Apuntes para un aprendiz de programador: App Inventor, programación en dispositivos móviles al alcance de todos

"Tools amplify your talent"
The Pragmatic Programmer

Introducción

Los dispositivos móviles "inteligentes" tales como teléfonos celulares y "tabletas" se hacen más comunes día con día. La mayoría de ellos se programan de forma tradicional. Sin embargo, mientras Apple ha escogido tener un modelo de control estricto para desarrollar aplicaciones para sus dispositivos (IPod, IPad & IPhone). Google por su parte ha sido mucho más abierto para el desarrollo de aplicaciones en su sistema operativo Android.

Con el App Inventor, Google acerca la programación de dispositivos móviles a mucha más gente que no es especialista. Inclusive puede ser usado para desarrollo de prototipos rápidos, simples y funcionales por lo que no es descartable totalmente su uso por parte de expertos. Disponible inicialmente de manera pública en diciembre de 2010. En agosto de 2011, Google anunció que iba a liberar el código del proyecto y que éste va a quedar a cargo del MIT Center for Mobile Learning que forma parte del Media Lab de la misma institución.

En un artículo previo (Fernández y Fernández, 2011) se mencionan algunas herramientas (Alice, Scratch, Lego Mindstorms, entre otras) para acercar la programación a los niños[1], donde recalcamos la importancia del pensamiento computacional: generar soluciones con pensamiento algorítmico es clave en cualquier disciplina y el entrenarse en pensamiento computacional ofrecería una herramienta invaluable a la gente (Wing, 2006). Del conjunto de herramientas mencionadas en (Fernández y Fernández, 2011), una de ellas es la base del App Inventor: Scratch. Scratch (Figura 1), que proporciona un lenguaje creado por el Lifelong Kindergarten group en el MIT Media Lab. Esta herramienta fue diseñada originalmente para ayudar a los jóvenes a mejorar sus procesos de aprendizaje, pero se ha convertido en una herramienta útil de programación (Maloney, 2004).

App Inventor

De igual manera, App Inventor es una herramienta útil de programación, con la ventaja adicional de permitir el desarrollo de aplicaciones para dispositivos móviles que usen el sistema operativo Android. Una característica interesante es que el desarrollo de la aplicación es en Web. Aunque es necesario instalar un módulo de software en la computadora, en el momento del desarrollo se ejecuta la última versión del App Inventor disponible en su sitio web[2] y los proyectos se guardan en línea.

El App Inventor consta de dos segmentos principales: un módulo Web y el editor de bloques de Android. El módulo en Web que se mencionaba, donde aparte de ser el punto de entrada tenemos acceso a nuestros proyectos y, una vez abierto un

1. Otra herramienta interesante es Greenfoot (Henriksen, 2004), un ambiente interactivo basado en Java para la enseñanza de programación. Ver: http://www.greenfoot.org/

2. Originalmente en http://App Inventor.googlelabs.com/ y ahora disponible en http://www.AppInventorbeta.com/

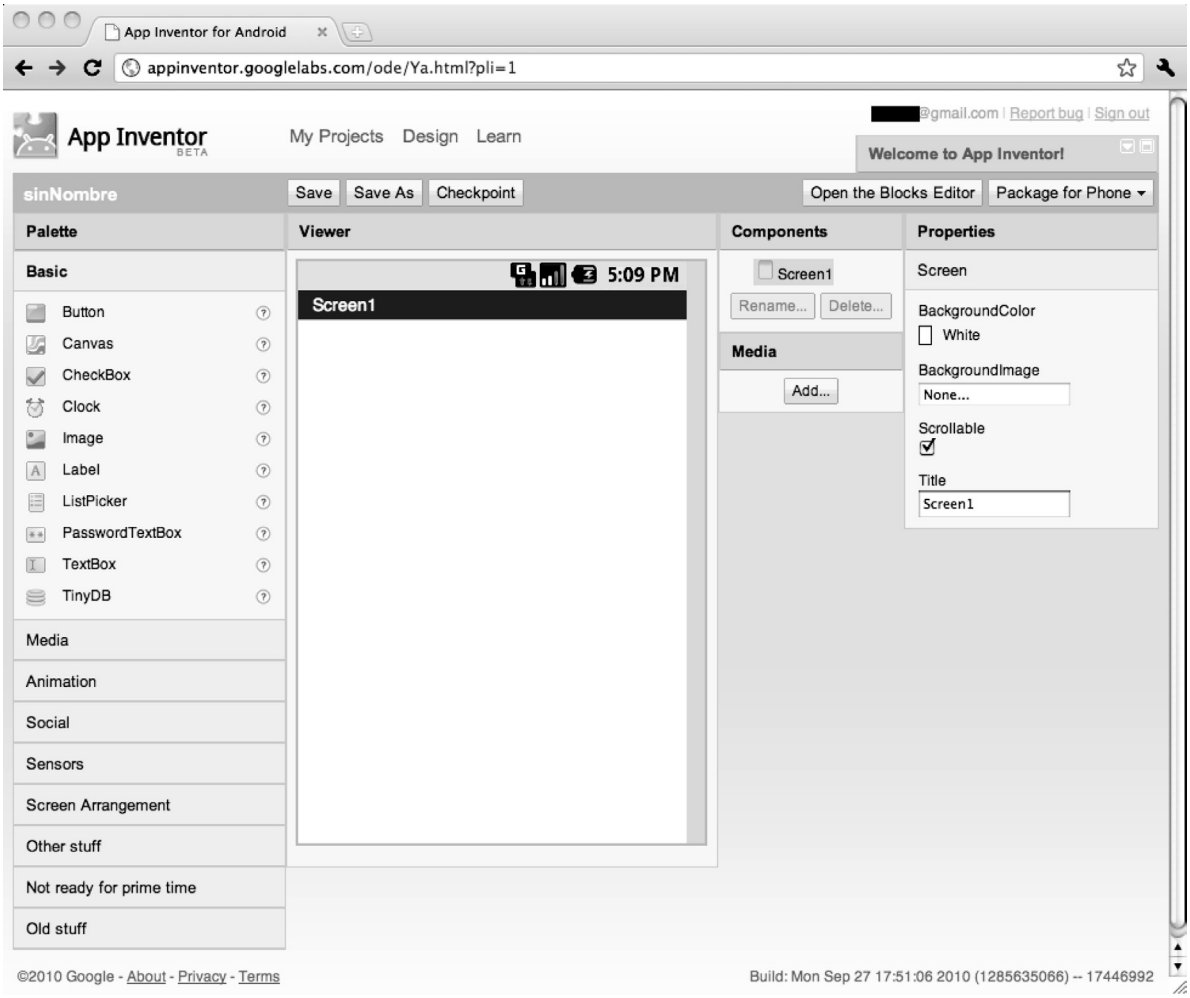

FIGURA 1. PANTALLA DE DISEÑO DE APP INVENTOR

proyecto, podemos entrar a la sección de diseño de nuestra aplicación. Esta sección es donde podemos añadir los componentes y configurarlos apropiadamente. Si se trata de componentes visuales, entonces definimos también el diseño de la interfaz. Para los familiarizados con desarrollo de aplicaciones mediante componentes visuales verán que es un concepto bastante similar. El segmento del editor de bloques se verá más adelante, por el momento basta con mencionar que ahí es donde los bloques se conectan cual piezas Lego, formando la lógica de la aplicación.

Una aplicación simple para envío de texto

Para poder explicar mejor lo simple que es desarrollar una aplicación para Android con el App Inventor vamos a desarrollar un pequeño ejemplo. Generaremos una aplicación que mande un mensaje de texto a un contacto del teléfono.

Diseñando la interfaz de usuario

Para iniciar, creamos un proyecto llamado EnviarTexto y nos aparecerá una imagen en blanco similar a la de la Figura 1. Ajustamos la propiedad del título de la aplicación, cambiando el texto Screen1 por el de "Enviar texto", como se aprecia en la Figura 2.

A continuación diseñamos nuestra pantalla, para lo cual necesitaremos los siguientes componentes: TextBox, Button, Texting, PhoneNumberPicket, TableArrangement, ImagePicker, Notifier y Label. Nuestra pantalla inicial con estos componentes agregados lucen aún sin configurar (salvo TableArrangement) como lo muestra la Figura 3.

El componente TableArrangement fue configurado para dar una mejor organización de los componentes.

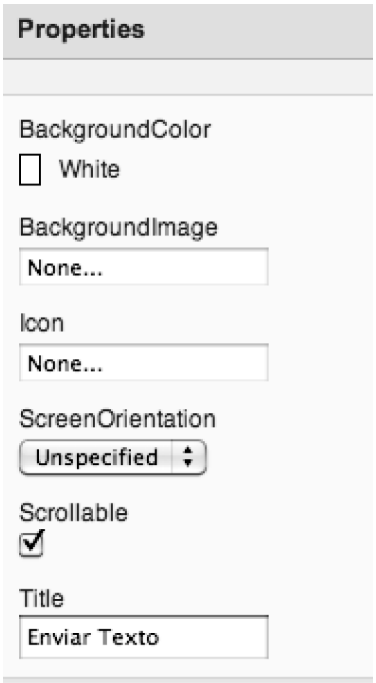

FIGURA 2. AJUSTANDO EL TÍTULO DE LA APLICACIÓN.

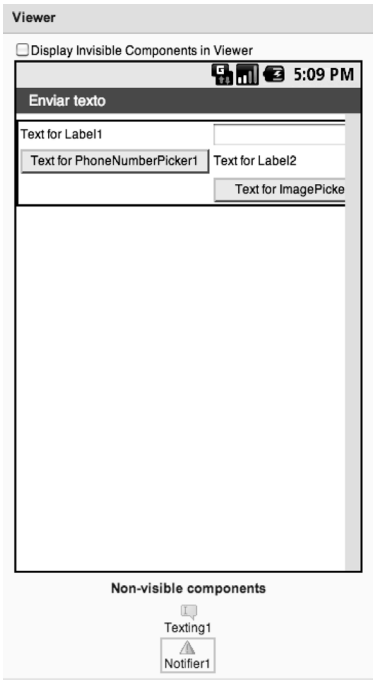

FIGURA 3. COMPONENTES AGREGADOS AÚN SIN CONFIGURAR.

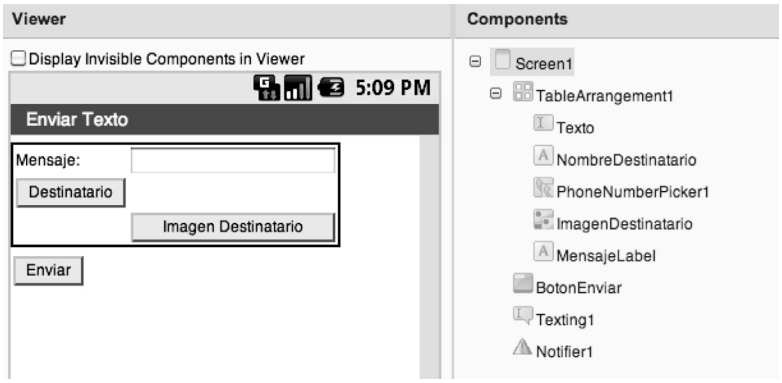

FIGURA 4. PANTALLA PARCIAL MOSTRANDO LOS COMPONENTES PERSONALIZADOS.

Componente	Nombre asignado	Otra personalización	Objetivo
Label	MensajeLabel	La propiedad *text* se cambia a "Mensaje:".	Mostrar texto "Mensaje:".
TextBox	Texto		Captura el texto para enviar.
Button	BotonEnviar	La propiedad *text* se cambia a "Enviar".	Envía el texto al destinatario
Texting	Texting1		Componente para enviar mensaje de texto.
PhoneNumber Picker	PhoneNumberPicket1	La propiedad *text* se cambia a "Destinatario".	Permite elegir al destinatario de nuestros contactos
Table Arrangement	TableArrangement1		Panel para organizar los elementos de la interfaz
Label	NombreDestinatario	Se borra todo texto de la propiedad *text*. La propiedad *Hint* se cambia a "Nuevo un mensaje"	Muestra el nombre del destinatario del mensaje.
ImagePicker	ImagenDestinatario	La propiedad *text* se cambia a "Imagen Destinatario".	Muestran la imagen del contacto elegido.
Notifier	Notifier1		Avisar al usuario el envío del mensaje.

TABLA 1. LISTADO DE COMPONENTES USADOS EN NUESTRA APLICACIÓN.

FIGURA 5. VISTA DEL ANDROID BLOCKS EDITOR

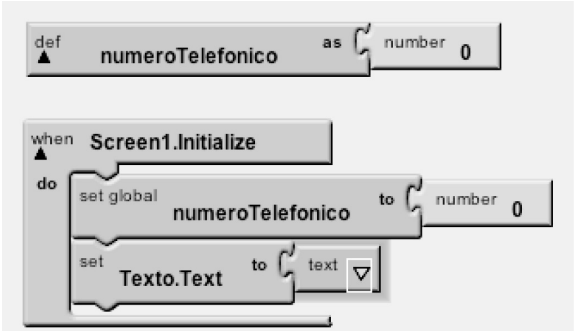

FIGURA 6. INICIALIZACIÓN DE LA APLICACIÓN.

FIGURA 7. SELECCIÓN DEL DESTINATARIO.

Esta y otras personalizaciones de los componentes se mencionan en la Tabla 1.

La pantalla de nuestra aplicación debe lucir como se muestra en la pantalla parcial de la Figura 4.

Agregando la lógica de la aplicación

Hasta ahora únicamente hemos diseñado la interfaz con el usuario. Para agregar el comportamiento de nuestra aplicación seleccionamos el botón "Open the Blocks Editor", lo que nos lleva a la descarga y ejecución de la última versión disponible del editor de bloques. Nos aparecerá una aplicación similar a la de la Figura 5, donde se muestran las diferentes secciones de bloques predefinidos y una sección (My Blocks) donde se encuentran los componentes (ahora llamados bloques) usados en la etapa anterior. Los bloques se arrastran al área principal donde vamos armando nuestros conjuntos de bloques generando la lógica de la aplicación.

Android Block Editor es además el encargado de realizar la conexión con el teléfono para probar la ejecución de la aplicación. Si no se cuenta con un teléfono, es posible conectarse con un emulador ejecutando éste desde Android Block Editor. Sin embargo hay que tener en cuenta que, al no ser un teléfono real, algunas funciones no están disponibles en el emulador, como es el envío y recepción de mensajes de texto.

El comportamiento que le vamos a definir a nuestra aplicación se basa en 3 partes generales:

1. Inicializar la aplicación.
2. Elección del destinatario.
3. Enviar mensaje de texto.

1. Inicializar la aplicación

En esta parte se preparan los datos para el inicio de la ejecución de la aplicación. Necesitamos una variable para almacenar el número telefónico, a la cual asignamos el valor de cero. Ahora, como el comportamiento de nuestra aplicación empieza con la visualización de nuestra única pantalla (Screen1), elegimos el bloque de inicialización de dicho elemento (Screen1.Initialize). Aquí vamos a especificar las acciones de asignar cero a numeroTelefonico y texto vacío al componente TextBox que llamamos Texto. La Figura 6 muestra como debe quedar este segmento de bloques.

2. Selección el destinatario

En esta parte, necesitamos indicar la lógica de la aplicación donde se selecciona el contacto al que se va a enviar el mensaje de texto. Un comportamiento parcial está predefinido por el tipo de componentes colocados en la pantalla. En el caso del componente TextBox, éste claramente tiene un comportamiento para recibir un texto, el texto a enviar en este caso. PhoneNumberPicker por su parte, tiene todo el comportamiento definido para ir a la lista de contactos del teléfono y elegir uno. En este caso, nosotros únicamente definimos lo que hacemos después de que el usuario ha elegido un contacto. Para esto elegimos el componente PhoneNumberPicker1 en la sección de MyBlocks de la barra del Android Blocks Editor. Para este componente queremos el bloque AfterPicking.

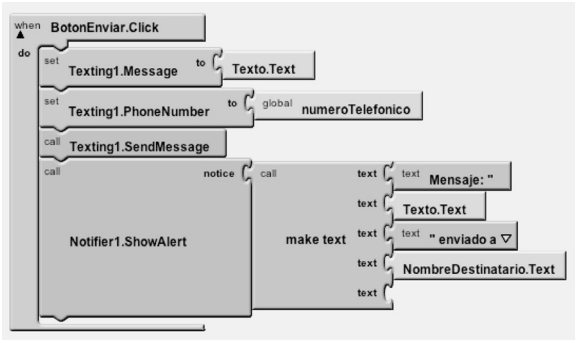

FIGURA 8. ENVÍO DE MENAJE Y NOTIFICACIÓN AL USUARIO.

Ahora, queremos definir el comportamiento para este evento, que básicamente se reduce a obtener la información del componente PhoneNumberPicker1, como se puede apreciar en la Figura 7:

- A NombreDestinatario.Text le asignamos ContactName para tener el nombre del contacto.
- A ImagenDestinatario.Image se le asigna Image para obtener la imagen del contacto si la hubiera.
- A nuestra variable numeroTelefonico se le asigna PhoneNumber.

3. Enviar mensaje de texto y notificar al usuario

En esta sección nos interesa definir el comportamiento para el envío del mensaje de texto al destinatario, confirmando al usuario de la aplicación el envío del mensaje. Al presionar "Enviar", la aplicación debe mandar el mensaje de texto al contacto elegido. Para indicar este comportamiento en nuestra aplicación, debemos seleccionar el componente BotonEnviar de la barra MyBlocks. De dicho componente queremos elegir el bloque Click. A continuación definimos el comportamiento específico dentro del evento Click. Primero, preparamos el mensaje, el cual esta en contenido en Texto.Text y debemos pasarlo al componente Texting para envío de texto (Texting1.Message). Del mismo modo, asignamos numeroTelefonico a Texting1.PhoneNumber. Ya tenemos el mensaje listo y lo enviamos ejecutando la operación SendMessage de Texting1.

En nuestro ejemplo, sólo queda notificar al usuario. Esto lo hacemos eligiendo ahora de MyBlocks el componente Notifier1 y ahí el bloque para la operación ShowAlert. Esta operación recibe un texto, pero nosotros vamos a introducir un texto compuesto para enviar la información del mensaje que se ha enviado y el destinatario. Para esto concatenamos el texto haciendo uso del componente make text, que se encuentra en la barra de Built-in en la sección de Text. La Figura 8 muestra el conjunto de bloques para la sección descrita en este apartado.

FIGURA 9. EJEMPLO DE EJECUCIÓN DE LA APLICACIÓN EN EMULADOR.

Comentarios finales

Muchas mejoras que se podrían añadir a nuestro pequeño programa; por ejemplo, nuestra aplicación no válida que se haya elegido a un contacto o que el texto no vaya vacío antes de enviarse. Pero hemos querido mostrar una aplicación simple y cualquier mejora queda como ejercicio para el lector. La Figura 9 muestra la aplicación ejecutándose en el emulador.

Al comenzar este artículo se mencionaron algunas opciones para iniciarse en la programación de aplicaciones. Como pudimos apreciar App Inventor es una forma simple e inovadora de realizar aplicaciones realmente funcionales en smartphones que usen Android. Sin embargo existen otras opciones más tradicionales que pueden ser consideradas.

Inicialmente Android soportaba directamente únicamente el lenguaje de Java. Actualmente, se cuenta con Android NDK[3] (Native Development Kit) soporta los

3. http://developer.android.com/sdk/ndk/index.html

lenguajes de C y C++. Adicionalmente, ya tiene tiene soporte para lenguajes script dinámicos, por medio del Scripting Layer for Android[4] (SL4A), que permite el desarrollo de aplicaciones con lenguajes como Python, Lua, Perl, Ruby y Javascript, entre otros.

Además, es posible usar diferentes ambientes de desarrollo. El más popular es Eclipse ADT[5] (Android Development Tools), el cual es un plugin que funciona dentro del IDE de Eclipse. Motodev Studio[6] es una herramienta también basada en Eclipse para el desarrollo de aplicaciones Android pero ofrecida por Motorola.

Carlos Alberto Fernández y Fernández[1], Héctor Gabriel Acosta Mesa[2], Nicandro Cruz Ramírez[2]

[1]Universidad Tecnológica de la Mixteca
[2]Universidad Veracruzana

Referencias

Fernández-y-Fernández, C. A.
2011 Apuntes para un aprendiz de programador: jugando con robots Lego, El Poder de la Computación, a tu Alcance, pp. 23-30.

Henriksen P., Kölling M. and Köolling M.
2004 Greenfoot: combining object visualisation with interaction, in Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications - OOPSLA '04, p. 73.

Maloney, J. Burd L., Kafai Y., Rusk N., Silverman B. and Resnick M.
2004 Scratch: A sneak preview,, pp. 104–109.

Wing J.
2006 Computational thinking, Communications of the ACM.

4. http://code.google.com/p/android-scripting/
5. http://developer.android.com/sdk/eclipse-adt.html
6. http://developer.motorola.com/docstools/motodevstudio/