Curso de App Inventor

Escuela Superior de Informática

Universidad de Castilla-La Mancha

Escuela Superior de Informática
de Ciudad Real

Escuela Superior de Informática

e-mail	esi@uclm.es
Teléfono	926 29 53 00
Web	http://www.esi.uclm.es

Este documento fue compuesto con LaTeX, y ha sido desarrollado a partir de una plantilla de Carlos González Morcillo y Sergio García Mondaray.

Resumen

Este librito que tienes en tus manos es el resultado del material docente del *Curso de App Inventor* impartido por la Escuela Superior de Informática de la Universidad de Castilla-La Mancha, especialmente orientado a estudiantes de Secundaria y Bachillerato.

Desde su primera versión completa publicada por *Google* en Diciembre de 2011, *App Inventor* ha sido una auténtica revolución. Gracias a *App Inventor* es posible crear programas empleando un interfaz de objetos visuales. Utilizando una potente metáfora de piezas de puzzle (similar a la aproximación de *Scratch* o *StarLogo*), como programador crearás fácilmente potentes aplicaciones para teléfonos y tabletas Android.

Disfruta y... Happy Hacking!

Índice general

1. Mi Primer Programa: *¡Hola, Programador!* **1**

1.1. Primeros pasos 2

1.2. Construyendo tu primera App 5

1.3. Probando la aplicación 9

2. El Xilófono de Colores **11**

2.1. ¿Qué voy a construir? 11

2.2. Diseño de los componentes 12

2.3. Crear el teclado 14

2.3.1. Primeras notas del teclado 14

2.3.2. Añadir el componente *Sound* 16

2.3.3. Conectar los sonidos a los botones 16

2.3.4. Inicializar los sonidos 19

2.3.5. El resto del teclado 19

3. Whack A Zombie **23**

3.1. ¿Qué voy a construir? 23

3.2. Diseño de la interfaz paso a paso 26

3.2.1. La pantalla de juego 26

3.2.2. La barra de energía y la puntuación 26

3.2.3. El sonido y el reloj del juego 28
3.3. Defínción del comportamiento del juego 29
3.3.1. Animación del zombie 29
3.3.2. Arranque del juego 33
3.3.3. Interacción con el juego 34
3.3.4. Reinicio juego 37

Capítulo 1

Mi Primer Programa: *¡Hola, Programador!*

Es una práctica habitual en todos los cursos de programación comenzar por un ejemplo de programa muy sencillo, que sea muy fácil de entender, y que de paso dé una idea de cómo funciona el nuevo lenguaje. En la jerga informática se le suele conocer como "hola, mundo", en honor al primer programa escrito en el lenguaje C de la historia, allá por el año 1975. El nombre de "hola, mundo" viene precisamente de que lo único que hacía ese programa era imprimir ese mensaje en la pantalla.

Siguiendo con esta práctica, lo que se propone en este capítulo es crear el "hola, mundo" para AppInventor. Pero como se trata de crear una App para un dispositivo móvil, y no un programa para un antiguo ordenador con un teclado y monitor monocromo, realizaremos una pequeña adaptación del original, que va a consistir en colocar un botón en la pantalla, de manera que al pulsarlo se muestre el mensaje "Hola AppInventor".

1.1. Primeros pasos

El primer paso para la creación de la App será iniciar el entorno de trabajo. Para ello abre tu navegador de internet habitual y dirígite a la siguiente URL:

http://appinventor.mit.edu

Verás cómo se carga la página de inicio de la aplicación, que en la parte inferior incluye tres botones: Teach, Explore e Invent, que dan acceso a recursos para los educadores, información y tutoriales, y a la creación de aplicaciones móviles, respectivamente. Puesto que nuestra intención es crear una aplicación, pulsaremos sobre el botón Invent.

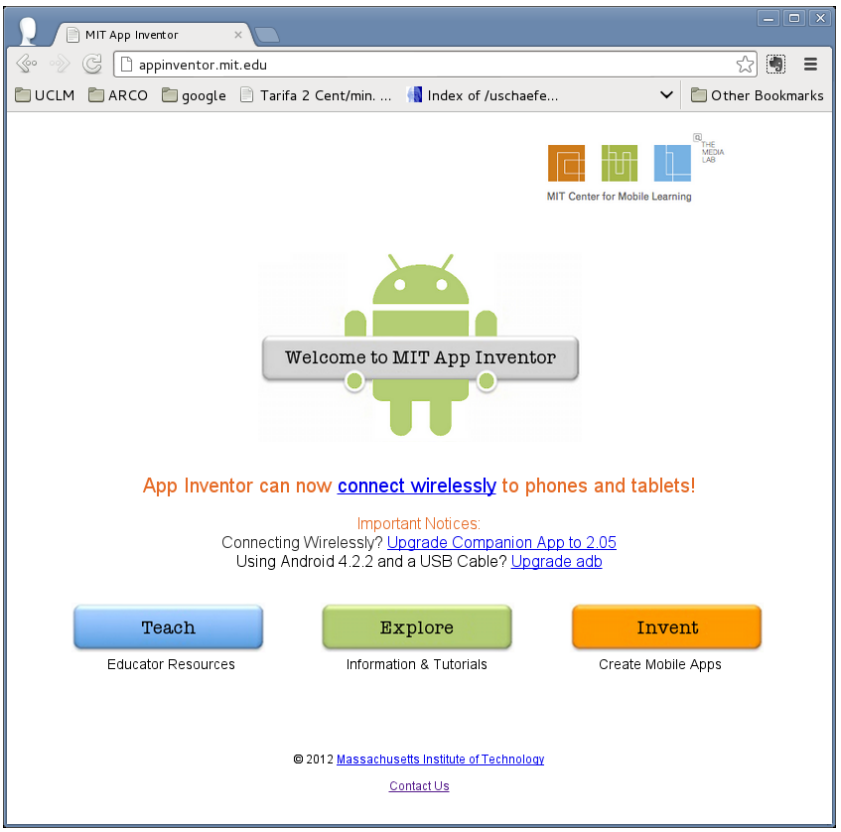

Figura 1.1: Página de inicio de AppInventor.

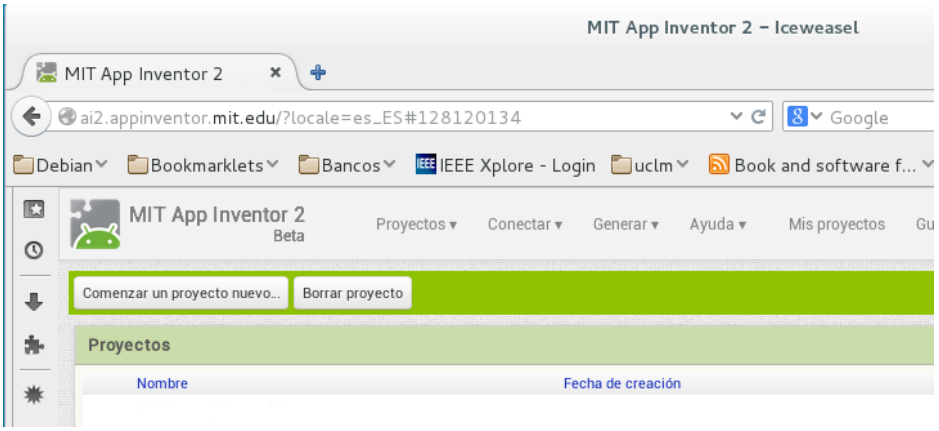

Figura 1.2: Página para la creación de proyectos.

AppInventor utiliza las cuentas de Google para el inicio de sesión, por lo que si llegados a este punto aun no la has iniciado, este será el momento de hacerlo. Si no dispones de una cuenta de Google puedes crearla también. A partir del momento en el que estés registrado, la web de AppInventor recordará todos tus proyectos cada vez que retornes a ella. Puesto que aun no hemos comenzado el desarrollo, la lista de proyectos estará vacía (figura 1.2).

Para comenzar a desarrollar la aplicación el primer paso consistirá en crear un nuevo proyecto. Para ello pulsa el botón Comenzar un proyecto nuevo... (figura 1.3), e introduce en el cuadro de diálogo que aparecerá a continuación el nombre del proyecto, que en este caso podría ser el siguiente: *hola_programador*.

Figura 1.3: Creación del nuevo proyecto.

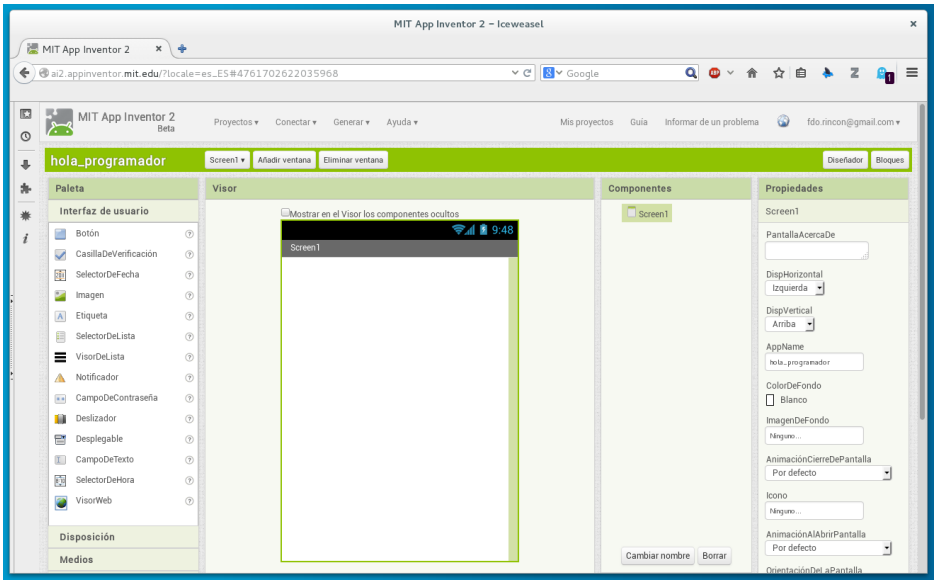

Figura 1.4: Estructura del entorno de edición.

Una vez creado el proyecto, aparecerá una nueva página, con un aspecto como el de la figura 1.4. Como puedes apreciar, en el centro de la página se dibuja la pantalla de un teléfono móvil. Sobre esta zona se colocarán los diferentes componentes gráficos de la aplicación, que se mostrarán con el aspecto que realmente tendrán en el dispositivo real.

En la parte izquierda de la pantalla se encuentra la paleta de componentes que pueden ser utilizados en la aplicación, organizados por categorías. Para este primer proyecto nos bastará con las que aparecen dentro de la categoría *Interfaz de usuario*.

En la parte derecha de la ventana hay 2 columnas más. La más próxima a la pantalla del móvil contiene la lista de los componentes que se han añadido a la aplicación. La otra muestra las propiedades (características) de aquel componente que esté seleccionado. Estas características sirven, por ejemplo, para cambiar el texto que aparece en el botón, cargar el fichero mp3 asociado a un sonido, etc.

El primer paso para el desarrollo de nuestra aplicación va a consistir en identificar qué componentes van a hacer falta. Para ello pensemos en qué es lo que debe hacer exactamente nuestro programa. Supongamos que tratas de explicárselo a un amigo. El diálogo podría ser algo como esto:

- Tú: Hoy he creado mi primera App.
- Amigo: ¡Ah sí! ¿Y qué hace?
- Tú: Te dice hola.
- Amigo: Pues no parece gran cosa.
- Tú: Ya, pero es que la primera tiene que ser fácil.
- Amigo: Bueno, ¿y cómo lo hace?
- Tú: Pues cuando pulsas un botón escribe en pantalla el texto hola, mundo.

Si revisas la última línea de la explicación verás que hacemos referencia a tres nombres: botón, pantalla y texto. La pantalla es el área de trabajo, y ya hemos visto que está en el centro de la ventana de trabajo. Así que los dos elementos que faltarían son el botón y el texto. Casi todas las aplicaciones hacen uso de estos elementos, así que forman parte de la paleta de interfaz de usuario. Búscalos en la columna de la izquierda, teniendo en cuenta que el componente de texto se llama Etiqueta.

1.2. Construyendo tu primera App

Ahora sí, vamos a pasar a construir la App. Para añadir el botón y el texto (etiqueta) a la aplicación, el procedimiento es muy sencillo. Tan sólo tienes que seleccionarlos en la paleta y arrastrarlos hasta la pantalla del móvil, soltándolos en su interior. El aspecto resultante será parecido al de la figura 1.5. No te preocupes de momento si no puedes controlar dónde se colocan, y si tampoco son muy atractivos. Eso lo resolveremos en próximos capítulos.

Haz clic ahora sobre el botón para seleccionarlo. Como verás quedará recuadrado, pero además se marcará en la columna de componentes, tal y como se aprecia en la figura 1.5. Además, en la columna de más a la derecha (columna *propiedades*) se pueden ver las características del botón. Cada vez que se selecciona algún elemento de la pantalla, las propiedades se actualizan para mostrar las correspondientes a ese elemento. Busca dentro de las propiedades la caja de texto llamada *Texto*. Haz clic con el ratón sobre ella, y teclea lo siguiente: *Púlsame*. Comprueba cómo al hacerlo ha cambiado el texto que aparecía sobre el botón.

Repite el mismo procedmiento para editar el texto asociado al componente Etiqueta1, es decir, busca su propiedad *Texto*, pero ahora borra el contenido. Al hacerlo te parecerá que ha desaparecido de la pantalla, pero no te preocupes, sigue allí, pero está vacío.

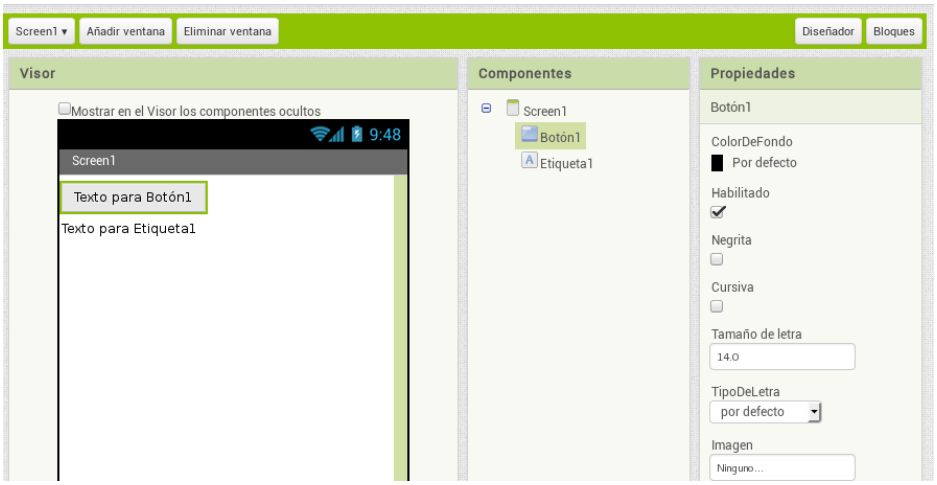

Figura 1.5: Listado de componentes y sus propiedades.

Regresemos a la columna de componentes. Aquí se muestra en todo momento todo aquello que hemos colocado en la pantalla. De momento verás que hay una lista que contiene las palabras Screen1, Botón1 y Etiqueta1. Esto se refiere a que en la pantalla del móvil, que se llama Screen1, hemos colocado un botón llamado Botón1 y un texto, llamado Etiqueta1. Los nombres de los componentes los pone por defecto AppInventor, pero suele ser buena práctica cambiarlos por otros que te den una pista de qué hacen en tu aplicación. En nuestro caso, vamos a renombrar Botón1 a BotonPulsame, mientras que Etiqueta1 pasará a ser TextoHola.

Para renombrar un componente primero debes hacer clic sobre su nombre en la lista de la columna *Componentes*, o sobre su dibujo en la pantalla. Después debes pulsar el botón Cambiar nombre que hay al pie de la columna *Componentes*. Al hacerlo aparecerá un cuadro de diálogo donde se indica el *Nombre anterior* y el que se quiere utilizar como *Nuevo nombre*. Renombra pues el botón y el texto a los nuevos BotonPulsame y TextoHola, y comprueba cómo en la lista de componentes aparecen también esos nuevos nombres.

Una vez colocados todos los elementos necesarios para completar nuestra aplicación, vamos a pasar a conectarlos entre sí para que hagan algo útil. Para ello necesitamos abrir una nueva ventana, el **editor de bloques**, y lo haremos pulsando el botón que con el texto *Bloques* encontrarás en la parte superior derecha del área de trabajo.

Si todo ha funcionado correctamente verás como aparece una nueva ventana, que será donde indicaremos qué queremos que haga el botón

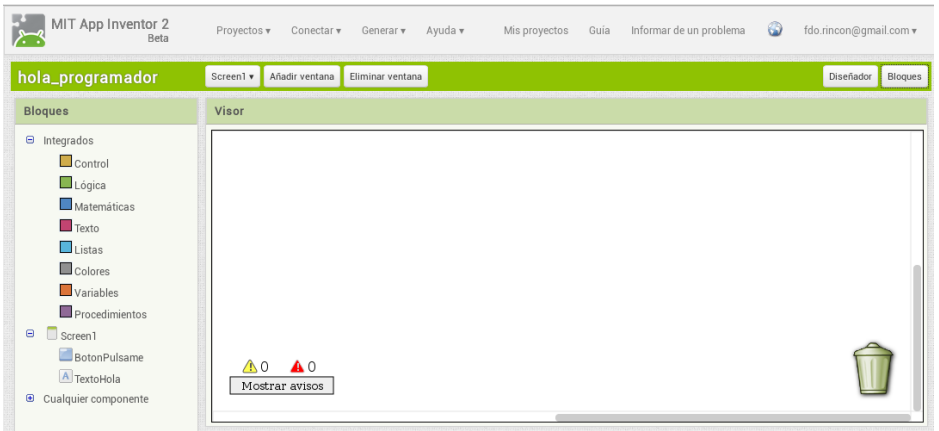

Figura 1.6: Editor de bloques.

al ser pulsado (figura 1.6).

La manera en la que se indica cuál debe ser el funcionamiento de la aplicación se inspira en los juegos de construcción, en los que se dispone de un conjunto de bloques de distintas formas y tamaños, y que únicamente encajan de cierta manera. Aunque estos bloques sean muy simples, mediante la combinación de muchos de ellos pueden construirse creaciones realmente complejas e impresionantes.

Busca a izquierda del editor de bloques la columna *Bloques*. Dentro de ella hay 3 categorías, que son: *Integrados*, *Screen1* y *Cualquier componente*. En la categorí *Screen1* encontrarás los componentes correspondientes al botón y la etiqueta. Haz clic sobre BotonHola y verás como a su derecha aparecen un montón de bloques de colores (figura 1.7).

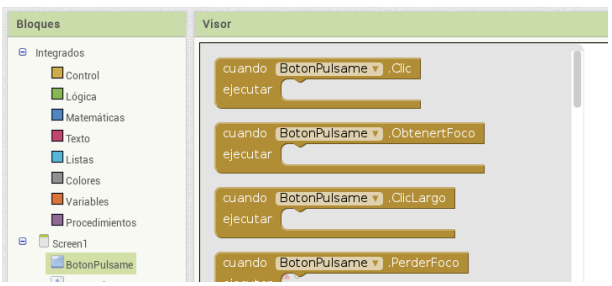

Figura 1.7: Bloques asociados al botón.

Elige el primero de ellos que tiene en negrita el texto **BotonPulsame.Clic**. En realidad si te fijas bien, el texto completo es **cuando BotonPulsame.Clic ejecutar**. Lo que en realidad quiere decir: "cuando hagas clic sobre el botón BotonPulsame . . . .". Selecciona este bloque y arrástralo hasta una zona libre de pantalla.

Podríamos completar la frase con algo así como "escribe el texto hola, programador en el componente TextoHola". Si seleccionas ahora el componente TextoHola, verás que en los bloques asociados que aparecen a su derecha hay precisamente uno con un mensaje muy similar al que buscamos: **poner TextoHola.Texto como**. Arrástralo hasta colocarlo dentro del bloque anterior, tal y como muestra la figura 1.8.

Figura 1.8: Composición del comportamiento.

Como puedes apreciar, el bloque de color verde (el bloque *poner*) está incompleto, y queda un hueco para encajar alguna cosa más, que en este caso debería ser el texto que queremos mostrar. Como los textos, números, condiciones, etc, son elementos básicos que se utilizan en todos los programas, hay una categoría especial que los contiene, y a la que se denomina *Integrados*. Busca dentro de ella la subcategoría *Texto*. Haz clic sobre ella y elige el primer bloque que aparece en la lista, en el que hay un hueco entrecomillado. Arrástralo después hasta conectarlo tal y como se indica en la figura 1.9.

Edita ahora el texto del bloque y teclea *hola, programador*. Vamos a leer ahora el texto que hemos construido con los bloques, y que dice lo siguiente: *cuando BotonPulsame.Clic ejecutar poner TextoHola.Texto como 'hola, programador'*. No hacen falta más explicaciones para saber qué es lo que hace la aplicación, ¿no?.

Figura 1.9: Comportamiento completo.

1.3. Probando la aplicación

Como último paso vamos a comprobar el correcto funcionamiento de la aplicación. Aquí tienes dos alternativas: puedes utilizar un dispositivo móvil Android, o si no dispones de ninguno, puedes hacer uso del emulador que se distribuye como parte del software de AppInventor, y que se parece mucho a un teléfono real.

Vamos a optar en primer lugar por probar la aplicación en el emulador, para lo cual tienes que seleccionar la opción *Conectar*, en la barra superior de la página. Despliégala y elije la opción *Emulador*. Al hacerlo aparecerá un mensaje que te pedirá paciencia, puesto que el emulador puede tardar un par de minutos en arrancar. Tras ese período de espera, tendrás en pantalla un móvil virtual como el de la figura 1.10.

Ya sólo queda poner en marcha la App, tarea que se realiza también desde el menú *Conectar*, en la barra superior de la página. Despliégalo y elije la opción emulador.

la ventaja del editor de bloques. Junto al botón New Emulator verás una caja de texto con la leyenda (conectar al dispositivo). Pincha sobre ella y elige, de entre las distintas opciones que aparecerán, la del emulador. Verás que el icono del teléfono que hay a la derecha comienza a parpadear en amarillo, hasta que pasa al color verde. En ese instante estará la aplicación cargada. Tan sólo tienes que desbloquear el emulador y probar tu App.

Puedes ahora probar a variar el texto que se muestra por cualquier otro de tu elección. En ese caso, como el emulador seguirá conectado al editor de bloques, tu app se cargará automáticamente sin que tengas que hacer nada más.

¡Enhorabuena, ya eres un desarrollador de Apps!

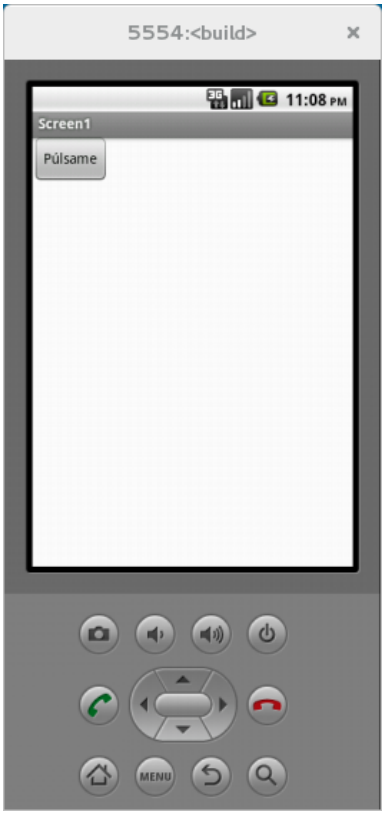

Figura 1.10: Emulador de dispositivos Android.

Figura 1.11: Aspecto de la aplicación en el emulador.

Capítulo 2

El Xilófono de Colores

En este capítulo vamos a explorar tus habilidades artísticas construyendo un sencillo Xilófono mediante App Inventor para tu teléfono móvil Android. Mediante esta aplicación podrás tocar tus canciones preferidas y algún capítulo más adelante incluso grabarlas para reproducirlas posteriormente.

2.1. ¿Qué voy a construir?

La figura 2.1 muestra una captura de pantalla del resultado final que obtendrás tras la realización de los pasos que se muestran en este tutorial.

Más concretamente, la aplicación que vas a construir te permitirá reproducir 8 notas musicales diferentes pulsando botones coloreados en la pantalla de tu dispositivo móvil.

A lo largo del desarrollo de este proyecto trabajarás con los siguientes conceptos y componentes:

- Uso de un componente Sound para reproducir ficheros de audio.

Figura 2.1: Aspecto final de la App xilófono.

- Uso del componente Clock para medir los intérvalos de tiempo entre acciones.
- Decisión sobre cuándo utilizar procedimientos.

Recuerda que para comenzar a desarrollar el proyecto, debes conectarte a la página web de APP Inventor e identificarte con tu cuenta de usuario de Google.

Después crea un nuevo proyecto y llámalo *xilofono*. Por último, abre el editor de bloques y conéctalo con tu emulador o con tu dispositivo móvil.

2.2. Diseño de los componentes

Esta App utiliza 12 componentes (tranquilo, 8 de ellos son las teclas del teclado del xilófono) que se muestran en la tabla 2.1. Como son bastantes y sería aburrido crearlos todos antes de empezar a escribir el

programa, vamos a dividir la App en sus diferentes funcionalidades, y la iremos construyendo parte por parte. Será necesario ir pasando del diseñador al editor de bloques varias veces.

Tipo	**Categ.**	**Nombre**	**Objetivo**
Botón	I.U.[a]	BotonDO	Reproducir un DO
Botón	I.U.	BotonRE	Reproducir un RE
Botón	I.U.	BotonMI	Reproducir un MI
Botón	I.U.	BotonFA	Reproducir un FA
Botón	I.U.	BotonSOL	Reproducir un SOL
Botón	I.U.	BotonLA	Reproducir un LA
Botón	I.U.	BotonSI	Reproducir un SI
Botón	I.U.	BotonDO8	Reproducir un DO (2ª octava)
Sonido	Medios	Nota	Reproducir las notas
Botón	I.U.	BotonPlay	Reproducir melodía almacenada
Botón	I.U.	BotonReset	Borrar memoria
Reloj	Sensores	Temporizador	Conseguir intervalos entre notas

Tabla 2.1: Lista completa de componentes utilizados en el juego *Xilófono de Colores*. Por cada componente se indica el tipo, la categoría en la que está, el nombre que tiene en el juego y su objetivo (para qué se ha utilizado).

[a] Interfaz de Usuario

2.3. Crear el teclado

En esta sección vamos a crear un teclado de 8 notas para contar con una escala completa más el DO de la siguiente escala.

2.3.1. Primeras notas del teclado

Vamos a crear las 2 primeras notas del teclado (DO y RE), que serán 2 botones.

1. En la categoría *Interfaz de usuario* del diseñador, encontrarás el componente Botón. Arrastra uno hasta la pantalla de la App, y renómbralo como *BotonDO* a través de la del botón Cambiar nombre en el área de componentes. Vamos a convertirlo en una barra larga morada, como una tecla de un xilófono real, para ello hay que establecer sus propiedades según la siguiente lista:
   - Cambia la propiedad *ColorDeFondo* a Violeta.
   - Cambia la propiedad *Texto* a 'DO'.
   - Cambiar el *Ancho* a '*Ajustar al contenedor*' para que ocupe todo el ancho de la pantalla.
   - Cambia el *Alto* a *40 pixels*.
2. Repite los pasos con un nuevo botón (BotonRE) poniéndolo debajo del anterior. Cambia el ancho, el alto, pon su propiedad *ColorDeFondo* a *Rojo* y cambia el texto por *RE*.

Más adelante completaremos el resto del teclado con más botones. Ahora mismo, tu proyecto debe parecerse a imagen que se muestra en la figura 2.2.

Figura 2.2: Primeras teclas del xilófono

2.3.2. Añadir el componente *Sound*

No se concibe un xilófono sin sonido ¿verdad? Entonces, crea un componente Sonido (renómbralo a Nota) y cambia su propiedad *IntervaloMinimo* a 0 milisegundos, así no tendremos que esperar entre un sonido y otro. Cuando arrastres el componente hacia el visor, observa cómo aparece en la parte inferior, al tratarse de un componente no visible.

Después tienes que cargar los archivos de sonido para poder reproducir las notas[1]). Debes guardar en tu ordenador una copia de todos los archivos wav (*1.wav*, *2.wav*, etc.). Después, en el diseñador, en el área de *Medios* (bajo el área de componentes) haz clic en *Subir archivo...* y busca en tu ordenador el fichero 1.wav, recién descargado. Después haz lo mismo para el *2.wav*, el *3.wav* y así sucesivamente. Deberías obtener algo similar a lo que se muestra en la figura 2.3.

Figura 2.3: Carga de archivos de audio.

2.3.3. Conectar los sonidos a los botones

El comportamiento que tenemos que programar implica que se reproduzca un fichero de audio cada vez que pulsemos un botón. Si pulsamos BotonDO debe sonar *1.wav*, si pulsamos BotonRE sonará *2.wav* y así sucesivamente. Vamos a utilizar el editor de bloques como se explica a continuación (hasta obtener un diseño como el de la figura 2.4):

[1]Disponibles en la URL http://www.esi.uclm.es/appinventor/download/cap6_xilofono.zip

1. En la categoría *Screen1* del menú de bloques pulsa BotonDO y arrastra al editor un bloque **cuando BotonDO.Clic**.

2. Pulsa *Nota* y arrastra un bloque **poner Nota.Origen como** dentro del bloque anterior.

3. Escribe *texto* sobre el área en blanco para que aparezca un bloque de texto. Cambia el valor de este bloque a **1.wav** y encájalo en el bloque **poner Nota.Origen como**.

4. Añade un bloque **llamar Nota.Reproducir** debajo de los anteriores.

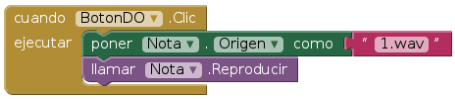

Figura 2.4: Bloque para que suene un sonido al pulsar un botón.

Haz lo mismo para el BotonRE, pero cambiando el texto del bloque *Texto* por *2.wav*. Debes haber conseguido algo similar a lo que se muestra en la figura 2.5.

Podríamos repetir el mismo proceso para el resto de botones, pero como verás resultaría algo muy repetitivo, ¿verdad? Bien, un comportamiento que se repite en múltiples ocasiones, es señal de que se debe utilizar un procedimiento. Un procedimiento sería equivalente a crear un bloque a partir de otros ya existentes, con la idea de no tener que repetir una y otra vez las mismas operaciones. En su lugar se sustituyen todas estas operaciones por la llamada al nuevo bloque creado. En ocasiones, además de llamar al bloque, es necesario pasar alguna información extra, y que se conoce con el nombre de argumento.

Vamos a crear pues un procedimiento que reproduzca una nota, para lo que deberá recibir un número como argumento de entrada. Dentro del procedimiento incluiremos las operaciones de seleccionar el fichero fuente de sonido (obtenido a partir del argumento de entrada), y la reproducción del sonido propiamente dicha:

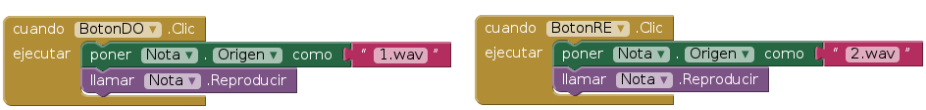

Figura 2.5: Más bloques de sonido.

Figura 2.6: Definición del procedimiento

1. en primer lugar, haz clic sobre la subcategoría *Procedimientos*, que pertenece a la categoría de bloques *Integrados*. Elige el bloque **como procedimiento ejecutar**, y arrástralo hasta el área de diseño.

2. Renombre el bloque a *reproducirNota*.

3. Para añadir el parámetro de entrada debes pulsar sobre el icono azul que hay en la parte superior izquierda del bloque. Verás cómo se despliega un bocadillo con 2 partes.

4. En la parte izquierda se muestra un bloque **entrada**, selecciónalo y arrástralo hasta incluirlo dentro del bloque **entradas** de la parte derecha. De esta forma se crea un nuevo argumento, que por defecto tiene el nombre *x*.

5. Ahora renombra el argumento a *número*, con lo que deberías obtener un bloque con el aspecto del de la figura 2.6.

Completaremos ahora la funcionalidad del nuevo bloque, reutilizando los bloques incluidos dentro del clic de los botones:

1. Arrastra el bloque **poner Nota.Origen como** de **BotonDO.Clic** al procedimiento a la derecha de la palabra **ejecutar** (el bloque de debajo se moverá a la vez).

2. Lleva el bloque de texto **1.wav** a la papelera.

3. Arrastra un bloque **unir** de la subcategoría *Texto* al hueco de **Nota.Origen**.

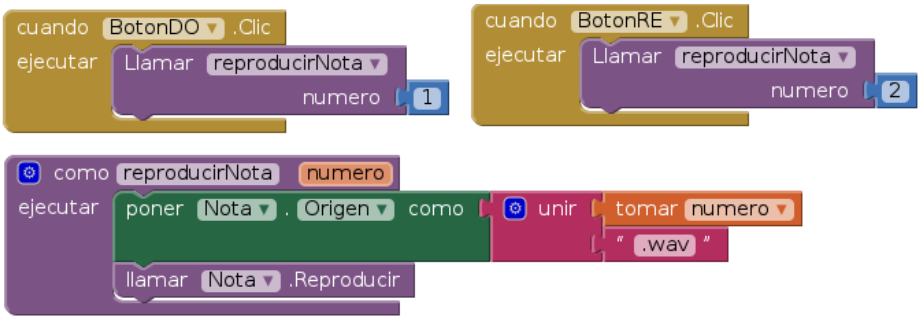

Figura 2.7: Procedimiento para tocar una nota.

4. Sitúa el cursor sobre el argumento *numero*, y verás cómo aparecen 2 nuevos bloques junto a él. Selecciona **tomar numero** y llévalo al primer hueco del bloque **unir**.
5. En el otro hueco, pon un bloque **texto** con el valor *.wav*

El último paso consiste en realizar la llamada al nuevo bloque (procedimiento) desde los bloques **Clic** de los botones. Encontrarás el bloque para la llamada en la subcategoría *Procedimientos*: **llamar reproducirNota numero**. En el hueco del número debes colocar el número **1** si se trata del boton DO, y el **2** para el RE. El aspecto final debe ser el de la figura 2.7.

2.3.4. Inicializar los sonidos

Si has tratado de ejecutar tu App, puede que te haya costado oír los sonidos porque hayan tardado mucho en cargar, o que te haya aparecido algún mensaje de error. Esto se debe a que Android necesita cargar los sonidos en memoria en tiempo de ejecución y eso tarda un tiempo. Esto se produce la priera vez que se le asigna un *Origen* concreto a la *Nota*, cosa que no sucede hasta que se pulsa cada botón. Una forma sencilla de resolverlo es utilizar el bloque de inicialización de la pantalla, que el primero en ser ejecutado al iniciarse la aplicación. Podemos incluir en este bloque una primera asignación para todos los posibles orígenes, tal y como se muestra en la figura 2.8. Prueba a hacerlo y verás como ahora funciona mejor.

2.3.5. El resto del teclado

Ya que tenemos 2 notas funcionando, te debería ser sencillo completar el teclado con las otras 6 notas que faltan. Créalas como se vio

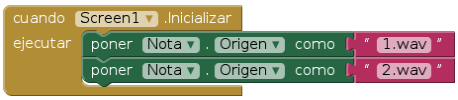

Figura 2.8: Inicialización de los ficheros de sonido.

anteriormente dándole los siguientes valores a las propiedades *Texto* y *ColorDeFondo* de cada botón de la siguiente lista:

- BotonMI : Rosa
- BotonFA : Naranja
- BotonSOL : Amarillo
- BotonLA : Verde
- BotonSI : Turquesa
- BotonDO8 : Azul

Quizá también te convenga cambiar la propiedad *ColorDeTexto* del ButtonDO8 a *Blanco* para que se vea mejor.

Fíjate en la figura 2.9. Comprueba que también le hemos cambiado el nombre a la App (propiedad *Title* de *Screen1*) y ahora se llama "Xilófono".

Ahora en el editor de notas debes completar los bloques **Clic** de cada botón con llamadas a *reproducirNota*. No te olvides de inicializar todos los ficheros de audio.

Si empiezas a quedarte sin espacio en el editor de bloques, puedes minimizar cada bloque haciendo clic con el botón derecho del ratón sobre él, y seleccionando la opción pulsando en el cuadro que aparece en la parte superior izquierda. La figura 2.10 muestra la situación actual con el procedimiento *ReproducirUnaNota* minimizado.

Con esto hemos completado el tutorial para la creación de tu propio xilófono para dispositivos móviles. Ahora sólo te queda deleitar a quien quieras con tus canciones favoritas.

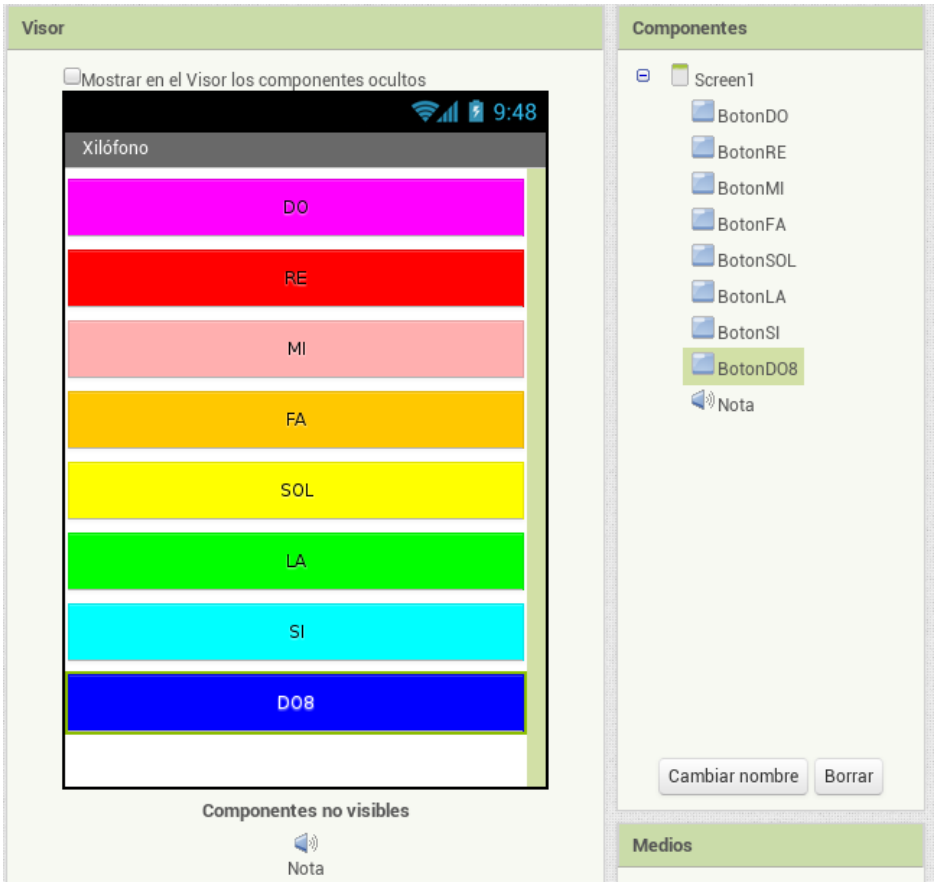

Figura 2.9: Teclado del xilófono completo.

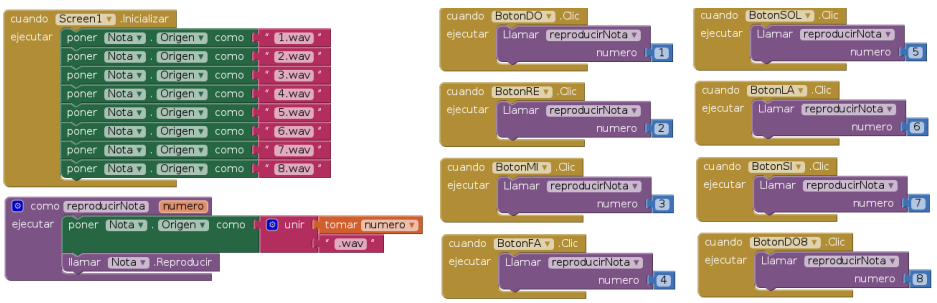

Figura 2.10: Bloques para el teclado completo.

Capítulo 3

Whack A Zombie

Este capítulo muestra cómo crear un sencillo juego tipo *whack-a-mole*[1] mediante App Inventor para tu teléfono móvil Android. Este juego está inspirado en el clásico juego arcade que tiene como objetivo *machacar* con un mazo a los topos que van saliendo de sus madrigueras para conseguir la mayor cantidad posible de puntos. A través de este sencillo ejemplo, comprobarás lo fácil que es crear un juego totalmente funcional.

3.1. ¿Qué voy a construir?

La figura 3.1 muestra una captura de pantalla del juego que serás capaz de desarrollar para Android al finalizar la lectura de este capítulo. Como puedes observar, el aspecto final de este juego, bautizado como ***Whack-A-Zombie***, está personalizado para *machacar* zombies en lugar de topos[2].

[1] http://en.wikipedia.org/wiki/Whac-A-Mole

[2] Los autores del libro tenemos más simpatía por los topos que por los zombies.

Figura 3.1: Captura de pantalla del juego *Whack-A-Zombie* discutido en este capítulo y ejecutado en un teléfono con Android 2.3. En la parte superior se muestra la barra de energía y la puntuación del jugador, las cuales se actualizan cuando pulsamos sobre la pantalla. En el centro se muestra la zona de juego, en la que el zombie se va moviendo de manera aleatoria cada cierto tiempo. En la parte inferior aparece el botón de Reset para comenzar a jugar de nuevo, reseteando la puntuación y la etiqueta de energía.

En concreto, la **funcionalidad** que implementarás será la siguiente:

- Visualización de un zombie en movimiento que, de manera aleatoria, aparece en diversas posiciones de la pantalla con un intervalo de un segundo.
- Tratamiento de la interacción táctil con el juego.
  - Si consigues *machacar* al zombie, entonces el juego reproducirá un sonido, el teléfono vibrará, el zombie aparecerá en otro lugar y la puntuación se incrementará en una unidad.
  - Si fallas al *machacar* al zombie, entonces el nivel de energía se verá reducido.
- Reseteo del juego, a través de un botón, para poder jugar de nuevo.
- Actualización de la información del juego, es decir, puntuación y nivel de energía.

A través del desarrollo del tutorial de este capítulo **aprenderás a manejar los siguientes componentes**:

- Etiqueta para mostrar texto en el teléfono.
- DisposiciónHorizontal para alinear etiquetas de texto.
- SpriteImagen para imágenes dinámicas con las que interactuar a través de la pantalla del teléfono.

Tipo	Categ.	Nombre	Objetivo
Lienzo	Dibujo	Lienzo	Contenedor para el sprite del zombie
SpriteImagen	Dibujo	Zombie	Zombie a *machacar*
Reloj	Sensores	Reloj	Control del movimiento del zombie
Botón	I.U.	BotonReset	Nuevo juego
Sonido	Medios	SonidoImpacto	Reproducir sonido y vibrar cuando se golpea al zombie
Sonido	Medios	SoundFin	Reproducir sonido al terminar el juego
Etiqueta	I.U.	EtiquetaEnergia	Mostrar el texto 'Energía'
Etiqueta	I.U.	EtiquetaPuntos	Mostrar el texto 'Puntos'
Etiqueta	I.U.	EtiquetaBarra	Mostrar la barra de energía
Etiqueta	I.U.	EtiquetaPuntuacion	Mostrar el número de puntos

Tabla 3.1: Lista completa de componentes utilizados en el juego *Whack-A-Zombie*. Por cada componente se indica el tipo, la categoría en la que encontrarlo, cual será su nombre en el juego *Whack-A-Zombie*, y su objetivo (para qué se ha utilizado).

- Lienzo como superficie para colocar el componente SpriteImageSn.
- Reloj para controlar cuándo cambiar al zombie de posición.
- Sonido para reproducir sonidos y generar vibraciones en el teléfono.
- Botón para iniciar un nuevo juego.

La tabla 3.1 muestra el listado completo de componentes utilizados para desarrollar el juego *Whack-A-Zombie*.

Además, serás capaz de **implementar la siguiente funcionalidad**:

- Mover al zombie de manera aleatoria.
- Detectar cuándo se ha *machacado* el zombie y cuándo no.
- Reproducir sonido y vibraciones en tu teléfono móvil.
- Manejar contadores para la energía y la puntuación.

3.2. Diseño de la interfaz paso a paso

En esta sección aprenderás a colocar todos los componentes que forman parte del juego *Whack-A-Zombie*. Para ello, describiremos paso a paso las distintas acciones que has de realizar para obtener el aspecto del juego final que se muestra en la figura 3.1.

3.2.1. La pantalla de juego

En primer lugar, arrastra un nuevo Lienzo disponible en la paleta *Dibujo y animación*. Puedes renombrarlo a *Lienzo*. A continuación, ajusta el *Ancho* y el *Alto* al valor *Ajustar al contenedor* para que se ajuste automáticamente a la resolución de tu teléfono móvil. Ahora ya puedes añadir la imagen *background.jpg* de fondo a través del campo *ImagenDeFondo*.

El siguiente paso consiste en añadir la imagen o *sprite* asociada a nuestro zombie. Para ello, arrastra un SpriteImagen desde la paleta *Dibujo y animación*. Cámbiale el nombre por el de Zombie. Ahora ya puedes establecer la imagen del zombie a través del campo *Foto*, utilizando *zombie.png*.

Finalmente, añade un botón *Reset* que servirá para reanudar el juego una vez terminado. Para ello, arrastra un nuevo Botón desde la paleta *Interfaz de usuario* hasta justo debajo del *Lienzo*. Puedes renombrarlo a BotonReset, y cambiar el texto que aparece por defecto.

En este punto, el resultado de la interfaz de tu juego debería ser similar al que se muestra en la figura 3.2, aunque quizás la posición del zombie sea otra distinta.

3.2.2. La barra de energía y la puntuación

A continuación, vamos a añadir al juego la barra de energía y el marcador de puntuación. Estos componentes se actualizarán a medida que el juego progrese según el número de aciertos y fallos a la hora de *machacar* el zombie.

Para ello, arrastra dos componentes de tipo DisposiciónHorizontal desde la paleta *Disposición* justo por encima del componente Lienzo. Este componente, como su nombre indica, nos va a servir para alinear, de manera horizontal, otros componentes en su interior. Renombra el de la parte superior a BarraVida, mientras que el otro será BarraPuntuacion. En ambos casos, puedes dejar el ancho y el alto a valores automáticos, ya que se ajustarán a continuación con los componentes que gestionen.

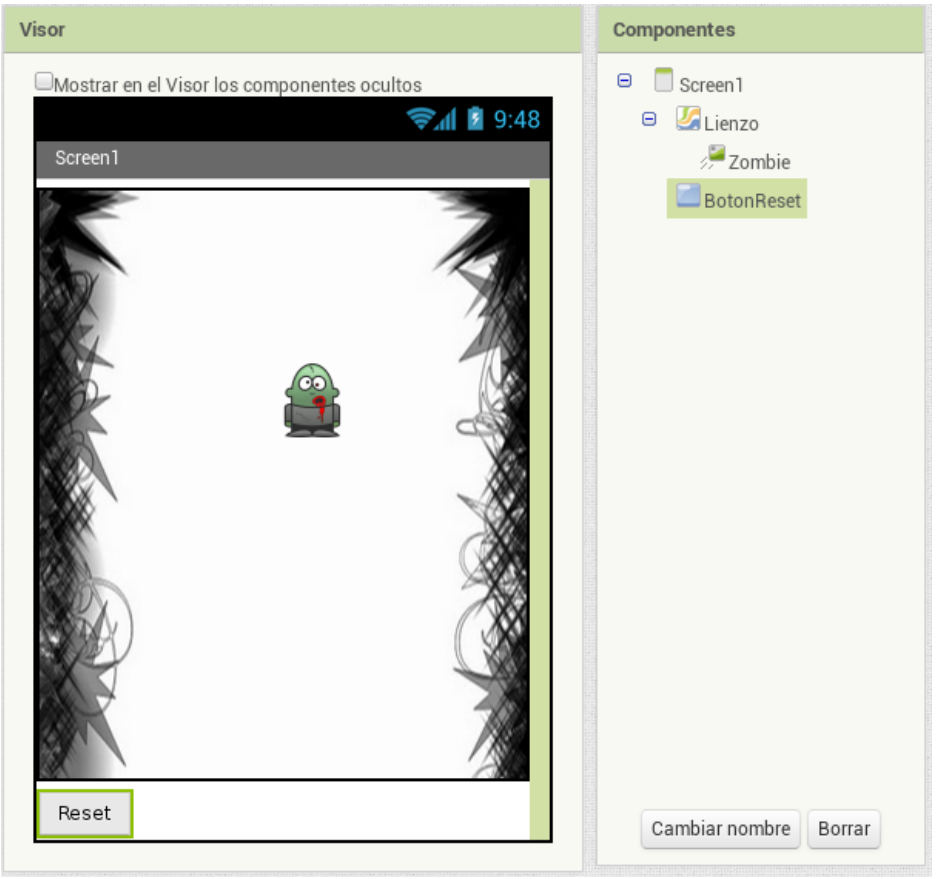

Figura 3.2: Interfaz gráfica inicial del juego *Whack-A-Zombie*.

Ahora, arrastra dos nuevas Etiquetas hasta el interior de BarraVida. La de la parte izquierda puedes renombrarla a EtiquetaEnergia, estableciendo las siguientes propiedades (parte derecha de la interfaz de App Inventor): *Negrita* activada, *Tamaño de letra* a 24.0, *TipoDeLetra* a *serif*, *Texto* a "*Energía:*" y, finalmente, *ColorDeTexto* a *Gris oscuro*.

Por otra parte, puedes renombrar la etiqueta de la parte derecha que acabas de integrar en BarraVida a EtiquetaBarra. En este caso, puedes establecer las siguientes propiedades: *ColorDeFondo* a *Azul*, *Ancho* a *100 pixeles* y *Alto* a *20 pixeles*. Borra después el contenido del campo *Texto*.

De nuevo, arrastra dos nuevas Etiquetas hacia el componente BarraPuntuacion. Para ambas, puedes utilizar las propiedades previamente establecidas para la etiqueta EtiquetaEnergia. Puedes renombrarlas

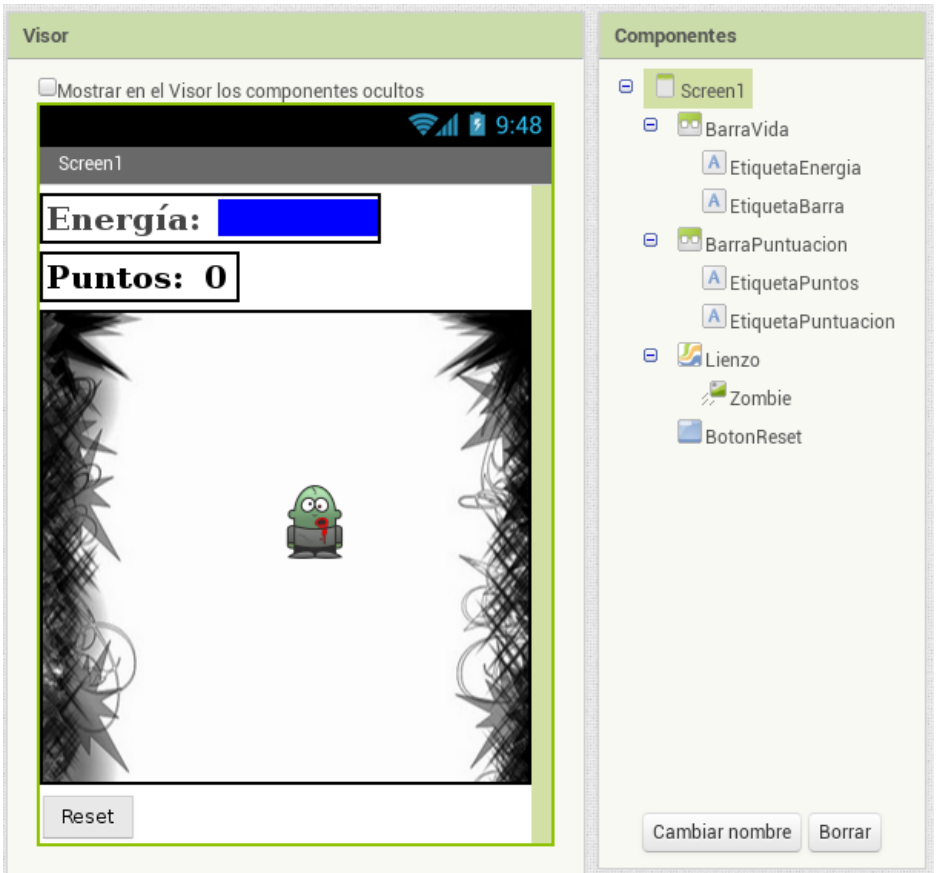

Figura 3.3: Interfaz gráfica inicial del juego *Whack-A-Zombie* integrando los componentes para la barra de energía y la puntuación.

a EtiquetaPuntos y EtiquetaPuntuacion y establecer el campo *Texto* a "*Puntos:*" y *0*, respectivamente.

En este punto, el resultado de la interfaz de tu juego debería ser similar al que se muestra en la figura 3.3.

3.2.3. El sonido y el reloj del juego

Llegados a este punto, el aspecto de la interfaz gráfica del juego ya está completado. Simplemente nos queda añadir tres componentes no visibles que servirán para reproducir sonidos en el juego y controlar cuándo éste termina.

En primer lugar, añade un nuevo Reloj desde la paleta *Sensores* y renómbralo a Reloj. Comprueba como el componente se sitúa bajo la interfaz del juego en el área de *Componentes no visibles*.

Por otra parte, añade dos nuevos Sonidos desde la paleta *Medios*. Renombra el primero a SonidoImpacto y asóciale el sonido *hit.wav* a través de la propiedad *Source*. Renombra el segundo a SonidoFin y asóciale el sonido *zombie.wav*. Estos dos componentes servirán para reproducir efectos de sonido al golpear un zombie y al terminar el juego, respectivamente.

La figura 3.4 muestra el aspecto final de la interfaz gráfica del juego *Whack-A-Zombie*. Ahora ya puedes añadir el comportamiento utilizando el editor de *Bloques*. Éste es precisamente el tema de la siguiente sección.

3.3. Defínción del comportamiento del juego

Una vez creados los distintos componentes del juego, el siguiente paso consiste en definir el comportamiento asociado a ellos, es decir, lo que debe ocurrir cuando el jugador interactúa con el juego. Básicamente, este comportamiento se puede resumir a través de los siguientes puntos:

- El zombie se moverá aleatoriamente por la pantalla de juego.
- El usuario intentará *machacar* al zombie. Ante una pulsación del jugador sobre el teléfono, pueden darse dos situaciones:
  1. Si el jugador pulsa sobre el zombie, entonces se actualiza la puntuación, se reproduce un sonido y se emite una vibración.
  2. Si el jugador falla al pulsar sobre el zombie, entonces se actualiza la barra de energía (reduciendo su tamaño).
- Una barra de energía vacía implica que el juego termina, finalizando la animación del zombie.
- El botón Reset permite reanudar el juego, es decir, establece la puntuación a 0, rellena la barra de energía y reanima al zombie.

3.3.1. Animación del zombie

Al igual que App Inventor ya proporciona funciones para, por ejemplo, reproducir un sonido mediante la función *play* del componente Sound, resulta muy interesante definir nuestras propias funciones o

procedimientos para poder utilizarlos cuando sea necesario. Un ejemplo podría ser el movimiento del zombie, es decir, podríamos crear un procedimiento que cada vez que fuera ejecutado recolocara al zombie en una nueva posición. Con esta idea en mente, hemos creado el procedimiento *MoverZombie*, cuyo diagrama de bloques se muestra en la figura 3.5.

Para poder mover el zombie correctamente, hay que tener en cuenta que su posición en la pantalla del teléfono no sea mayor que su resolución. En otras palabras, la posición de nuestro zombie en todo momento no debe estar fuera de las dimensiones del componente Lienzo que creamos previamente. La figura 3.6 muestra los valores de las coordenadas X e Y considerados para llevar a cabo el movimiento del zombie.

La construcción del diagrama de bloque se iniciará arrastrando un bloque de tipo **como procedimiento ejecutar** de la categoría *Procedimientos*, tal y como se muestra en la figura 3.5. Renómbralo a a *MoverZombie*. Ahora ya puedes integrar una llamada al bloque **llamar Zombie.MoverA**, que encontrarás al seleccionar dentro del área de bloques al componente Zombie.

En este punto, ya tenemos disponible la **base para mover el zombie** a una nueva posición. Ahora es necesario, precisamente, generar de manera aleatoria dicha posición, es decir, es necesario generar un nuevo valor para las coordenadas X e Y del zombie, teniendo en cuenta que no sobrepasen las dimensiones de la pantalla.

Haz *clic* sobre la categoría *Matemáticas* dentro de los bloques *Integrados*. Ensambla ahora un bloque del tipo **entero aleatorio entre y** al parámetro *x* de **Zombie.MoverA**. El valor aleatorio de la nueva posición en X del zombie ha de ser un entero comprendido entre 1 y la anchura del canvas (restando el tamaño del zombie para evitar que pueda quedar oculto). Para ello, ensambla un bloque del tipo **número** (también *Matemáticas* en el parámetro **entre** del **entero aleatorio** que acabas de añadir y asígnale un valor de 1.

Por otra parte, ensambla un bloque de resta, como se muestra en la figura 3.5, donde la primera parte tenga la anchura del canvas y la segunda la anchura del zombie en la rama *y* del **entero aleatorio**.

Finalmente, para generar la coordenada Y del zombie puedes seguir el mismo esquema, utilizando simplemente la altura del Lienzo y del Zombie. Deberías obtener un resultado similar al que se muestra en la figura 3.5.

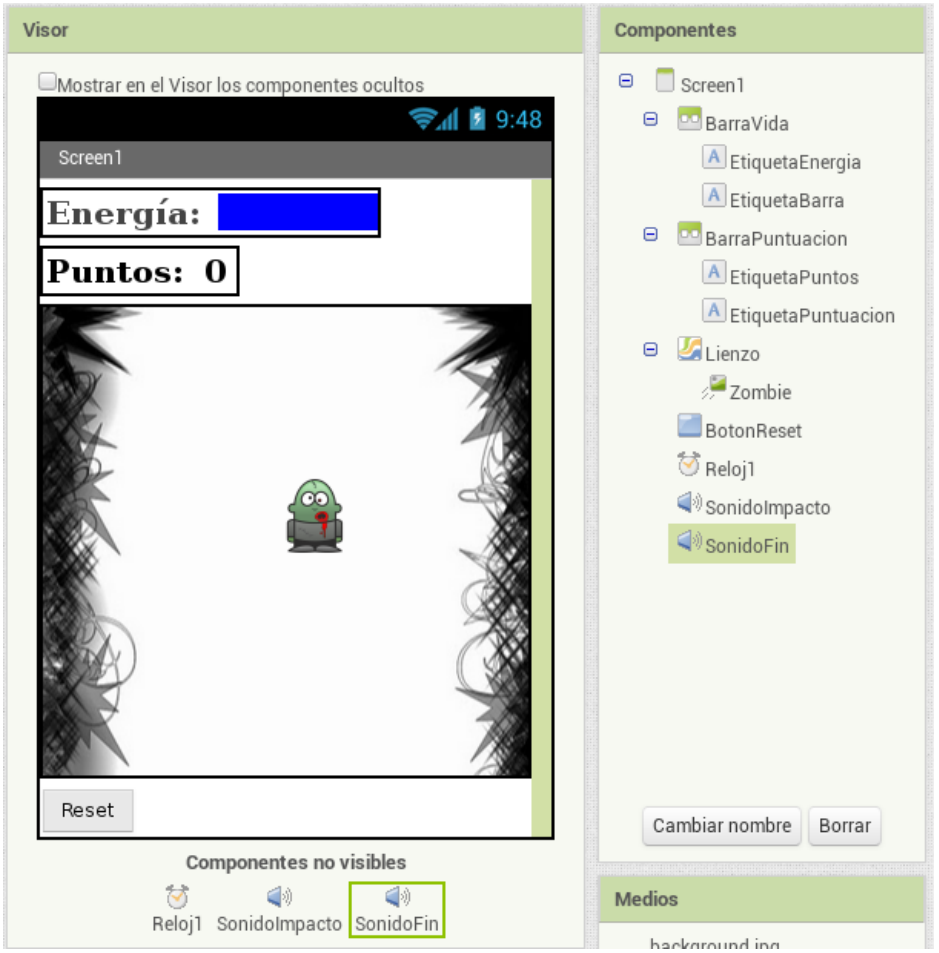

Figura 3.4: Aspecto final del diseño del juego *Whack-A-Zombie*.

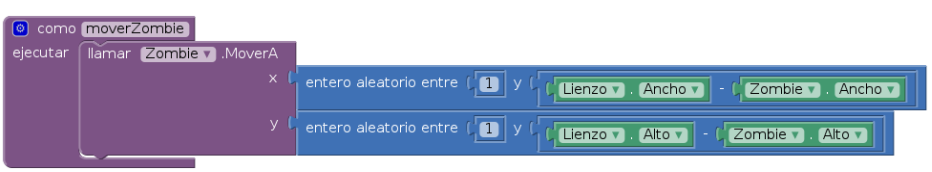

Figura 3.5: Procedimiento *MoverZombie* para colocar el zombie, de manera aleatoria, en distintas posiciones de la pantalla.

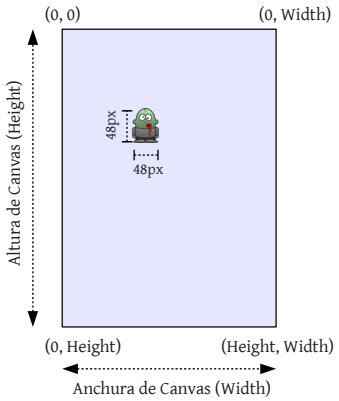

Figura 3.6: Representación gráfica de la posición del zombie dentro del Lienzo. Las dimensiones de este componente están representadas por su *Ancho* y *Alto*. La coordenada de la parte superior izquierda del Lienzo está representada por el valor $(0, 0)$, es decir, 0 unidades en el eje X y 0 unidades en el eje Y. La coordenada de la parte inferior derecha está representada por el valor $(Ancho, Alto)$, es decir, *Ancho* unidades en el eje X y *Alto* unidades en el eje Y.

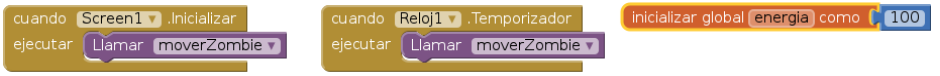

Figura 3.7: Arranque de la aplicación y asignación de valores iniciales.

3.3.2. Arranque del juego

Una vez modelado el bloque para mover al zombie, ya es posible utilizarlo. En este contexto, el comportamiento esperado del juego que estás desarrollando se basa en que el zombie comience a moverse justo al arrancar la aplicación. Para ello, utiliza el bloque **Screen1.Inicializar**, tal y como se muestra en la parte izquierda de la figura 3.7 (búscalo haciendo clic sobre la categoría *Screen1* dentro del área de bloques. Después, ensambla en este bloque el procedimento **MoverZombie**, disponible en la categoría *Procedimientos*.

Además, tienes que asegurarte que **el zombie continúe moviéndose** sin parar cada cierto tiempo sobre la pantalla mediante el componente Reloj definido anteriormente. Este componente tenía por defecto un valor de intervalo de 1000 milisegundos, es decir, 1 segundo[3]. Así, el zombie cambiará de posición cada segundo, dificultando de esta manera el *golpeo* por parte del jugador. Este comportamiento se especifica fácilmente mediante el componente **Reloj.Temporizador**. Ahora, tan sólo tienes que ensamblar en el mismo el bloque **MoverZombie** que definiste previamente. Justo en este punto puedes apreciar la utilidad de **MoverZombie**. El resultado debería ser similar al que se muestra en la parte central de la figura 3.7.

Por otra parte, también suele ser común establecer unos **valores de inicio** para algunos elementos en función de la aplicación que se está desarrollando. En el caso del *Whack-A-Zombie*, tienes que definir la cantidad de energía o vida inicial. Esta energía disminuirá cuando falles al golpear al zombie. Para definir la variable energia, ve a la categoría *Variables* y arrastra un bloque **inicializar global como**. A continuación, si haces clic sobre **variable**, podrás renombrarla a *energia*. Ahora, sólo falta asignarle el valor inicial, que será de 100. Puedes hacerlo a partir del bloque **número** de la categoría desde *Matemáticas*, o simplemente tecleando el valor 100 y pulsando el retorno de carro sobre una zona vacía de la pantalla. El resultado esperado se muestra en la parte derecha de la figura 3.7.

[3]Puedes ajustar este valor para que el zombie se mueva tan rápido como quieras.

3.3.3. Interacción con el juego

Como recordarás de la sección 3.2, creaste dos etiquetas, nombradas EtiquetaPuntuacion y EtiquetaBarra, para mostrar al jugador de *Whack-A-Zombie* los puntos que va acumulando y la cantidad de energía restante. ¡Recuerda que si el jugador consume su energía, entonces el juego terminará!

En este punto vamos a definir la lógica necesaria para **actualizar las etiquetas** en función de la interacción del usuario con el juego. Para ello, usarás el bloque **cuando Lienzo.Tocar**, que indica que el jugador tocó la pantalla, las coordenadas X e Y del toque (para nosotros son irrelevantes) y si alguno de los *sprites*, como nuestro zombie, fue tocado (esto sí que nos interesa). La figura 3.8 muestra el código necesario para actualizar las dos etiquetas mencionadas anteriormente y para reproducir un efecto sonoro al terminar el juego. No te preocupes por la longitud del bloque, ya que lo vamos a analizar paso a paso.

Como hemos comentado, el bloque **Lienzo.Tocar** comprueba si el toque alcanzó algún *sprite* o imagen del juego, como nuestro zombie, cuando se interactúa con la pantalla del móvil. Debido a que en *Whack-A-Zombie* sólo tenemos un *sprite* (Zombie), entonces sólo existe la opción de alcanzar a éste. **Si el jugador pulsa sobre el *sprite* Zombie**, entonces debería ocurrir lo siguiente:

- La puntuación se incrementará en 1.
- El juego reproducirá un sonido para que el jugador sepa que ha alcanzado al zombie.
- El teléfono móvil vibrará para que el jugador tenga algo más de realimentación.

Para modelar este comportamiento, puedes arrastrar el bloque **Lienzo.Tocar**. El resultado debería ser similar al de la parte superior de la figura 3.8.

A continuación, realiza los siguientes pasos:

1. Ensambla en la rama *ejecutar* de **Lienzo.Tocar** un bloque de control del tipo **si entonces**. Este bloque nos servirá para controlar que el juego continúe sólo si el jugador tiene energía. En este punto, ¡ya deberías dominar a la perfección dónde se encuentran dichos bloques!
2. Ensambla en la rama de comprobación de la condición del anterior bloque de control un bloque $>$ (mayor que). Este bloque lo vamos

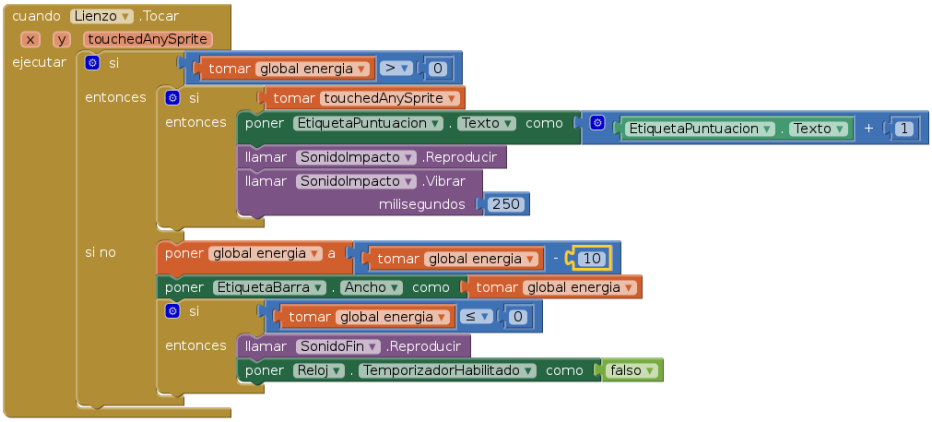

Figura 3.8: Interacción con la pantalla del juego. Control de energía y actualización, en su caso, de la puntuación.

a utilizar para comprobar si el nivel de energía, (es decir, la variable energia es mayor que 0. Ensambla el bloque **tomar global energia** desde *Variables* en la primera parte de dicho bloque. Ahora ensambla un 0 en la segunda parte.

3. Ensambla en la rama **entonces** otro bloque del tipo **si entonces**. Este bloque nos va a servir para definir el comportamiento en caso de que acertemos al golpear al zombie o, en caso contrario, que fallemos. Para ello, ensambla en la rama junto al **si** el bloque **tomar touchedAnySprite**, que te aparecerá al situar el cursor sobre el argumento del mismo nombre que encontrarás en la cabecera del bloque.

Antes de pasar a añadir el código asociado a golpear o fallar sobre el zombie, deberías comparar tu código con el de la figura 3.8 para comprobar que todo es correcto.

Ahora ya podemos añadir el código relativo al **golpeo del zombie**. Para ello, sigue los siguientes pasos:

1. Ensambla el bloque **poner EtiquetaPuntuacion.Texto como**, en la rama **entonces** del anterior bloque condicional. Completa el bloque, ensamblando una suma (en categoría *Matemáticas*, o simplemente pulsando + sobre una zona vacía del área de trabajo), y encaja como términos **EtiquetaPutuación.Texto** y **1**. Esto nos permite incrementar en 1 la puntuación cuando se golpea al zombie.

2. Ensambla el bloque **SonidoImpacto.Reproducir** para que se reproduzca el sonido que previamente asociamos a éste en la sección 3.2.

3. Ensambla el bloque **SonidoImpacto.Vibrar** con un valor de 250 en la rama *milisegundos*. De este modo, el teléfono vibrará durante un cuarto de segundo cuando golpees al zombie.

La otra cara de la moneda está representada por el código relativo al **fallo a la hora de golpear al zombie**. Para ello, sigue los siguientes pasos:

1. pulsa en el cuadrado azul del primer bloque **si entonces** para añadir la rama del **si no**, ya que esta no aparece por defecto. Al hacerlo verás un bocadillo con dos partes. Arrastra el bloque **si no** de la izquierda hasta el de la izquierda, encajándolo dentro del **si**. Ahora el condicional debería incluir las dos ramas.

2. Ensambla un bloque **poner global energia a** desde *Variables* en la rama **si no** del bloque condicional y asóciale una resta donde el primer término sea la variable energia y el segundo el número 10. Este bloque permite reducir el valor de la variable energia.

3. Ensambla un bloque **poner EtiquetaBarra.Ancho a** y asígnale el valor que tenga la variable energia, tal y como se muestra en la figura 3.8. Así, la longitud de la barra de energía se actualizará conforme el valor de la variable vaya cambiando.

La última parte de la rama **si no** del bloque condicional nos va a servir para controlar la **parada del juego**. Éste debería terminar cuando el jugador se quede sin barra de energía. En otras palabras, el juego debería acabar cuándo la variable energia alcance un valor igual o inferior a 0. Para ello:

1. Ensambla un bloque **si entonces** y añade como condición un bloque <= que permita comparar el valor de energia y el número 0.

2. Ensambla un bloque **SonidoFin.Reproducir** en la rama **entonces**. De este modo, el juego reproducirá un sonido cuando finalice.

3. Ensambla un bloque **poner Reloj.TemporizadorHabilitado como** y asígnale un valor **falso** desde la categoría *Lógica*. Esto permite detener el movimiento del zombie, ya que éste está controlado por un valor *cierto* en el bloque que se muestra en la figura 3.7. Si establecemos este valor a falso, entonces la rama **ejecutar** de dicho bloque, es decir, la que ejecuta *MoverZombie* no se ejecutará y, por lo tanto, el zombie no se moverá.

En este punto, ya deberías tener un juego completamente funcional y tu código debería ser similar al de la figura 3.8. La última parte que nos queda por discutir es el *reset* del juego para dar lo opción al jugador de seguir jugando a *Whack-A-Zombie* todas las veces que quiera.

3.3.4. Reinicio juego

Con el objetivo de realizar un *reset* de *Whack-A-Zombie* es necesario ejecutar las siguientes acciones:

- Reestablecer la energía del jugador al máximo.
- Poner la puntuación a 0.
- Volver a poner al zombie en movimiento.

Figura 3.9: Reseteo del juego e interacción con el zombie (sonido de golpe y vibración del teléfono).

La figura 3.9 muestra el código necesario para efectuarlas. Este código se ejecutará cuando el jugador pulse el botón Reset que está en la parte inferior del juego, tal y como se indica en la figura 3.1 al inicio de este capítulo. Para añadir este último bloque, sigue los siguientes pasos:

1. Añade un bloque **cuando BotonReset.Clic**. Precisamente, la palabra *Clic* hace referencia a pulsar dicho botón.

2. Ensambla el bloque **poner global energia** y asígnale un valor de 100, exactamente igual que en la inicialización de la sección 3.3.2.

3. Ensambla el bloque **poner EtiquetaBarra.Ancho** al valor de energía, exactamente igual que en la sección 3.3.3.

4. Ensambla el bloque **poner EtiquetaPuntuacion.Texto** con un valor de 0.

5. Ensambla el bloque **poner Reloj.TemporizadorHabilitado** a *cierto* (*Built-in->Logic*). Esto permite que el zombie empiece a moverse de nuevo, exactamente igual que al inicio de la aplicación.

En este punto, deberías comparar tu código con el que se muestra en la figura 3.9. Ahora, ya puedes probar tu juego y disfrutar de él en tu teléfono móvil.