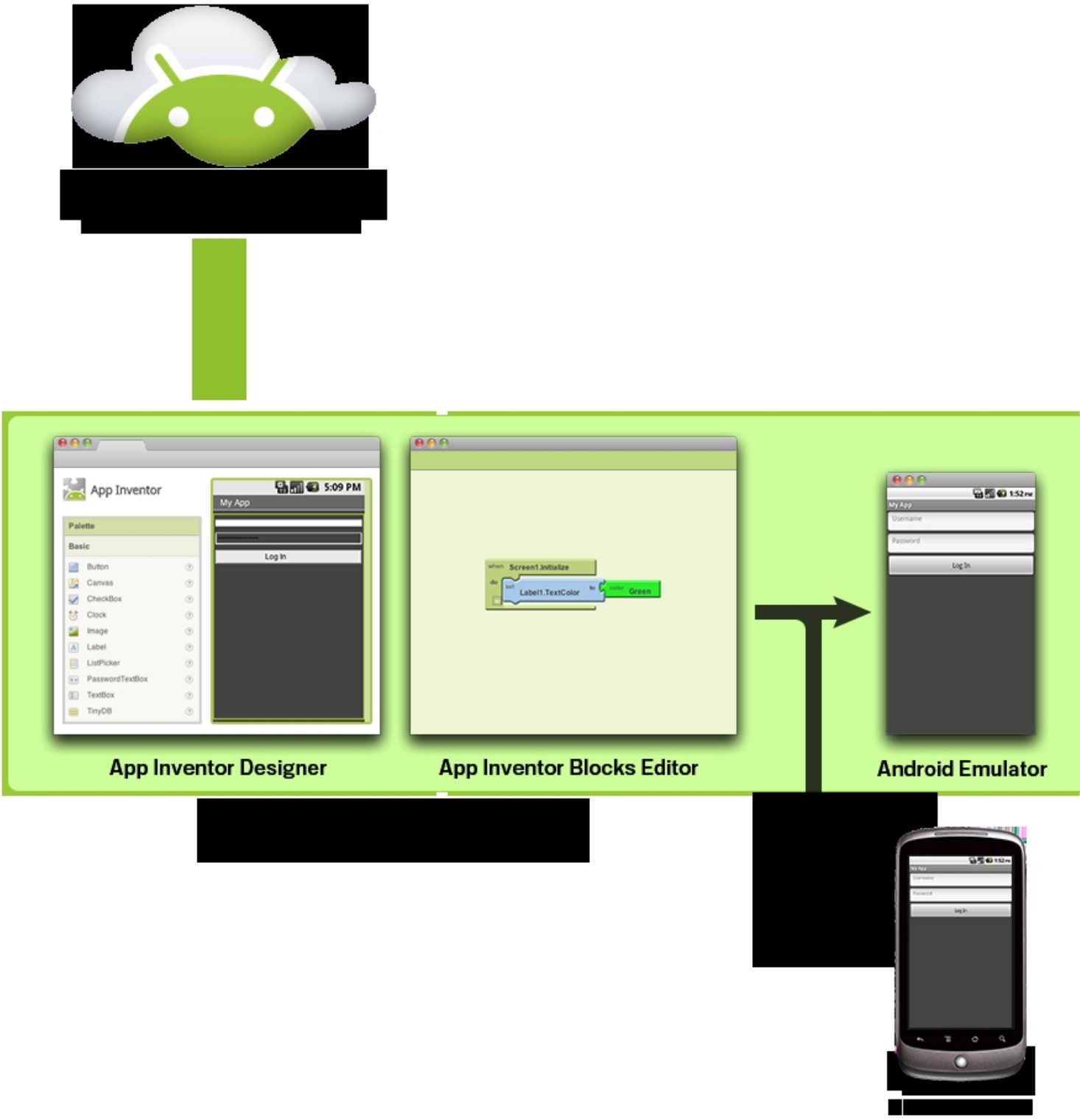
Casi todos los lenguajes de programación tienen una sintaxis que al principiante le suena a chino. Son una mezcla de inglés y extrañas reglas de sintaxis. Como ejemplo, el programa más pequeño que se puede hacer en Java es algo así:

```
class Hola
{
    public static void main(String [] args)
```

```
    {
        System.out.println("Hola mundo");
    }
}
```

Es evidente que al  aprender a programar la sintaxis provoca una curva de aprendizaje significativa. Aunque hay lenguajes más limpios en este sentido que Java, como Python, siempre requiere invertir mucha práctica y tiempo en aprender los comandos, sus reglas de escritura, sentido de los distintos signos de puntuación, etc. Todo este tiempo es tiempo no empleado en aprender a diseñar algoritmos para resolver problemas, tal y como pedía el primero de los objetivos de la asignatura de TIC.

Con App Inventor se aprende a programar usando bloques de programación. Estos bloques están hechos con elementos comunes a la mayoría de los lenguajes de programación existentes. Se colocan bloques para construir bucles, condiciones, variables, etc. que permiten pensar lógicamente y solucionar los problemas de forma metódica, sin perder el tiempo en encontrar el punto y coma o los dos puntos que están donde no deben y producen errores de compilación o ejecución.

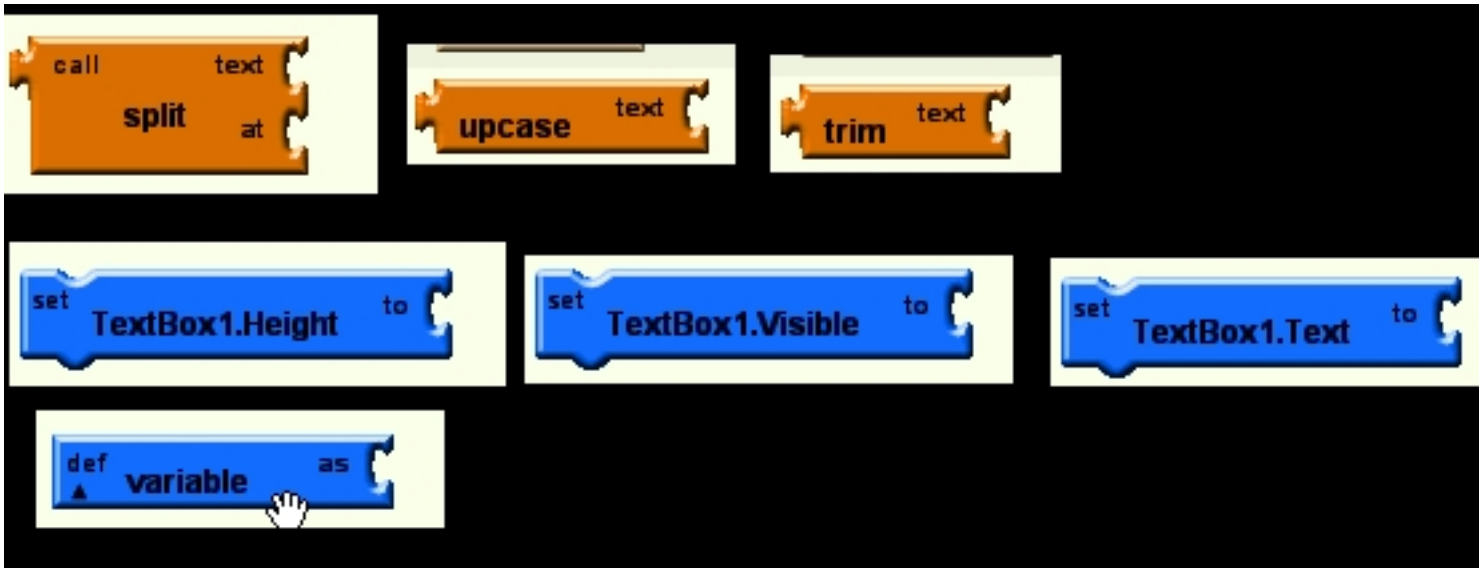



## Elementos de programación en App Inventor

Veamos algunos de esos bloques de App Inventor:

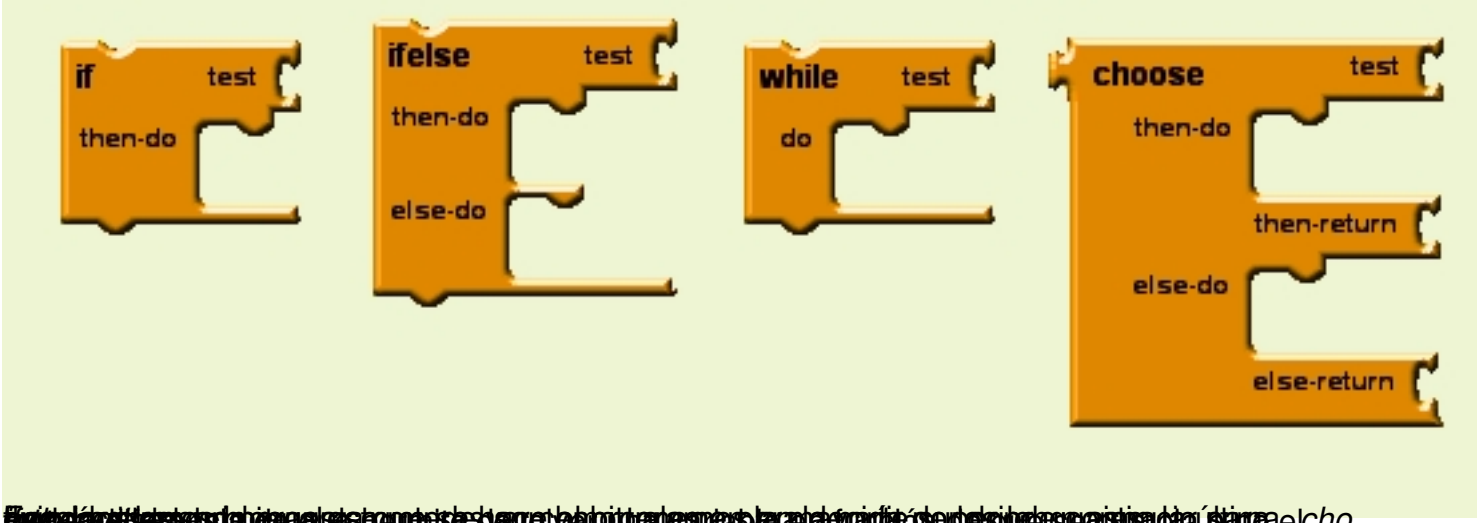
## Sentencias

Al programar necesitamos a menudo decirle al ordenador que haga algo. En App Inventor existen numerosos bloques que son sentencias de programación. Se distinguen rápidamente porque expresan una acción a realizar con un verbo en imperativo. Los más habituales son *call* , *set* y *def* (abreviatura de define). Además, los bloques que son sentencias sólo están disponibles en dos colores: azul para modificar variables o propiedades de un objeto y naranja para llamar a funciones. En la imagen siguiente se ven algunos ejemplos de las sentencias para dividir un texto en partes, convertirlo a mayúsculas, cambiar la altura, texto o visibilidad de un recuadro de texto o definir una variable.



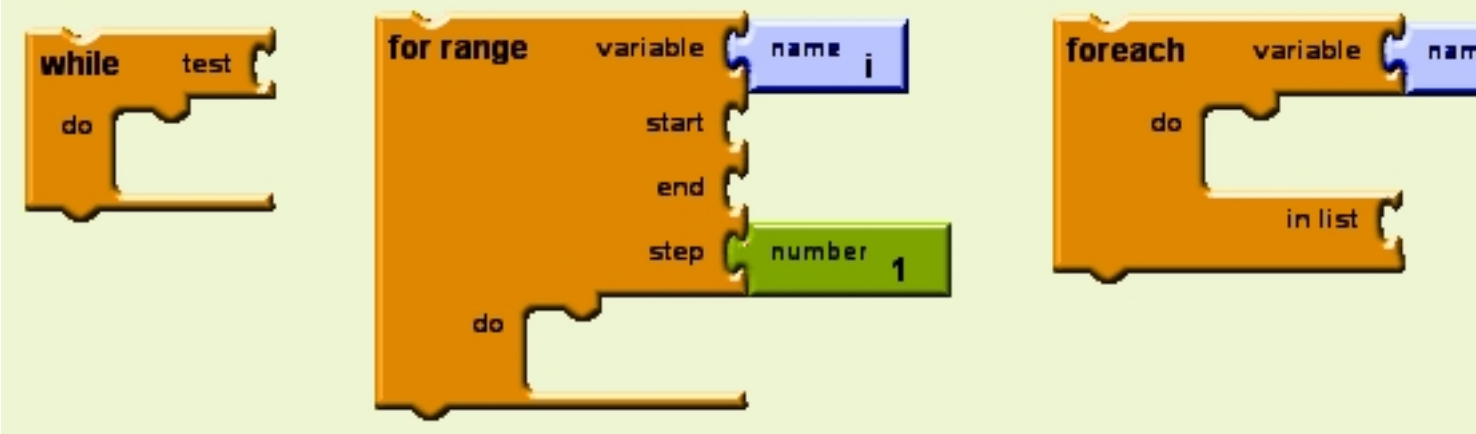

## Condiciones

Con frecuencia necesitamos realizar distintas acciones en función de que ocurra o no algo. En App Inventor para condicionar nuestro programa tenemos las estructuras *if-then, if-then-else , while y choose* :




## Bucles

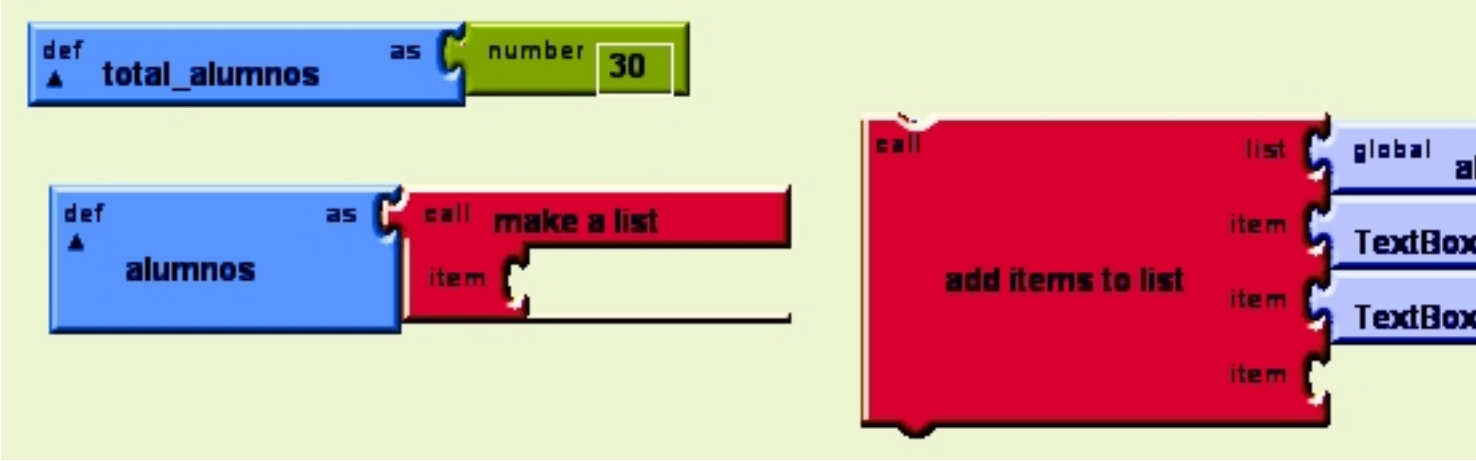
Un bucle provoca la ejecución repetida de varias sentencias. En App inventor disponemos de los bucles *while*, *for range* y *foreach*.




El primero ejecutará el bloque de sentencias que se encajen en el apartado do mientras se cumpla la condición que refleje el test. *for range* es el típico bucle "*for*" de otros lenguajes de programación y, como tal, tiene las opciones de elegir el nombre de la variable que itera, su comienzo, fin y el tamaño de los saltos de iteración. Finalmente,
*foreach*
es un iterador sobre los elementos de una lista.

## Variables

Disponemos de las herramientas para definir y cambiar variables. El tipo puede ser numérico o un texto y “se declara” al asignarle un valor por primera vez.

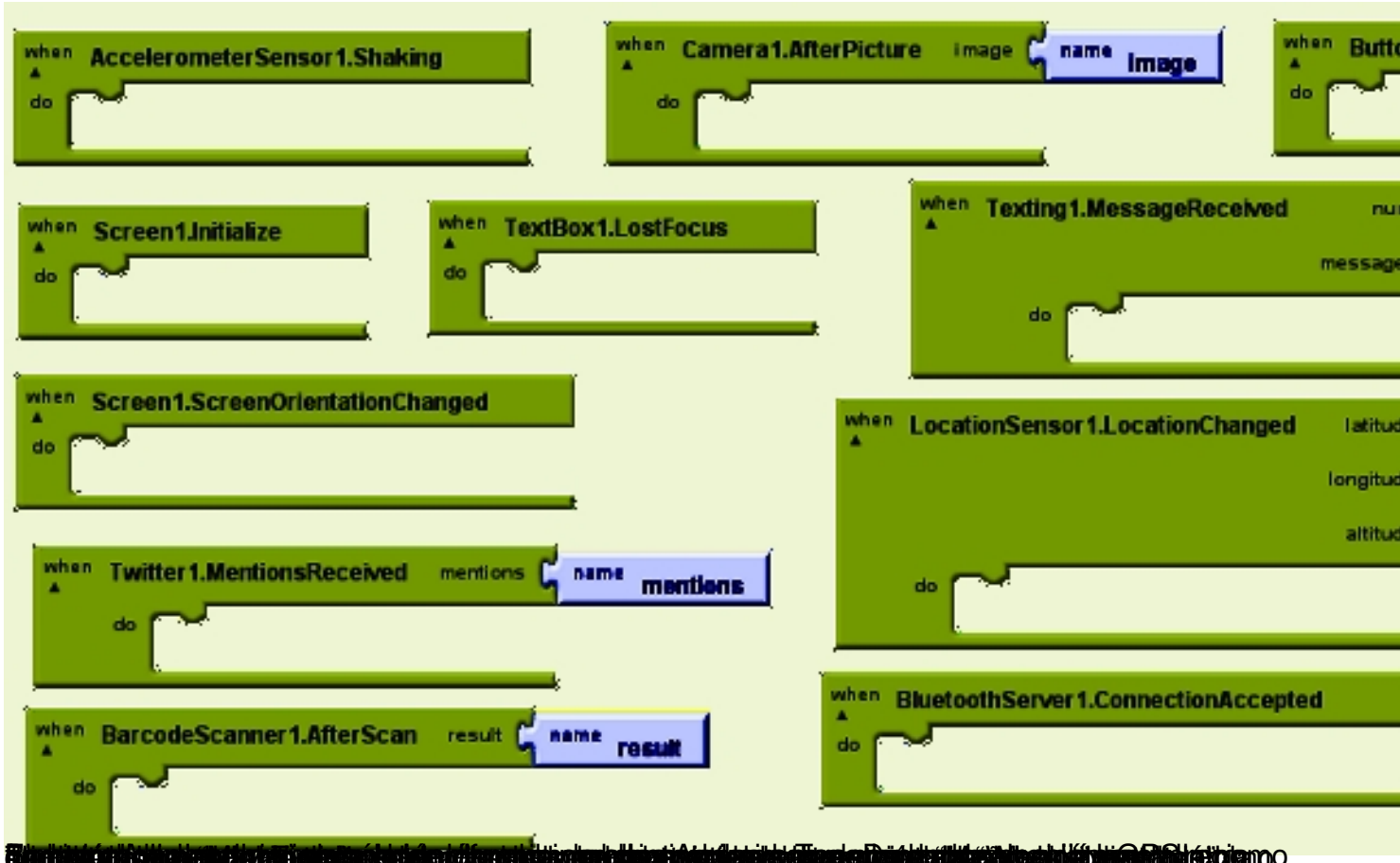



En el gráfico anterior se ve como se declara una variable llamada “total_alumnos” y se le da un valor numérico de 30. Además se ven dos bloques muy importante en App Inventor. Su lenguaje cuenta con una estructura de datos llamada *list* (lista) que es similar a los Arrays de otros lenguajes de programación. Se ve el bloque con el que se define la lista “alumnos” (se define vacía al no añadirle ningún *item*) y también se ve el bloque en el que se le añaden dos elementos a la lista alumnos. En este ejemplo se le añade el texto que contienen los cuadros de texto “TextBox1” y “TextBox2”.

## Eventos

Es indispensable que los programas que sean capaces de responder a la interacción del usuario con el interfaz. Es ahí donde toman su importancia los bloques de App Inventor, para ejecutar acciones en respuesta a los eventos que se produzcan en el teléfono o el tablet.

El nombre de los distintos eventos dependen del objeto que los provoca. Un botón avisa de cuando se ha hecho clic sobre él, mientras que el acelerómetro avisa de cuando se ha agitado el móvil, la mensajería de cuando se ha recibido un mensaje, el objeto de Twitter de cuando hemos sido mencionados en esa red social, la cámara de fotos de cuando se ha hecho una foto y así sucesivamente.




## Como usar App Inventor

App Inventor requiere que el ordenador tenga alguno de estos sistemas operativos:

- GNU/Linux: Ubuntu 8+, Debian 5+
- Macintosh (con procesador Intel): Mac OS X 10.5, 10.6
- Windows: Windows XP, Windows Vista, Windows 7

Necesita también de conexión a Internet (los programas y las aplicaciones se cargan siempre desde Internet) y uno de los siguientes navegadores web con estas versiones mínimas:

- Mozilla Firefox 3.6, sin la extensión NoScript instalada
- Apple Safari 5.0
- Google Chrome  4.0
- Microsoft Internet Explorer 7

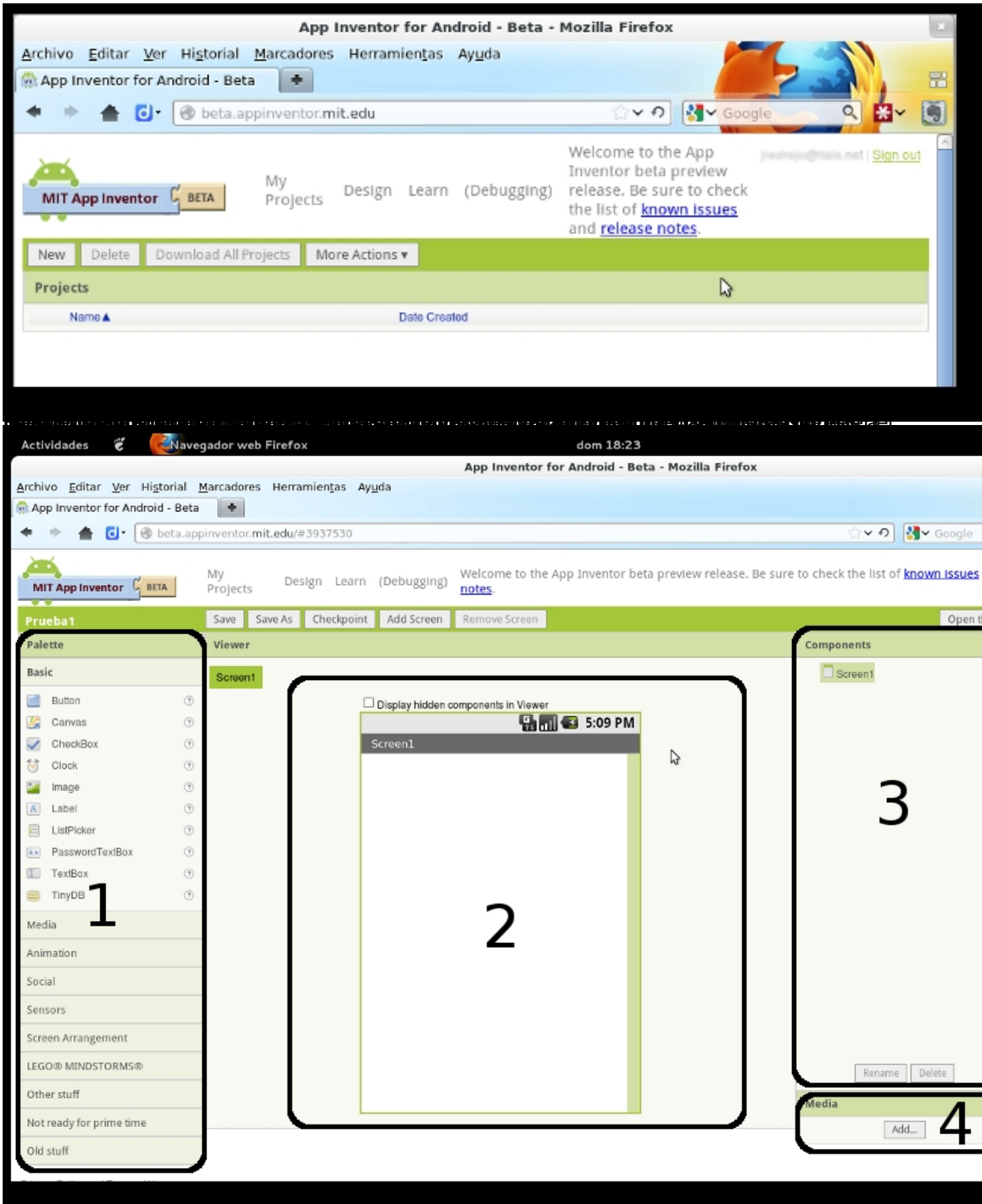
Se necesita también una cuenta de correo electrónico de Gmail y Java Web Start instalado en el ordenador.

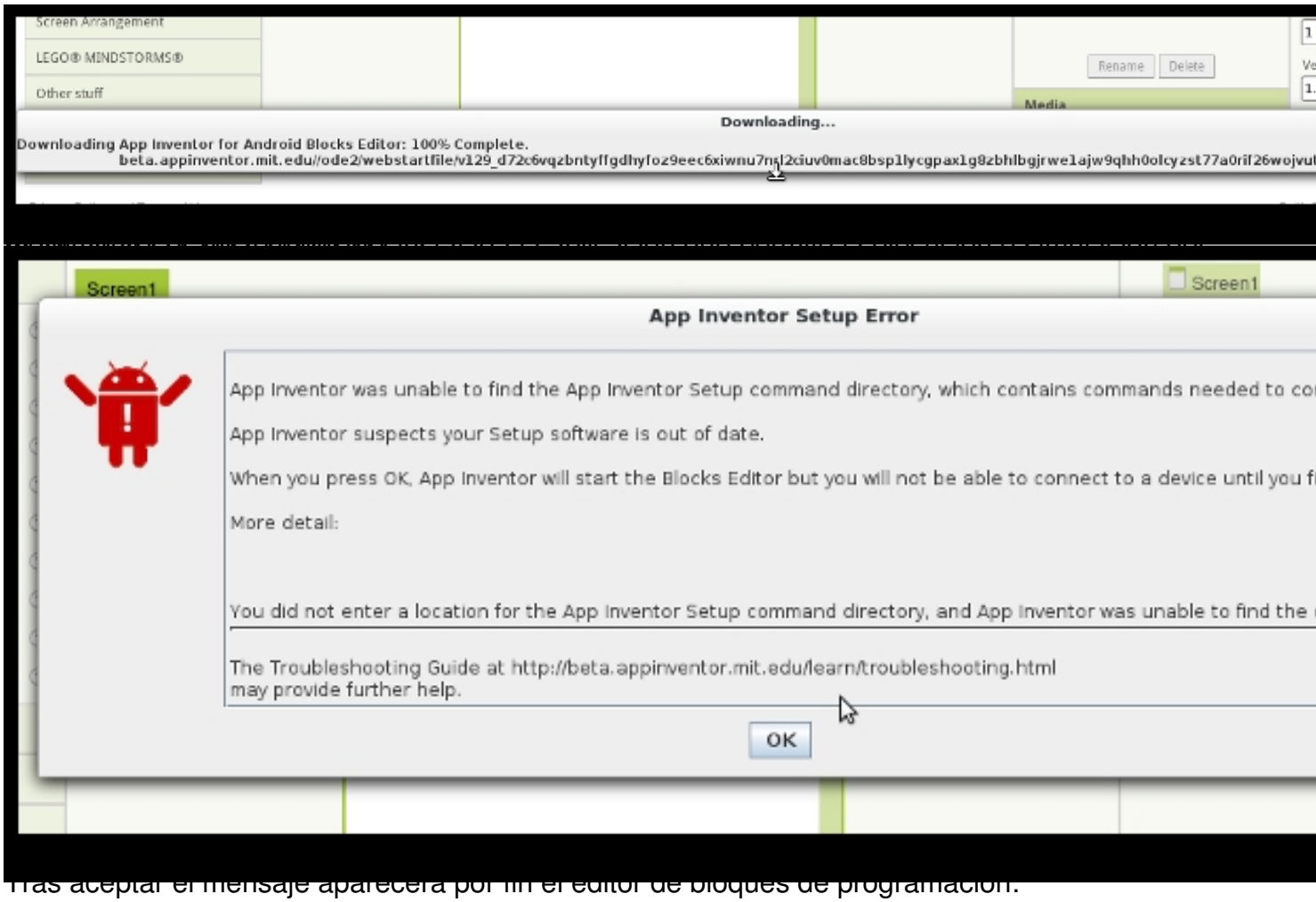
El software para ejecutar los programas se descarga de http://appinventor.mit.edu/explore/content/install-app-inventor-software.html , donde se elegirá una opción u otra dependiendo del sistema operativo que se esté usando. En el caso de Macintosh o Windows se dispone del típico instalador para estos entornos. En sistemas Gnu/Linux Debian o derivados se proporcionan los paquetes deb de instalación [4] o la opción de instalar directamente desde un archivo comprimido tar.gz [5]. Nota importante: En el caso de instalar en un sistema Linux de 64 bits es necesario asegurarse de que están instalados algunos paquetes para que el software funcione. En [6] se puede ver la lista de paquetes necesarios.

Este software permitirá no solo ejecutar los programas en nuestro móvil o en el emulador, sino que dispone además de un intuitivo interfaz de depuración que ayuda a encontrar errores de programación. Para que App Inventor pueda comunicarse desde el ordenador con el móvil usando el cable USB es necesario ir a los ajustes del móvil ->Opciones de desarrollador y activar la opción “Depuración de USB”.

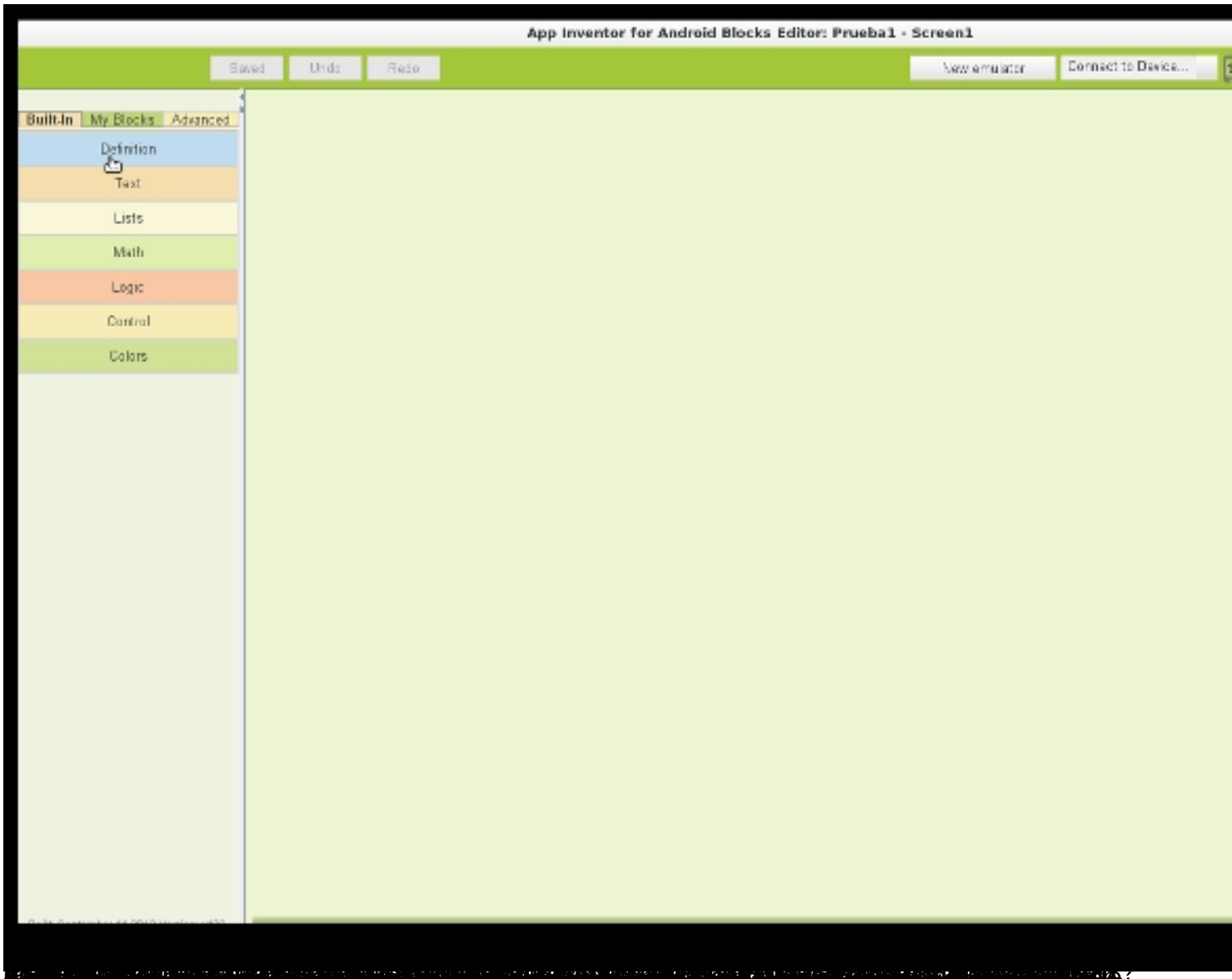
Importante: Es conveniente instalar este software, pero no necesario para hacer los programas. Si no se instala saldrá un mensaje de error cada vez que se carga la aplicación de desarrollo avisando de que no encuentra el entorno de emulación, pero se puede realizar el programa igualmente.

Si el ordenador está conectado a Internet, dispone del sistema operativo y el navegador adecuado y tiene Java instalado ya se puede empezar a programar. En el navegador hay que introducir la dirección http://beta.appinventor.mit.edu/ . Entonces pedirá nuestros datos de acceso al correo de Gmail y, si es la primera vez que se accede, pedirá permiso para usar esa cuenta de correo. Una vez que se le concede el permiso se carga la siguiente pantalla:


App Inventor for Android - Beta - Mozilla Firefox
Archivo Editar Ver Historial Marcadores Herramientas Ayuda
App Inventor for Android - Beta
beta.appinventor.mit.edu
MIT App Inventor BETA
My Projects
Design
Learn
(Debugging)
Welcome to the App Inventor beta preview release. Be sure to check the list of known issues and release notes.
New
Delete
Download All Projects
More Actions
Projects
Name
Date Created
Actividades
Navegador web Firefox
dom 18:23
beta.appinventor.mit.edu/#3937530
Prueba1
Save
Save As
Checkpoint
Add Screen
Remove Screen
Palette
Basic
Button
Canvas
CheckBox
Clock
Image
Label
ListPicker
PasswordTextBox
TextBox
TinyDB
Media
Animation
Social
Sensors
Screen Arrangement
LEGO® MINDSTORMS®
Other stuff
Not ready for prime time
Old stuff
Viewer
Screen1
Display hidden components in Viewer
5:09 PM
Components
Rename
Delete
Add...
1
2
3
4




Tras aceptar el mensaje aparecerá por fin el editor de bloques de programación.




## Programando

La mejor forma de ilustrar la forma de trabajar con App Inventor es mediante un ejemplo hecho paso a paso. En la página oficial hay un enlace bien visible llamado “Teach” donde se puede acceder a decenas de ejemplos y tutoriales de uso. Es casi un clásico empezar con la aplicación “Hello Purr” en la que un gatito maulla al pulsar en la pantalla [7] . Se pueden encontrar por Internet también ejemplos avanzados
[8]
que usan bases de datos y el reconocimiento de voz
[9]
o desarrollos curriculares completos. En concreto, hay varios
[10]

[11]
que están orientados para alumnos del primer curso universitario en facultades de ciencias, y

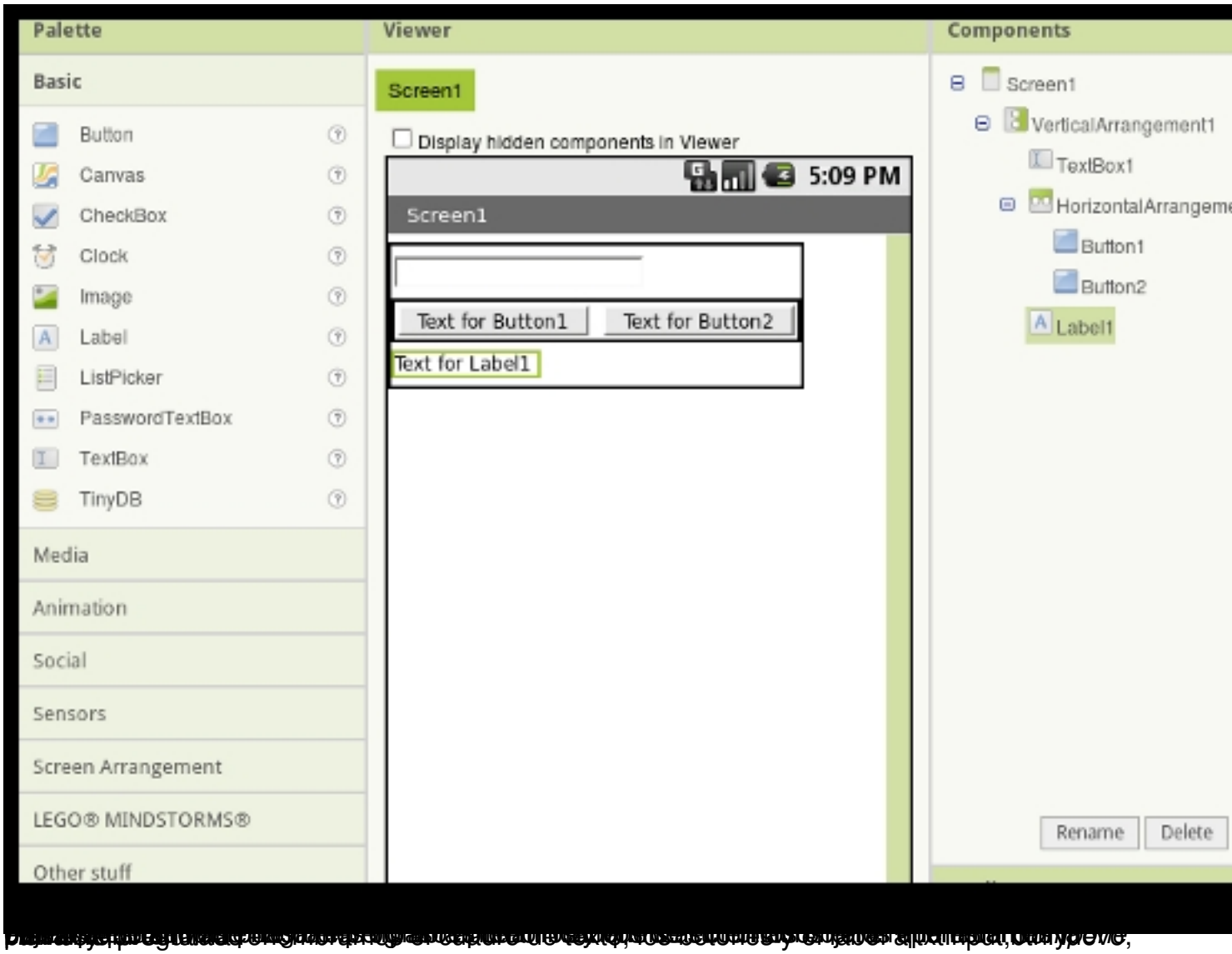
con pequeñas adaptaciones para disminuir la dificultad de los proyectos propuestos pueden ser usados en 1º de Bachillerato en la asignatura de TIC.

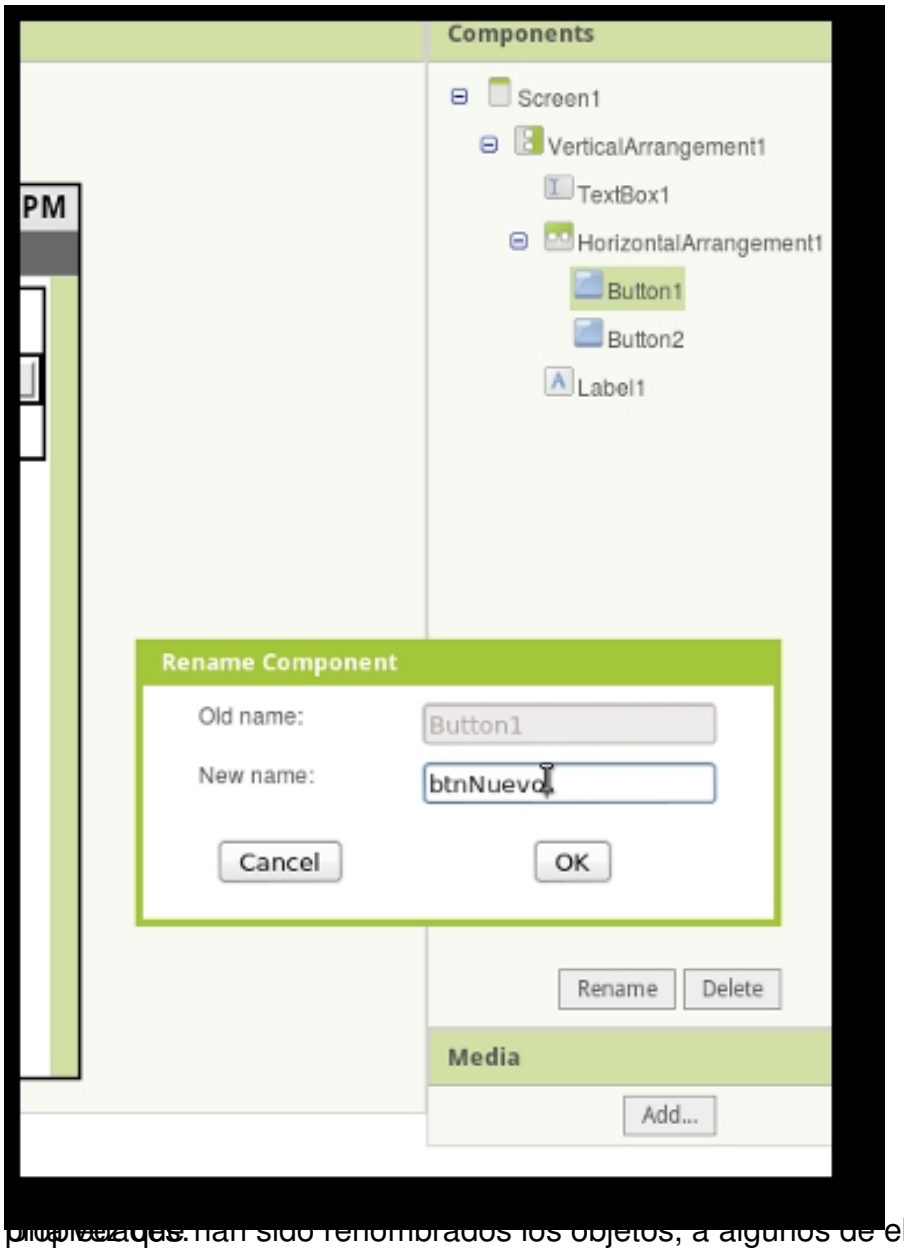
En lugar de seguir el criterio habitual en los tutoriales que muestran cómo hacer aplicaciones visuales para los móviles, a continuación se expondrá cómo hacer una aplicación con un interfaz sencillo en el que prime más la parte de solución de problemas. Para ello , vamos a hacer el ejercicio típico en los cursos de lenguajes de programación de decir cuál el mayor número de entre tres o más.

Primero hay que hacer el diseño en el navegador, usando el diseñador de App Inventor. En la paleta, en la sección “Screen Arrangement” se escoje un “VerticalArrangement” y se suelta sobre la pantalla del Viewer. El componente VerticalArrangement es para forzar una distribución vertical de los componentes que se suelten sobre él. Así se asegura que la disposición de los elementos en la pantalla es en vertical, independientemente del tamaño y los componentes que pongamos.

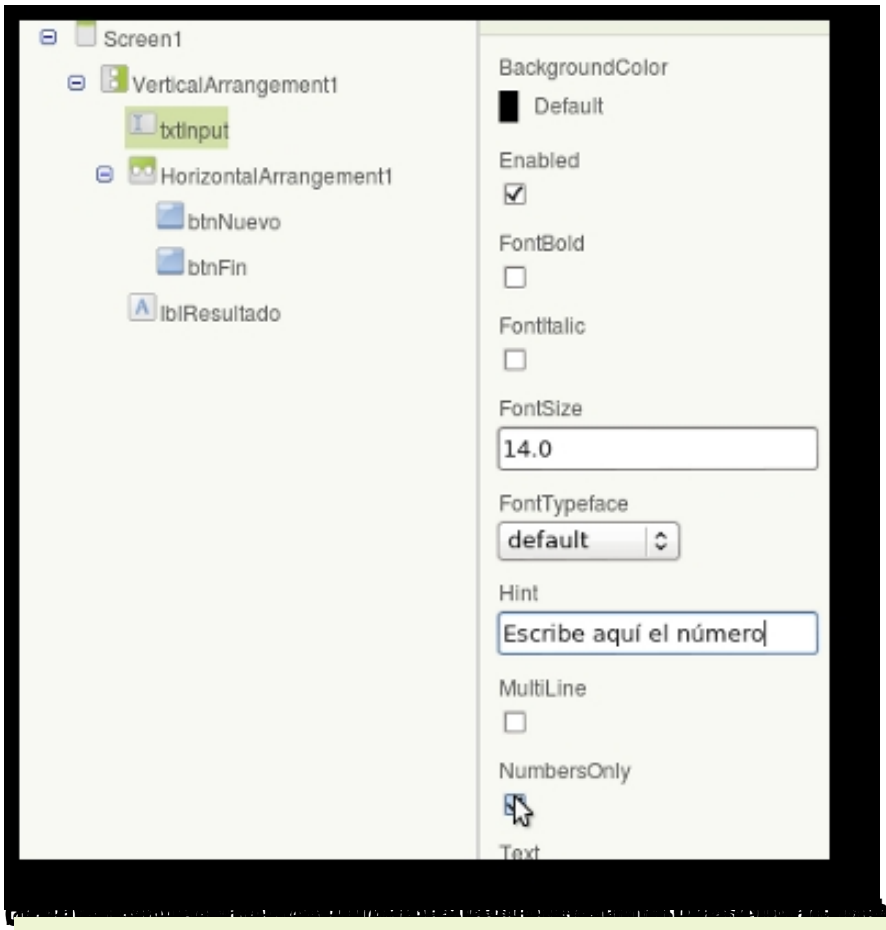
Después se añade, sobre el VerticalArrangement1, un TextBox (de la pestaña Basic), un HorizontalArrangement y un Label. Después se añaden dos Button sobre el HorizontalArrangement con lo que se asegura que los botones van a estar siempre colocados uno al lado del otro, en horizontal.

Debe quedar una pantalla como la siguiente:

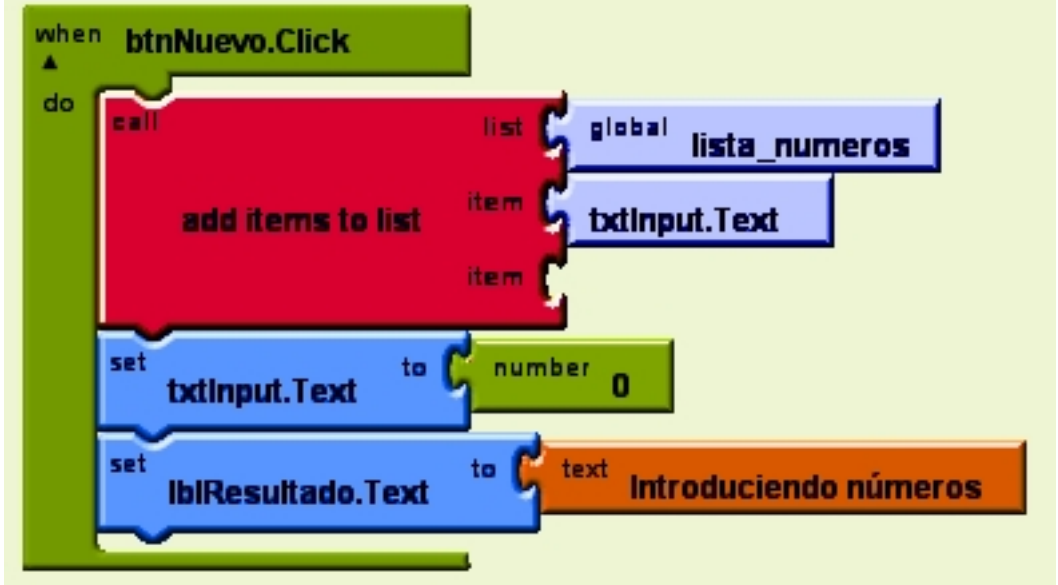

Palette
Basic
Button
Canvas
CheckBox
Clock
Image
Label
ListPicker
PasswordTextBox
TextBox
TinyDB
Media
Animation
Social
Sensors
Screen Arrangement
LEGO® MINDSTORMS®
Other stuff
Viewer
Screen1
Display hidden components in Viewer
5:09 PM
Screen1
Text for Button1
Text for Button2
Text for Label1
Components
Screen1
VerticalArrangement1
TextBox1
HorizontalArrangeme
Button1
Button2
Label1
Rename
Delete



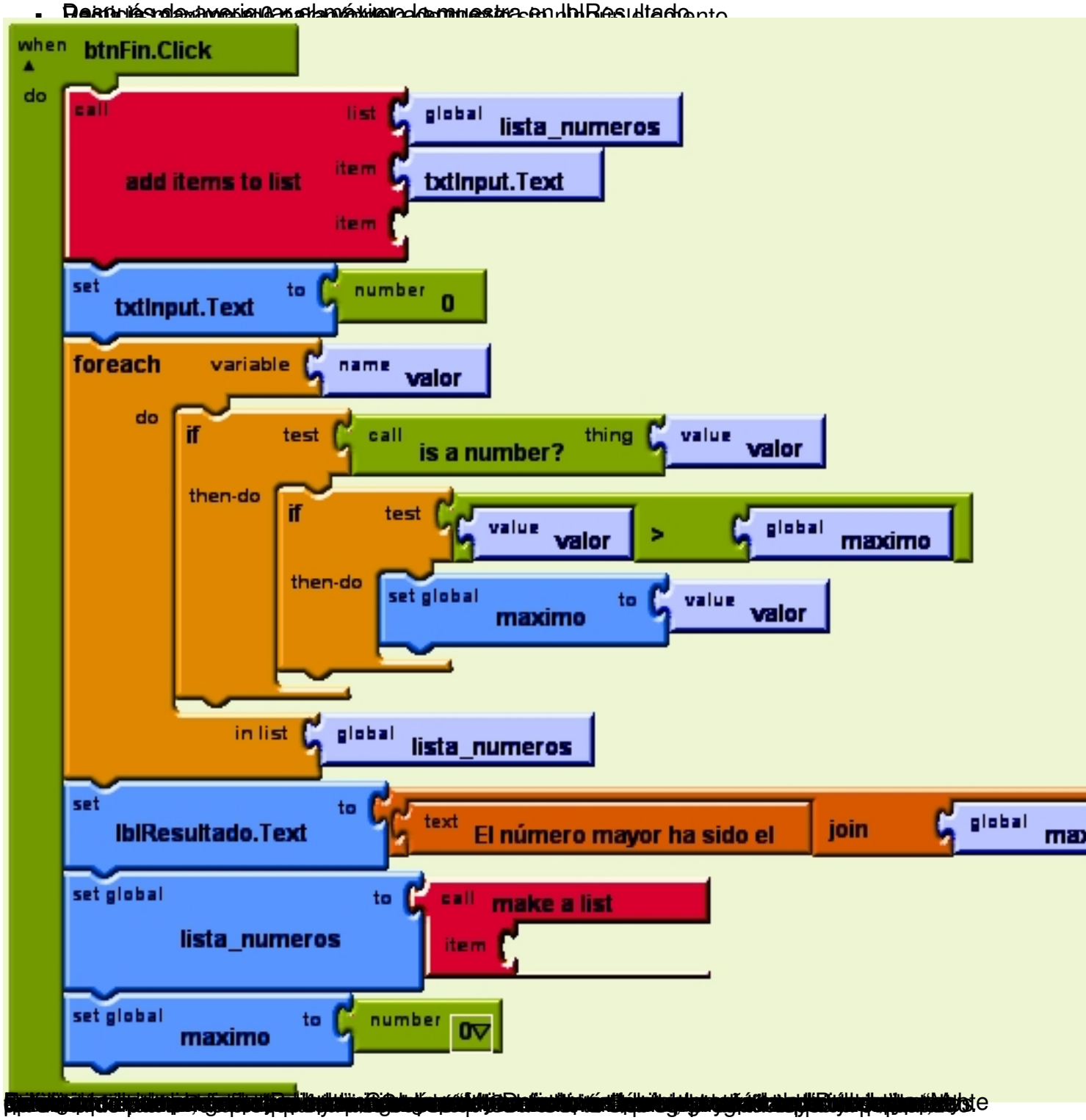

propiedades. han sido renombrados los objetos, a algunos de ellos se le cambian las

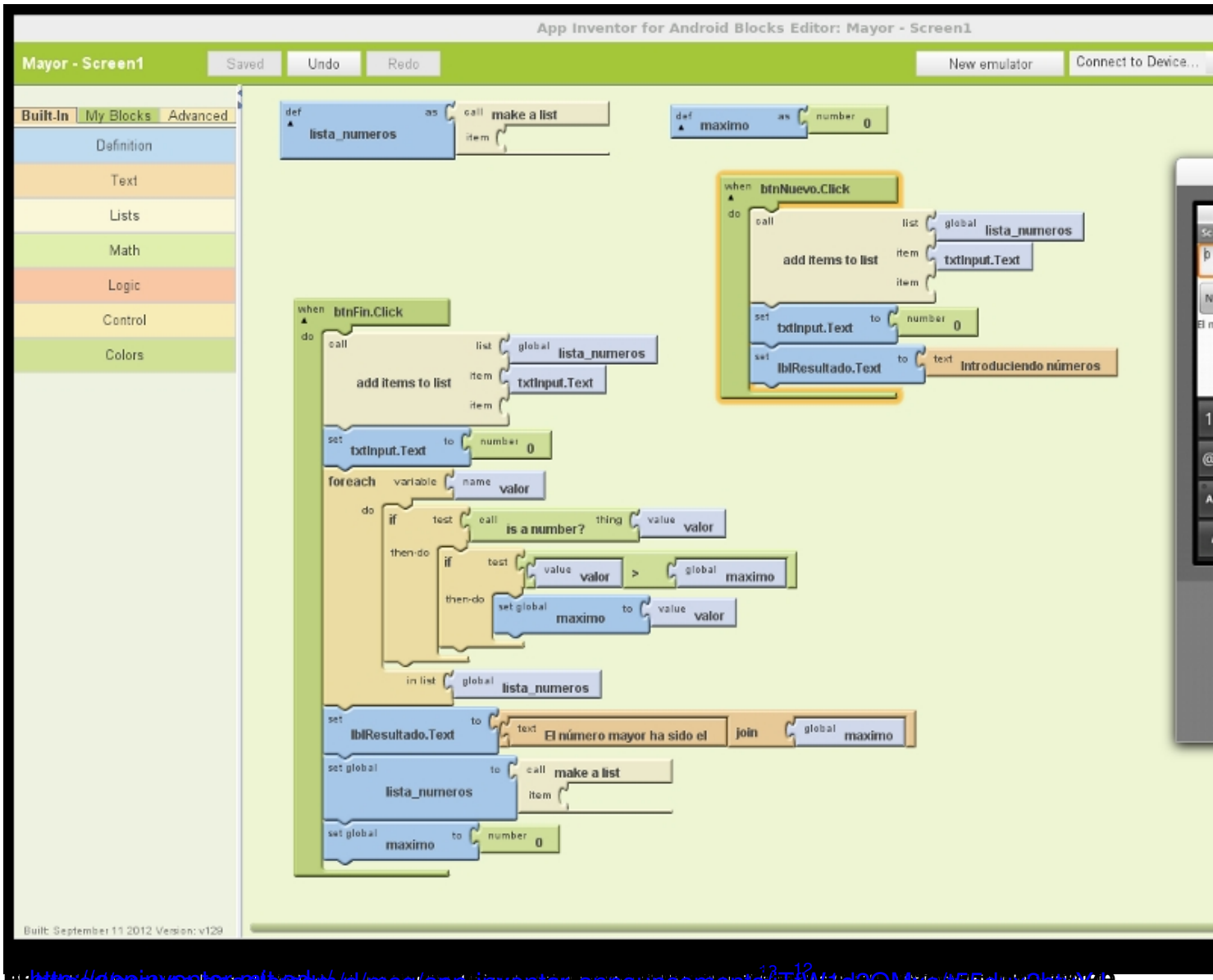

Screen1
VerticalArrangement1
txtInput
HorizontalArrangement1
btnNuevo
btnFin
lblResultado
BackgroundColor
Default
Enabled
FontBold
FontItalic
FontSize
14.0
FontTypeface
default
Hint
Escribe aquí el número
MultiLine
NumbersOnly
Text

def
lista_numeros
as
call
make a list
item
def
maximo
as
number
0


when
btnNuevo.Click
do
call
add items to list
list
global
lista_numeros
item
txtInput.Text
item
set
txtInput.Text
to
number
0
set
lblResultado.Text
to
text
Introduciendo números

- Después de averiguar el máximo lo muestra en lblResultado




App Inventor for Android Blocks Editor: Mayor - Screen1
Mayor - Screen1
Saved
Undo
Redo
New emulator
Connect to Device...
Built-In
My Blocks
Advanced
Definition
Text
Lists
Math
Logic
Control
Colors
def lista_numeros as call make a list item
def maximo as number 0
when btnNuevo.Click
do call add items to list list global lista_numeros item txtInput.Text item
set txtInput.Text to number 0
set lblResultado.Text to text Introduciendo números
when btnFin.Click
do call add items to list list global lista_numeros item txtInput.Text item
set txtInput.Text to number 0
foreach variable name valor
do if test call is a number? thing value valor
then-do if test value valor > global maximo
then-do set global maximo to value valor
in list global lista_numeros
set lblResultado.Text to text El número mayor ha sido el join global maximo
set global lista_numeros to call make a list item
set global maximo to number 0
Built: September 11 2012 Version: v129

[1] [2] [3]

[1] [2] [3]