

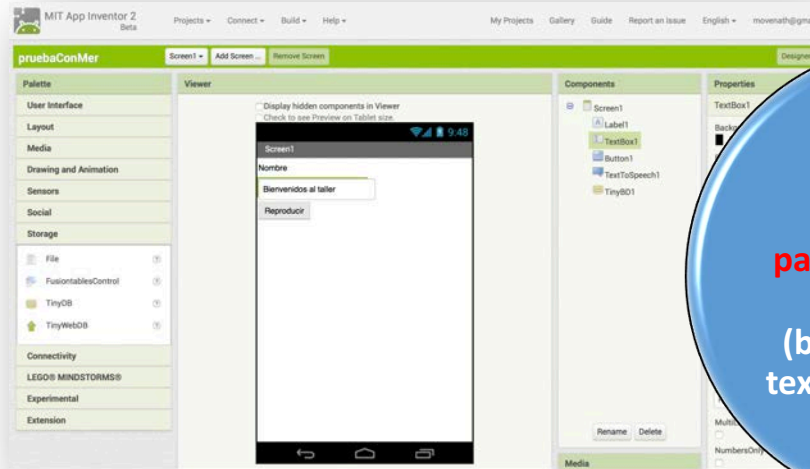
Aplicaciones Android con App Inventor



¿Qué es App Inventor?

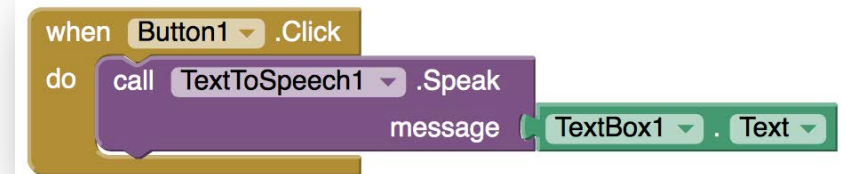
- Una herramienta online desarrollada por el MIT para implementar aplicaciones en el S.O. Android.
- Para utilizarla es preciso tener una cuenta de Google
 - Como cualquier producto de Google, los proyectos están disponibles en cualquier ordenador donde se inicie sesión en Google.
- Es muy visual
 - El entrenamiento necesario para programar aplicaciones completamente funcionales es mínimo.

¿Qué es App Inventor (II)?



1. Diseñar y construir las **pantallas** (interfaz) de la app (botones, cajas de texto, imágenes, etc)

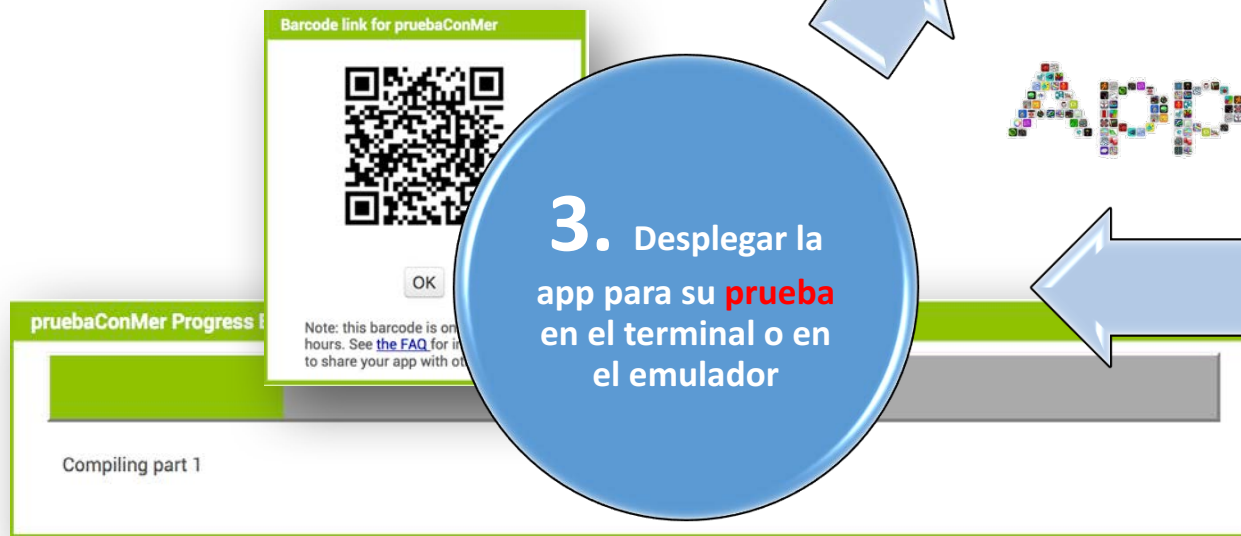
<http://ai2.appinventor.mit.edu/>



2. Definir el **comportamiento** de la app (que hace la app cuando se pulsa un botón, se hace una foto, ...).

App

3. Desplegar la app para su **prueba** en el terminal o en el emulador



Comenzar a trabajar con App Inventor (I)

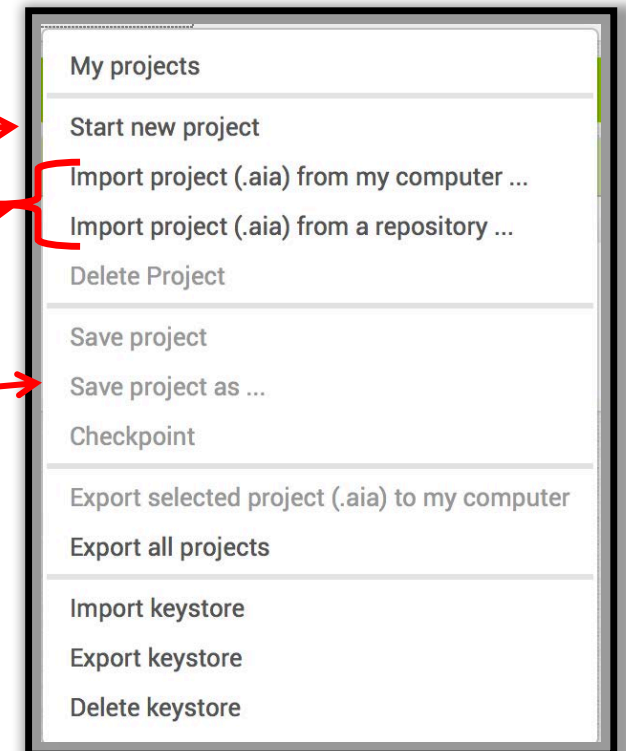
- Se pueden probar las aplicaciones de dos formas
 - Mediante un emulador
 - Directamente en el teléfono



- Para trabajar con el emulador
 - Para la instalación:
<http://appinventor.mit.edu/explore/ai2/setup-emulator>
 - Una vez instalado, ejecutar el programa **aiStarter**

Comenzar a trabajar con App Inventor (II)

- Cada app se desarrolla como un proyecto
 - Podemos **crear** un proyecto nuevo
 - **Cargar/ importar** un proyecto existente
 - En el disco duro local
 - En la nube : Google drive
 - **Salvar** un proyecto
 - En el disco duro local
 - En la nube : Google drive



<http://ai2.appinventor.mit.edu/>



My Projects		
Name	Date Created	Date Modified ▼
<input type="checkbox"/> MiPrimeraApp	Mar 1, 2015 3:48:24 PM	Mar 1, 2015 3:48:24 PM
<input type="checkbox"/> text2speech	Feb 15, 2015 5:09:21 PM	Feb 18, 2015 10:22:44 AM
<input type="checkbox"/> Rebote	Feb 15, 2015 4:30:22 PM	Feb 15, 2015 5:05:48 PM
<input type="checkbox"/> MiPrimeraAPP	Jul 1, 2014 12:36:00 PM	Feb 15, 2015 4:28:53 PM

Pantallas de App Inventor

- Pantalla **Designer**
 - Aquí vamos diseñando el aspecto de nuestra aplicación, añadiendo los **componentes** necesarios (botones, etiquetas, campos de texto, sensores...)
 - Algunos elementos no aparecen en la pantalla, como los sensores
- Pantalla **Blocks**
 - Aquí controlamos la lógica de los distintos componentes que se han añadido, mediante **bloques** lógicos que funcionan como piezas de un puzzle

Pantalla Designer (I)

The screenshot displays the MIT App Inventor 2 Beta interface. At the top, the navigation bar includes the MIT App Inventor logo, the project name 'Proyecto', and various menu options: Projects, Connect, Build, Help, My Projects, Gallery, Guide, Report an Issue, English, and Mercedes.Amor.Pinilla@gmail.com. Below the navigation bar, the main workspace is divided into four panels: Palette, Viewer, Components, and Properties.

- Palette:** Contains categories for User Interface, Layout, Media, Drawing and Animation, and Sensors. The Sensors category is expanded, showing a list of sensors: AccelerometerSensor, BarcodeScanner, Clock, GyroscopeSensor, LocationSensor, NearField, OrientationSensor, Pedometer, and ProximitySensor.
- Viewer:** Shows a preview of the screen. It includes a header bar with the text 'Screen1' and a status bar at the top right displaying the time '9:48' and various icons. Below the header is a large white area representing the screen content. At the bottom, there is a navigation bar with three icons: a back arrow, a home button, and a recent apps button. Above the viewer, there are checkboxes for 'Display hidden components in Viewer' and 'Check to see Preview on Tablet size'.
- Components:** Shows a list of components for the screen, currently containing 'Screen1'. Below the list are 'Rename' and 'Delete' buttons. At the bottom of this panel is an 'Upload File ...' button.
- Properties:** Shows the properties for the selected component 'Screen1'. The properties include: AboutScreen (a text input field), AlignHorizontal (Left: 1), AlignVertical (Top: 1), AppName (Proyecto), BackgroundColor (White), BackgroundImage (None...), CloseScreenAnimation (Default), Icon (None...), OpenScreenAnimation (Default), ScreenOrientation (Unspecified), Scrollable (checkbox), ShowListsAsJson (checkbox), ShowStatusBar (checkbox, checked), Sizing (Fixed), and Title (Screen1).

Pantalla Designer (II)

The screenshot shows the MIT App Inventor 2 Beta interface. The top navigation bar includes 'Projects', 'Connect', 'Build', 'Help', 'My Projects', 'Gallery', 'Guide', 'Report an Issue', 'English', and a user profile 'Merrill Pinilla@gmail.com'. Below this is a green header for the project 'Proyecto', with buttons for 'Screen1', 'Add Screen...', and 'Remove Screen'. On the right side of the header are 'Designer' and 'Blocks' buttons. A red arrow points to the user profile in the top right.

The main workspace is divided into four panels:

- Palette (Red border):** Contains categories like 'User Interface', 'Layout', 'Media', 'Drawing and Animation', and 'Sensors'. The 'Sensors' section is expanded, listing: AccelerometerSensor, BarcodeScanner, Clock, GyroscopeSensor, LocationSensor, NearField, OrientationSensor, Pedometer, and ProximitySensor. Other categories include Social, Storage, Connectivity, LEGO® MINDSTORMS®, Experimental, and Extension.
- Viewer (Blue border):** Shows a mobile device preview for 'Screen1'. It includes checkboxes for 'Display hidden components in Viewer' and 'Check to see Preview on Tablet size.' The preview shows a status bar with signal, Wi-Fi, and battery icons, and a time of 9:48. The bottom navigation bar has back, home, and recent apps icons.
- Components (Yellow border):** Shows a list of components for 'Screen1', currently containing only 'Screen1'. Below the list are 'Rename' and 'Delete' buttons. A 'Media' section at the bottom has an 'Upload File ...' button.
- Properties (Purple border):** Shows the configuration for 'Screen1'. Properties include: AboutScreen (text input), AlignHorizontal (Left: 1), AlignVertical (Top: 1), AppName (Proyecto), BackgroundColor (White), BackgroundImage (None...), CloseScreenAnimation (Default), Icon (None...), OpenScreenAnimation (Default), ScreenOrientation (Unspecified), Scrollable (checkbox), ShowListsAsJson (checkbox), ShowStatusBar (checked), Sizing (Fixed), and Title (Screen1).

Pantalla Designer (III)

- En la pestaña **Palette** se encuentran los componentes que podemos añadir a nuestra aplicación.
- Los componentes se arrastran a la pantalla activa (**Viewer**)
 - Por defecto hay una única pantalla (Screen1)
 - Se pueden añadir más pantallas
 - Por ejemplo, una pantalla para la bienvenida a la aplicación, otra para la configuración, otra para la pantalla principal...
 - Podemos seleccionar la pantalla activa

Pantalla Designer (IV)

- En la pestaña **Components** se listan todos los componentes que se han añadido a cada pantalla.
- En la pestaña **Properties** se pueden configurar las propiedades del componente seleccionado.
- En la pestaña **Media** podemos subir archivos necesarios para nuestra aplicación (imágenes, vídeo, audio).

Pantalla Designer (V)

- Tipos de componentes:
 - **Interfaz**: botones, etiquetas, campos de texto...
 - **Layout**: para organizar los componentes de interfaz en horizontal, vertical o en tablas.
 - **Media**: video, sonido...
 - **Drawing and animation**: para insertar componentes animados
 - **Sensors**: acelerómetro, localización GPS, orientación...

Pantalla Designer (VI)

- Tipos de componentes:
 - **Social**: acceso a llamadas de teléfono, mensajes de texto, e-mail, twitter...
 - **Storage**: para almacenar datos de manera persistente y que estén disponibles entre distintas ejecuciones de la aplicación
 - **Connectivity**: para conectar con otras aplicaciones, bluetooth...
 - **Lego Mindstorm**: para robots de Lego
 - **Experimental**
 - **Extension**

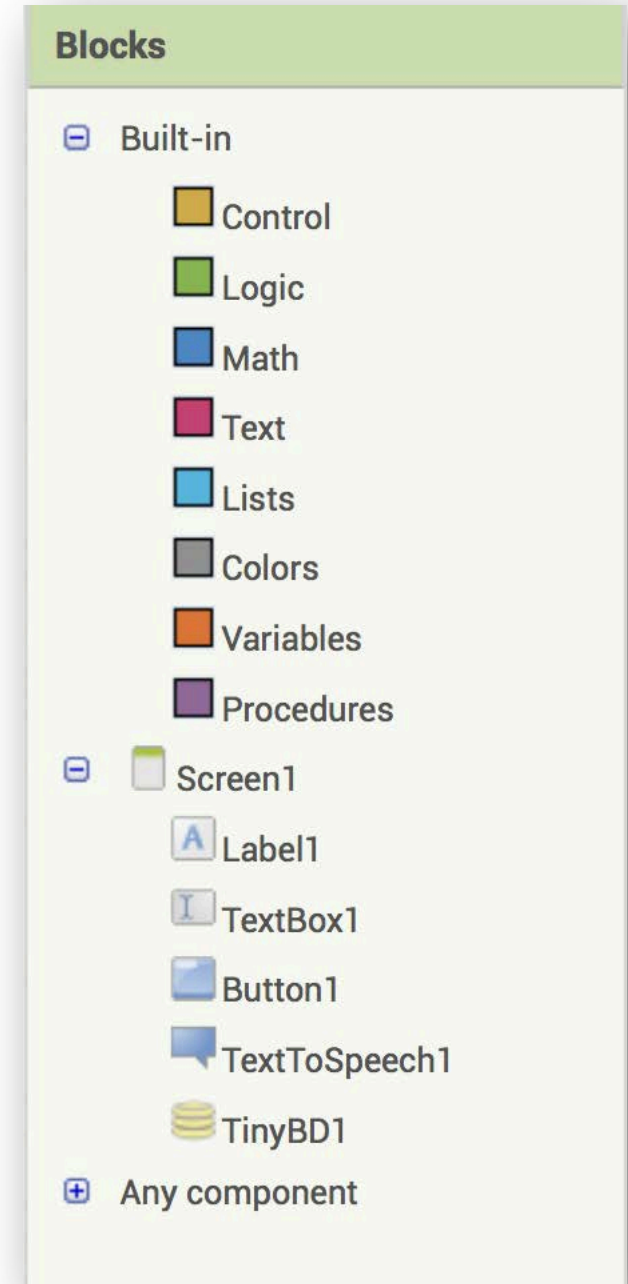
Pantalla Blocks (I)

The image shows the MIT App Inventor 2 web interface. At the top left, the MIT App Inventor 2 logo and 'Beta' version are displayed. The top navigation bar includes 'Project', 'Connect', 'Build', and 'Help' menus. On the right side of the top bar, there are links for 'My Projects', 'Guide', 'Report an Issue', and a user profile 'manuela.ruiz.montiel'. A red arrow points to the 'Blocks' tab in the top right corner of the interface.

The main workspace is divided into two sections: 'Blocks' on the left and 'Viewer' on the right. The 'Blocks' section contains a 'Built-in' category with sub-categories: Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Below these are 'Screen1' and 'Any component'. At the bottom of the 'Blocks' section are 'Rename' and 'Delete' buttons. The 'Viewer' section is currently empty. At the bottom of the 'Viewer' section, there are warning indicators (a yellow triangle with '0' and a red triangle with '0') and a 'Show Warnings' button. A trash can icon is located in the bottom right corner of the 'Viewer' section.

Pantalla Blocks (II)

- Bloques Built-in. Los más importantes:
 - **Control** (if, bucles...)
 - **Logic** (true, false, and/or...)
 - **Math** (números, comparaciones, operaciones...)
 - **Texto** (concatenar, longitud, reemplazar...)
 - **Variables** (globales y locales)
 - **Procedures** (llamadas a métodos)



Pantalla Blocks (III)

- Bloques asociados a nuestros componentes
 - Para cada componente, nos aparecerá una paleta con los posibles bloques asociados.

- Por ejemplo, para las etiquetas tenemos únicamente **getters** y **setters**



- Para los botones, además de getters y setters tenemos bloques de control que **controlan eventos** (un posible evento es, por ejemplo, que el botón sea pulsado).
 - Otros componentes más sofisticados (sensores, bases de datos, actividades) incluyen también **llamadas a procedimientos**. Por ejemplo, en el caso de una base de datos, podemos llamar al procedimiento para almacenar un valor.

Pantalla Blocks (III)

- Bloques asociados a nuestros componentes
 - Para los botones, además de getters y setters tenemos bloques de control que **controlan eventos** (un posible evento es, por ejemplo, que el botón sea pulsado)



- Otros componentes más sofisticados (sensores, bases de datos, actividades) incluyen también **llamadas a procedimientos**. Por ejemplo, en el caso de una base de datos, podemos llamar al procedimiento para almacenar un valor



Primera App

Text2Speech

Bienvenidos al taller



Text2Speech

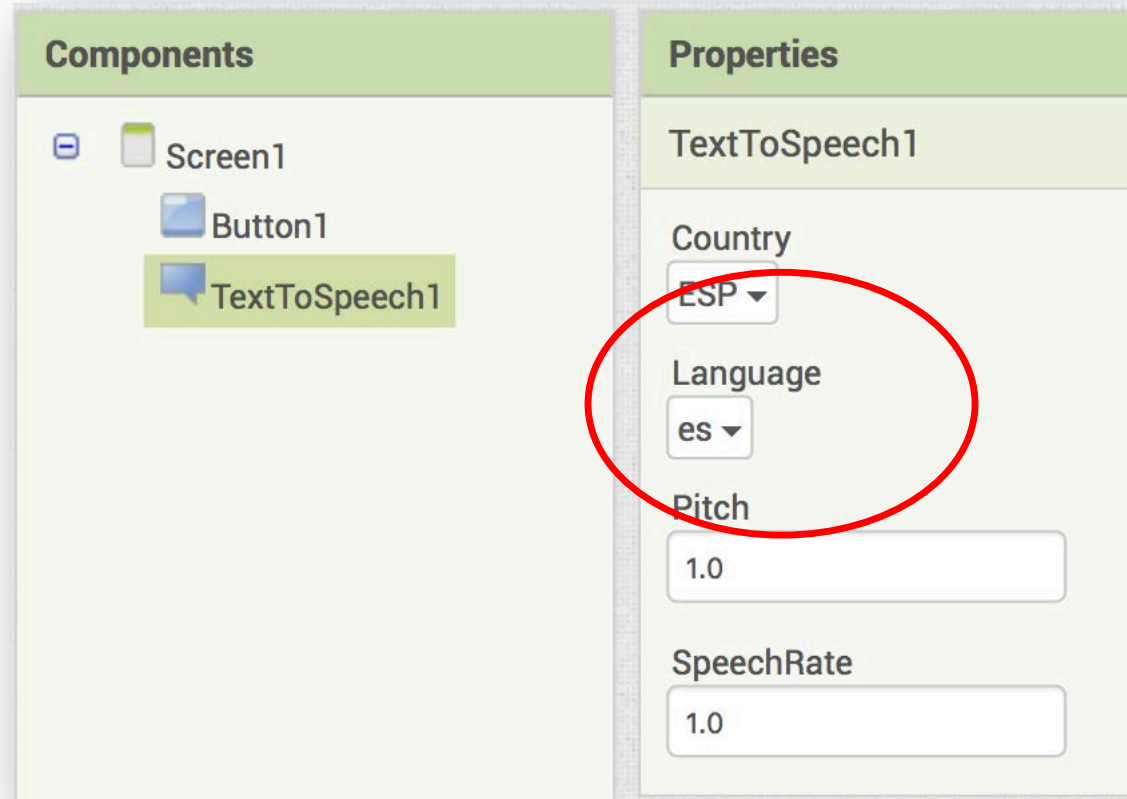
- Modelamos la interfaz añadiendo componentes a la pantalla.

The screenshot displays a mobile application development environment with three main panels: Viewer, Components, and Properties.

- Viewer:** Shows a mobile screen titled "Screen1" with a button labeled "Press here!!". The screen includes a status bar at the top with signal, battery, and time (9:48) indicators, and an Android-style navigation bar at the bottom. A "Non-visible components" section at the bottom of the viewer shows a "TextToSpeech1" component, which is circled in red.
- Components:** A tree view on the right shows the hierarchy: "Screen1" containing "Button1" and "TextToSpeech1". Below this panel are "Rename" and "Delete" buttons, and a "Media" section with an "Upload File ..." button.
- Properties:** A detailed property panel for "Button1" is shown on the far right. It includes various settings such as "BackgroundColor" (Default), "Enabled" (checked), "FontSize" (14.0), "FontTypeface" (default), "Height" (Automatic...), "Width" (Automatic...), "Image" (None...), "Shape" (default), and "ShowFeedback" (checked). The "Text" property is set to "Press here!!" and is circled in red.

Text2Speech

- Modelamos la interfaz añadiendo componentes a la pantalla.



Text2Speech

- Modelamos el comportamiento

The screenshot displays the MIT App Inventor 2 Beta web interface. The browser address bar shows `ai2.appinventor.mit.edu`. The top navigation bar includes the MIT App Inventor logo, the text "MIT App Inventor 2 Beta", and several menu items: "Projects", "Connect", "Build", "Help", "My Projects", "Gallery", "Guide", "Report an Issue", "English", and a user email "movenath@gmail.". Below this is a green header bar for the project "Text2Speech", which contains a "Screen1" dropdown, "Add Screen ..." and "Remove Screen" buttons, and a "Designer" button.

The main workspace is divided into two panels: "Blocks" on the left and "Viewer" on the right. The "Blocks" panel shows a "Built-in" category with sub-categories: Control, Logic, Math, Text (highlighted), Lists, Colors, Variables, and Procedures. The "Viewer" panel shows a visual programming block: a "when Button1 .Click" block containing a "do call TextToSpeech1 .Speak message" block with the text "Taller de AppInventor". At the bottom of the viewer, there are warning indicators (0 yellow and 0 red triangles) and a "Show Warnings" button. On the right side of the viewer, there are icons for a backpack and a trash can.

Text2Speech

- Y la probamos...



The screenshot shows the MIT App Inventor 2 Beta interface. At the top, there are navigation buttons: 'Projects', 'Connect', 'Build', and 'Help'. The 'Connect' button is highlighted with a blue border. A red box surrounds the 'Connect' button and its dropdown menu, which includes the following options: 'AI Companion', 'Emulator', 'USB', 'Reset Connection', and 'Hard Reset'. Two red arrows point from the 'Emulator' option in the menu to the 'Emulator' button in the top navigation bar and to a virtual Android emulator window at the bottom right. The emulator window shows a screen with the text 'Press here!'. In the background, a code block is visible with the text 'when do call TextToSpeech1.Speak message'. On the left side, there is a 'Speech' section and a 'Viewer' section. At the bottom left, there is a 'It-in' section with a list of categories: Control, Logic, Math, Text, Lists, Colors, and Variables.

Text2Speech

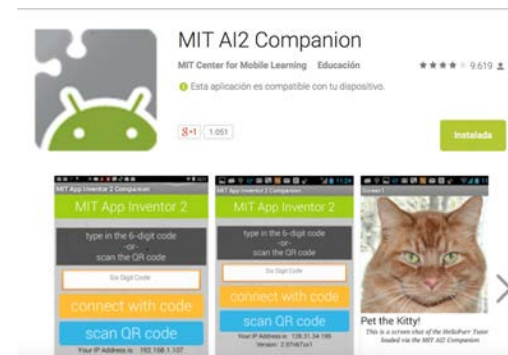
- Y la probamos...
 - ✓ En el emulador
 - ✓ En un móvil
 - ✓ Instalar primer la app AI2 companion
 - ✓ <https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3&hl=es>
 - ✓ <http://appinventor.mit.edu/explore/ai2/setup-device-wifi.html>



Build your project on
your computer

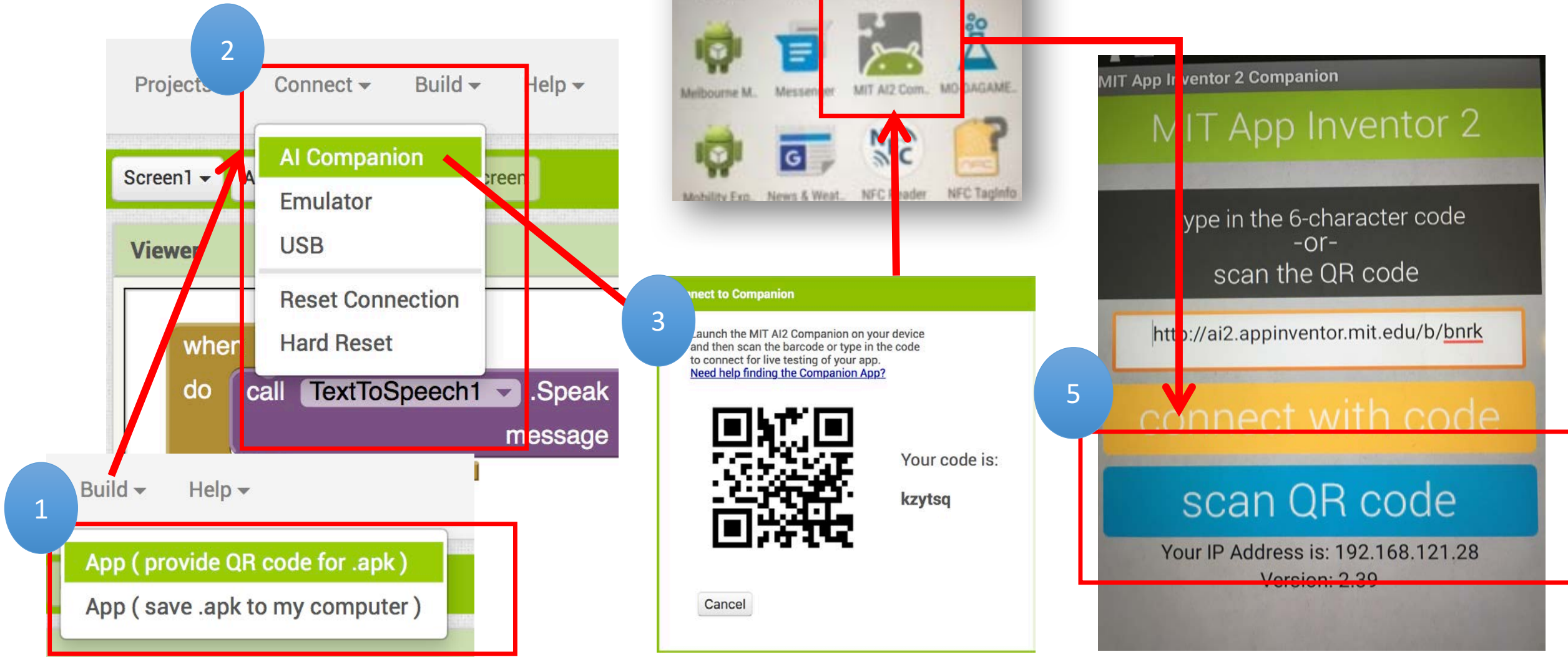


Test it in real-time on
your device



Text2Speech

- En el móvil...



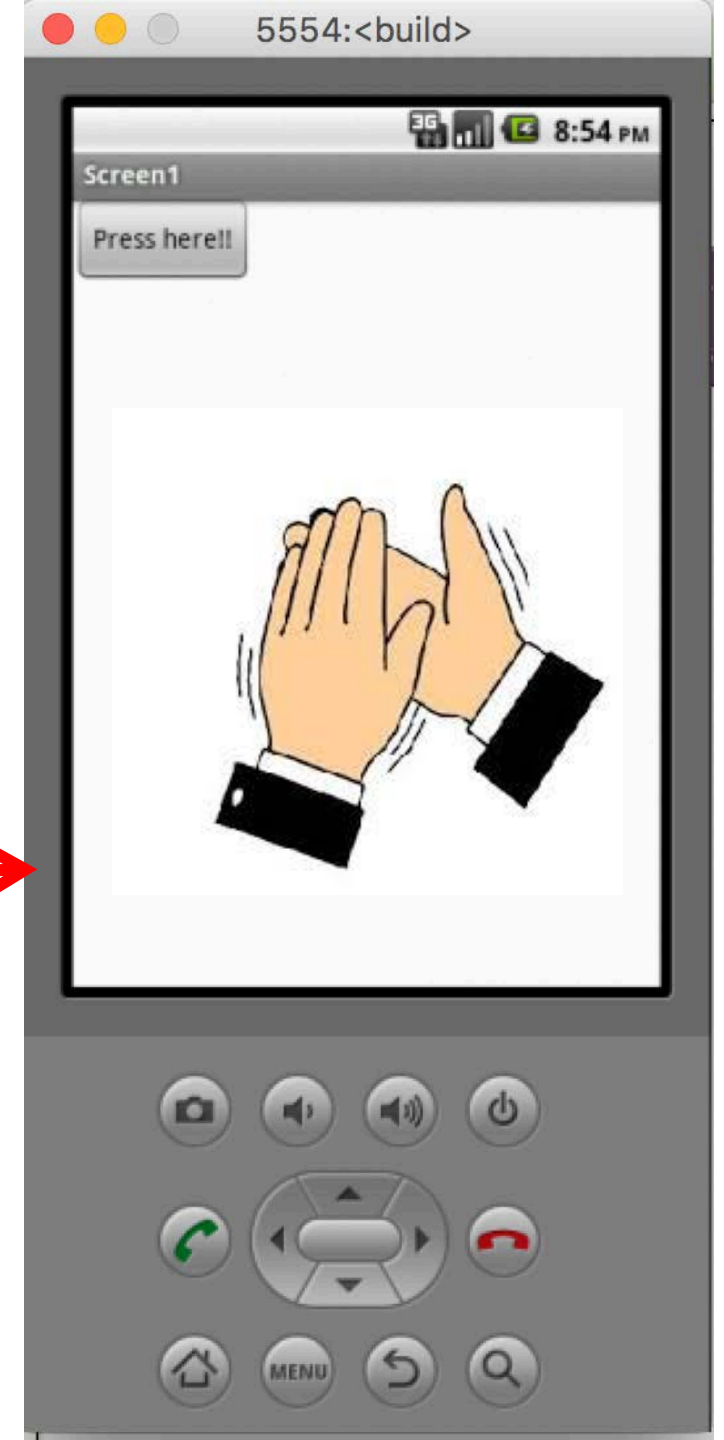
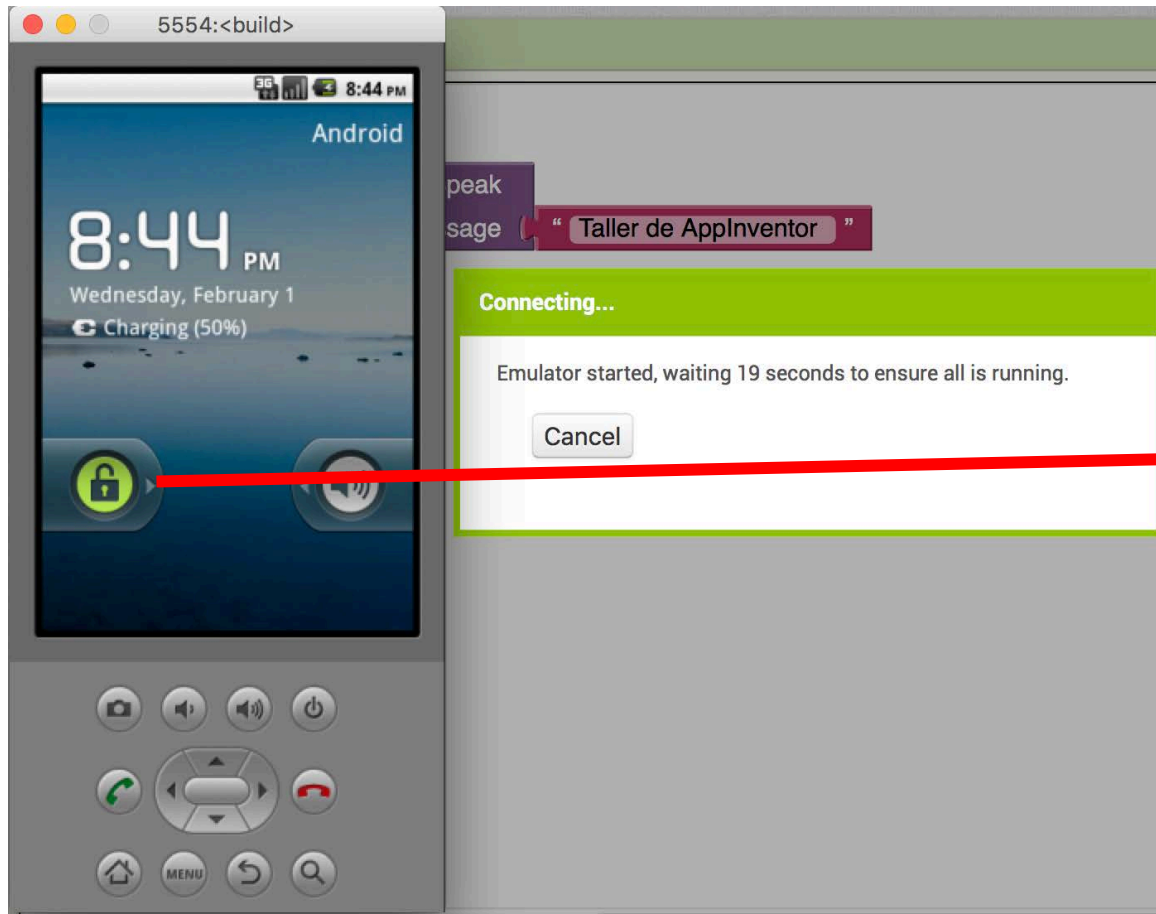
Text2Speech

- Con el emulador...

The image shows a screenshot of an Android emulator interface. On the left, a context menu is open, listing options: AI Companion, Emulator (highlighted in green), USB, Reset Connection, and Hard Reset. A blue circle with the number '1' is next to the menu. A red arrow points from the 'Emulator' option to the emulator window. The emulator window displays the time 8:44 PM, the date Wednesday, February 1, and a charging status of 50%. Below the screen is a virtual keypad. On the right, a dialog box titled 'Connecting...' is visible, with the text 'Emulator started, waiting 19 seconds to ensure all is running.' and a 'Cancel' button.

Text2Speech

- Y la probamos...



Primera App: **Modificación 1**

Text2Speech

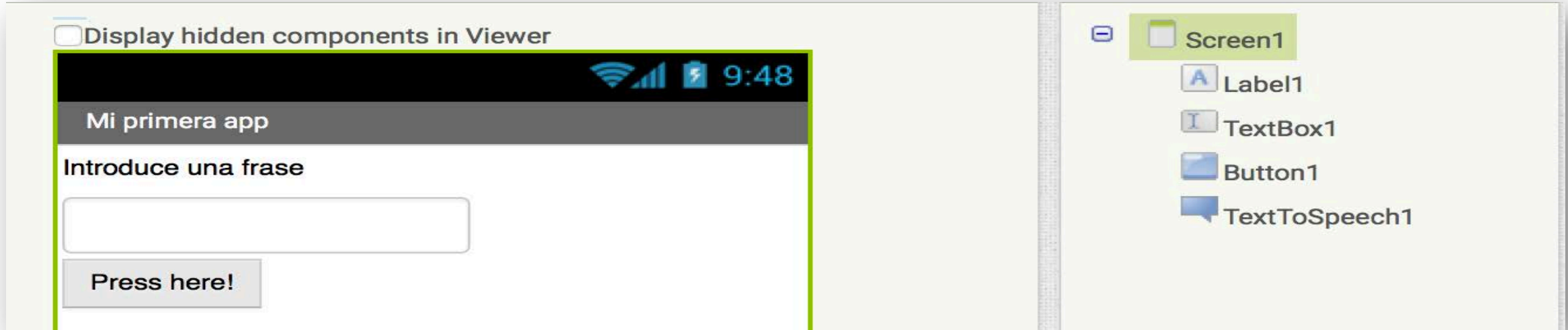
Bienvenidos al taller



Text2Speech:

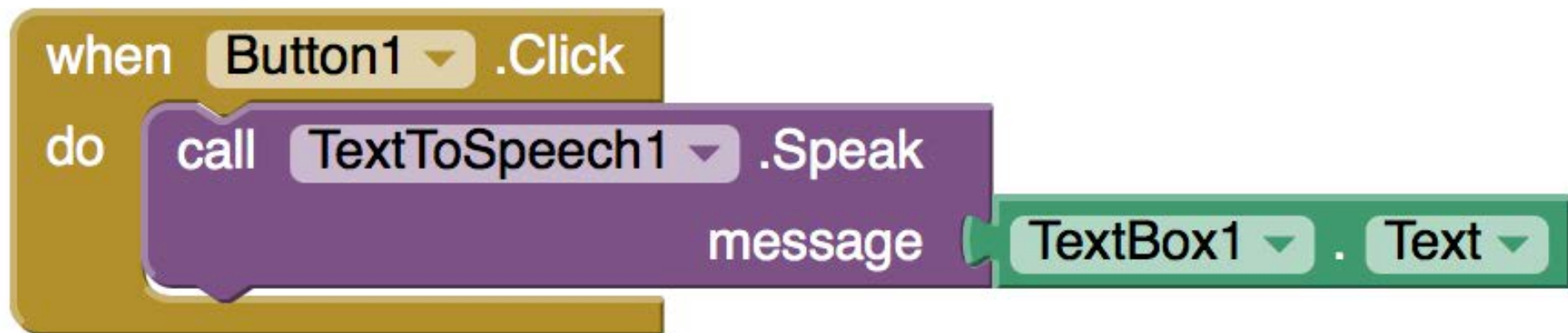
Introducir un mensaje para que la app lo reproduzca como un audio

- Tomamos como referencia nuestra primera app
- Añadimos un componente **Label** y un componente **TextBox** en la pantalla de diseño

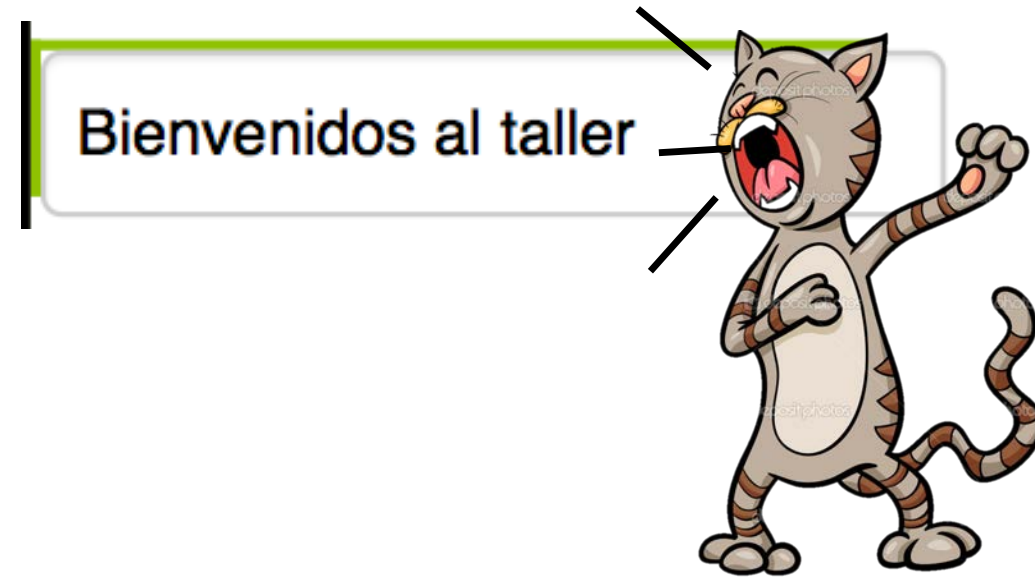


Text2Speech: Introducir un mensaje para que la app lo reproduzca como un audio

- Modificamos las propiedades para personalizar los componentes.
- Añadimos el siguiente **comportamiento**



Text2Speech: Introducir un mensaje para que la app lo reproduzca como un audio



Primera App: **Modificación 2**

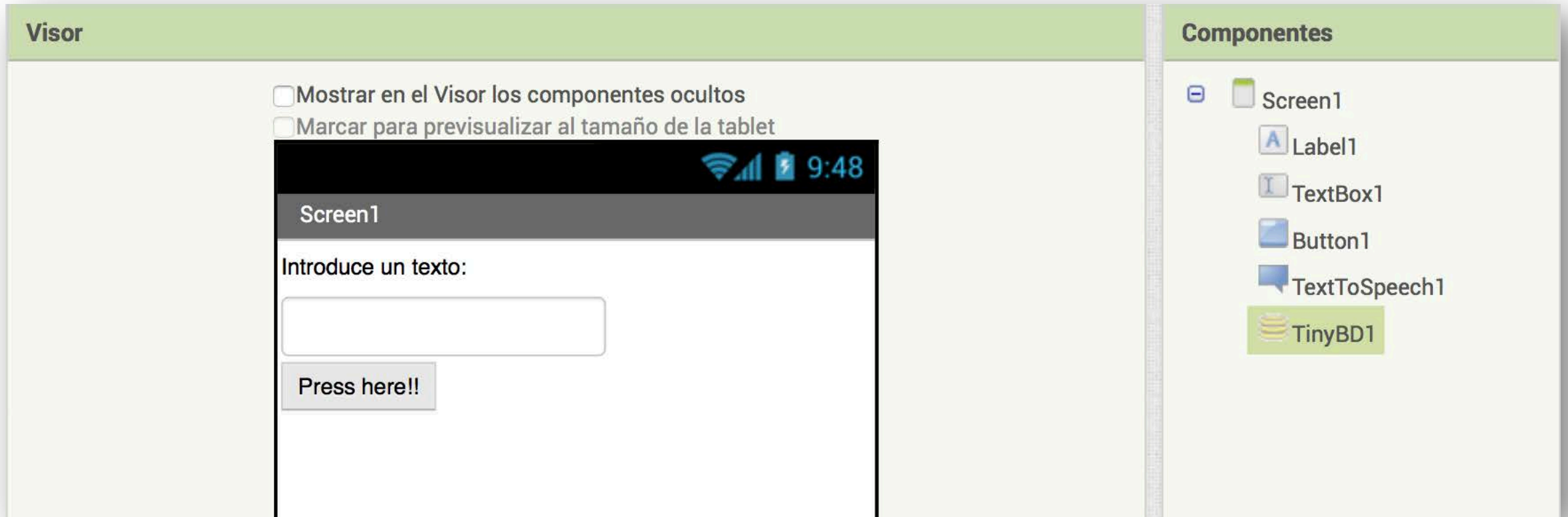
Text2Speech

Bienvenidos al taller



Text2Speech: Que guarde el último mensaje escrito.

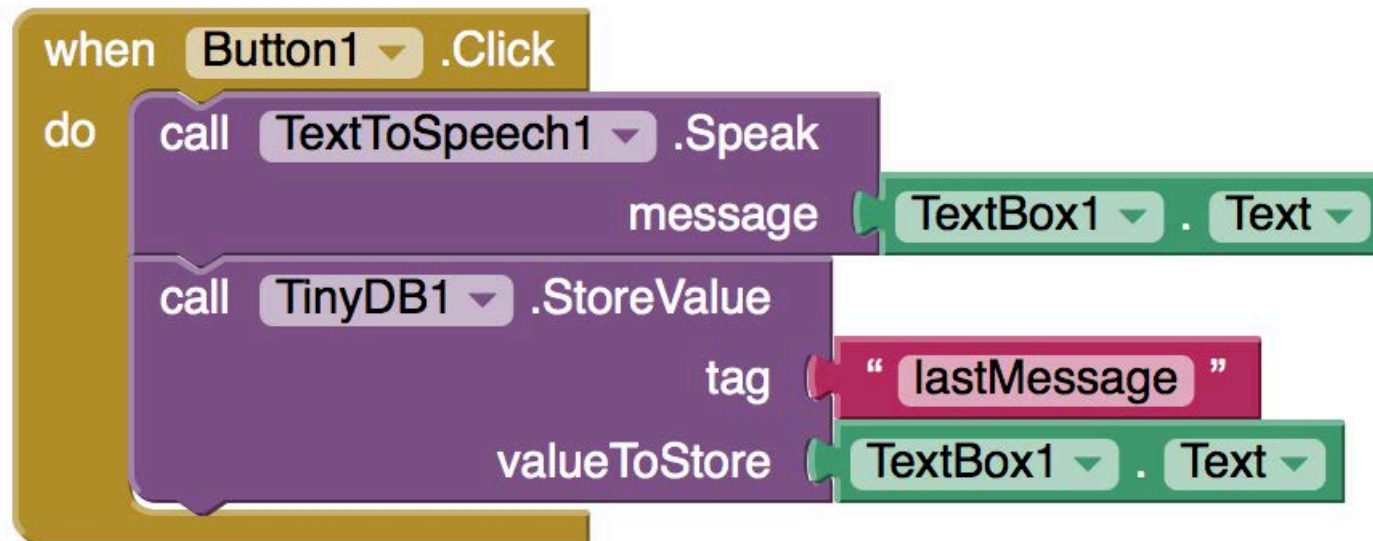
- Usamos el componente del menú **Storage TinyDB**
- Añadimos un nuevo componente **TinyDB** (es no visible)



The screenshot displays the Android Studio development environment. On the left, the 'Visor' (Preview) pane shows a mobile application interface. At the top, there are two unchecked checkboxes: 'Mostrar en el Visor los componentes ocultos' and 'Marcar para previsualizar al tamaño de la tablet'. Below these, a simulated mobile screen is shown with a status bar at the top displaying signal strength, Wi-Fi, battery, and the time 9:48. The screen content includes a title bar 'Screen1', a label 'Introduce un texto:', a text input field, and a button labeled 'Press here!!'. On the right, the 'Componentes' (Components) pane lists the available UI elements for the app: 'Screen1' (the root container), 'Label1', 'TextBox1', 'Button1', 'TextToSpeech1', and 'TinyBD1'. The 'TinyBD1' component is highlighted with a green background, indicating it is the selected component.

Text2Speech: Que guarde el último mensaje escrito.

- Añadimos el comportamiento...



Primera App: **Modificación 3**

Text2Speech

Bienvenidos al taller



Text2Speech: ...que al iniciar la aplicación aparezca, en el cuadro de texto, la última frase escrita.

- Añadimos el comportamiento...

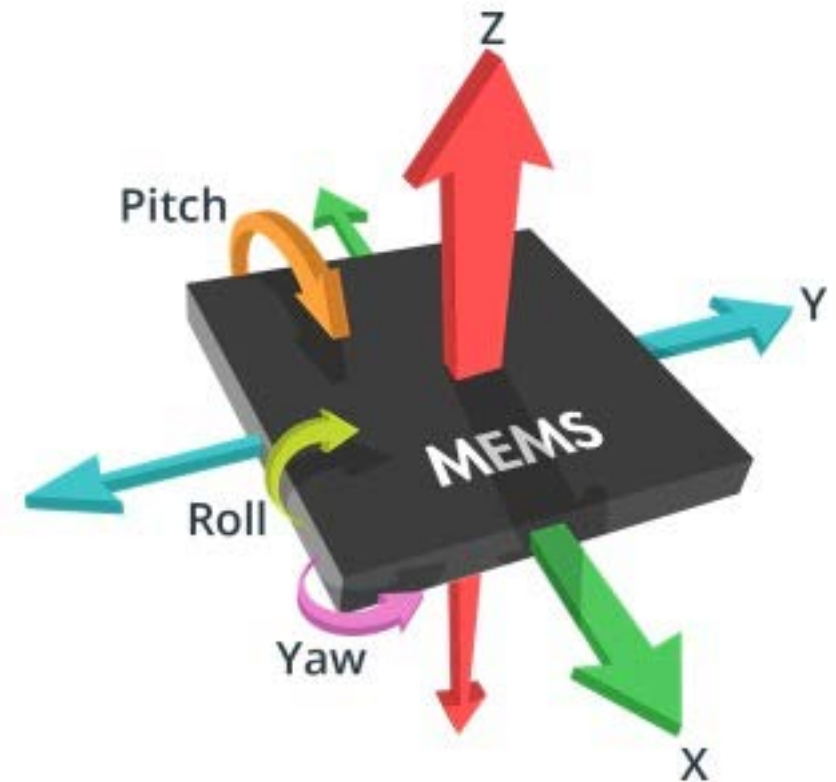


```
when Screen1 .Initialize
do
  set TextBox1 . Text to
  call TinyDB1 .GetValue
    tag "lastMessage"
    valueIfTagNotThere "Ninguna frase almacenada"
```

The image shows a sequence of Scratch-style code blocks. The first block is a yellow 'when Screen1 .Initialize' block. Inside its 'do' area, there is a green 'set TextBox1 . Text to' block. This block is connected to a purple 'call TinyDB1 .GetValue' block. The 'call' block has two arguments: 'tag "lastMessage"' and 'valueIfTagNotThere "Ninguna frase almacenada"'. The 'tag' argument is connected to a pink block containing the string 'lastMessage'. The 'valueIfTagNotThere' argument is connected to a pink block containing the string 'Ninguna frase almacenada'.

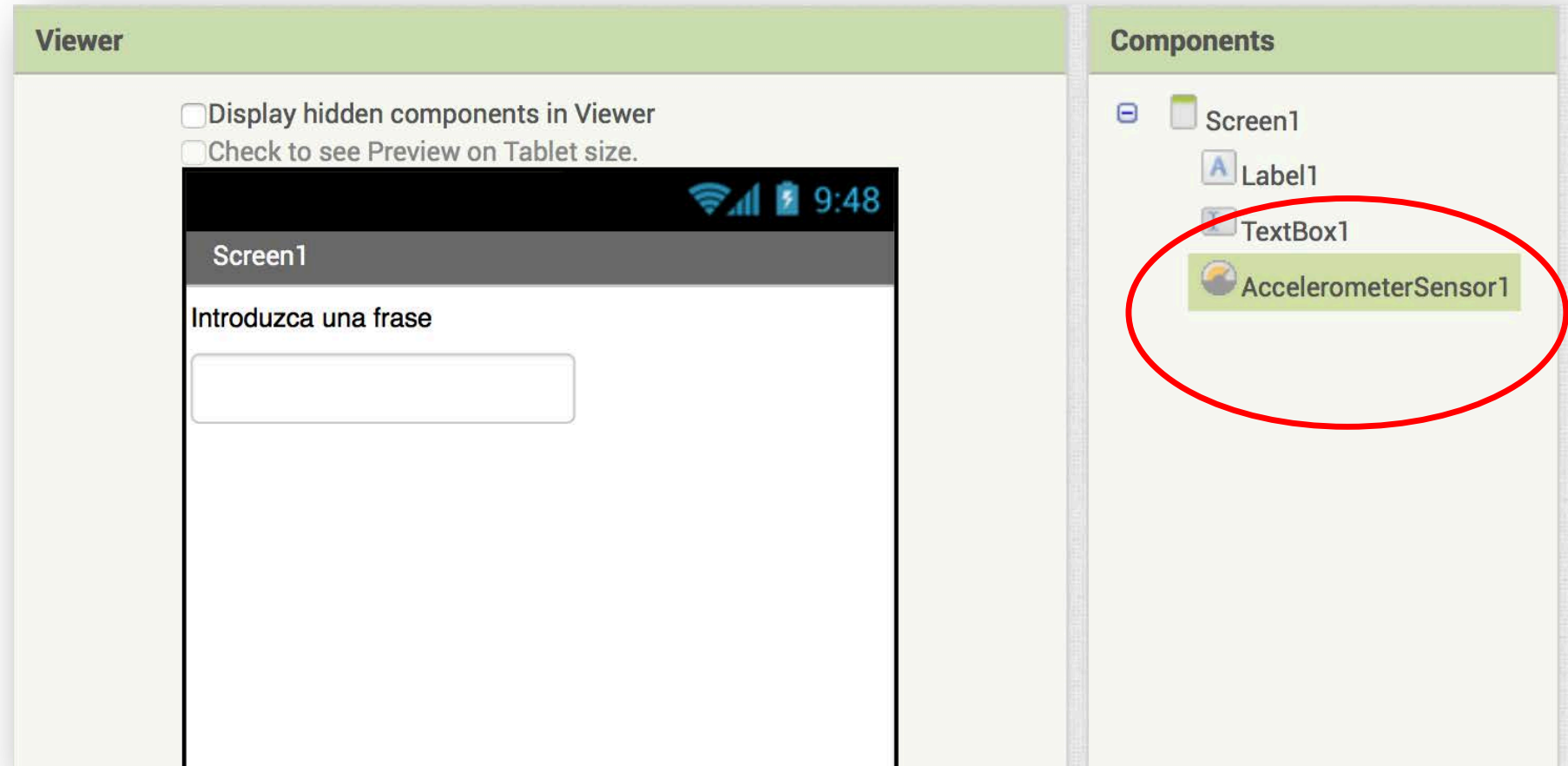
Segunda App

Acelerómetro



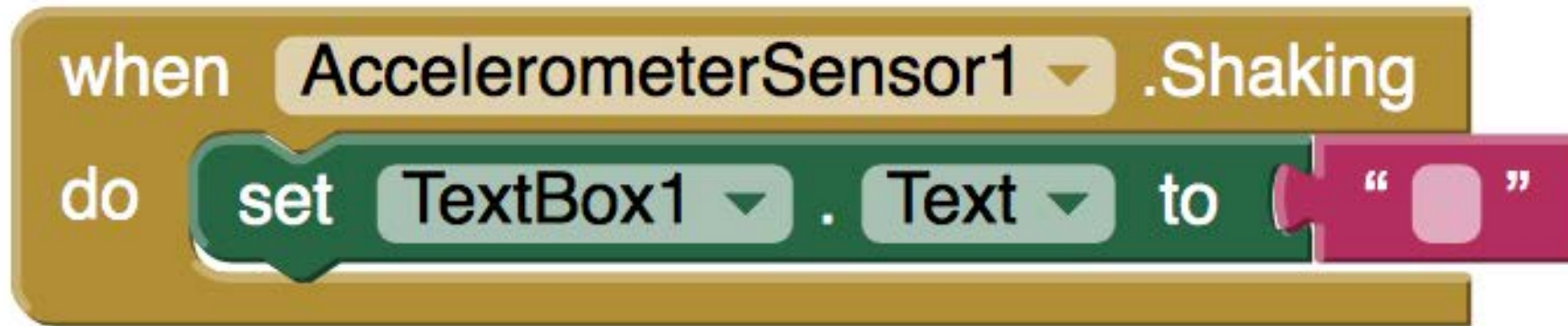
Acelerómetro

- Modelamos la interfaz añadiendo componentes a la pantalla.
- Añadimos un sensor que detecte el movimiento.



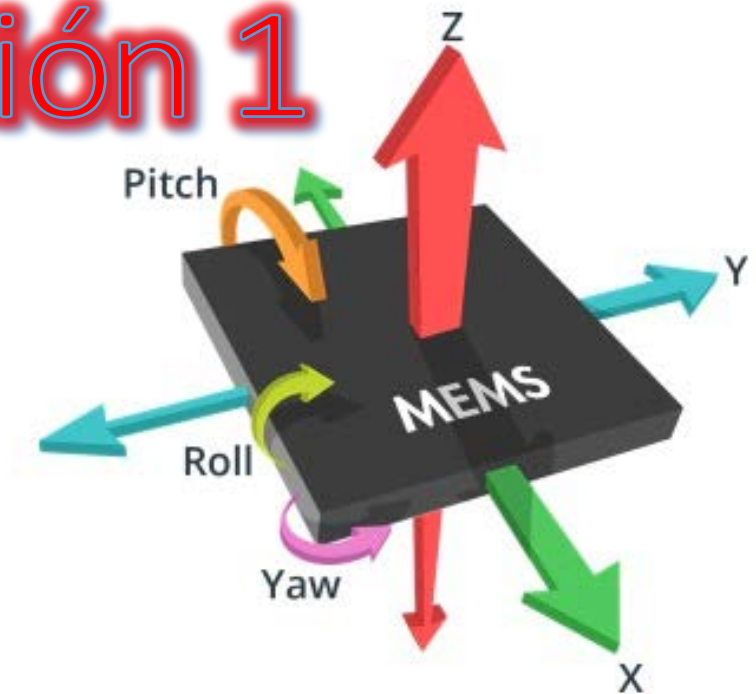
Acelerómetro

- Modelamos el comportamiento.



Segunda App: **Modificación 1**

Acelerómetro + Notificador



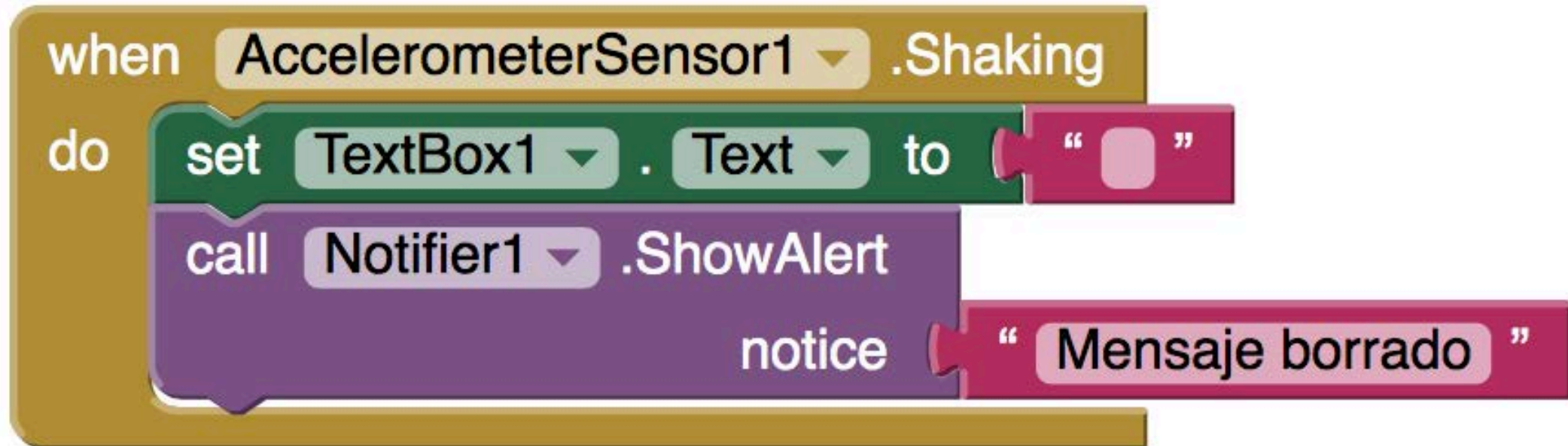
Acelerómetro + Notificador

- Modelamos la interfaz añadiendo componentes a la pantalla.

The image shows the Android Studio interface. On the left, a preview window displays a mobile app screen titled "Screen1" with a status bar at the top showing signal strength, Wi-Fi, battery, and the time 9:48. Below the status bar is a header "Screen1" and a text input field with the placeholder "Introduzca una frase". At the bottom of the preview, the Android navigation bar is visible, circled in red. Below the preview, the "Non-visible components" section shows "AccelerometerSensor1" and "Notifier1", with "Notifier1" highlighted by a green box. On the right, the "Component palette" shows a tree view with "Screen1" containing "Label1", "TextBox1", "AccelerometerSensor1", and "Notifier1". The "Notifier1" component is highlighted in green. Below the palette are "Rename" and "Delete" buttons. At the bottom right, there is a "Media" section with an "Upload File ..." button.

Acelerómetro

- Modelamos el comportamiento.



Tercera App

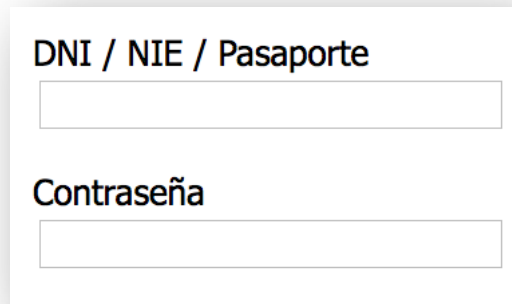
Login



Login

- Pantalla de inicio que solicite al usuario su identificador y su contraseña.
- Modelamos la interfaz

1

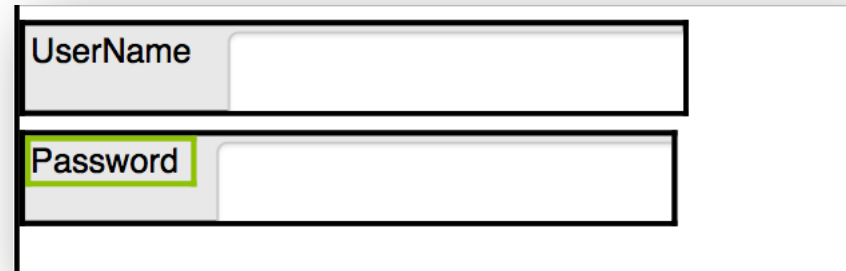


A form model with two input fields. The first field is labeled "DNI / NIE / Pasaporte" and the second field is labeled "Contraseña".

DNI / NIE / Pasaporte

Contraseña

2



A form model with two input fields. The first field is labeled "UserName" and the second field is labeled "Password". The "Password" label is highlighted with a green border.

UserName

Password

Login

- Layout => Organizar los elementos dentro de la interfaz de usuario

The screenshot displays a mobile application development environment with three main panels: Palette, Viewer, and Components.

- Palette:** Located on the left, it is divided into sections. The 'User Interface' section is expanded to show the 'Layout' category, which is highlighted with a red box. The 'Layout' section contains five options: 'HorizontalArrangement', 'HorizontalScrollArrangement', 'TableArrangement', 'VerticalArrangement', and 'VerticalScrollArrangement'. The 'HorizontalArrangement' option is selected, and two red arrows point from it to the 'UserName' and 'Password' text boxes in the viewer.
- Viewer:** The central panel shows a preview of the application. At the top, there is a status bar with icons for Wi-Fi, signal strength, battery, and the time '9:48'. Below the status bar is a header labeled 'Screen1'. The main content area contains two text input fields: 'UserName' and 'Password'. The 'Password' field is highlighted with a green box.
- Components:** Located on the right, it shows a hierarchical tree of the application's components. The root is 'Screen1', which contains 'HorizontalArrangement1'. Under 'HorizontalArrangement1' are 'User' and 'TextBox1'. Below that is 'HorizontalArrangement2', which contains 'Password' and 'TextBox2'. The 'Password' component is highlighted with a green box.

Tercera App: **Modificación 1**

Login



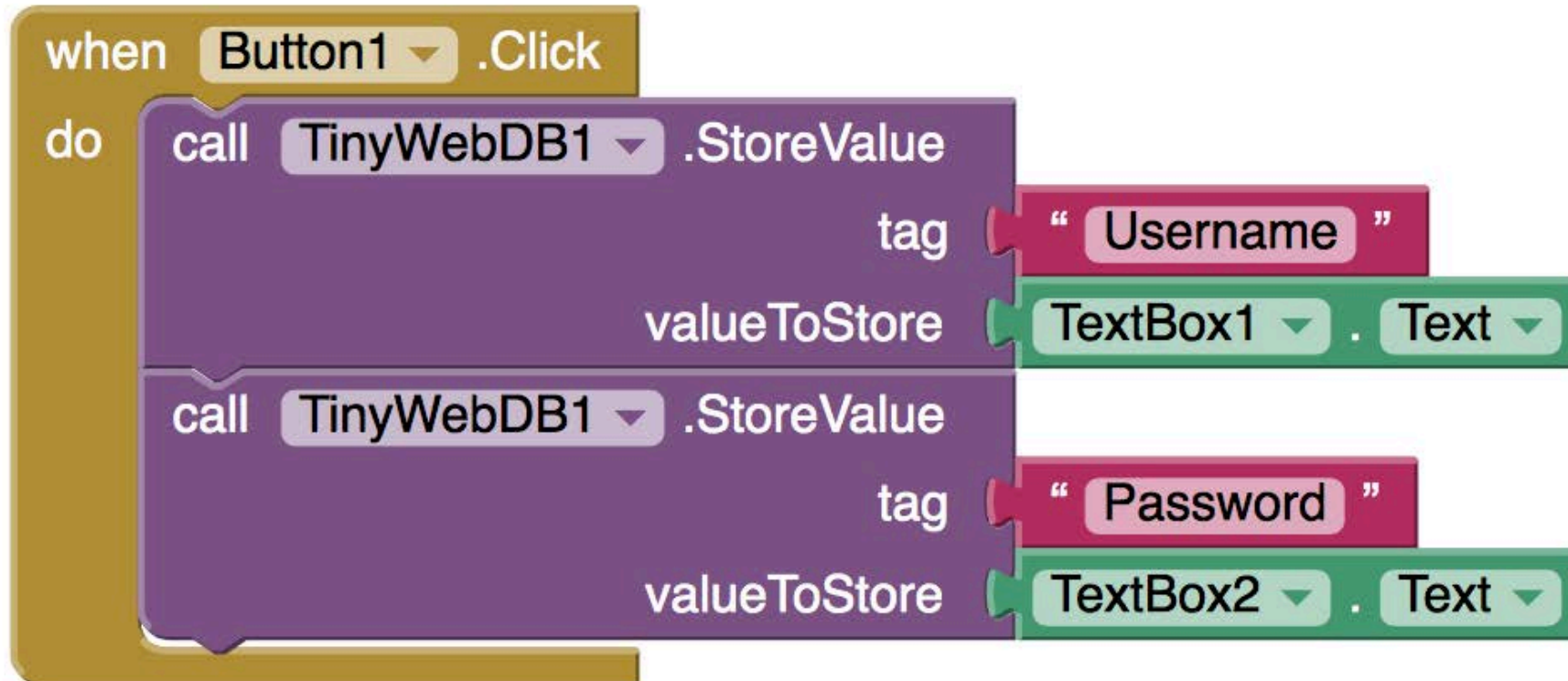
Login: Que le permita registrarse, es decir, el username y password del usuario se almacenarán en el móvil.

- Añadimos elementos a la interfaz...

The image shows the Android Studio IDE interface. On the left is the 'User Interface' component palette with categories like 'User Interface', 'Layout', 'Media', 'Drawing and Animation', and 'Sensors'. The 'User Interface' category is expanded, showing various widgets such as Button, CheckBox, DatePicker, Image, Label, ListPicker, ListView, Notifier, PasswordTextBox, Slider, Spinner, TextBox, TimePicker, and WebViewer. In the center is a preview of a mobile screen titled 'Screen1' with a status bar at the top showing signal strength, Wi-Fi, and the time 9:48. The screen contains a registration form with two text boxes labeled 'UserName' and 'Password', and a 'Register' button below them. A red arrow points from the 'Register' button in the preview to the 'Button' widget in the component palette. Another red arrow points from the 'Layout' category in the component palette to the 'Non-visible components' section at the bottom of the preview, which contains a 'TinyWebDB1' component. On the right side of the interface is a 'Media' section with an 'Upload File ...' button. At the bottom right of the preview area are 'Rename' and 'Delete' buttons.

Login: Que le permita registrarse, es decir, el username y password del usuario se almacenarán en el móvil.

- Modelamos el comportamiento



Login: Que le permita registrarse, es decir, el username y password del usuario se almacenarán en el móvil.

- Y probamos la aplicación...



Cuarta App

Foto

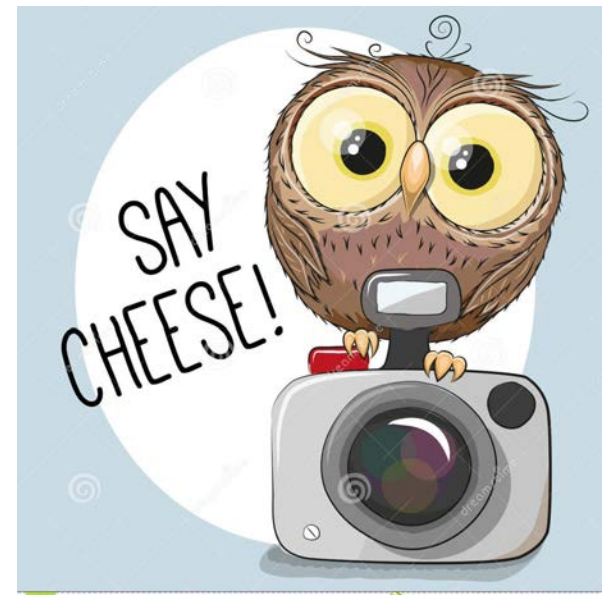


Foto: ...que haga una foto y la muestre a continuación.

- Necesitamos los componentes: Botón, Camera e ImageViewer.

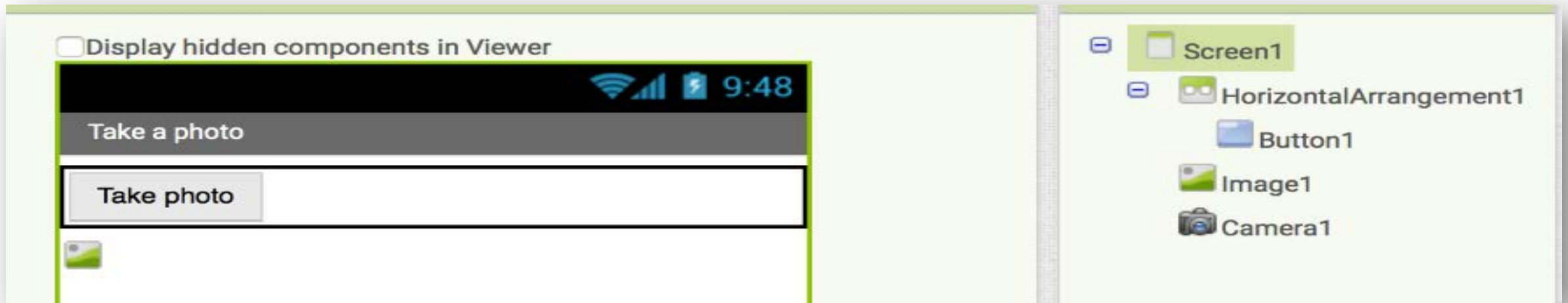
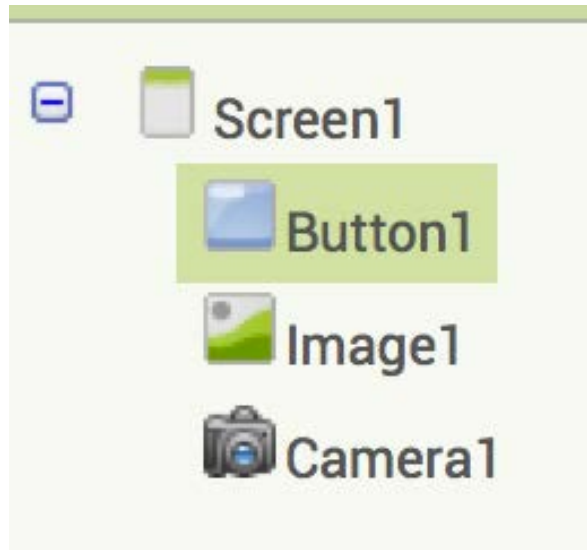


Foto: ...que haga una foto y la muestre a continuación.

- Y la muestra a continuación.



```
when Button1 .Click
do call Camera1 .TakePicture

when Camera1 .AfterPicture
image
do set Image1 . Picture to get image
set Image1 . Visible to true
```

Cuarta App: **Modificación 1**

Foto

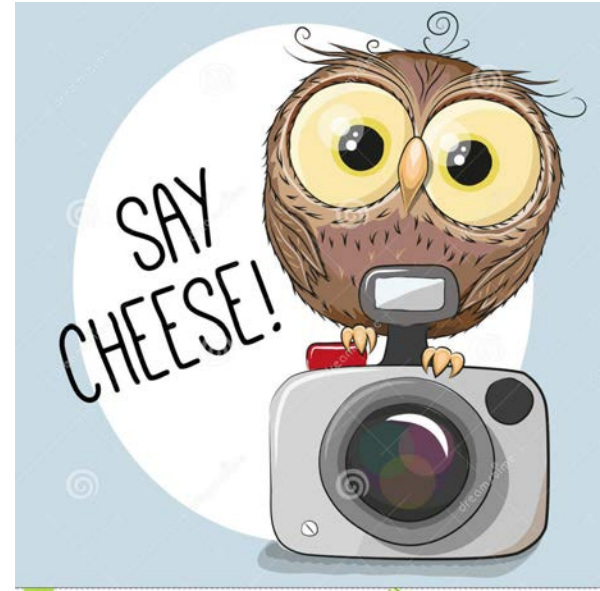


Foto: ...que muestre la foto en otra pantalla

- Añadimos un nuevo screen y movemos UIImageView a la otra pantalla.

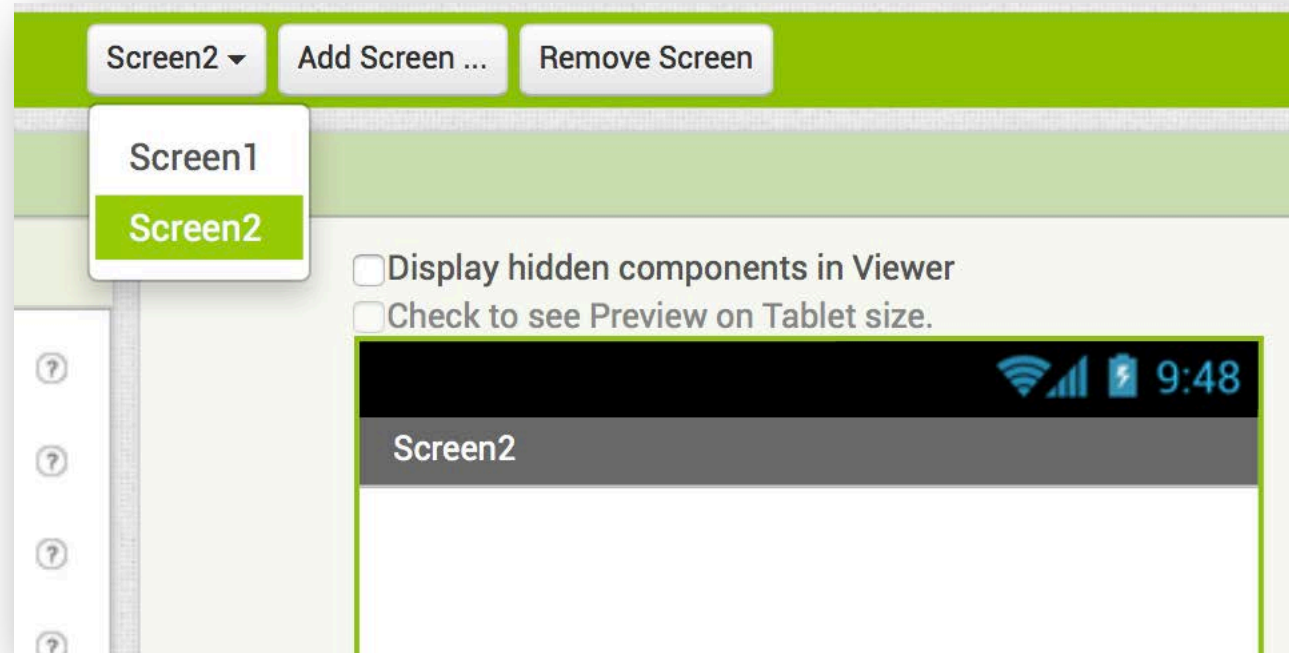


Foto: ...que muestre la foto en otra pantalla

- Modelamos las interfaces

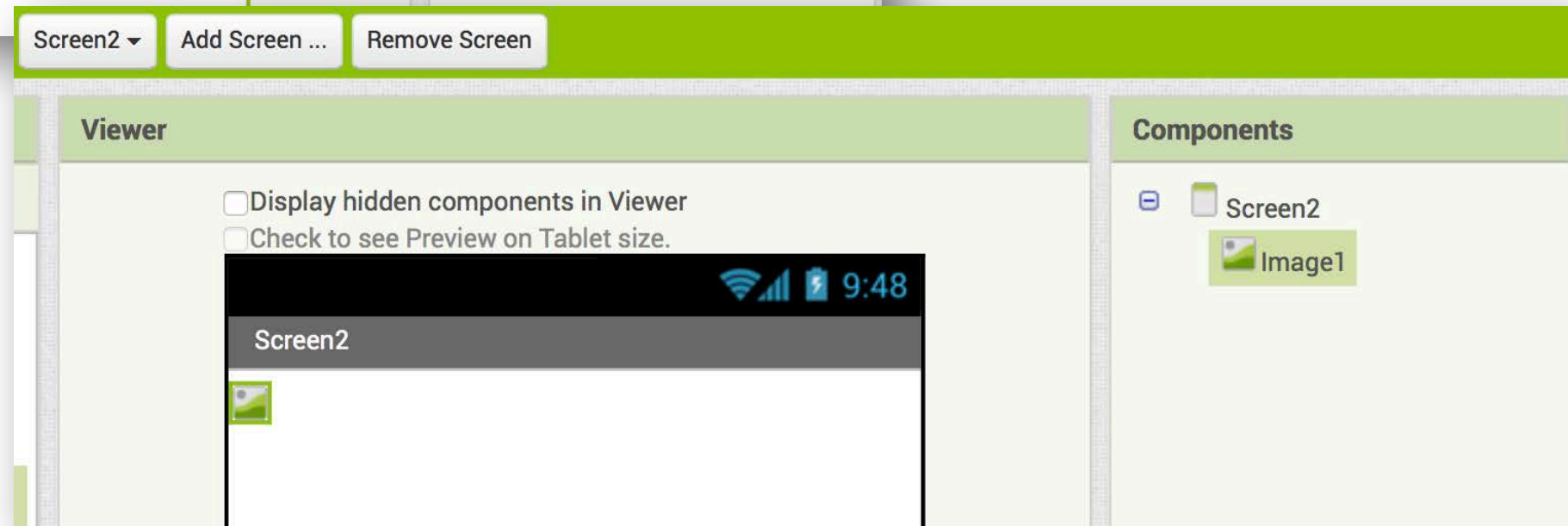
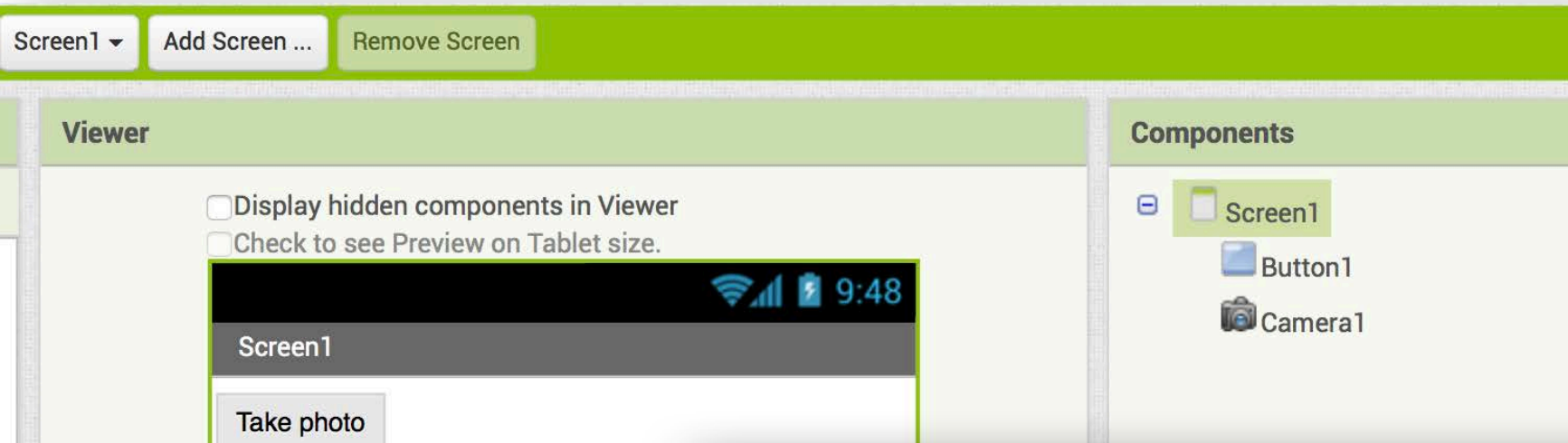
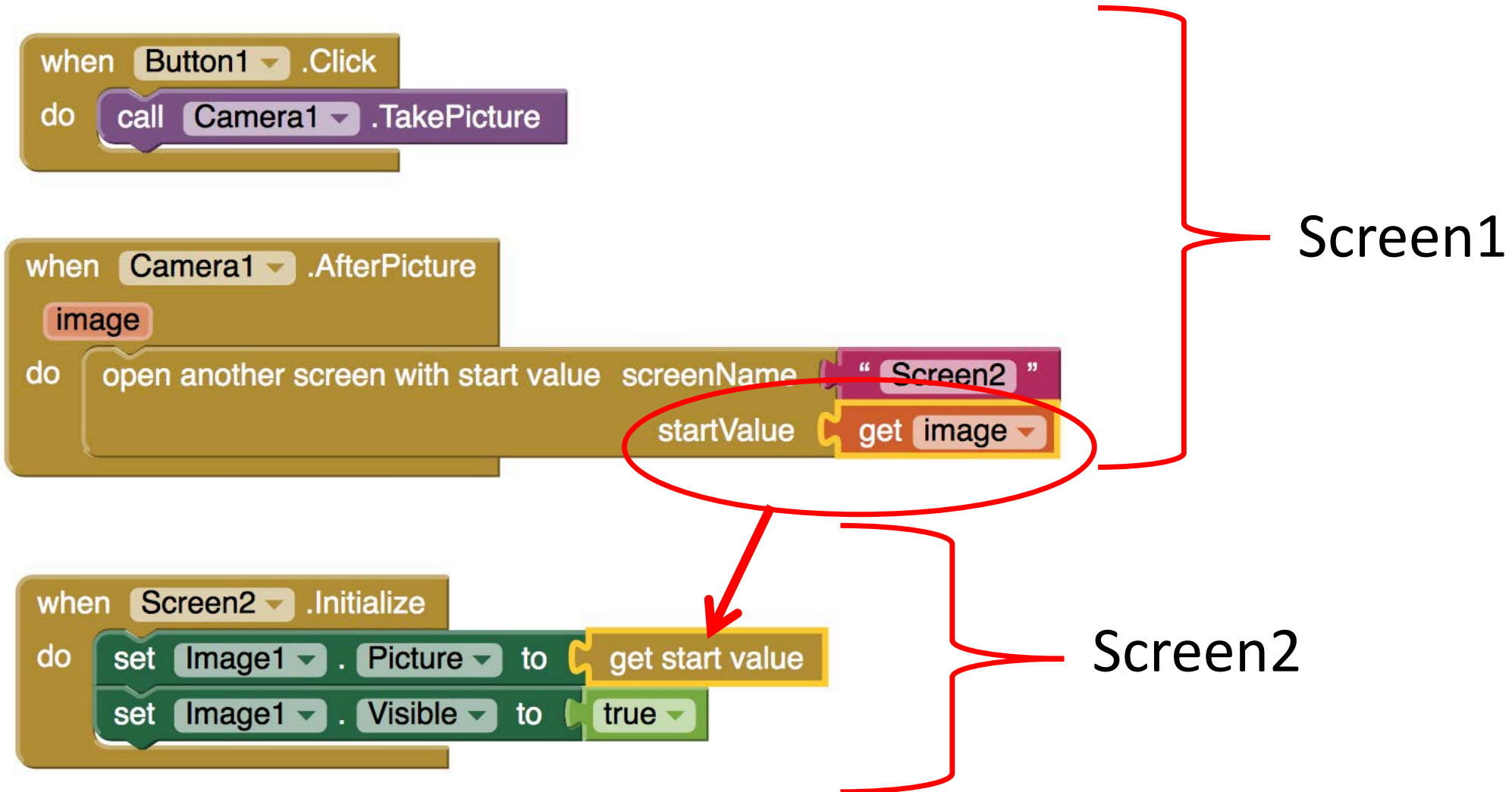


Foto: ...que muestre la foto en otra pantalla

- Modelamos el comportamiento



Quinta App

Códigos QR



QR: Aplicación que escanee un código QR y muestre su contenido en un cuadro de texto

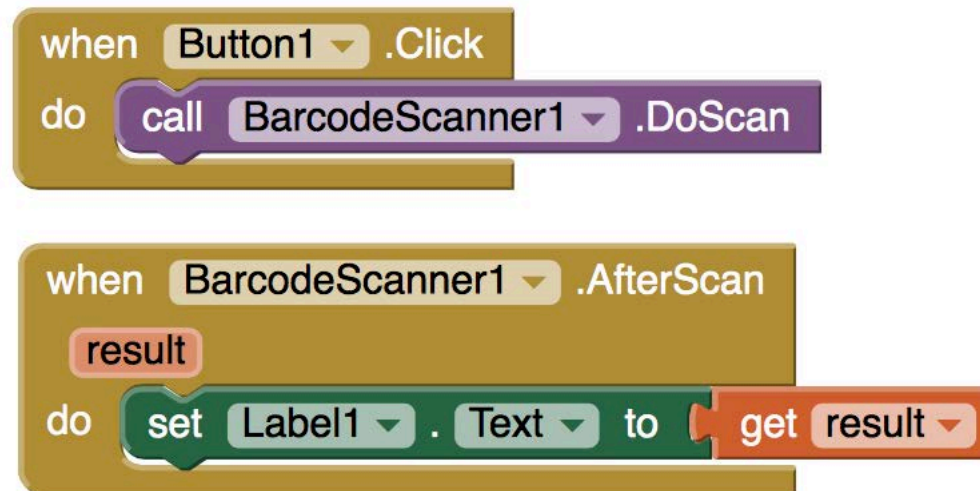
- Muestra el código leído en la etiqueta **MuestraCodigoLabel**
- La aplicación se inicia con el botón **LeeCodigoButton**



<http://www.codigos-qr.com/generador-de-codigos-qr/>

QR: Aplicación que escanee un código QR y muestre su contenido en un cuadro de texto

- Cuando el usuario lo pulsa (hace **Click**) llama a **BarcodeScanner.DoScan** que pone en funcionamiento el lector de códigos.
- Cuando este termina, establecemos el valor de la propiedad **Text** de la etiqueta **MuestraCodigoLabel** con el resultado de la lectura.



```
when Button1 .Click
do call BarcodeScanner1 .DoScan

when BarcodeScanner1 .AfterScan
result
do set Label1 . Text to get result
```

The image shows two Scratch-style code blocks. The first block is a 'when clicked' block for 'Button1' containing a 'call BarcodeScanner1 .DoScan' block. The second block is a 'when BarcodeScanner1 .AfterScan' block containing a 'set Label1 . Text to get result' block, where 'get result' is connected to the 'to' parameter of the 'set' block.

Nota:



- Para poder utilizar el lector de códigos, la aplicación "escáner de código de barras" de ZXing debe estar instalada en el teléfono. Esta aplicación está disponible de forma gratuita en el Android Market.

Quinta App: **Modificación 1**

Códigos QR

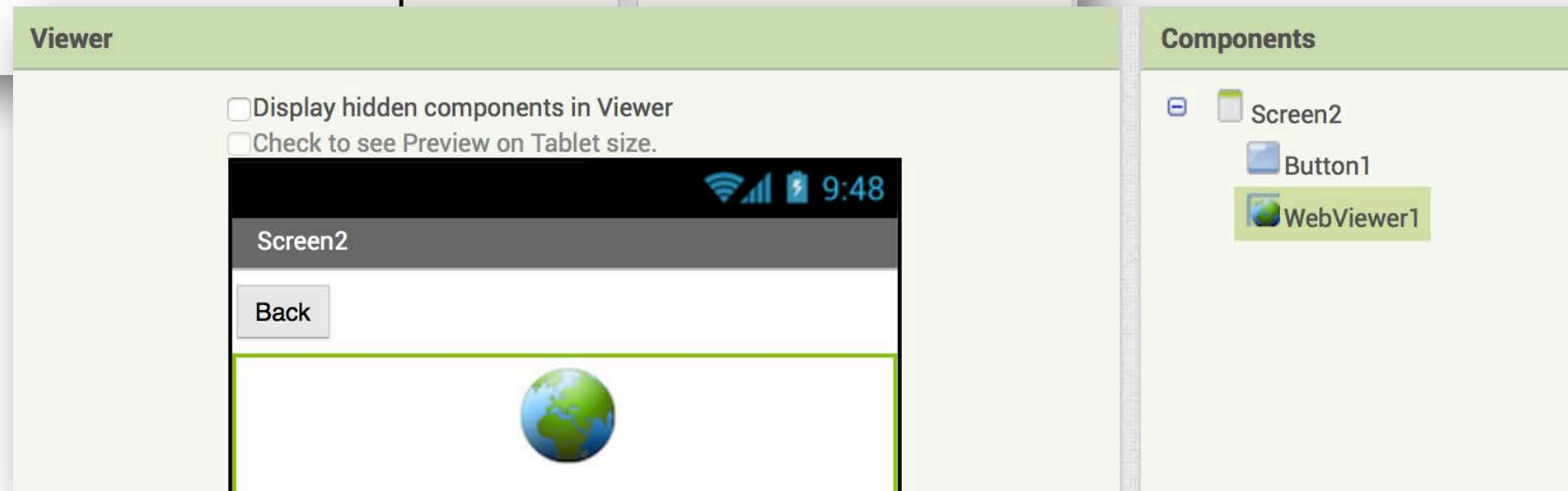
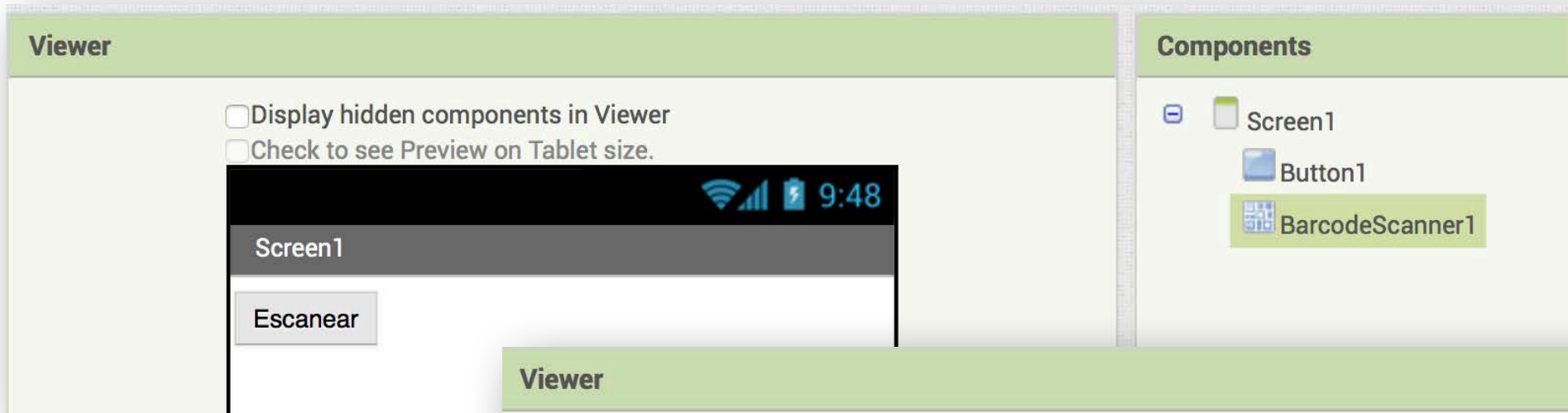


QR: ...que escanee un código QR y abra una página Web en otra pantalla

- En la pantalla principal añadimos un botón y un componente lector de código, denominado “**BarcodeScanner**”.
- Tenemos que crear otro pantalla “Screen” → Botón “add Screen”.
- Añadimos a la nueva pantalla un botón para volver a la primera pantalla, y un componente **WebView**
- Cada pantalla tiene su comportamiento basado en bloques.

QR: ...que escanee un código QR y abra una página Web en otra pantalla

- Modelamos las interfaces.



QR: ...que escanee un código QR y abra una página Web en otra pantalla

```
when GetQR .Click  
do call BarcodeScanner1 .DoScan
```

Screen1

```
when BarcodeScanner1 .AfterScan  
result  
do open another screen with start value screenName "Screen2"  
startValue BarcodeScanner1 .Result
```

```
when Screen2 .Initialize  
do call WebViewer1 .GoToUrl  
url get start value
```

Screen2

```
when Button1 .Click  
do open another screen screenName "Screen1"
```

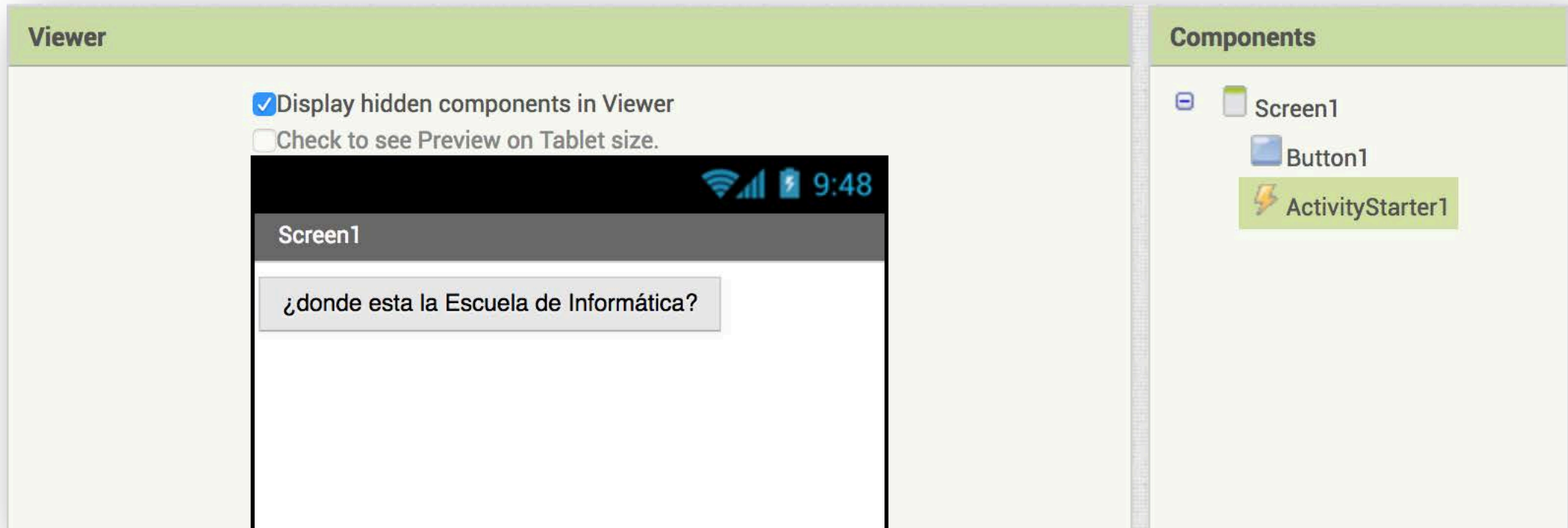
Sexta App

Google Maps



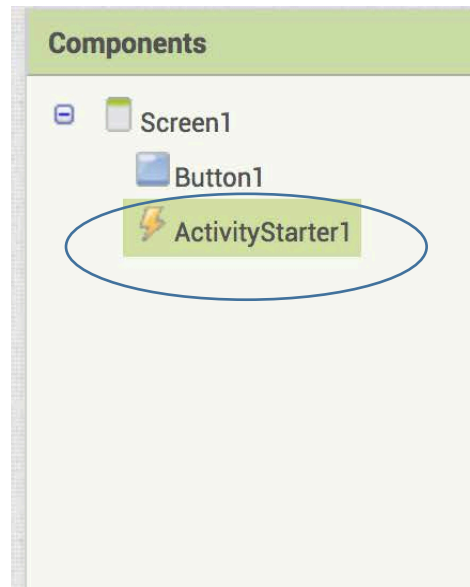
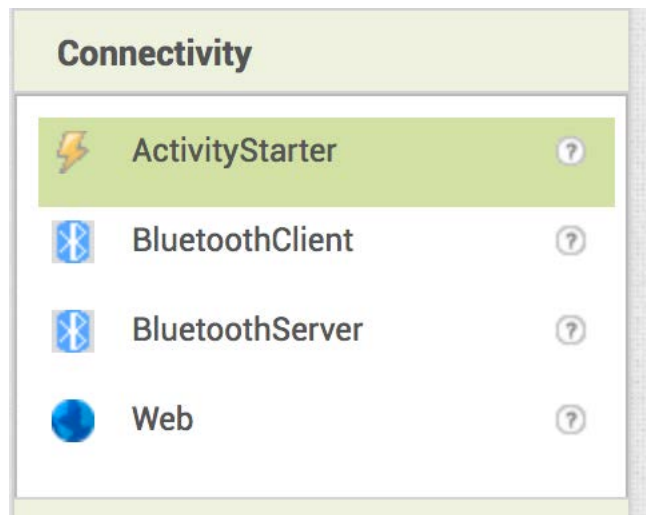
Google Maps: ...que nos muestre nuestra ubicación en un mapa.

- La Escuela de Informática
- La aplicación se inicia cuando preguntas donde está la Escuela ...



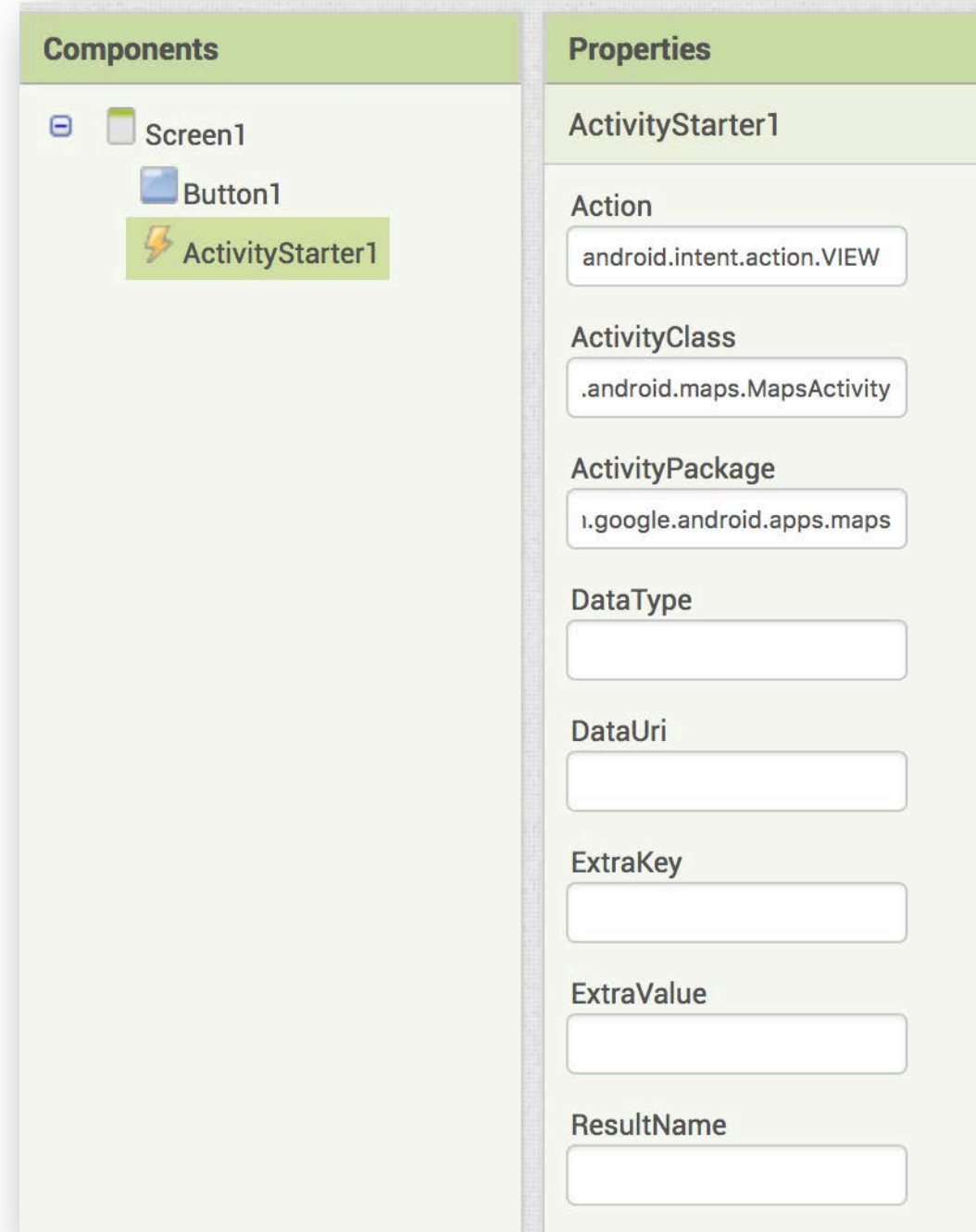
Google Maps: ...que nos muestre nuestra ubicación en un mapa.

- Para mostrar ubicaciones en el mapa usamos la API de Google maps
- Necesitamos el componente ActivityStarter (no visual)
 - Inicia otra app de Android desde nuestra app
 - En la propiedades se configuran las propiedades especificas de la app a lanzar



Propiedades de Activity Starter

- Para llamar a Google Maps:
 - Action:
 - **android.intent.action.VIEW**
 - ActivityClass
 - **com.google.android.maps.MapActivity**
 - ActivityPackage
 - **com.google.android.apps.maps**



The screenshot displays the Android Studio interface. On the left, the 'Components' panel shows a tree view with 'Screen1' containing 'Button1' and 'ActivityStarter1'. The 'ActivityStarter1' component is highlighted. On the right, the 'Properties' panel shows the configuration for 'ActivityStarter1'.

Property	Value
Action	android.intent.action.VIEW
ActivityClass	.android.maps.MapActivity
ActivityPackage	i.google.android.apps.maps
DataType	
DataUri	
ExtraKey	
ExtraValue	
ResultName	

Blocks → Comportamiento

- Indicamos nuestra ubicación
 - Universidad de Málaga: **Escuela Técnica Superior de Ingeniería Informática**

```
initialize global ubicacion to " Universidad de Málaga: Escuela Técnica Superior de Ingeniería Informática "
```

```
when Button1 .Click  
do set ActivityStarter1 . DataUri to join " http://maps.google.com?q="  
get global ubicacion
```

Sexta App: **Modificación 1**

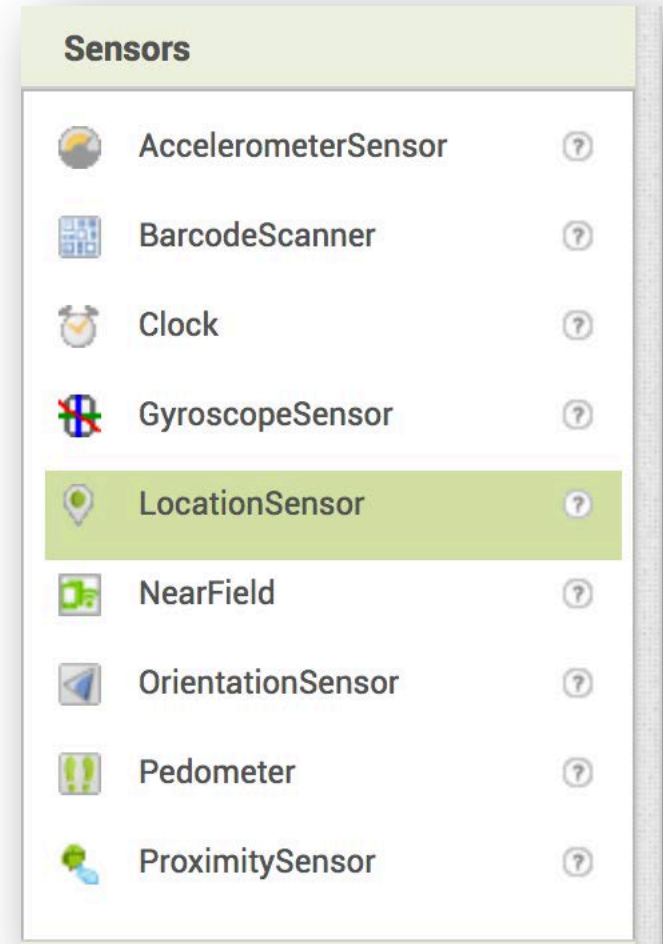
Google Maps





Mostrar ubicación actual

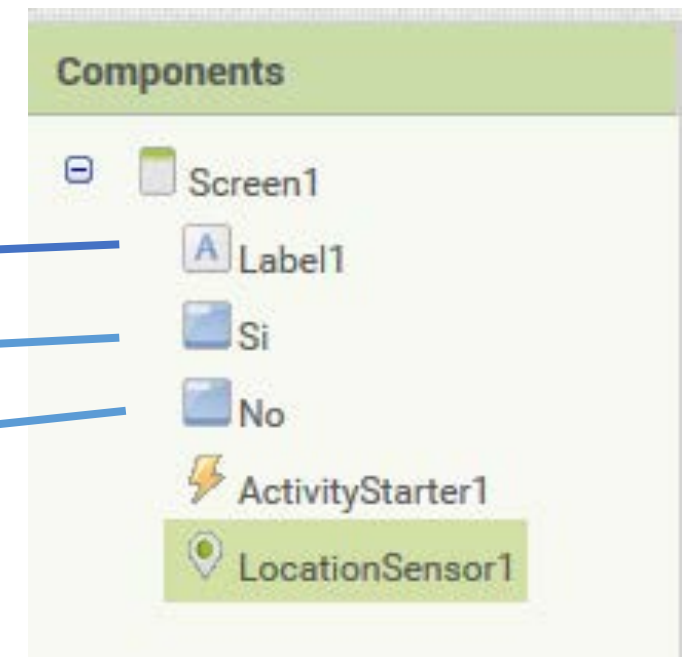
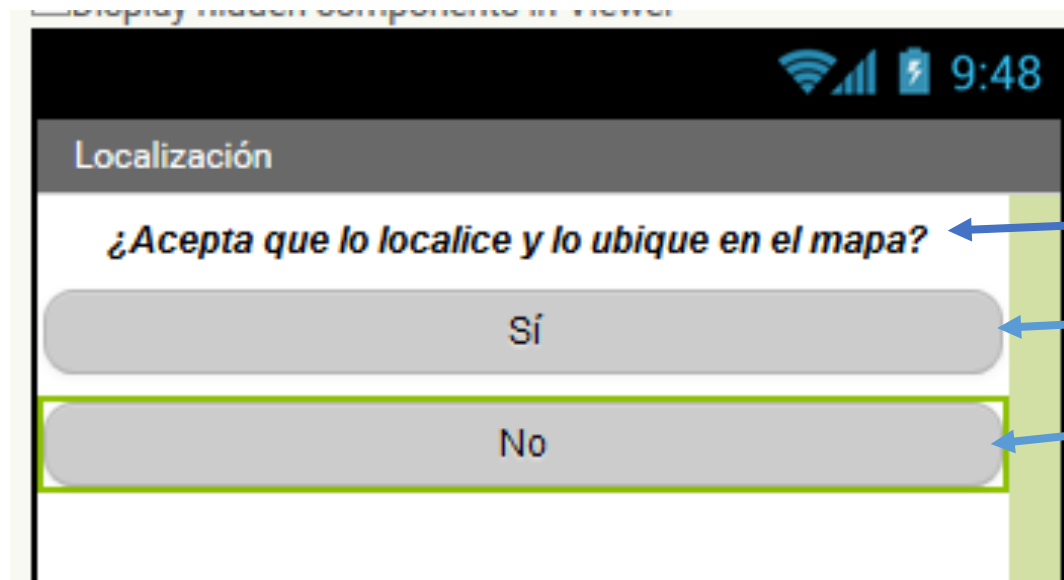
- En lugar de escribir nuestra ubicación, la obtendremos a través del componente **LocationSensor**
 - Captura la información de longitud y latitud que proporciona el GPS (o de otro método interno de Android)
 - Necesita una variable para almacenar la posición
 - Hay que tratar el evento de cambio de posición (**LocationChanged**)
 - Lo usaremos con **Activity Starter** para iniciar la aplicación google maps en el móvil y hacer que google maps muestre la posición actual



Google Maps: ...que nos muestre nuestra ubicación en un mapa.

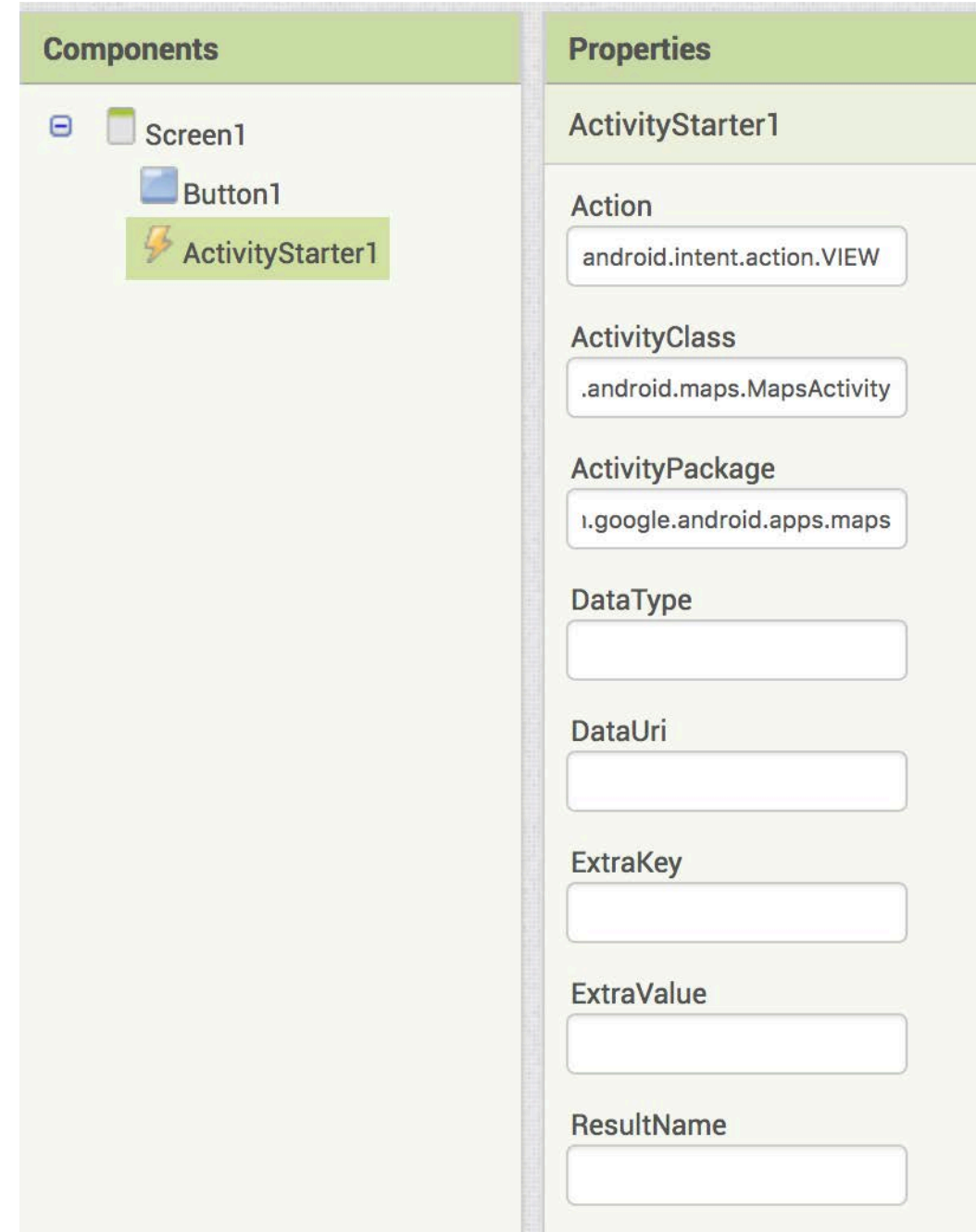


- Accede a los datos de localización del móvil (GPS)
- La aplicación se inicia cuando aceptas que te localicen ...



Propiedades de Activity Starter

- Para llamar a Google Maps:
 - Action:
 - android.intent.action.VIEW
 - ActivityClass
 - com.google.android.maps.MapActivity
 - ActivityPackage
 - com.google.android.apps.maps

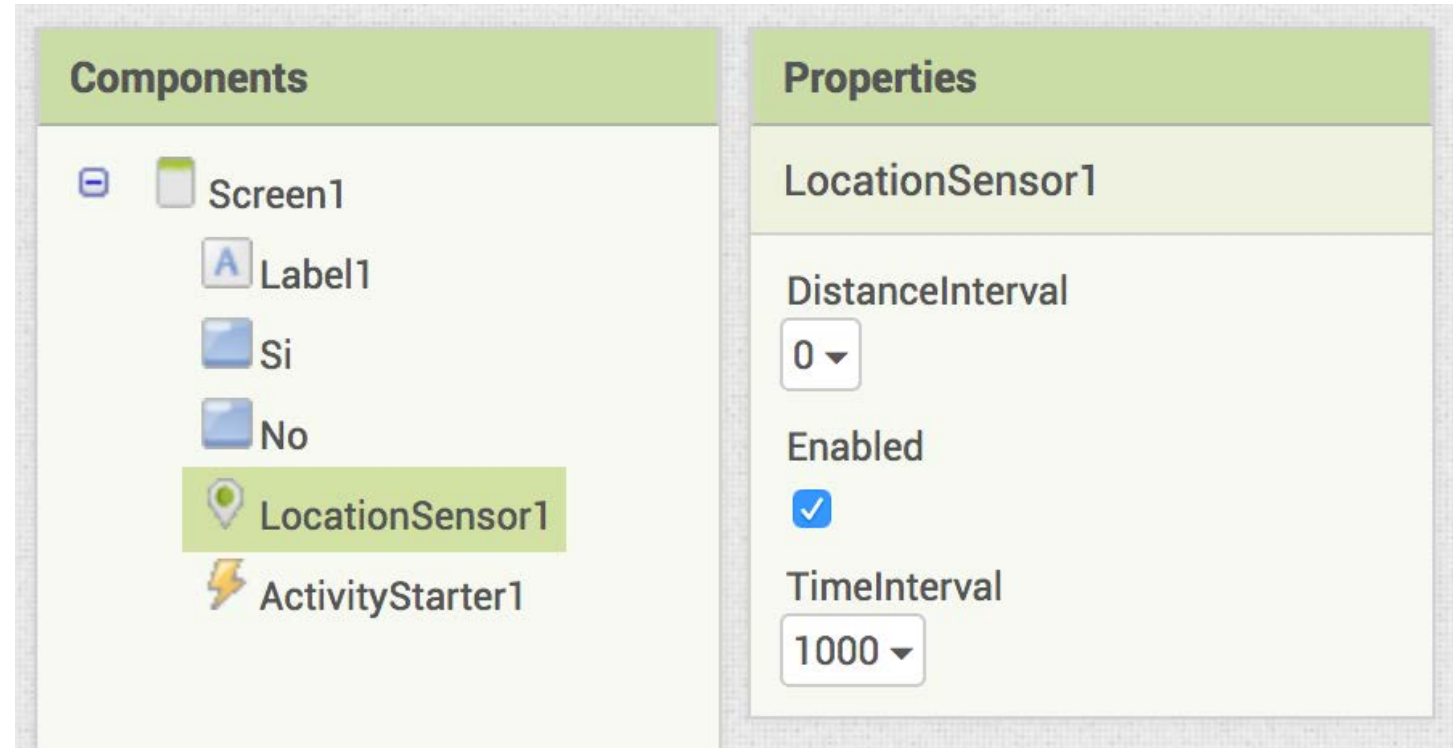


The screenshot displays the Android Studio interface with the 'Components' and 'Properties' panels. The 'Components' panel shows a hierarchy: Screen1 (containing Button1 and ActivityStarter1). The 'Properties' panel is titled 'ActivityStarter1' and lists the following properties:

- Action:** android.intent.action.VIEW
- ActivityClass:** .android.maps.MapActivity
- ActivityPackage:** i.google.android.apps.maps
- DataType:** (empty text box)
- DataUri:** (empty text box)
- ExtraKey:** (empty text box)
- ExtraValue:** (empty text box)
- ResultName:** (empty text box)

Mostrar ubicación actual

- Obtenida a través del componente **LocationSensor**
 - Captura la información de longitud y latitud que proporciona el GPS (o de otro método interno de Android)
 - Lo usaremos con **ActivityStarter** para iniciar la aplicación google maps en el móvil y hacer que google maps muestre la posición actual



Blocks → Comportamiento

```
initialize global ubicacionActual to " Universidad de Málaga: Escuela Técnica Superior de Ingeniería Informática "
```

```
when Si .Click
```

```
do set ActivityStarter1 . DataUri to join " http://maps.google.com?q="
```

```
get global ubicacionActual
```

```
call ActivityStarter1 .StartActivity
```

```
when LocationSensor1 .LocationChanged
```

```
latitude longitude altitude speed
```

```
do set global ubicacionActual to LocationSensor1 . CurrentAddress
```

```
when No .Click
```

```
do close application
```

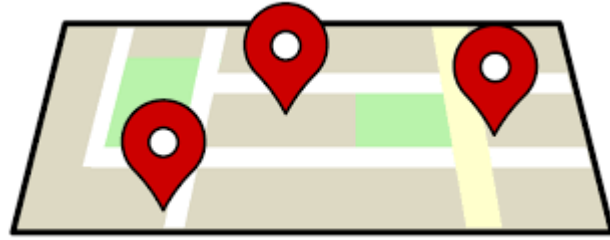
Almacenamos la ubicación en una variable.
Gestionamos el evento LocationChanged

Sexta App: **Modificación 2**

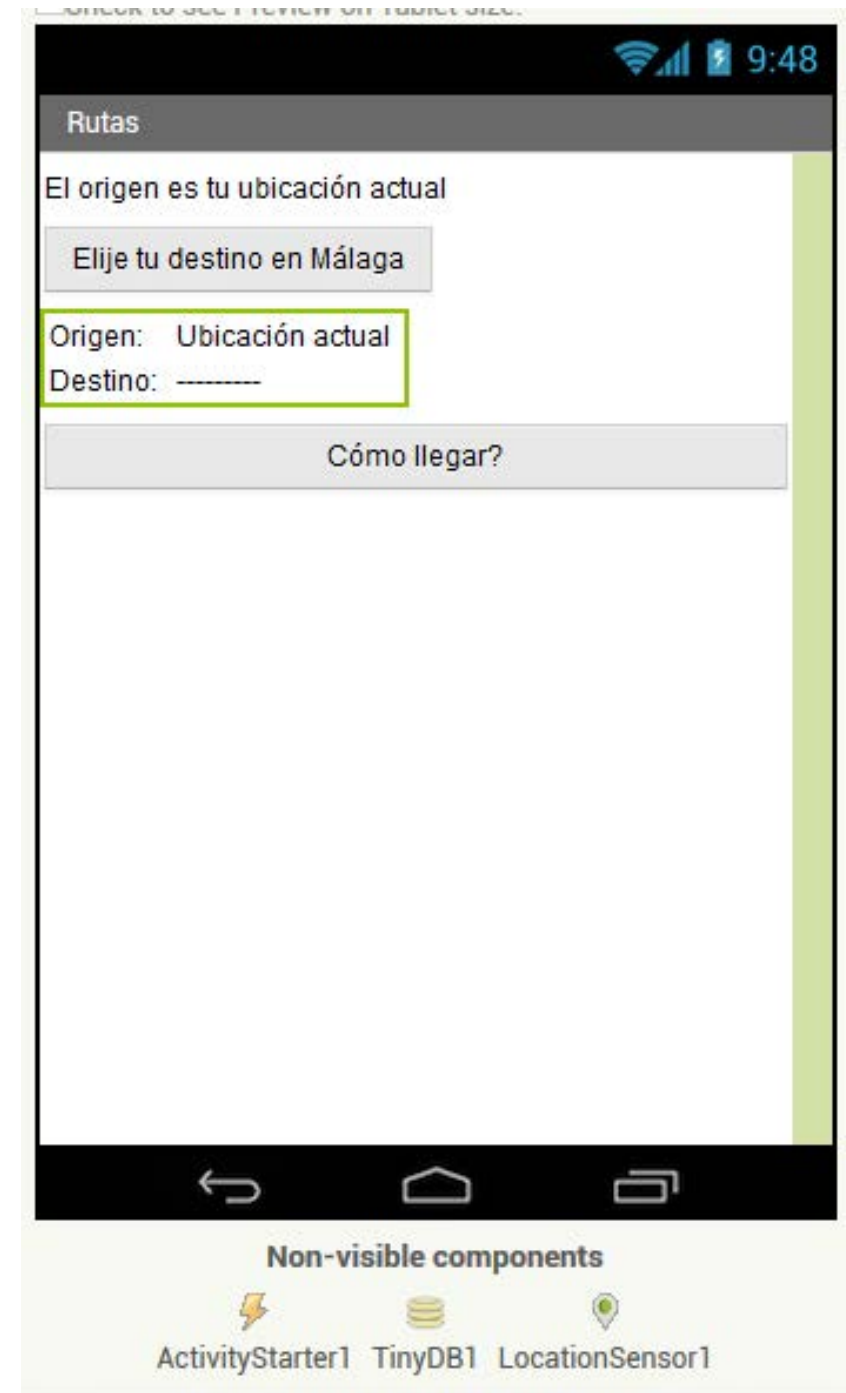
Ruta



App Ruta



- Nos muestra la ruta entre dos puntos
- El origen de la ruta es nuestra ubicación actual
- El destino lo seleccionamos de una lista
- Cuando el botón de “Cómo llegar” se pulse, se debe iniciar una actividad (componente ActivityStarter) para abrir la aplicación de Google maps y nos muestre la ruta.





App Ruta

- Diseño de la interfaz

Viewer

Display hidden components in Viewer
 Check to see Preview on Tablet size.

Rutas

El origen es tu ubicación actual

Elije tu destino en Málaga

Origen: Ubicación actual

Destino: -----

Cómo llegar?

Components

- Screen1
 - Label3
 - ListPicker1
 - TableArrangement1
 - Label4
 - Label5
 - Label6
 - Label7
 - Button1
 - ActivityStarter1
 - TinyDB1
 - LocationSensor1

Rename Delete

Media

Upload File ...

Properties

ListPicker1

BackgroundColor
Default

ElementsFromString

Enabled

FontBold

FontItalic

FontSize
14.0

FontTypeface
default ▾

Height
Automatic...

Width
Automatic...

Image
None...

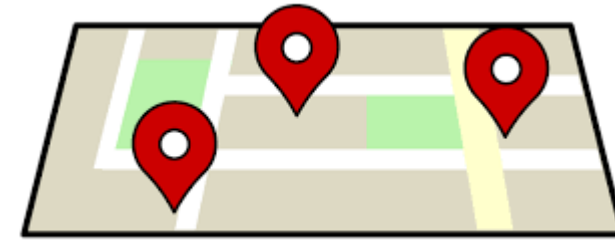
ItemBackgroundColor
Black

ItemTextColor
White

Selection

Non-visible components

- ActivityStarter1
- TinyDB1
- LocationSensor1



App Ruta

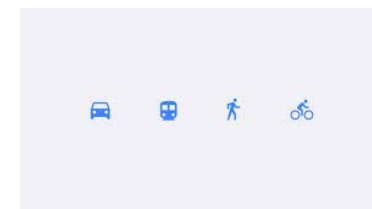
- El origen de la ruta es nuestra ubicación actual

```
initialize global origen to "Universidad de Málaga: Escuela Técnica Superior de Ingeniería Informática"
```

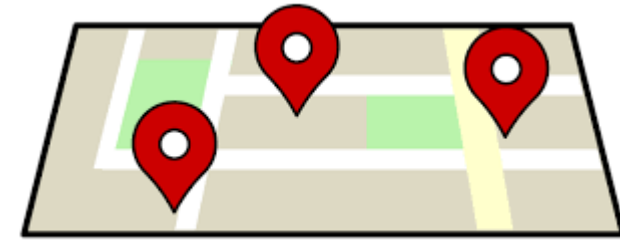
```
initialize global destino to ""
```

- El destino lo seleccionamos de **una lista**
 - Creación de una lista
 - Componentes predefinidos ("built-in) List

```
initialize global Destinations to [make a list ["Alcazaba de Málaga", "Catedral de Málaga", "Museo Picasso de Málaga"]]
```



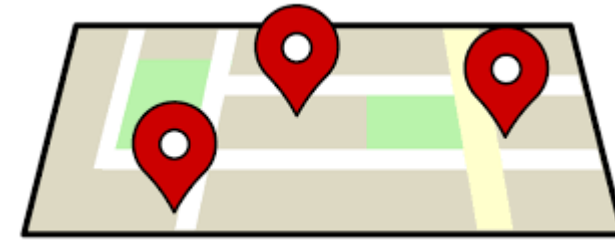
App Ruta



- El destino lo seleccionamos de **una lista**
 - Añadir componentes a una lista seleccionable → Componente ListPicker (en User Interface)

```
when Screen1 .Initialize
do
  call TinyDB1 .StoreValue
  tag "lista"
  valueToStore get global Destinations
  set ListPicker1 . Elements to get global Destinations
  set Label5 . Text to get global origen
  set Label7 . Text to get global destino
  set Button1 . Enabled to false
```


App Ruta



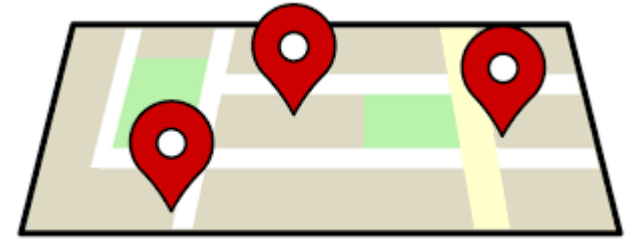
- El destino lo **seleccionamos de una lista**

- Selección de la lista: Evento AfterPicking

- Establecemos el destino al elemento de la lista seleccionado y habilitamos el botón de calcular ruta

```
when ListPicker1 .AfterPicking
do
  set Label7 . Text to ListPicker1 . Selection
  set global destino to ListPicker1 . Selection
  set Button1 . Enabled to true
```

App Ruta

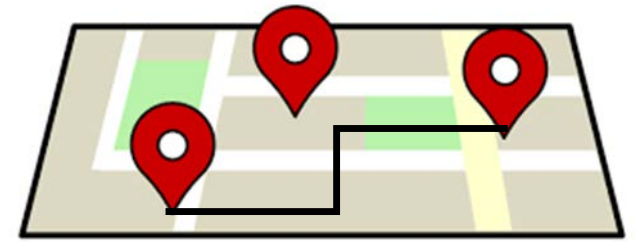


- Cuando el botón de “Cómo llegar” se pulse, se debe iniciar una actividad (componente **ActivityStarter**) para abrir la aplicación de Google maps y nos muestre la ruta.
- Los parámetros para tal acción son los siguientes:
 - a. **Action: android.intent.action.VIEW**
 - b. **ActivityClass: com.google.android.maps.MapActivity**
 - c. **ActivityPackage: com.google.android.apps.maps**

Para proporcionar las coordenadas de origen y destino a la aplicación, debemos modificar el “DataUri” de la actividad antes de iniciarla.

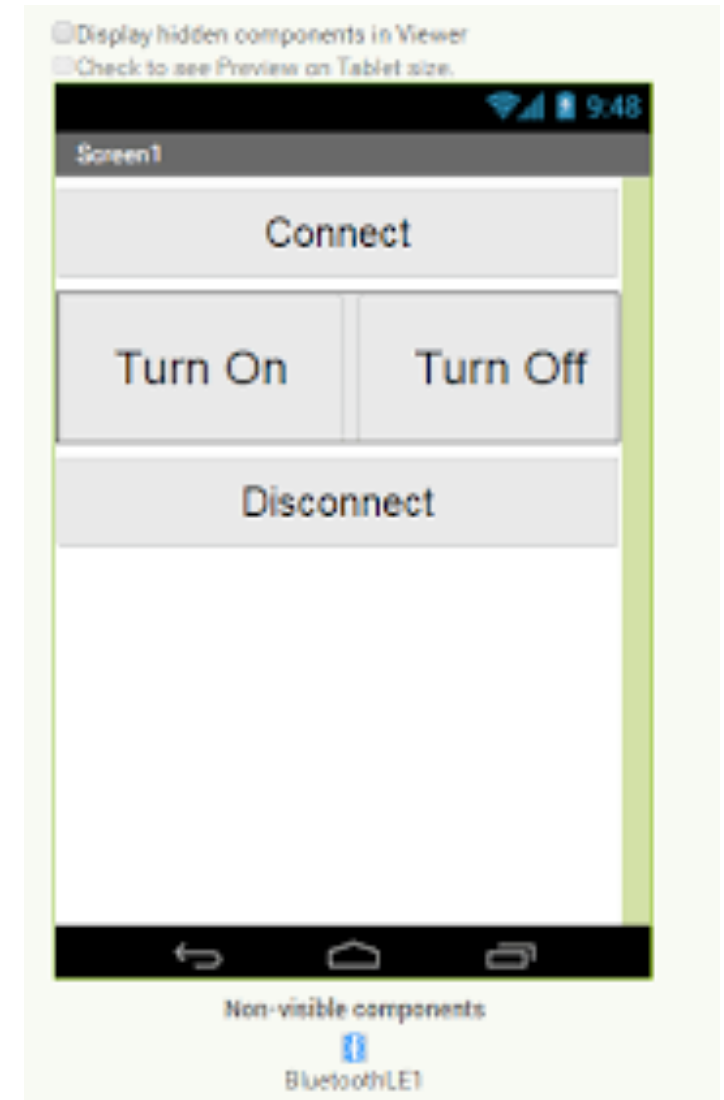
Esta uri es la siguiente: **“http://maps.google.com/maps/?saddr=” + latitudOrigen + “,” + longitudOrigen + “&daddr=” latitudDestino + “,” + longitudDestino**

App Ruta



```
when Button1 .Click
do
  set ActivityStarter1 . DataUri to
  join (
    " http://maps.google.com/?saddr= "
    get global origen
    " &daddr= "
    get global destino
  )
  call ActivityStarter1 .StartActivity
```

Internet de las Cosas: Tecnologías y Aplicaciones



NFC

- Near Field Communication

- NFC, tecnología inalámbrica de corto alcance
- Funciona en la banda de los 13.56 MHz
- Deriva de las etiquetas RFID
- NFC es una plataforma abierta pensada desde el inicio para teléfonos y dispositivos móviles.
- Su tasa de transferencia puede alcanzar los 424 kbit/s → destinada a la comunicación instantánea, es decir, identificación y validación de dispositivos/usuarios.
- Velocidad de comunicación, que es casi instantánea sin necesidad de emparejamiento previo.
- El alcance de la tecnología NFC es muy reducido (rango < 1 cm).
- Su uso es transparente a los usuarios
- Los equipos con tecnología NFC son capaces de enviar y recibir información al mismo tiempo.
- La tecnología NFC puede funcionar en dos modos:
 - Activo, en el que ambos equipos con chip NFC generan un campo electromagnético e intercambian datos.
 - Pasivo, en el que solo hay un dispositivo activo y el otro aprovecha ese campo para intercambiar la información.



Usos

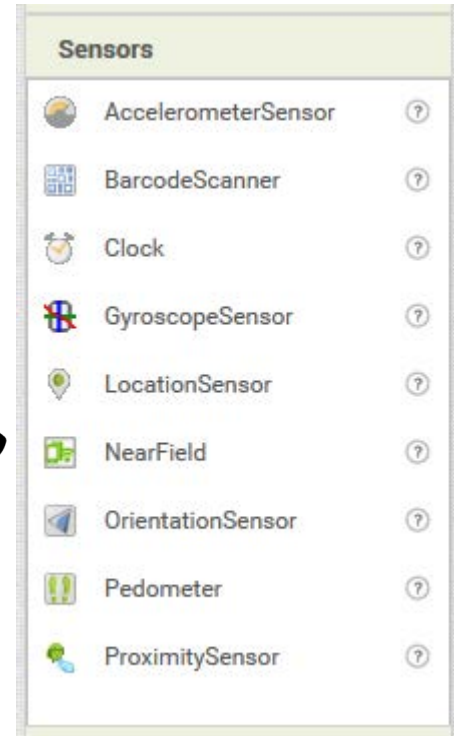


Usos

- La premisa básica a la que se acoge el uso de la tecnología NFC es aquella situación en la que es necesario un intercambio de datos de forma inalámbrica.
- Lo usos que más futuro tienen son la identificación, la recogida e intercambio de información y sobre todo, el pago.
- Identificación:
 - el acceso a lugares donde es precisa una identificación podría hacerse simplemente acercando nuestro teléfono móvil o tarjeta con chip NFC a un dispositivo de lectura.
 - Los nuevos D.N.I. 3.0 tienen NFC
 - mediante esta tecnología podemos pasar nuestra identificación al móvil, para ello pondremos nuestro nuevo DNI **cerca del móvil** y mediante transmisión electromagnética intercambiarán datos.
 - Los abonos de autobús/METRO son un ejemplo.
- Recogida/intercambio de datos:
 - Google es el principal protagonista de este uso, pues en combinación con las etiquetas RFID, utilidades como marcar dónde estamos, recibir información de un evento o establecimiento son inmediatas.
- Pago con el teléfono móvil:
 - La comodidad de uso y gasto asociado a nuestra factura o una cuenta de banco.
- Sobre la implantación de la tecnología en dispositivos móviles ...

NFC en APPInventor

Componente NearField



ai2.appinventor.mit.edu/



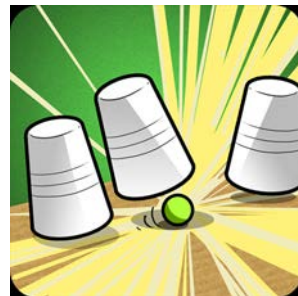
- Las aplicaciones creadas con el componente NFC no detectarán etiquetas NFC mientras que en el modo de desarrollo.
- Para probar la aplicación, hay que descargar el APK a su teléfono.
- El componente NearField sólo funciona en la pantalla Screen1

Shell Game

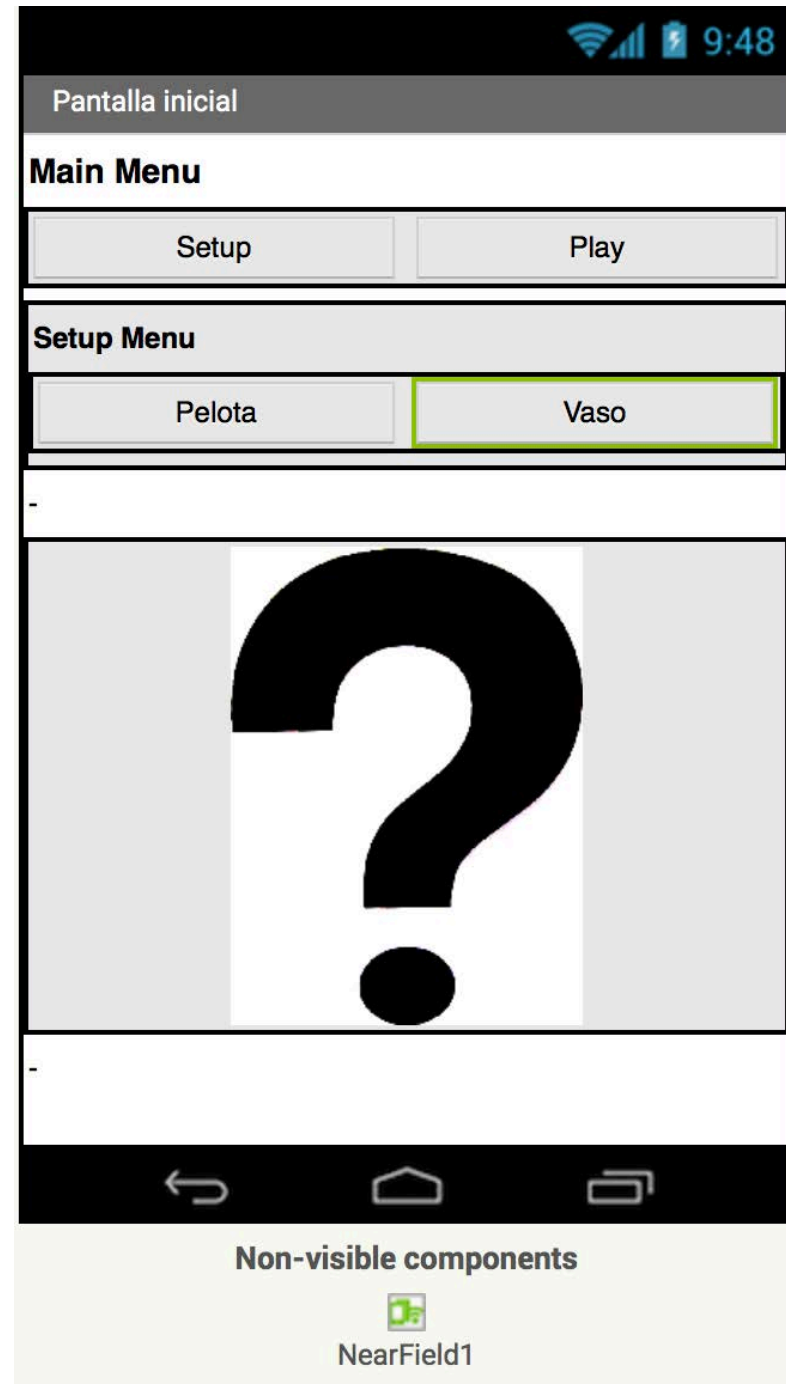
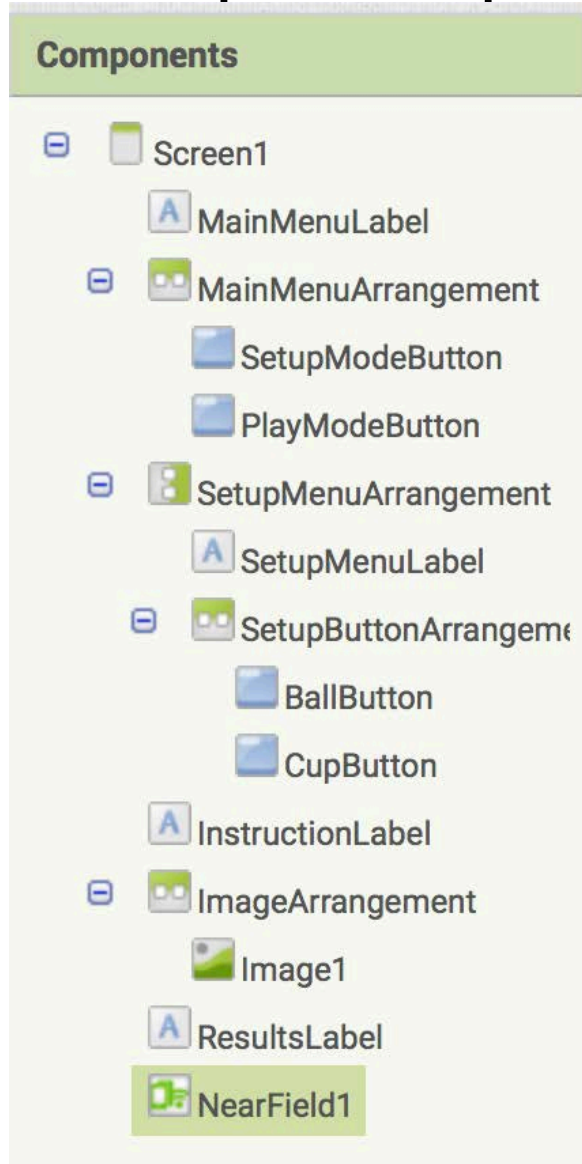
¿Dónde está la bolita?



- Objetivo
 - Construir una aplicación que simula un juego consistente en adivinar en cuál de tres cubiletes se esconde un objeto
 - En las etiquetas NFC se escriben tres imágenes (mensajes) en el modo de configuración.
 - En el modo juego se leen, intentando adivinar donde está el objeto.



Pantalla principal: Screen1



Recursos Multimedia

(cambiar por otras imágenes)



Media

- icon.png
- pingpongball.jpg
- questionmark.png
- redcup.jpg

Upload File...



Comportamiento: Inicialización de variables

initialize global SetupMode to true

initialize global NFCBallText to "ball"

initialize global SetupInstruction2Text to "Ponga el móvil sobre una de las etiquetas"

initialize global SetupInstruction1Text to "Elija Pelota o Vaso para empezar"

initialize global SetupInstruction3Text to "Pulse de los botones arriba para configurar el contenido de la tarjeta"

initialize global QuestionPic to "questionmark.png"

initialize global CupPic to "redcup.jpg"

initialize global PlayInstructionText to "Adivina que etiqueta esconde la pelota poniendo el teléfono encima de una de ellas"

initialize global PlayResultErrorText to "Lo siento, no reconozco la etiqueta que se ha escaneado , esta etiqueta dice :"

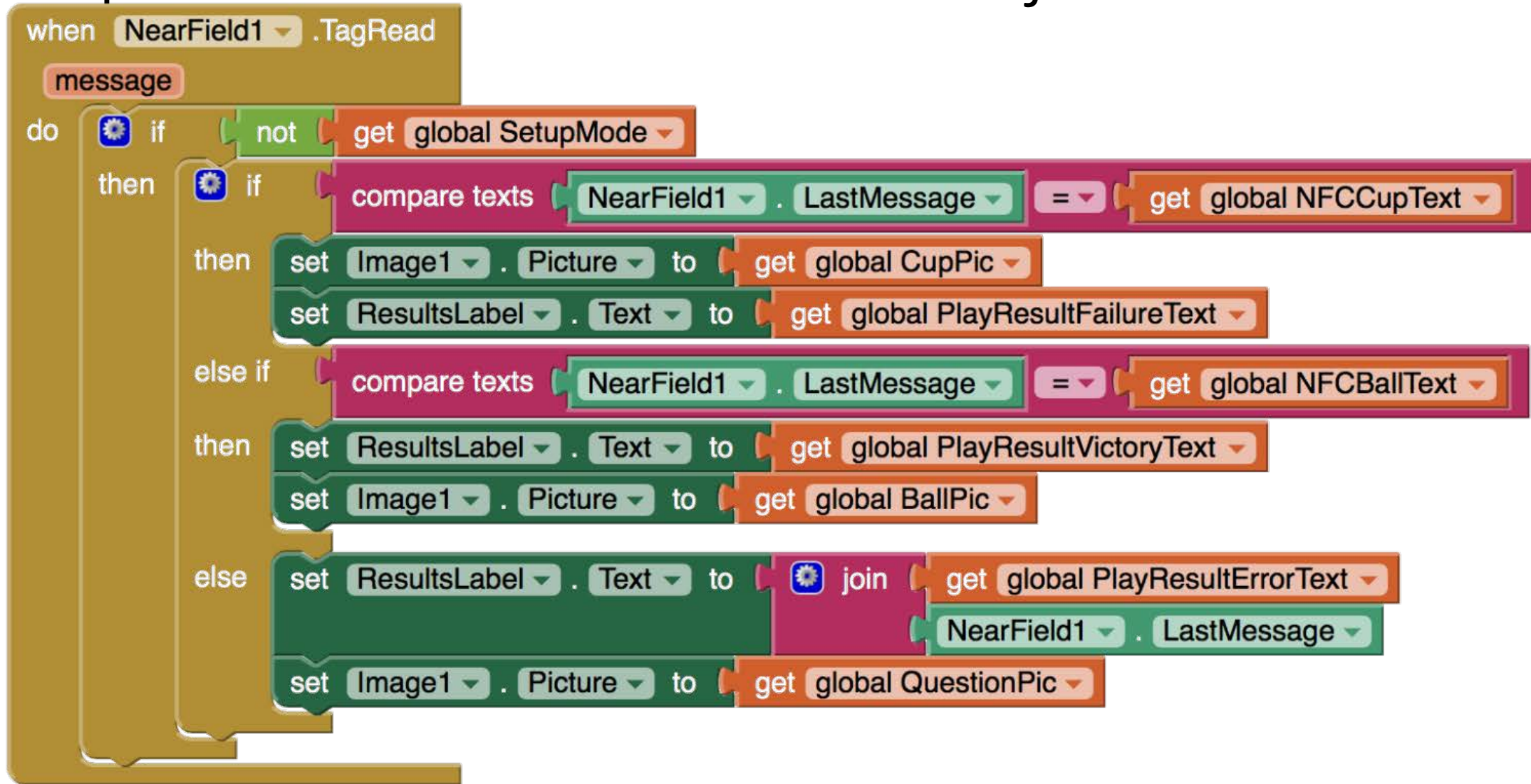
initialize global NFCCupText to "cup"

initialize global BallPic to "pingpongball.jpg"

initialize global PlayResultVictoryText to "Enhorabuena! Encontraste la bola!!"

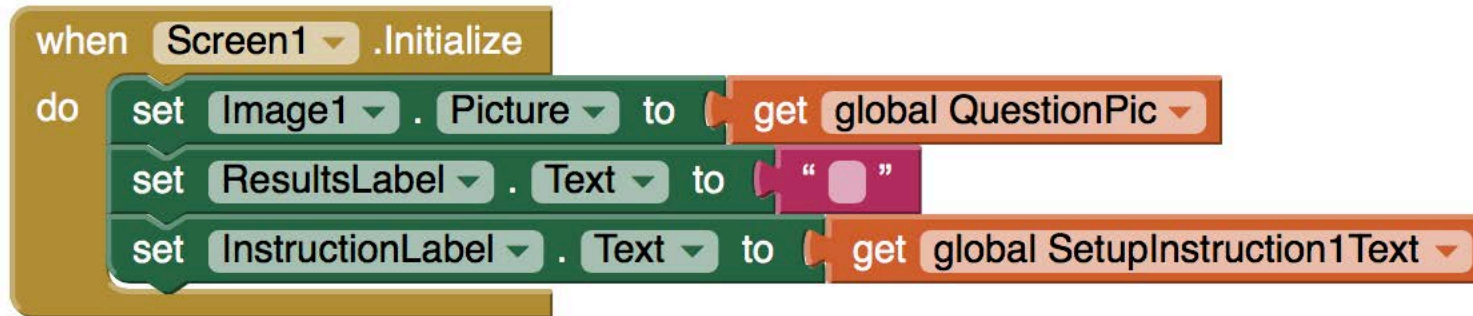
initialize global PlayResultFailureText to "Lo siento, inténtalo de nuevo!"

Comportamiento: Lectura de tarjeta



Comportamiento: Inicialización

```
when Screen1 .Initialize
do
  set Image1 . Picture to get global QuestionPic
  set ResultsLabel . Text to ""
  set InstructionLabel . Text to get global SetupInstruction1Text
```

A Scratch code block for screen initialization. It starts with a 'when Screen1 .Initialize' block. Inside a 'do' loop, there are three 'set' blocks: 'set Image1 . Picture to get global QuestionPic', 'set ResultsLabel . Text to ""', and 'set InstructionLabel . Text to get global SetupInstruction1Text'.

Comportamiento: Modos

```
when SetupModeButton .Click
do
  set Image1 . Picture to get global QuestionPic
  set ResultsLabel . Text to " "
  set InstructionLabel . Text to get global SetupInstruction1Text
  set SetupMenuArrangement . Visible to true
  set global SetupMode to true
```

```
when PlayModeButton .Click
do
  set Image1 . Picture to get global QuestionPic
  set ResultsLabel . Text to " "
  set InstructionLabel . Text to get global PlayInstructionText
  set SetupMenuArrangement . Visible to false
  set global SetupMode to false
  set NearField1 . ReadMode to true
```

Comportamiento: Configuración

when BallButton .Click

do

- set InstructionLabel . Text to get global SetupInstruction2Text
- set Image1 . Picture to get global BallPic
- set NearField1 . ReadMode to false
- set NearField1 . TextToWrite to get global NFCBallText

when CupButton .Click

do

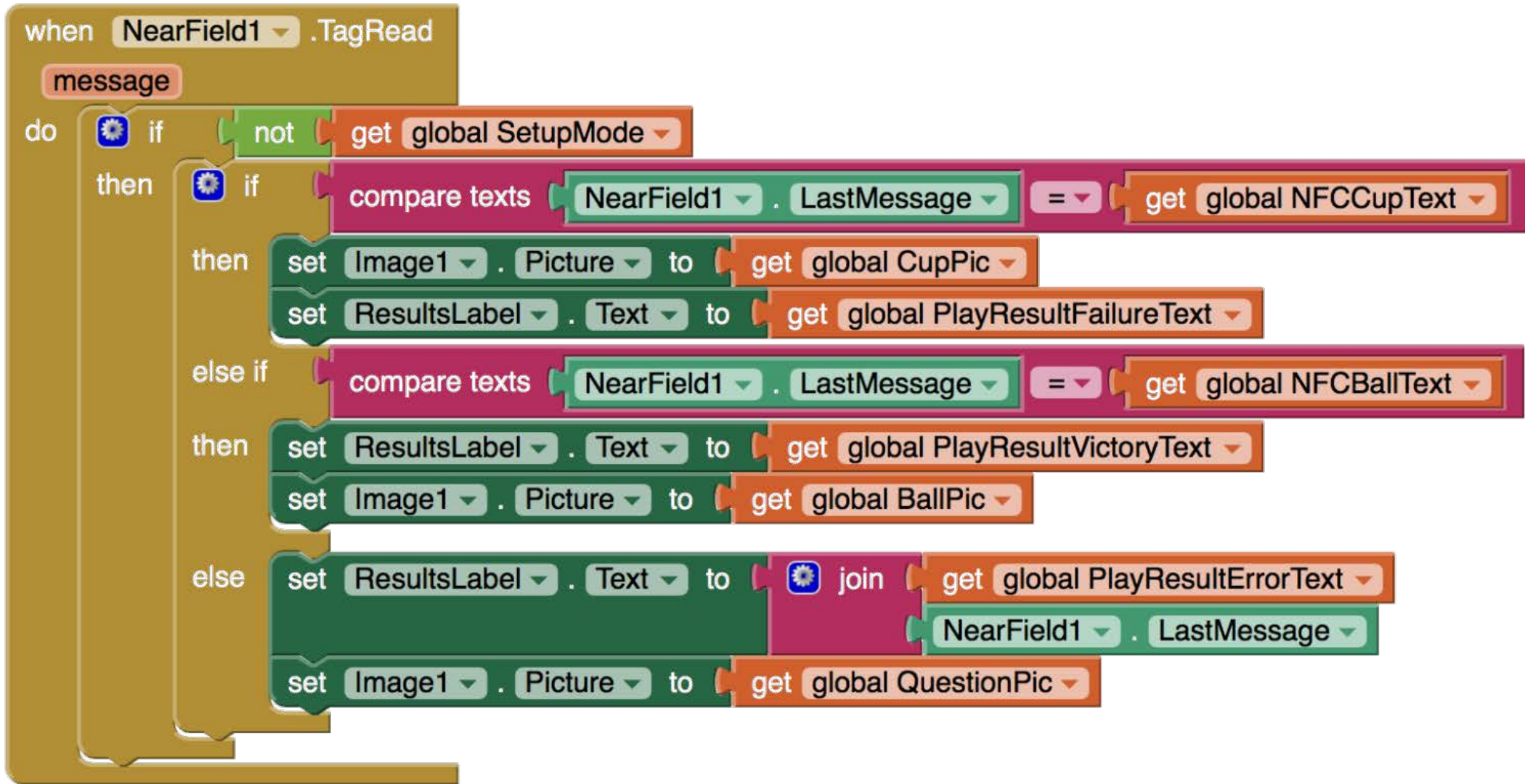
- set InstructionLabel . Text to get global SetupInstruction2Text
- set Image1 . Picture to get global CupPic
- set NearField1 . ReadMode to false
- set NearField1 . TextToWrite to get global NFCCupText

when NearField1 .TagWritten

do

- set Image1 . Picture to get global QuestionPic
- set InstructionLabel . Text to get global SetupInstruction3Text
- set NearField1 . ReadMode to true

Comportamiento: Modo Juego





Comunicación via Bluetooth | 2.0

Componentes Bluetooth en AppInventor

Connectivity

- ⚡ ActivityStarter
- 📶 BluetoothClient
- 📶 BluetoothServer
- 🌐 Web

Properties

BluetoothServer1

CharacterEncoding
UTF-8

DelimiterByte
0

HighByteFirst

Secure

Properties

BluetoothClient1

CharacterEncoding
UTF-8

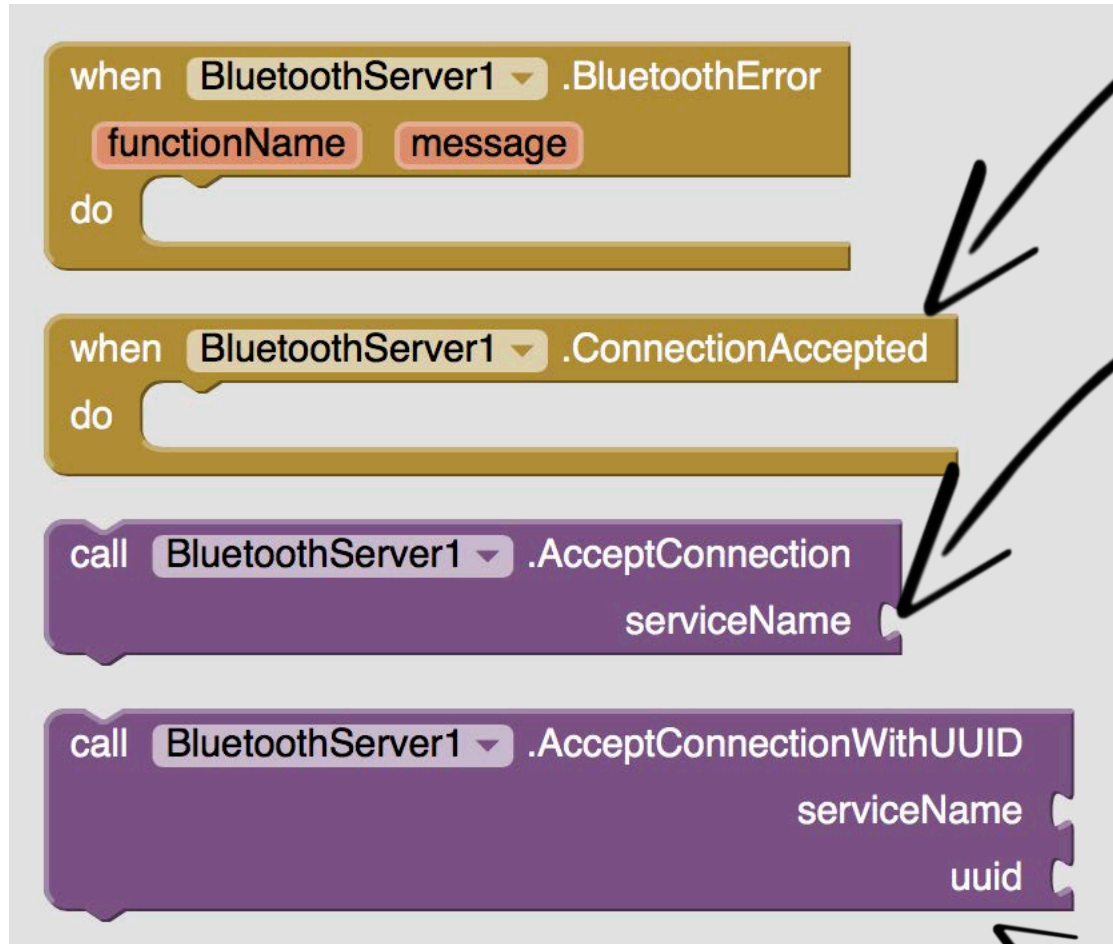
DelimiterByte
0

HighByteFirst

Secure

Hay más propiedades que se pueden consultar y modificar desde el comportamiento

BluetoothServer: Comportamiento



Evento que indica que se ha aceptado una conexión

Acepta una conexión Serial Port Profile (SPP). El parámetro es el nombre del servicio publicado.

Acepta una conexión Serial Port Profile (SPP) con un UUID específico

BluetoothServer: Comportamiento

The screenshot displays the class `BluetoothServer1` with the following elements:

- `BluetoothServer1` . Available
- `BluetoothServer1` . CharacterEncoding
- set `BluetoothServer1` . CharacterEncoding to
- `BluetoothServer1` . DelimiterByte
- set `BluetoothServer1` . DelimiterByte to
- `BluetoothServer1` . Enabled
- `BluetoothServer1` . HighByteFirst
- set `BluetoothServer1` . HighByteFirst to
- `BluetoothServer1` . IsAccepting
- `BluetoothServer1` . IsConnected
- `BluetoothServer1` . Secure
- set `BluetoothServer1` . Secure to
- Show Warnings
- `BluetoothServer1`

call `BluetoothServer1` .BytesAvailableToReceive

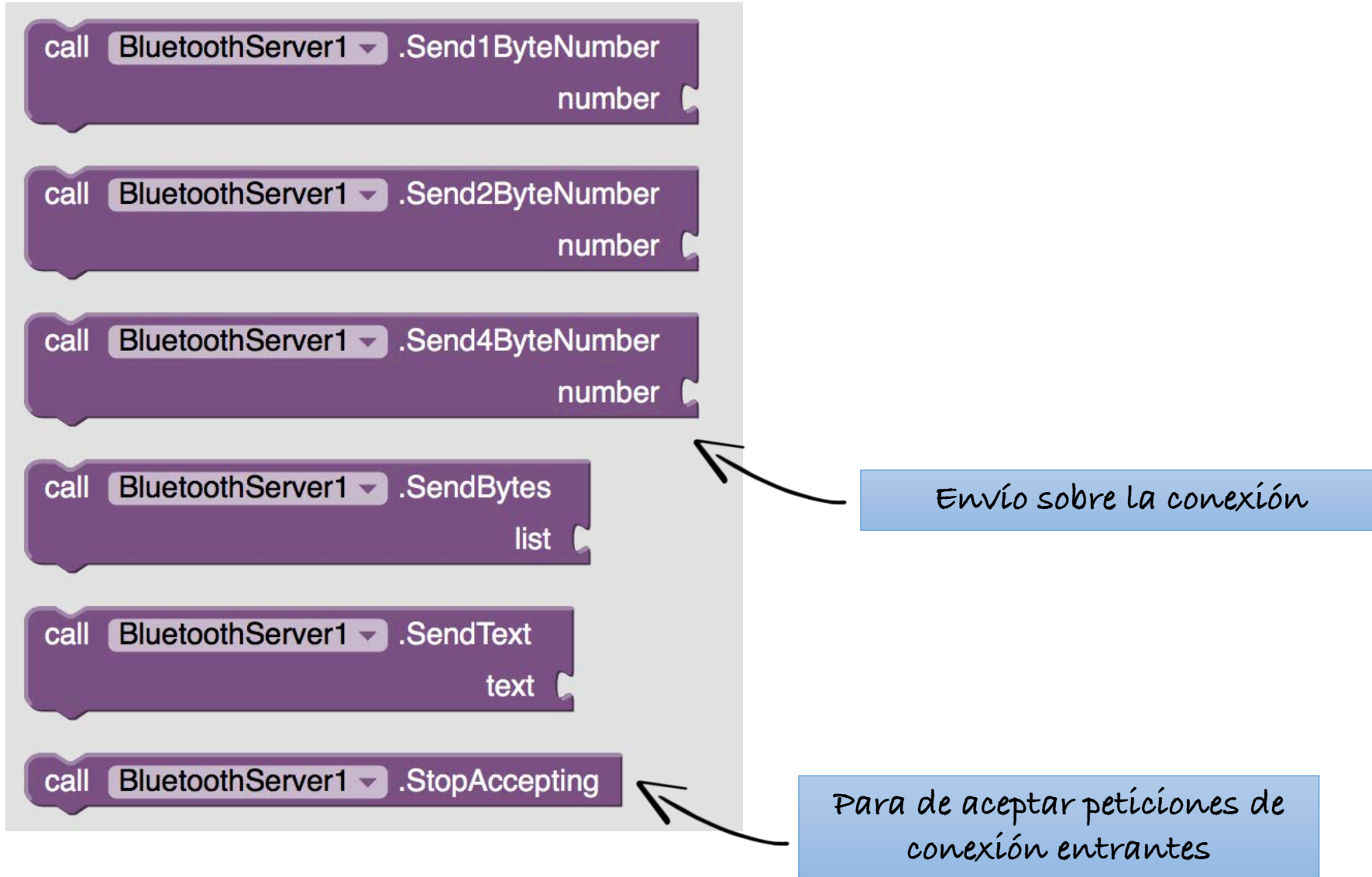
call `BluetoothServer1` .Disconnect

Devuelve un nº estimado de bytes que puede recibir sin bloquear

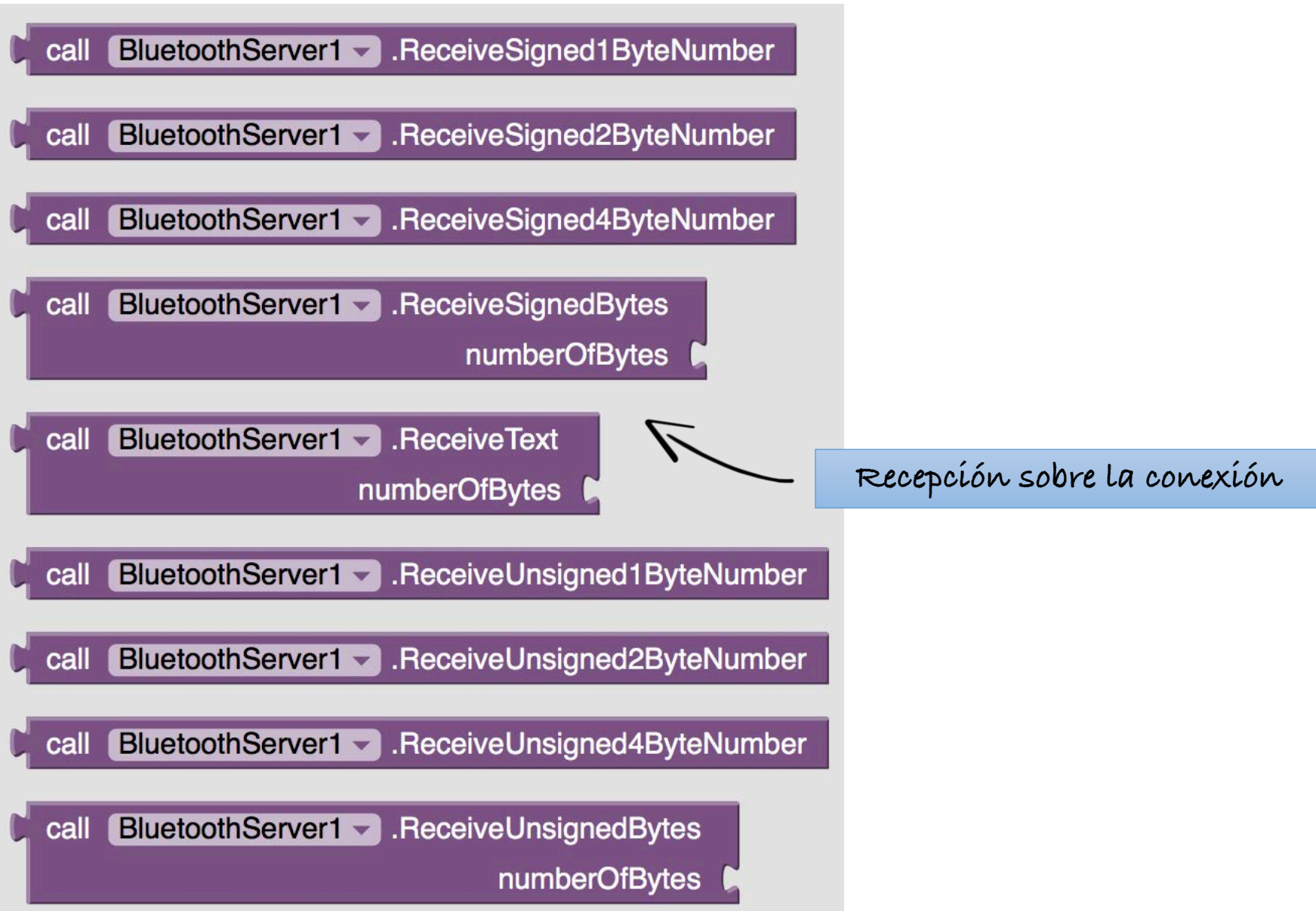
Se desconecta del dispositivo Bluetooth conectado

Propiedades que se pueden consultar y modificar desde el comportamiento

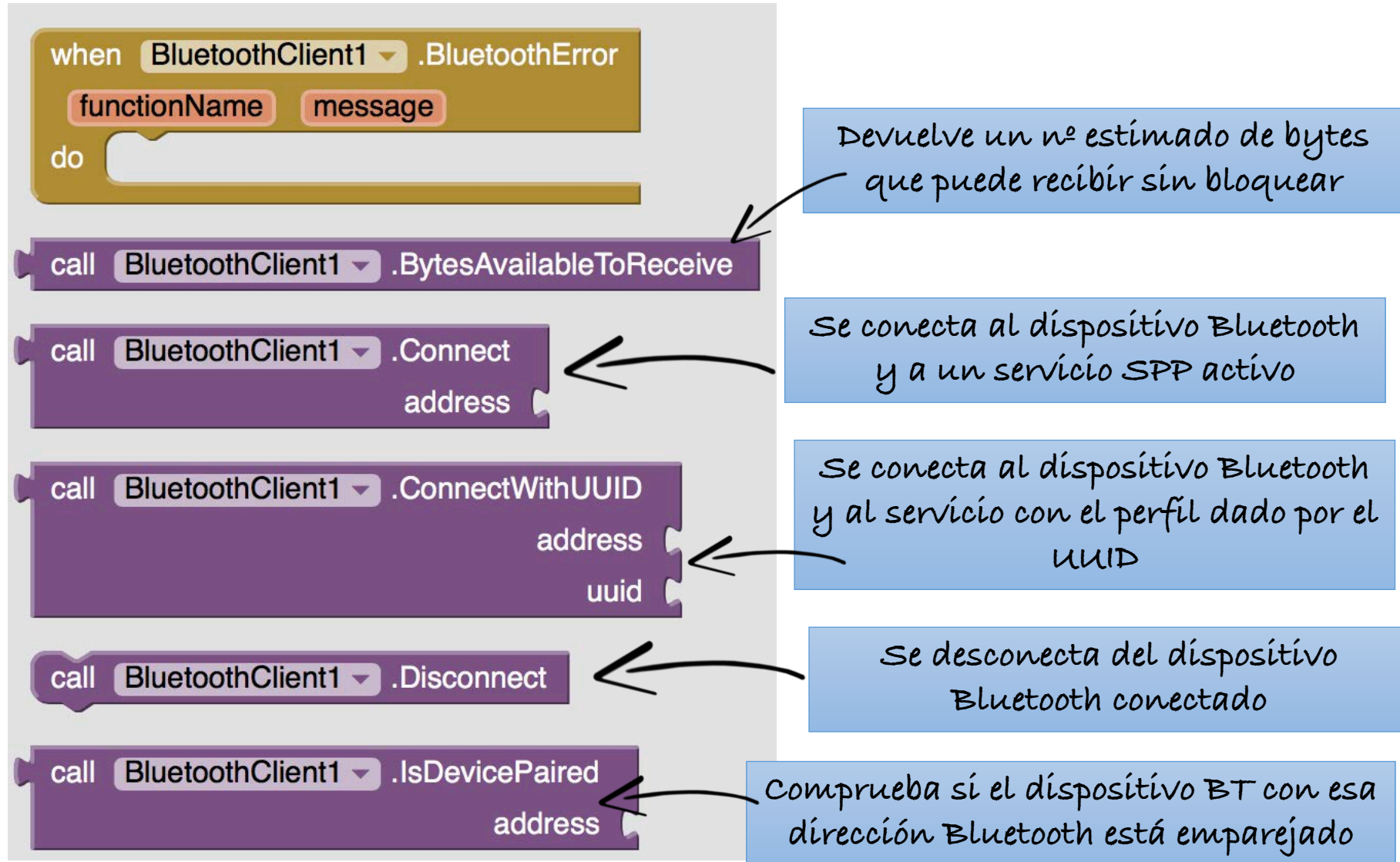
BluetoothServer: Comportamiento



BluetoothServer: Comportamiento



BluetoothClient: Comportamiento



BluetoothClient: Comportamiento

BluetoothClient1 . AddressesAndNames

BluetoothClient1 . Available

BluetoothClient1 . CharacterEncoding

set BluetoothClient1 . CharacterEncoding to

BluetoothClient1 . DelimiterByte

set BluetoothClient1 . DelimiterByte to

BluetoothClient1 . Enabled

BluetoothClient1 . HighByteFirst

set BluetoothClient1 . HighByteFirst to

BluetoothClient1 . IsConnected

BluetoothClient1 . Secure

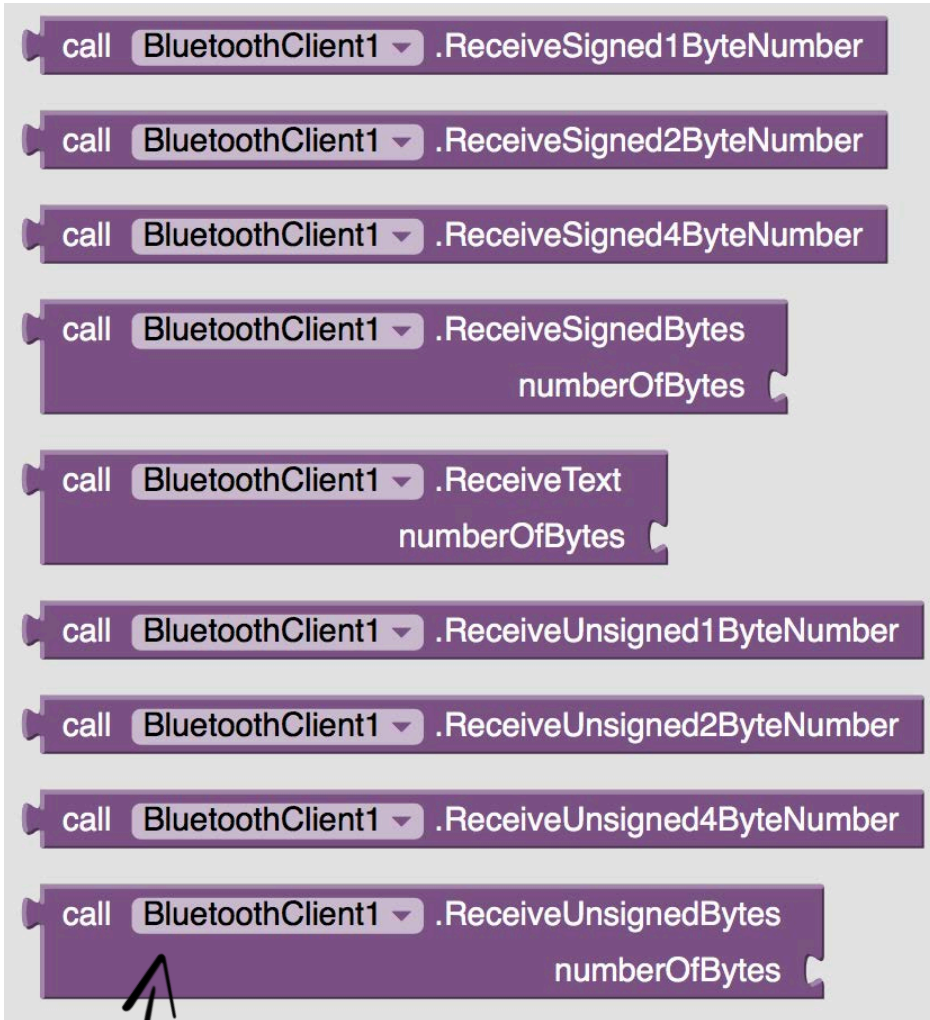
set BluetoothClient1 . Secure to

Show Warnings

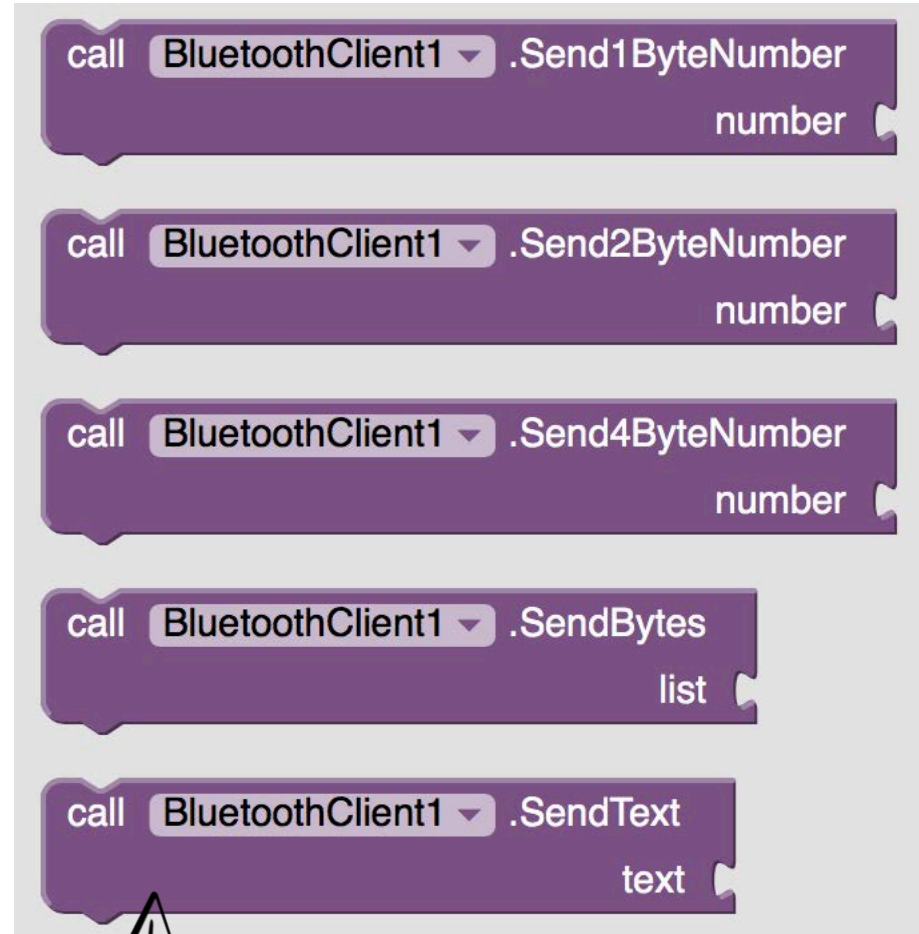
BluetoothClient1

Propiedades que se pueden consultar y modificar desde el comportamiento

BluetoothClient: Comportamiento

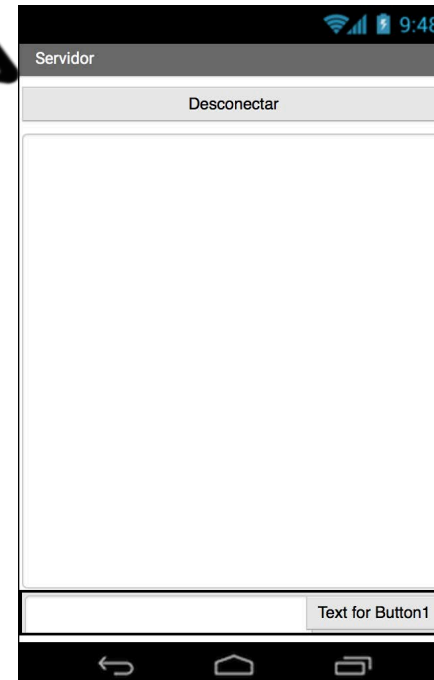
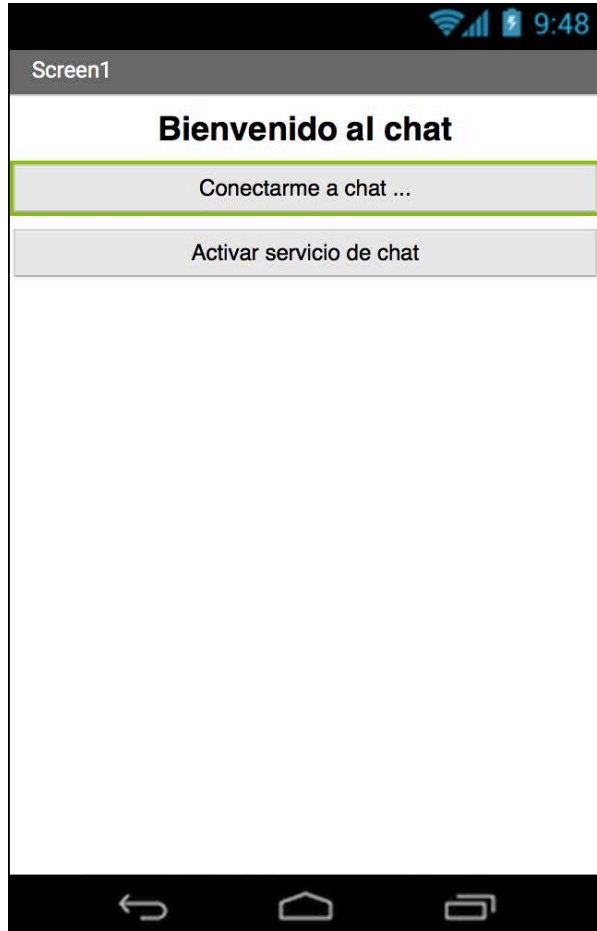


Recepción sobre la conexión



Envío sobre la conexión

Chat Bluetooth



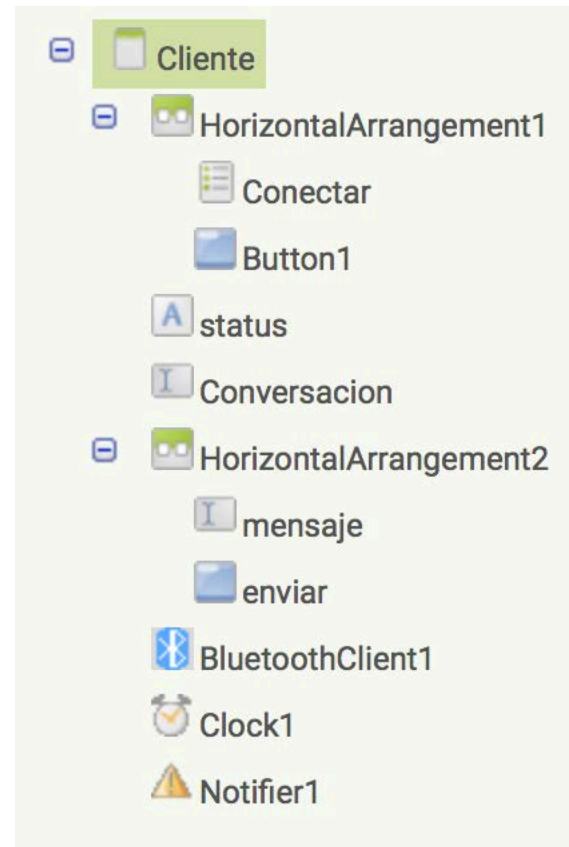
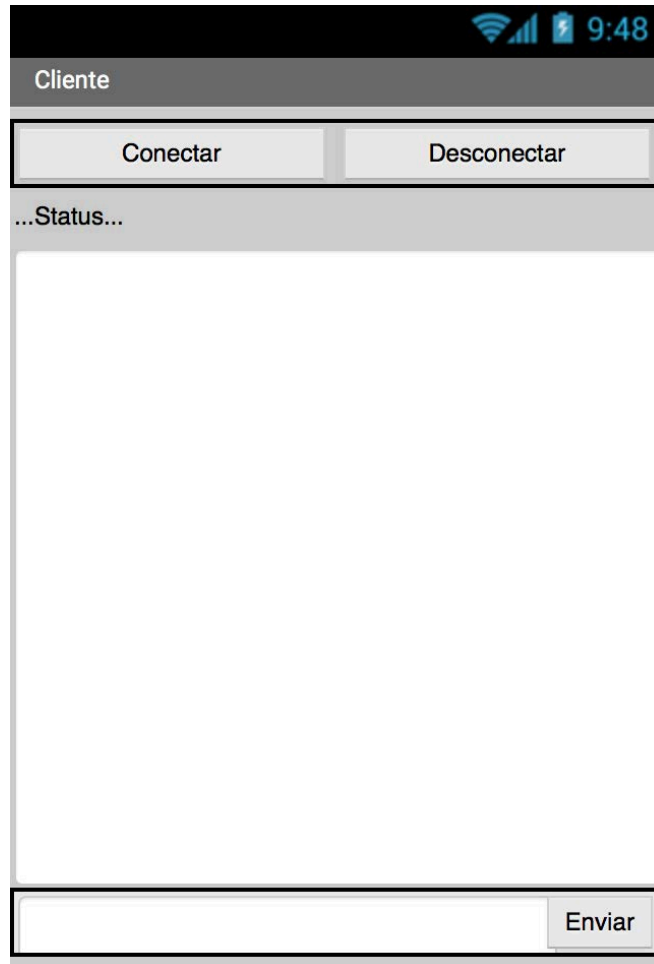
Screen1

```
when btConectar ▾ .Click  
do open another screen screenName " Cliente "
```

```
when btActivar ▾ .Click  
do open another screen screenName " Servidor "
```

```
when Screen1 ▾ .BackPressed  
do close application
```

Ejercicio práctica: Cliente



```
when ListPicker1 .BeforePicking
do set ListPicker1 . Elements to BluetoothClient1 . AddressesAndNames
```

```
when ListPicker1 .AfterPicking
do if call BluetoothClient1 .Connect
    address ListPicker1 . Selection
    then set lblStatus . Text to " Conectado "
```

Antes

```
when btDesconectar .Click
do if BluetoothClient1 . IsConnected
    then call BluetoothClient1 .Disconnect
        close screen
```

```
when Cliente .BackPressed
do close screen
```

Después

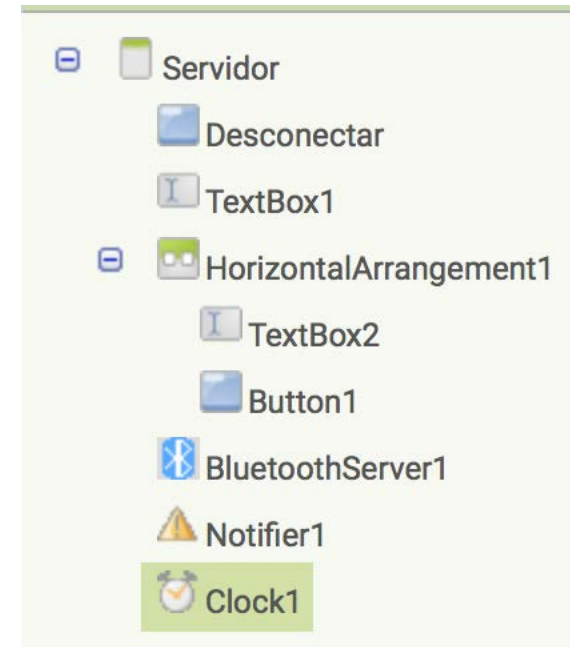
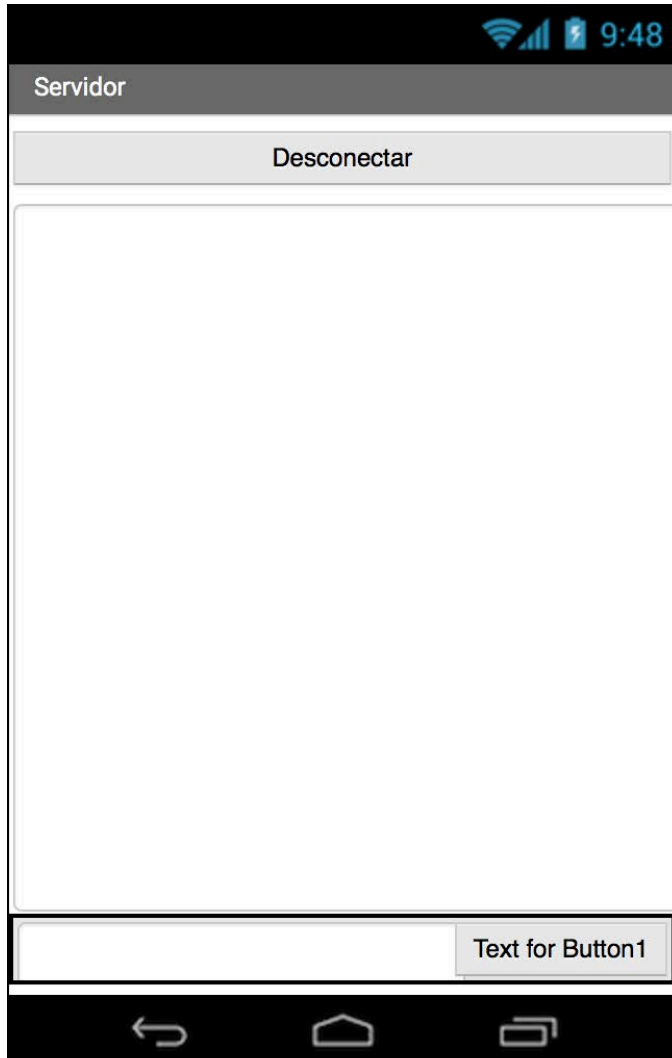
Envío

Recepción

```
when btEnviar .Click
do
  if BluetoothClient1 .IsConnected
  then
    call BluetoothClient1 .SendText
      text join CampoDeTexto1 .Text
      "\n"
    set TextBox1 .Text to join join TextBox1 .Text
      "\n"
      CampoDeTexto1 .Text
    set lblStatus .Text to "Mensaje enviado"
    set CampoDeTexto1 .Text to ""
```

```
when Clock1 .Timer
do
  if BluetoothClient1 .IsConnected
  then
    if call BluetoothClient1 .BytesAvailableToReceive > 0
    then
      set TextBox1 .Text to join TextBox1 .Text
        "\n"
        call BluetoothClient1 .ReceiveText
          numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
      set lblStatus .Text to "Mensaje recibido"
```

Ejercicio práctica: Servidor



Antes

```
when Servidor .Initialize
do
  call ServidorBluetooth1 .AcceptConnection
  serviceName "michat"
```

Después

```
when btDesconectar .Click
do
  call ServidorBluetooth1 .StopAccepting
  call ServidorBluetooth1 .Disconnect
  close screen
```


Envío

```
when btEnviar .Click
do
  if BluetoothClient1 .IsConnected
  then
    call BluetoothClient1 .SendText
      text join CampoDeTexto1 . Text
            "\n"
    set TextBox1 . Text to join join TextBox1 . Text
                              "\n"
                              CampoDeTexto1 . Text
    set lblStatus . Text to "Mensaje enviado"
    set CampoDeTexto1 . Text to ""
```

Recepción

```
when Clock1 .Timer
do
  if BluetoothClient1 .IsConnected
  then
    if call BluetoothClient1 .BytesAvailableToReceive > 0
    then
      set TextBox1 . Text to join TextBox1 . Text
                                "\n"
                                call BluetoothClient1 .ReceiveText
                                  numberOfBytes call BluetoothClient1 .BytesAvailableToReceive
      set lblStatus . Text to "Mensaje recibido"
```

BluetoothServer1

Comunicación con Arduino via Bluetooth

APP: Encender un LED desde el móvil

- Crearemos una app que con un botón controle un LED
 - Botones para conectar y desconectar el móvil/tablet de Arduino
 - Un botón para encender y otro para apagar el LED



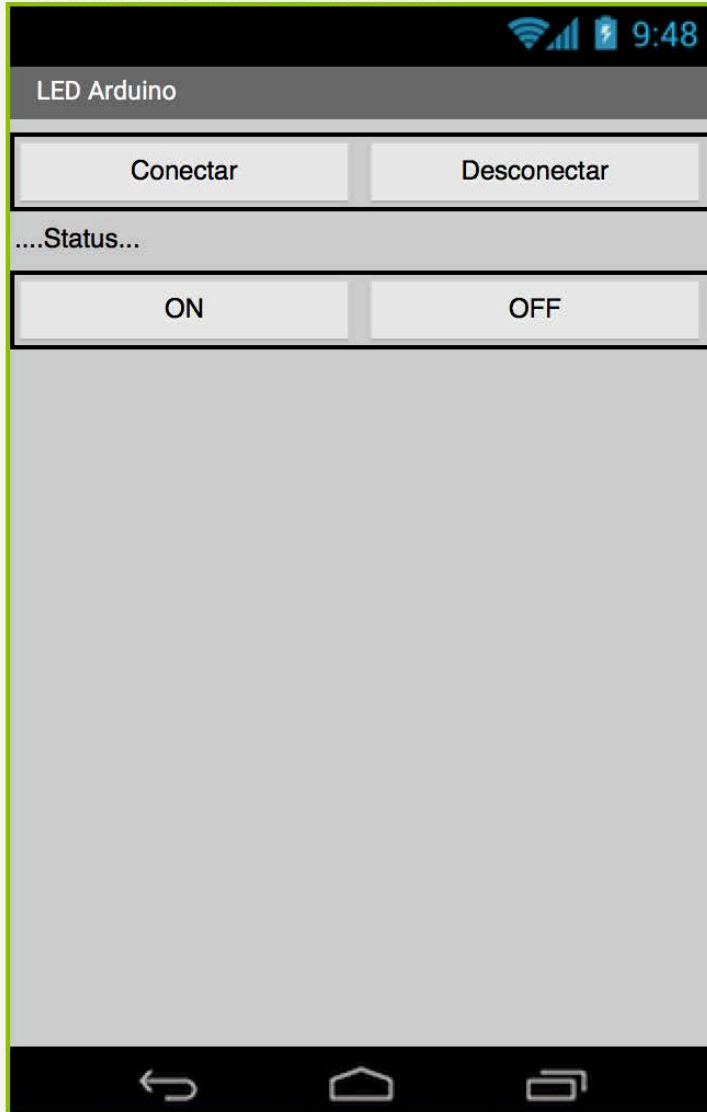
Código de Arduino

```
char led; // donde leemos orden
void loop() {
  if (BT.available())
  {
    led=(BT.read());
    // Si recibe el dato "a" el led se
    enciende

    if (led=='e')
    {
      digitalWrite(13, HIGH);
      BT.println("LED on");
    }
    if (led=='a')
    // Si recibe el dato "b" el led se apaga
    {
      digitalWrite(13, LOW);
    }
  }
}
```

Display hidden components in Viewer

Check to see Preview on Tablet size.



Non-visible components

 
BluetoothClient1 Aviso

A screenshot of the Design Studio component palette. The palette is organized into a tree structure. At the top is 'Screen1'. Underneath it are two 'HorizontalArrangement' containers. The first 'HorizontalArrangement1' contains 'Conectar' and 'Desconectar' buttons, and an 'Estado' label. The second 'HorizontalArrangement2' contains 'ON' and 'OFF' buttons. Below these are a 'BluetoothClient1' component and an 'Aviso' warning icon. At the bottom of the palette are 'Rename' and 'Delete' buttons. A 'Media' section is visible at the very bottom with an 'Upload File ...' button.

Diseño de la app

Comportamiento

Deshabilitar botones
de encendido,
apagado y
desconectar

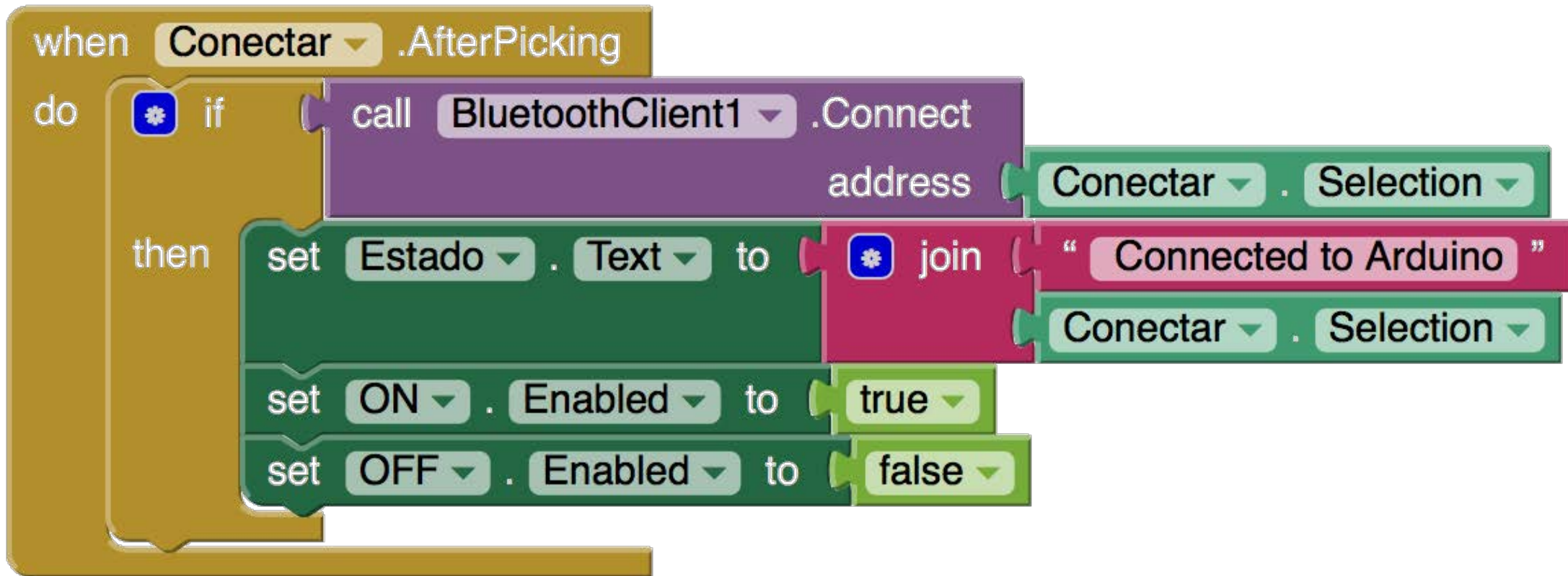
```
when Screen1.Initialize
do
  set ON.Enabled to false
  set OFF.Enabled to false
  set Desconectar.Enabled to false
```

Inicializar lista de
dispositivos y
avisar si BT no está
activado en el móvil

```
when Conectar.BeforePicking
do
  if BluetoothClient1.Available
  then
    set Conectar.Elements to BluetoothClient1.AddressesAndNames
  else
    call Aviso.ShowAlert
    notice "Bluetooth no Activado!!"
```

Comportamiento

Cuando se seleccione un dispositivo de la lista nos conectamos



Comportamiento

Comportamiento de los botones ON, OFF, y Desconectar

```
when OFF .Click
do
  set OFF . Enabled to false
  call BluetoothClient1 .SendText
  text "a"
  set ON . Enabled to true
```

```
when ON .Click
do
  set ON . Enabled to false
  call BluetoothClient1 .SendText
  text "e"
  set OFF . Enabled to true
```

```
when Desconectar .Click
do
  call BluetoothClient1 .Disconnect
  set Estado . Text to "Desconectado"
```

! 0 ! 0

Show Warnings



¡Gracias por vuestra participación!

