

Aplicaciones Android con App Inventor



Redes Inalámbricas

¿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? (2)

- Con esta herramienta, se puede crear una app en tres pasos:



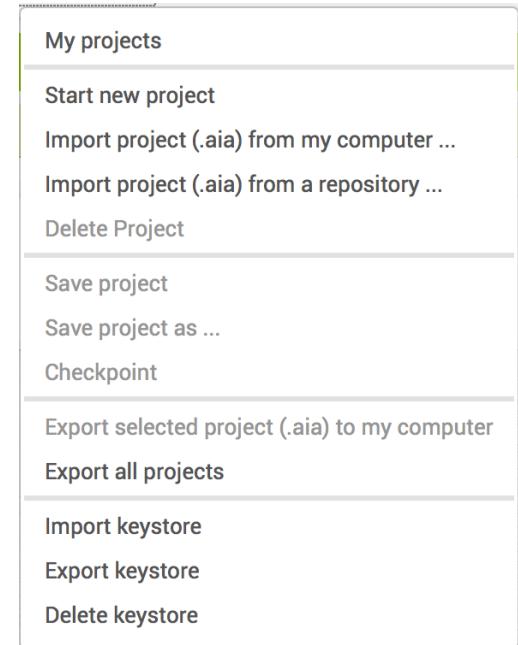
Comenzar a trabajar con App Inventor

- 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.html>
 - Una vez instalado, ejecutar el programa aiStarter



App inventor 2

- 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



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)

MIT App Inventor 2 Beta

Project ▾ Connect ▾ Build ▾ Help ▾

My Projects Guide Report an Issue manuela.ruiz.montiel@gmail.com ▾

DondeEstaMiCoche Screen1 ▾ Add Screen ... Remove Screen Designer Blocks

Palette

User Interface

- Button
- CheckBox
- Clock
- Image
- Label
- ListPicker
- Notifier
- PasswordTextBox
- Slider
- TextBox
- WebViewer

Layout

Media

Drawing and Animation

Sensors

Social

Storage

Viewer

Display hidden components in Viewer

Screen1

Components

Screen1

Rename Delete

Properties

Screen1

AlignHorizontal Left

AlignVertical Top

BackgroundColor White

BackgroundImage None...

CloseScreenAnimation Default

Icon None...

OpenScreenAnimation Default

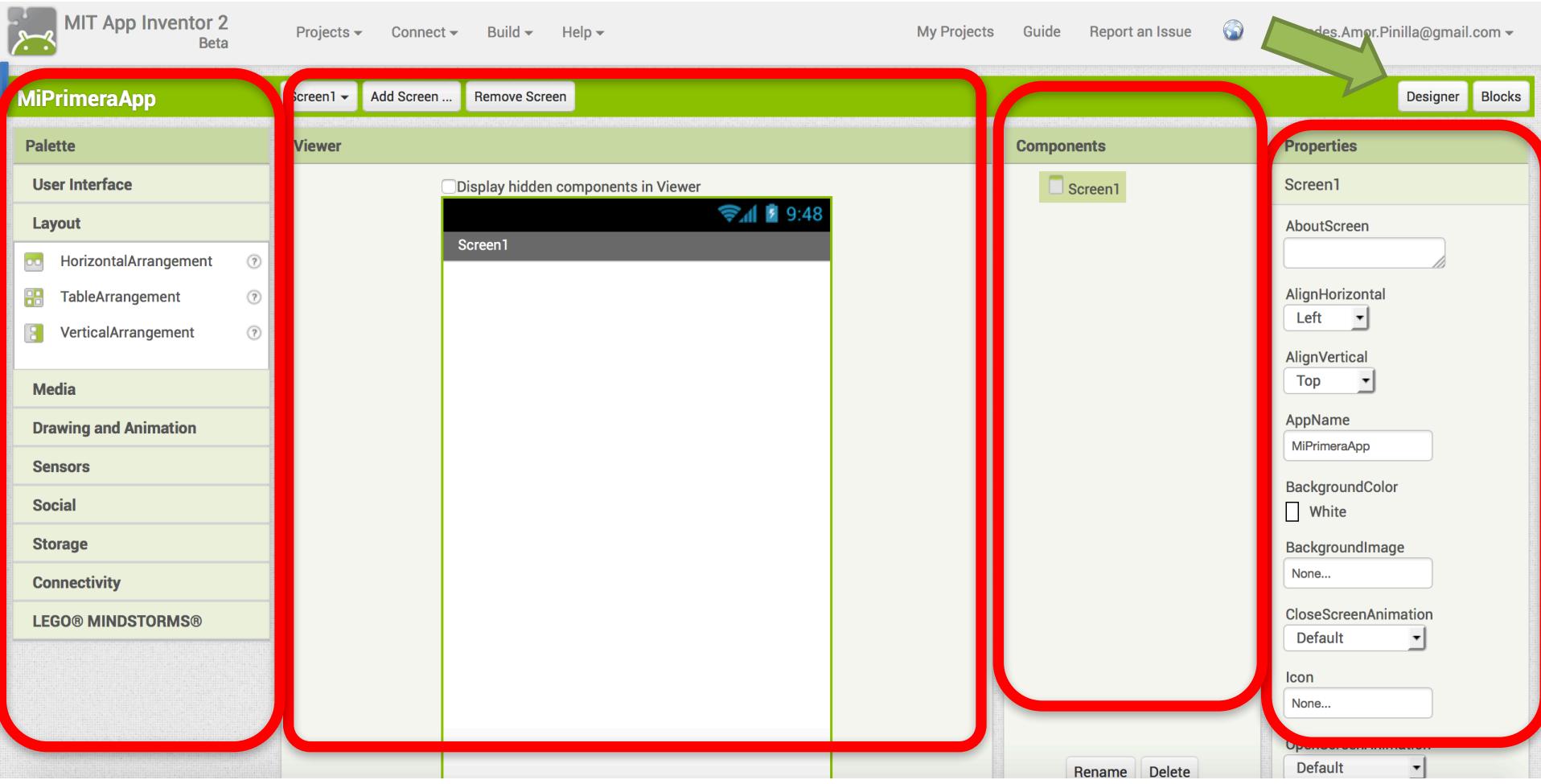
ScreenOrientation Unspecified

Scrollable

Title

Media

The screenshot shows the MIT App Inventor 2 Designer interface. The top navigation bar includes 'Project', 'Connect', 'Build', 'Help', 'My Projects', 'Guide', 'Report an Issue', and a user email. Below the navigation is a toolbar with tabs for 'Screen1', 'Add Screen ...', 'Remove Screen', 'Designer', and 'Blocks'. The main workspace is divided into four panels: 'Palette' (User Interface, Layout, Media, Drawing and Animation, Sensors, Social, Storage), 'Components' (listing 'Screen1'), 'Properties' (showing properties for 'Screen1' like AlignHorizontal, AlignVertical, BackgroundColor, etc.), and 'Viewer' (a preview window showing a mobile screen with the title 'Screen1'). The 'User Interface' section of the palette is expanded, listing various UI components like Button, CheckBox, Clock, Image, Label, etc.



Pantalla Designer (II)

- 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
- 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 (III)

- 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...
 - 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

Pantalla Blocks (I)

The screenshot shows the MIT App Inventor 2 Designer interface. At the top, there is a navigation bar with the MIT App Inventor 2 logo, the text "Beta", and links for "Project", "Connect", "Build", and "Help". To the right of the navigation bar are links for "My Projects", "Guide", "Report an Issue", and an email address "manuel.a.ruiz.montiel@gmail.com". Below the navigation bar, the project title "DondeEstaMiCoche" is displayed, along with buttons for "Screen1", "Add Screen ...", and "Remove Screen". The main workspace is titled "Viewer" and contains a large empty area. In the bottom left corner of the workspace, there are two small status indicators: a yellow warning icon with "0" and a red error icon with "0", followed by a "Show Warnings" button. On the far right edge of the workspace, there is a green trash can icon. On the left side of the interface, there is a sidebar titled "Blocks" which lists various block categories: Built-in (Control, Logic, Math, Text, Lists, Colors, Variables, Procedures), Screen1, and Any component. At the bottom left of the sidebar, there are "Rename" and "Delete" buttons. The bottom of the interface has a green bar labeled "Media".

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)

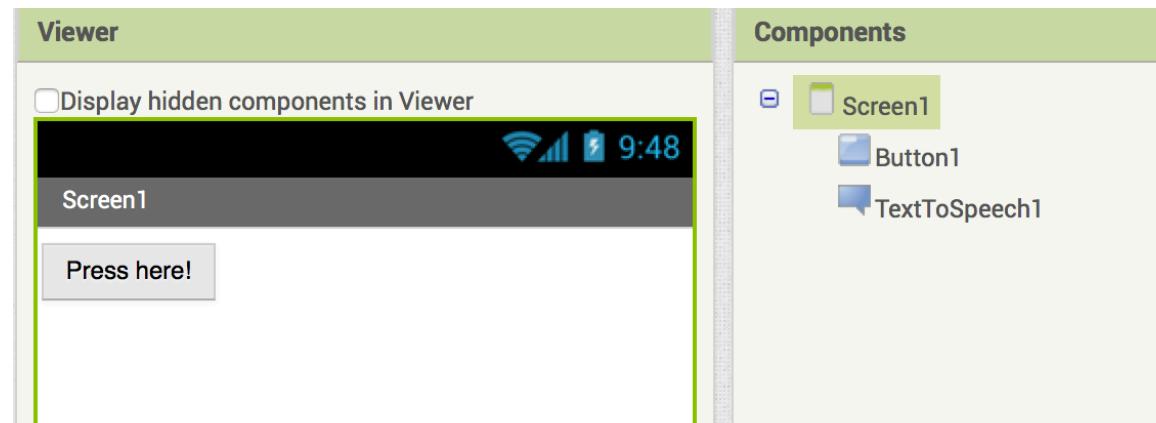
- Bloques asociados a nuestros componentes
 - Para cada componente, nos aparecerá una paleta con los posibles bloques asociados
 - Por ejemplo, para las etiquetas tenemos únicamente **getters** y **setters**
 - Para los botones, además de getters y setters tenemos bloques de control que **controlan eventos** (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

Primera APP

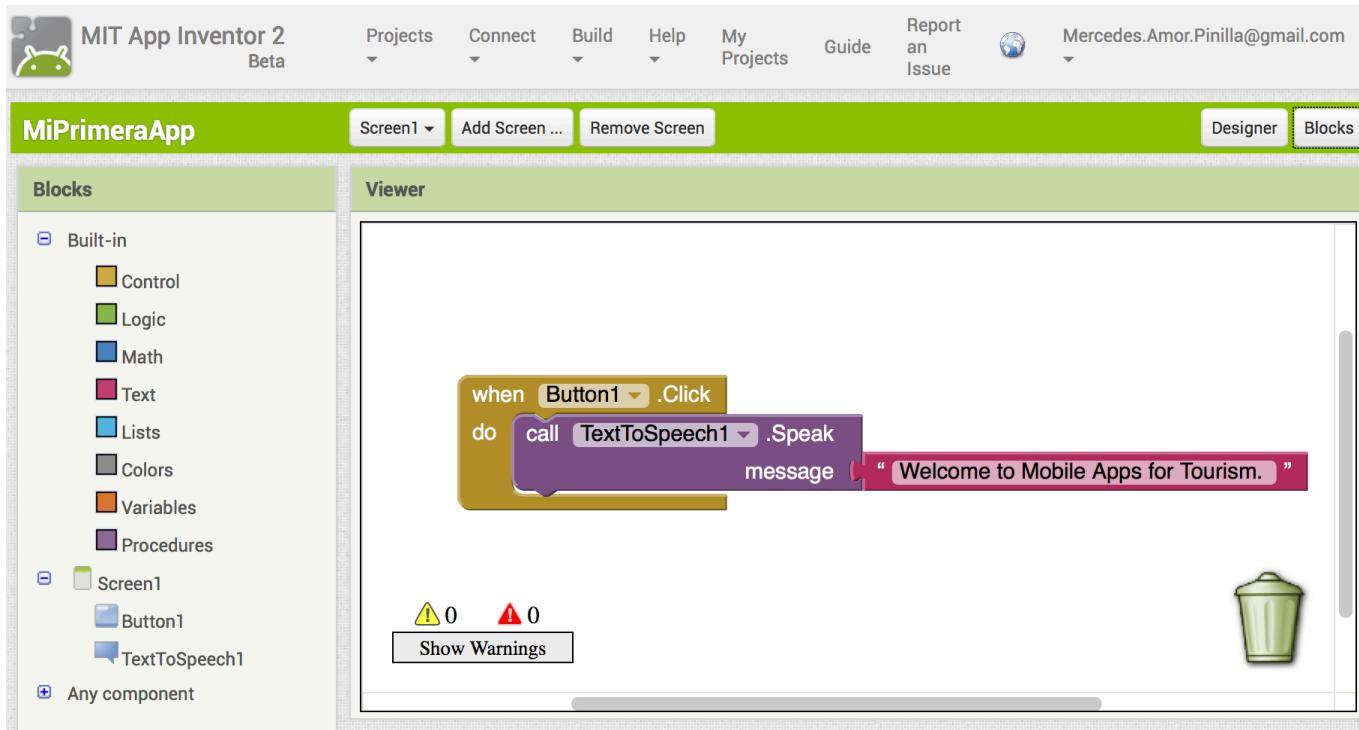
Text2Speech

1º añadimos componentes a la pantalla

- Un componente Button de UserInterface
- Un componente TextToSpeech de Media



Añadimos comportamiento ...



Y la probamos ...

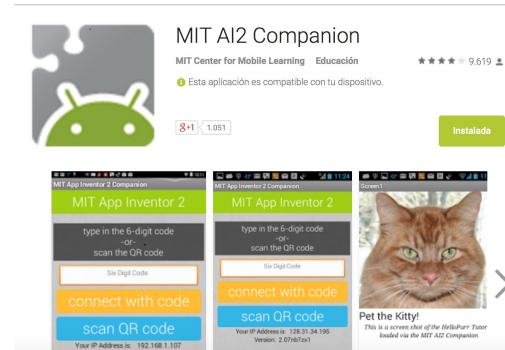
- ✓ En el emulador
- ✓ En un móvil
 - ✓ Instalar primer la app AI2 companion
 - ✓ <https://play.google.com/store/apps/details?id=edu.mit.appinventor.ai companion3&hl=es>
 - ✓ <http://appinventor.mit.edu/explore/ai2/setup-device-wifi.html>



Build your project on
your computer



Test it in real-time on
your device





Pequeñas apps para ...

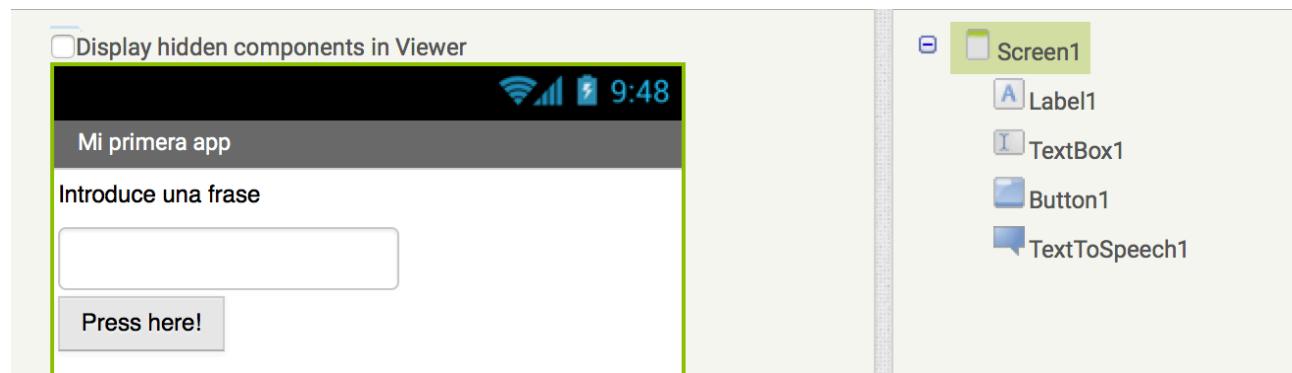
Ejemplos de AppInventor

Introducir un mensaje para

Que la aplicación lo reproduzca como un audio

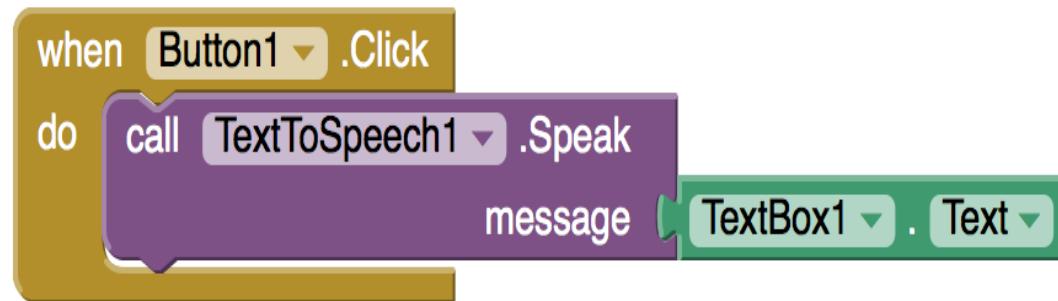
Tomamos como referencia nuestra primera app

Añadimos un componente Label y un componente TextBox en la pantalla de diseño



Text2Speech

Modificamos las propiedades para personalizar los componentes
Añadimos el siguiente comportamiento

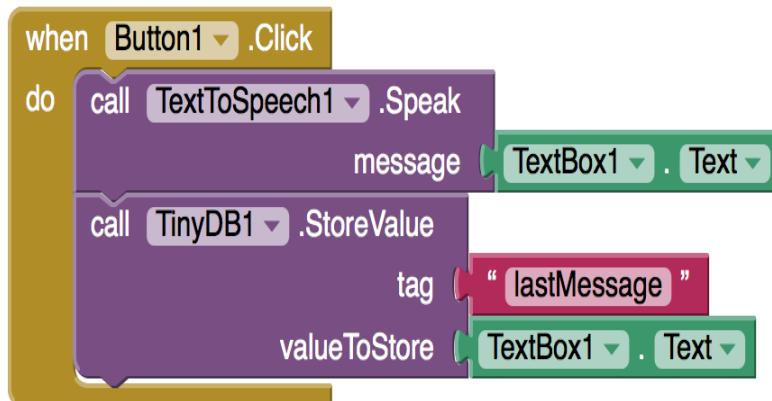


Que guarde el último mensaje escrito

Usamos el componente del menú Storage TinyDB

Añadimos un nuevo componente TinyDB (es no visible)

Añadimos el siguiente comportamiento

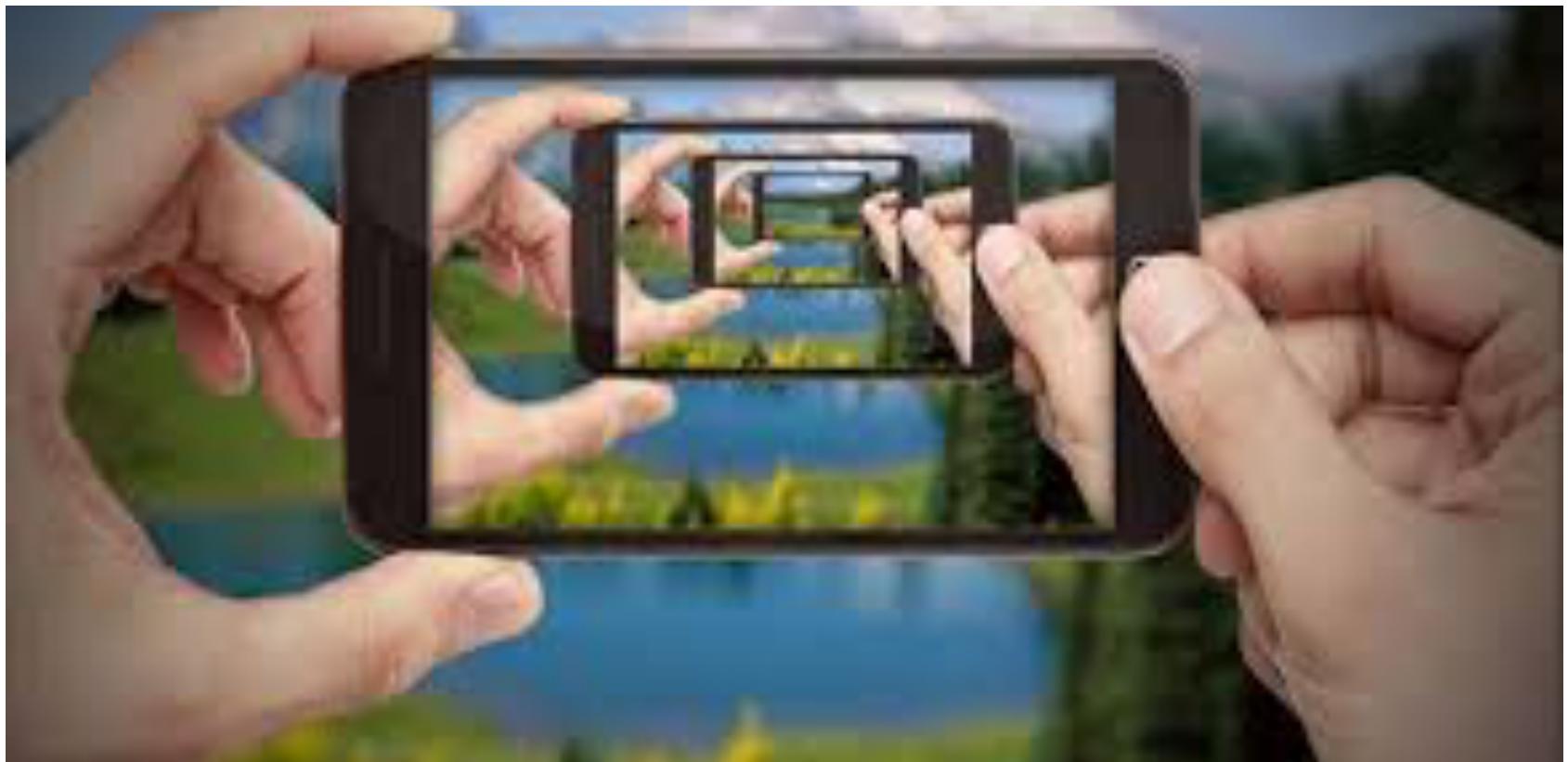


....queremos que al iniciar la aplicación

Aparezca la última frase escrita

Añadimos el siguiente comportamiento

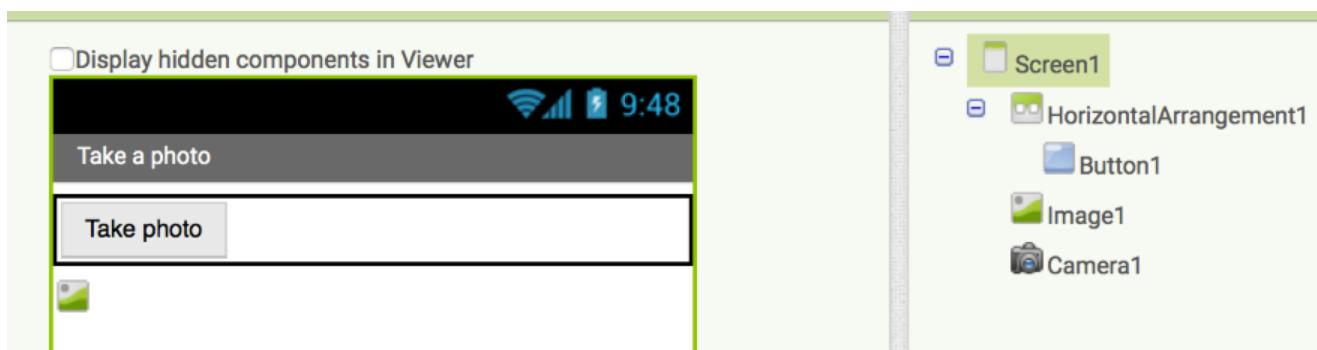




Captura de imagen y códigos QR

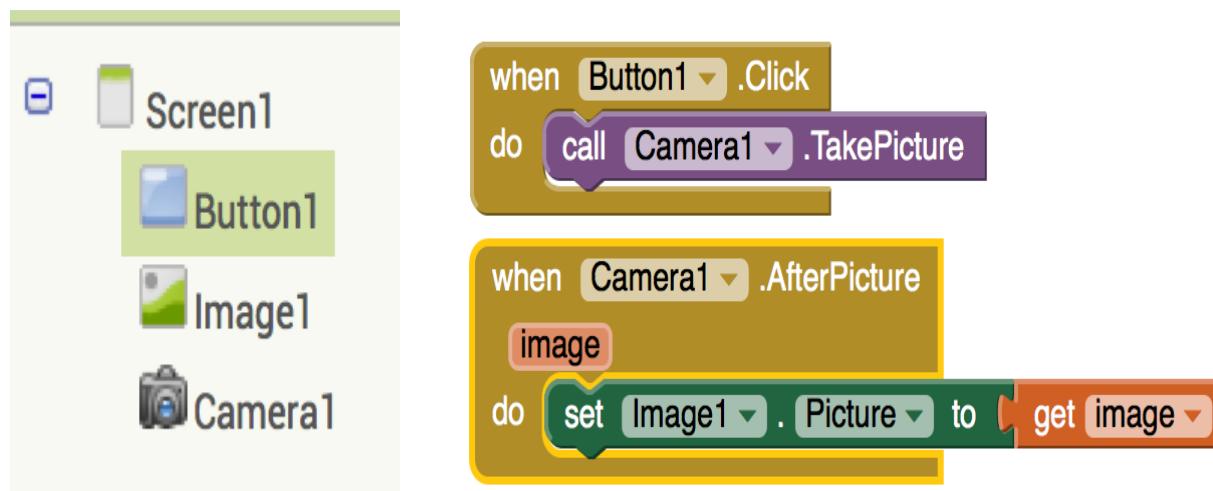
App que hace una foto y la muestra a continuación

Necesitamos los componentes: Botón, Camera e ImageViewer



App que hace una foto ...

Y la muestra a continuación.



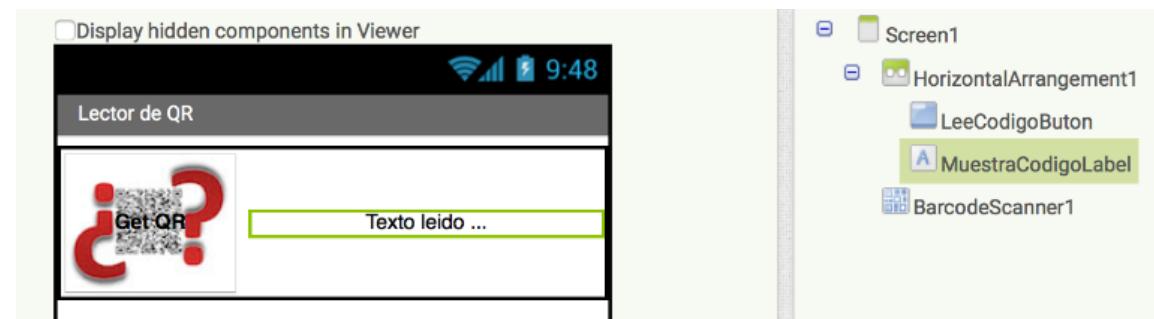
Definimos comportamiento

The image shows a Scratch script consisting of two events:

- when Button1 .Click**:
do **call Camera1 .TakePicture**
- when Camera1 .AfterPicture**:
image
do **set [Image1 . Picture] to [get image]**
set [Image1 . Visible] to [true]

APP que escanea un código QR y

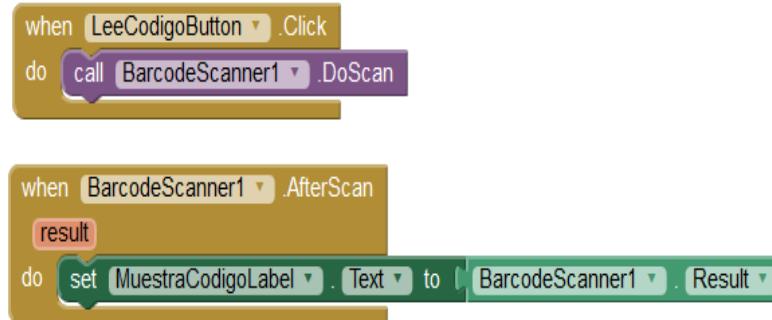
- Muestra el código leído en la etiqueta MuestraCodigoLabel
- La aplicación se inicia con el botón LeeCodigoButton



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

Comportamiento

- 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.



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.

APP que escanea un código QR y abre una página Web en otra pantalla

1. En la pantalla principal añadimos un botón y un componentelector de código, denominado “BarcodeScanner”
2. Tenemos que crear otra pantalla “Screen” → Botón “add Screen”
3. Añadimos a la nueva pantalla un botón para volver a la primera pantalla, y un componente WebViewer
4. Cada pantalla tiene su comportamiento basado en bloques.

```
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 "
```

Ejercicios

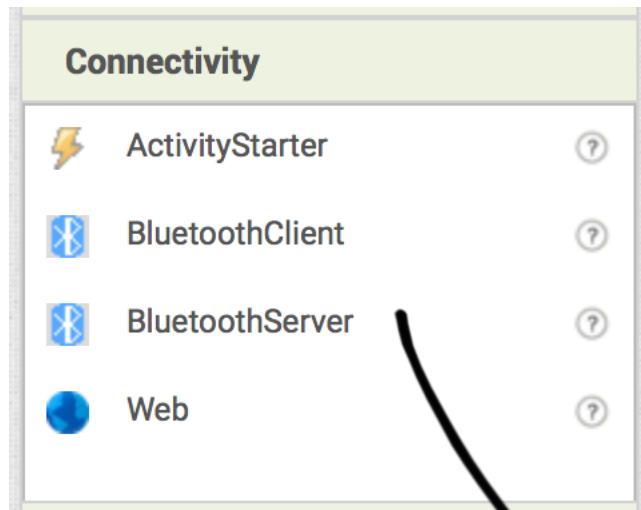


- Crea una pantalla de inicio que solicite al usuario un identificador y una contraseña.
- Crea una pantalla que permita al usuario registrarse. Los datos usuario/contraseña se almacenarán en el móvil.
- Modifica la APP que toma una foto y la muestre, pero que la muestre en otra pantalla.



Comunicación via Bluetooth

Componentes Bluetooth en AppInventor



Properties

BluetoothClient1

CharacterEncoding: UTF-8

DelimiterByte: 0

HighByteFirst:

Secure:

Properties

BluetoothServer1

CharacterEncoding: UTF-8

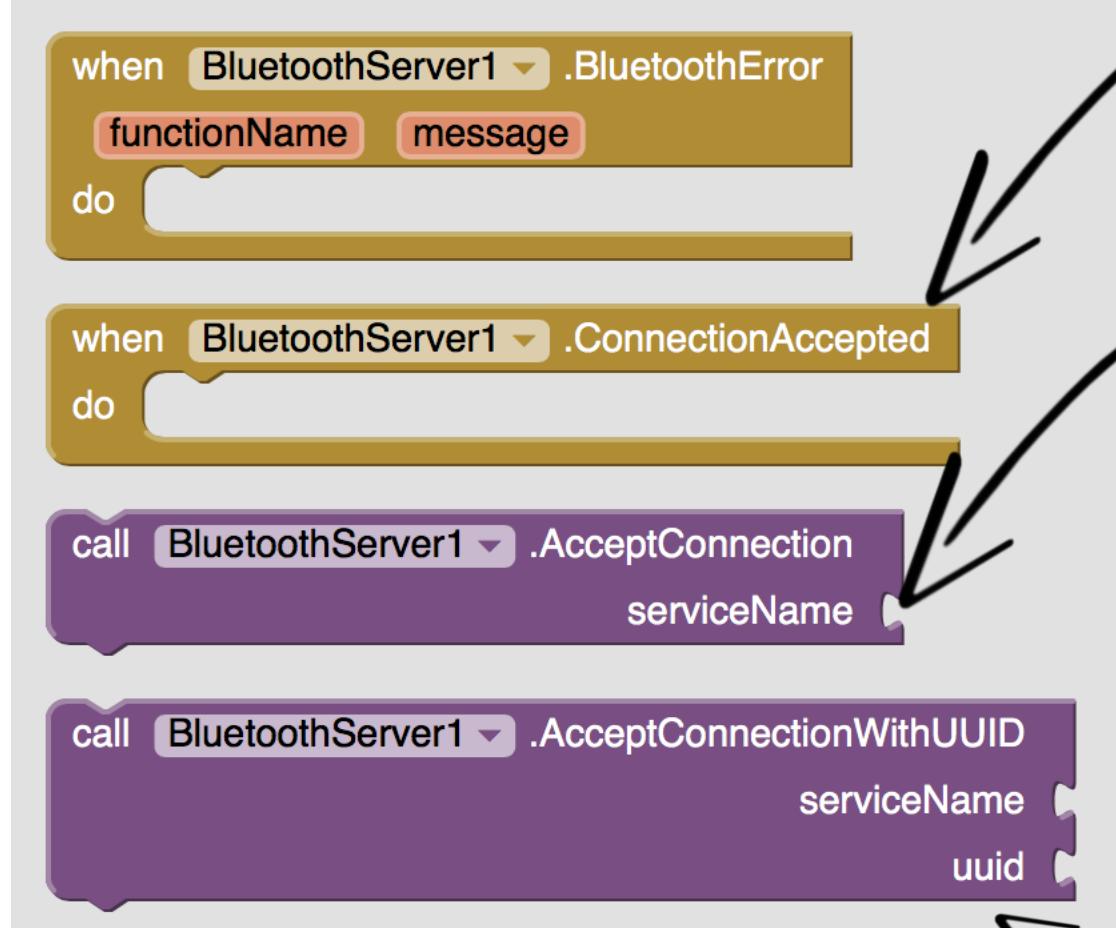
DelimiterByte: 0

HighByteFirst:

Secure:

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

BluetoothServer: Comportamiento

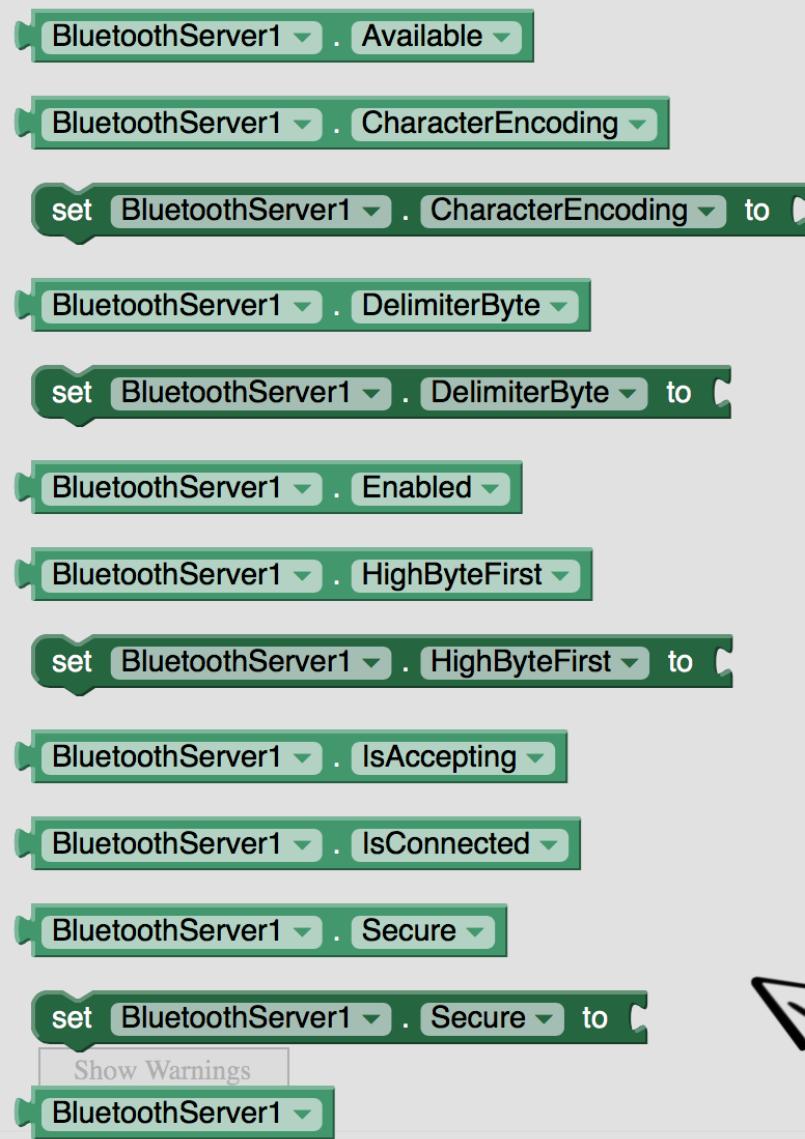


Evento que indica que se ha aceptado una conexión

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



BluetoothServer: Comportamiento

```
call BluetoothServer1 .Send1ByteNumber  
number
```

```
call BluetoothServer1 .Send2ByteNumber  
number
```

```
call BluetoothServer1 .Send4ByteNumber  
number
```

```
call BluetoothServer1 .SendBytes  
list
```

```
call BluetoothServer1 .SendText  
text
```

```
call BluetoothServer1 .StopAccepting
```

Envío sobre la conexión

Para de aceptar peticiones de conexión entrantes

BluetoothServer: Comportamiento

```
call BluetoothServer1 .ReceiveSigned1ByteNumber
```

```
call BluetoothServer1 .ReceiveSigned2ByteNumber
```

```
call BluetoothServer1 .ReceiveSigned4ByteNumber
```

```
call BluetoothServer1 .ReceiveSignedBytes
```

 numberOfBytes

```
call BluetoothServer1 .ReceiveText
```

 numberOfBytes

Recepción sobre la conexión

```
call BluetoothServer1 .ReceiveUnsigned1ByteNumber
```

```
call BluetoothServer1 .ReceiveUnsigned2ByteNumber
```

```
call BluetoothServer1 .ReceiveUnsigned4ByteNumber
```

```
call BluetoothServer1 .ReceiveUnsignedBytes
```

 numberOfBytes

BluetoothClient: Comportamiento

```
when BluetoothClient1 .BluetoothError  
  functionName message  
do
```

Devuelve un nº estimado de bytes que puede recibir sin bloquear

```
call BluetoothClient1 .BytesAvailableToReceive
```

Se conecta al dispositivo Bluetooth y a un servicio SPP activo

```
call BluetoothClient1 .Connect  
  address
```

Se conecta al dispositivo Bluetooth y al servicio con el perfil dado por el UUID

```
call BluetoothClient1 .ConnectWithUUID  
  address  
  uuid
```

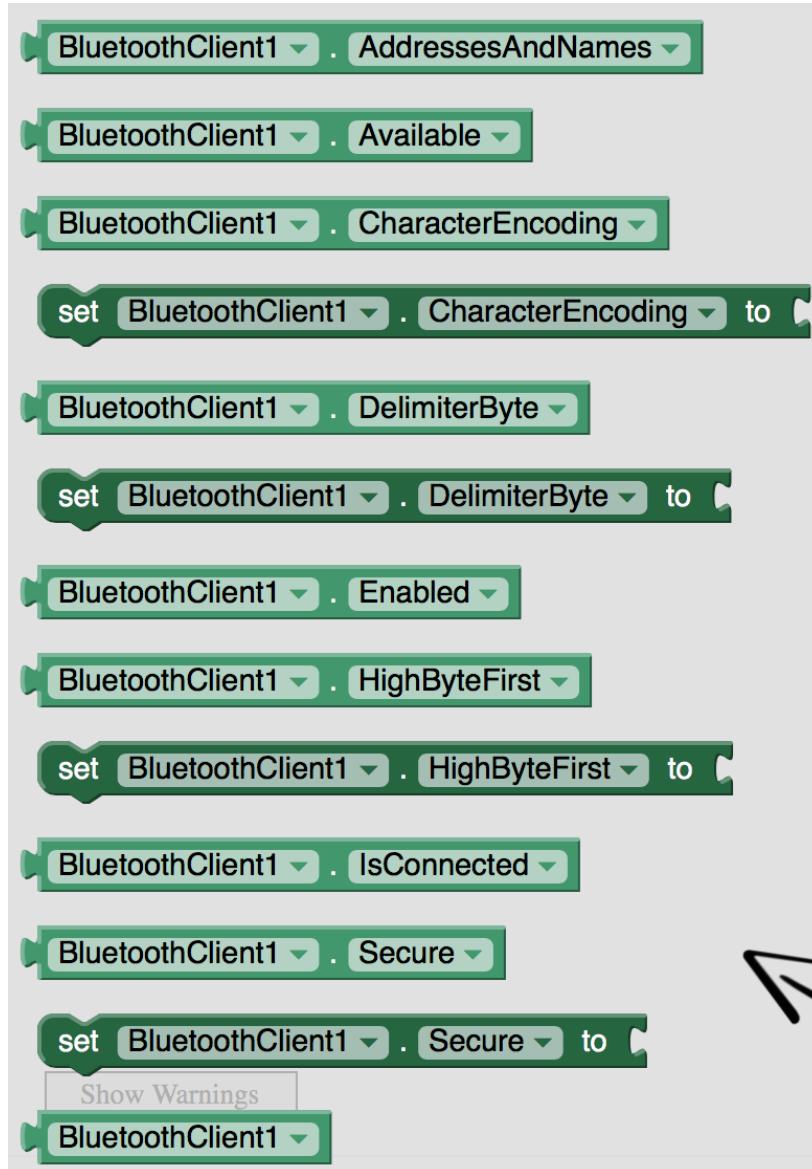
Se desconecta del dispositivo Bluetooth conectado

```
call BluetoothClient1 .Disconnect
```

Comprueba si el dispositivo BT con esa dirección Bluetooth está emparejado

```
call BluetoothClient1 .IsDevicePaired  
  address
```

BluetoothClient: Comportamiento



Propiedades que se pueden consultar y modificar desde el comportamiento

BluetoothClient: Comportamiento

```
call BluetoothClient1 .ReceiveSigned1ByteNumber
```

```
call BluetoothClient1 .ReceiveSigned2ByteNumber
```

```
call BluetoothClient1 .ReceiveSigned4ByteNumber
```

```
call BluetoothClient1 .ReceiveSignedBytes  
    numberOfBytes
```

```
call BluetoothClient1 .ReceiveText  
    numberOfBytes
```

```
call BluetoothClient1 .ReceiveUnsigned1ByteNumber
```

```
call BluetoothClient1 .ReceiveUnsigned2ByteNumber
```

```
call BluetoothClient1 .ReceiveUnsigned4ByteNumber
```

```
call BluetoothClient1 .ReceiveUnsignedBytes  
    numberOfBytes
```

Recepción sobre la conexión

```
call BluetoothClient1 .Send1ByteNumber  
    number
```

```
call BluetoothClient1 .Send2ByteNumber  
    number
```

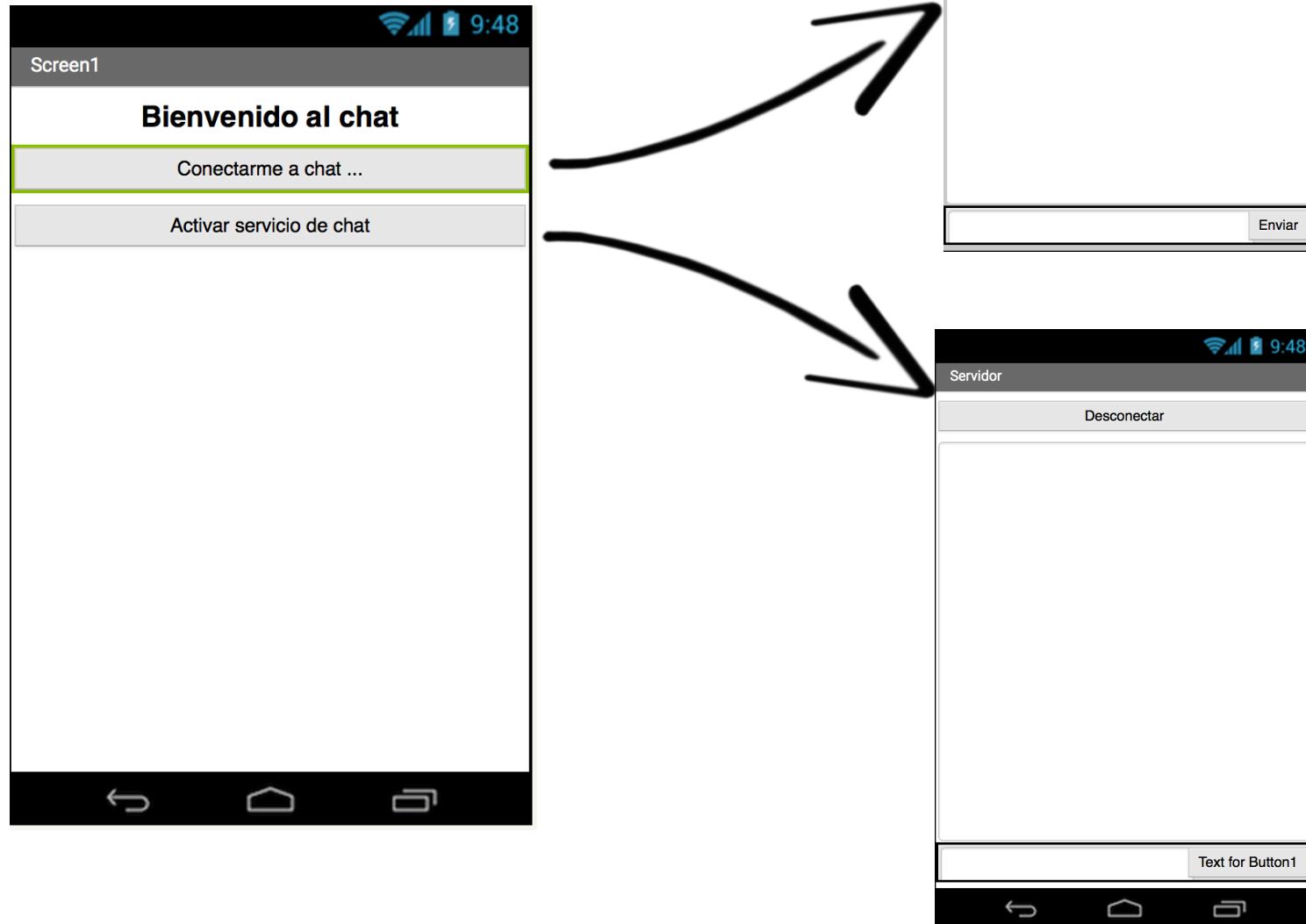
```
call BluetoothClient1 .Send4ByteNumber  
    number
```

```
call BluetoothClient1 .SendBytes  
    list
```

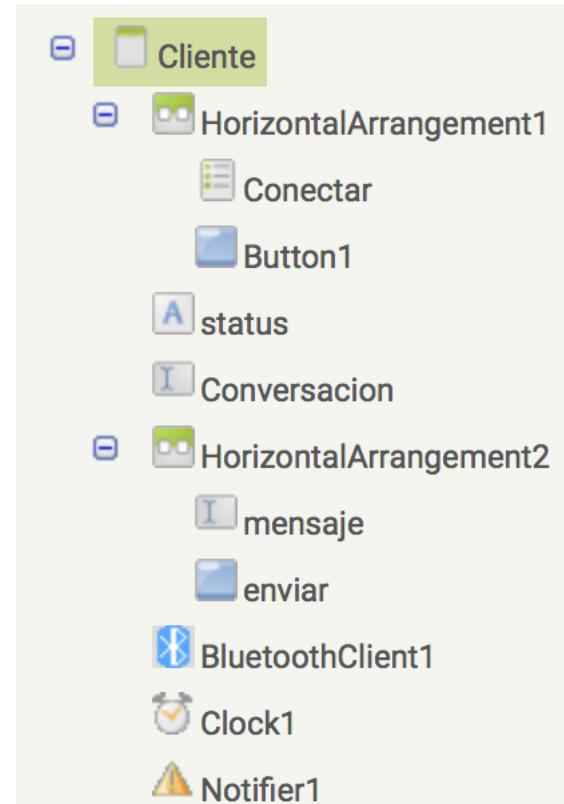
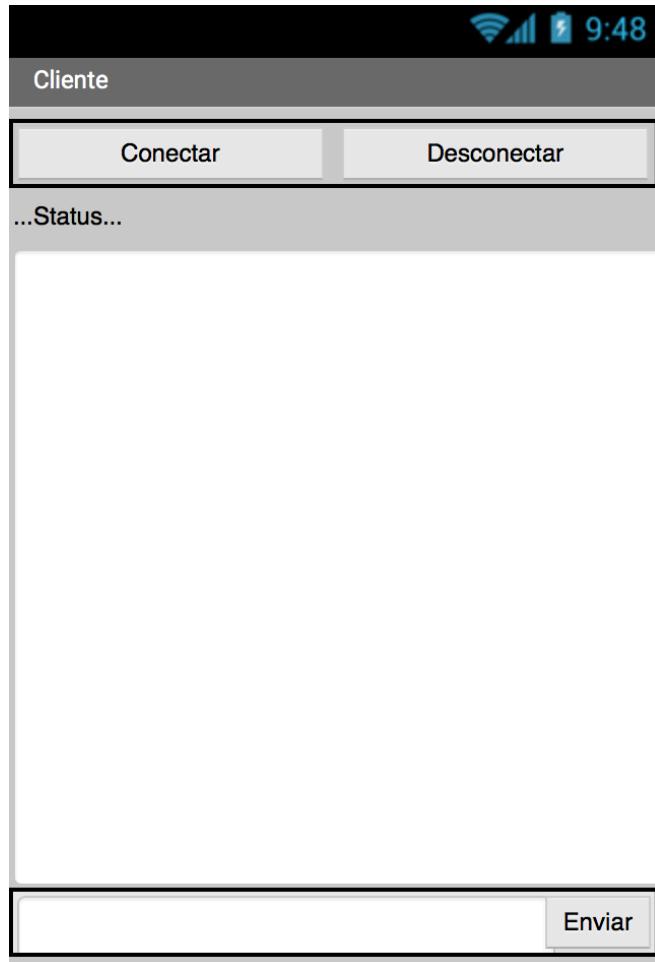
```
call BluetoothClient1 .SendText  
    text
```

Envío sobre la conexión

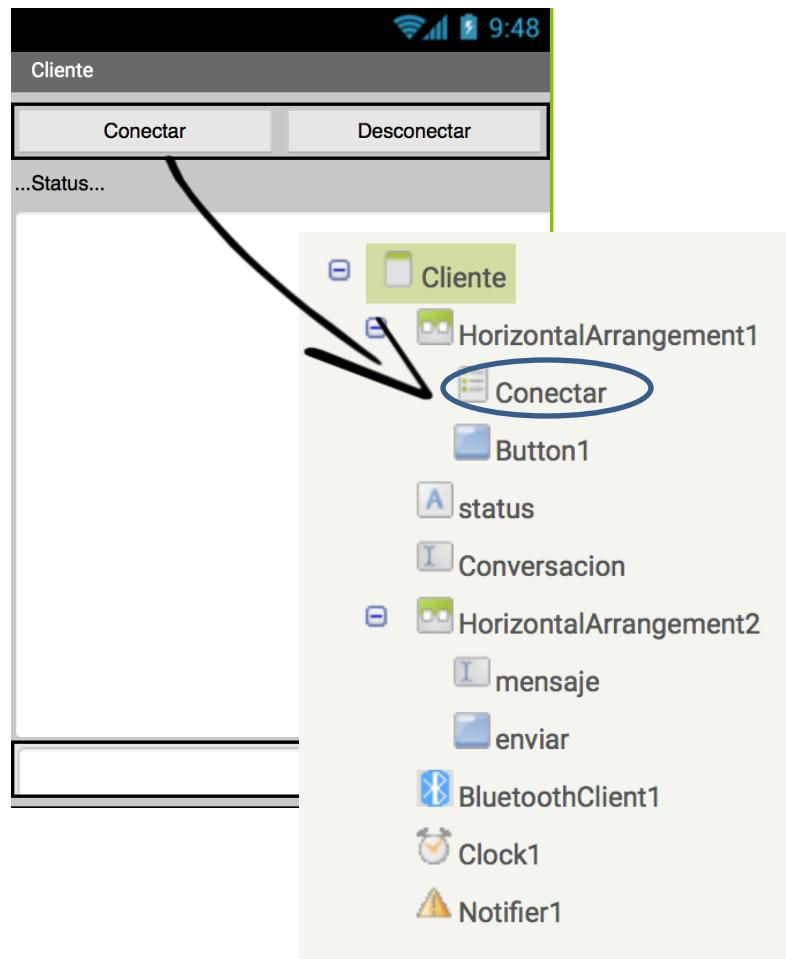
Ejercicio práctica



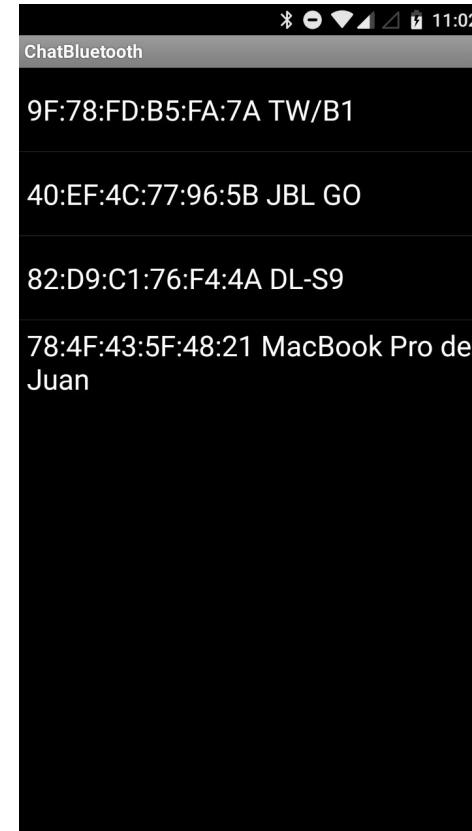
Ejercicio práctica: Cliente



Ejercicio práctica: Cliente – conectarse al servidor



Componente ListPicker (para buscar el servidor y seleccionarlo de entre los dispositivos disponibles)



Ejercicio práctica: Cliente – conectarse al servidor

The image shows a Scratch script for a 'Cliente' project. On the left, the stage tree lists the following components:

- Cliente
- HorizontalArrangement1
 - Conectar (highlighted with a blue oval)
 - Button1
- status
- Conversacion
- HorizontalArrangement2
 - mensaje
 - enviar
- BluetoothClient1
- Clock1
- Notifier1

A black arrow points from the 'Conectar' button in the stage tree to the 'when Conectar .BeforePicking' event in the script area.

Pestaña Blocks para ListPicker

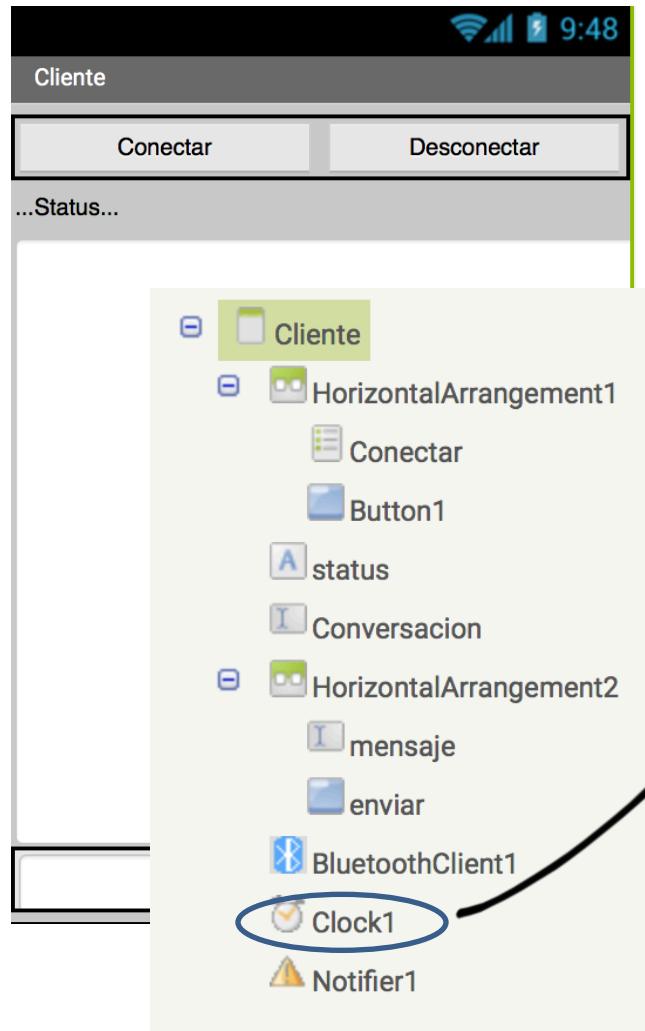
when Conectar .BeforePicking

```
when Conectar .BeforePicking
do [set Conectar . Elements to BluetoothClient1 . AddressesAndNames]
```

when Conectar .AfterPicking

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

Ejercicio práctica: Cliente – recibir datos



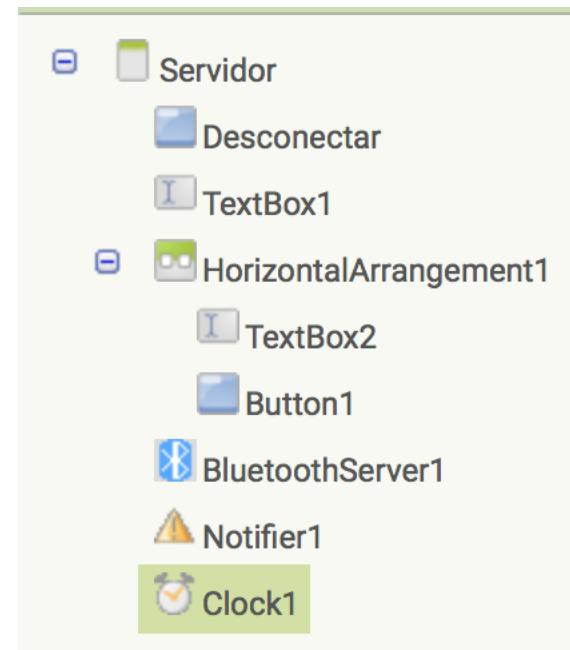
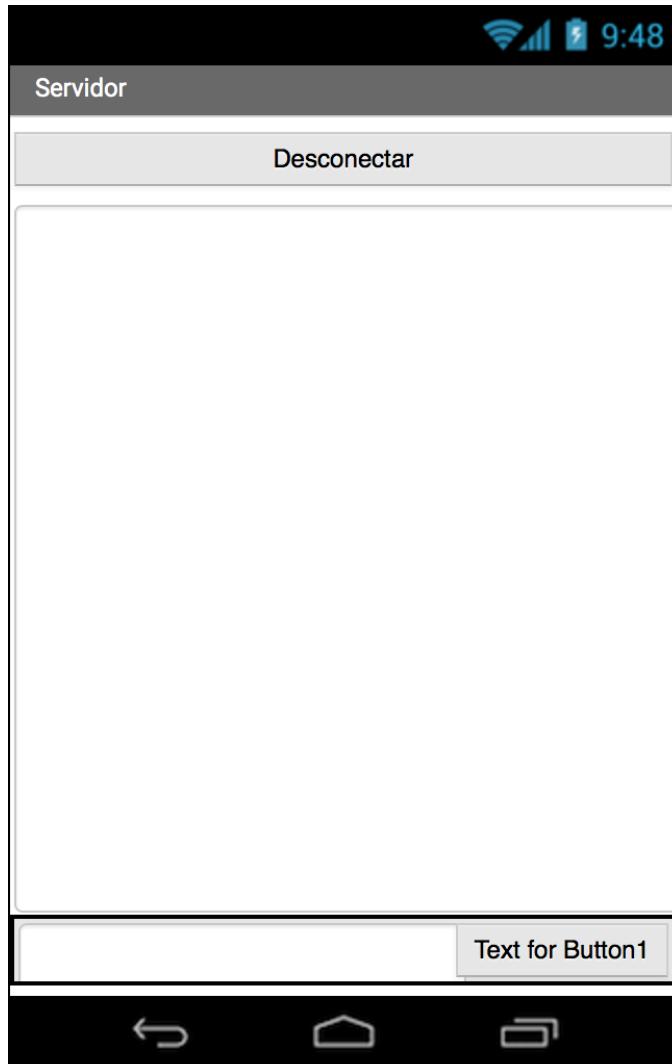
Temporizador para la recepción periódica de mensajes



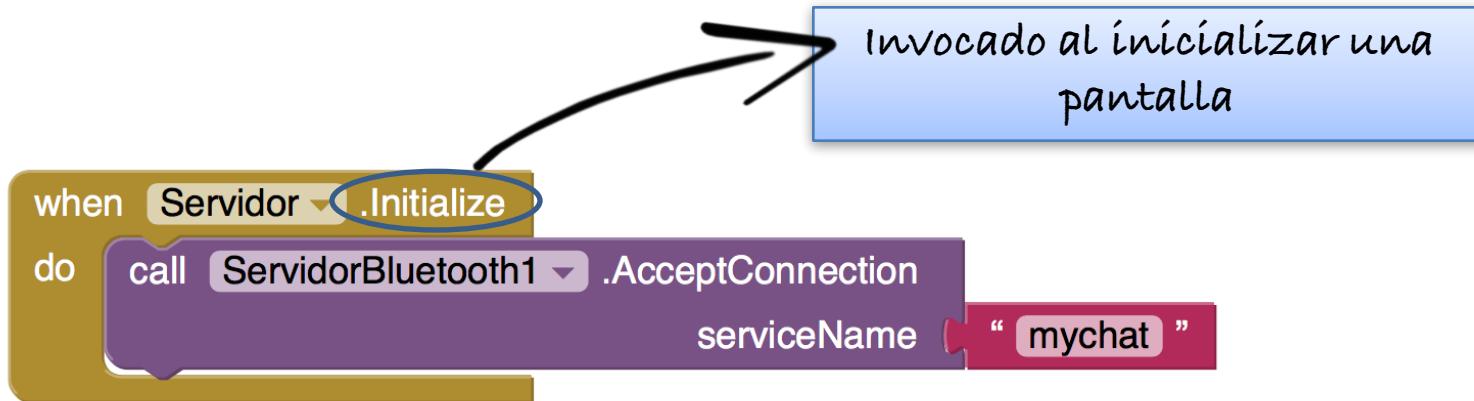
Insertar bloques de recepción y actualización de la GUI aquí

El envío de mensajes se realiza cuando se pulsa el botón de "enviar"

Ejercicio práctica: Servidor



Ejercicio práctica: Servidor – inicialización y desconexión del servicio



Es necesario también desconectar cuando se pulsa el botón
(además del cliente)

El resto (envío y recepción de mensajes) es similar en cliente y servidor