Programa Tus Ideas

Día 1

1.1. Tutorial: Hola Gatito

Típicamente el primer programa que se realiza para probar un nuevo computador o lenguaje de programación muestra el mensaje "Hola Mundo" para mostrar que todo está conectado y funcionando correctamente. En AppInventor incluso las aplicaciones más simples pueden hacer mucho más que sólo mostrar mensajes: ellas pueden reproducir sonidos y reaccionar cuando el usuario toca el dispositivo. Así que vamos a comenzar con algo mucho más emocionante: tu primera aplicación se llamará ***Hola Gatito***. En general todas las aplicaciones se definen por sus pantallas visibles—lo que se conoce como la *interfaz de usuario*—y su comportamiento. La pantalla de la aplicación se muestra en la Figura 1.1, y su comportamiento será como sigue:

- *Cuando* la imagen es presionada *entonces* se emite el sonido de un maullido y el dispositivo vibra.
- *Cuando* el dispositivo es agitado *entonces* se emite el sonido de un maullido.

Figura 1.1: Pantalla de aplicación ***Hola Gatito***

Qué Aprenderás

Siguiendo este tutorial aprenderás a:

- Construir aplicaciones en AppInventor seleccionando componentes y diciéndoles qué hacer y cuándo hacerlo.
- Usar el Diseñador de Componentes para seleccionar componentes. Algunos componentes son visibles en la pantalla del dispositivo mientras que otros no lo son.

- Agregar archivos multimedia (sonidos e imágenes) a las aplicaciones, subiéndoles a AppInventor desde tu computador.
- A usar el Editor de Bloques para ensamblar bloques de código que definen el comportamiento de los componentes, y que por lo tanto en conjunto definen el comportamiento de la aplicación.
- Probar las aplicaciones directamente en los dispositivos, lo que te ayudará a ver cómo las aplicaciones se comportan, paso a paso, a medida que las construyes.
- Empaquetar las aplicaciones que construyes para instalarlas en un dispositivo.

El Ambiente de Desarrollo AppInventor

Para comenzar a programar con App Inventor debes acceder al sitio `http://ai2.appinventor.mit.edu`.[1] Esto abrirá la última versión de App Inventor, publicada en Diciembre de 2013. Algunas personas llaman a esta aplicación App Inventor 2, pero su nombre formal sigue siendo App Inventor, y la versión anterior se conoce como App Inventor Classic. Nosotros usaremos siempre la nueva versión. El entorno de programación App Inventor tiene 3 partes esenciales:

- El Diseñador de Componentes, que se muestra en la Figura 1.2. Se usa para seleccionar los componentes de la aplicación y especificar sus propiedades.
- El Editor de Bloques, que se muestra en la Figura 1.3. Se usa para especificar cómo se comportan los componentes (por ejemplo, qué pasa cuando se presiona un botón).
- Un dispositivo Android que te permite ejecutar y probar las aplicaciones a medida que las vas desarrollando. Si no tienes un dispositivo Android disponible, puedes probar las aplicaciones usando el Emulador Android IF ►*Appendix sobre el emulador o referencia*◄

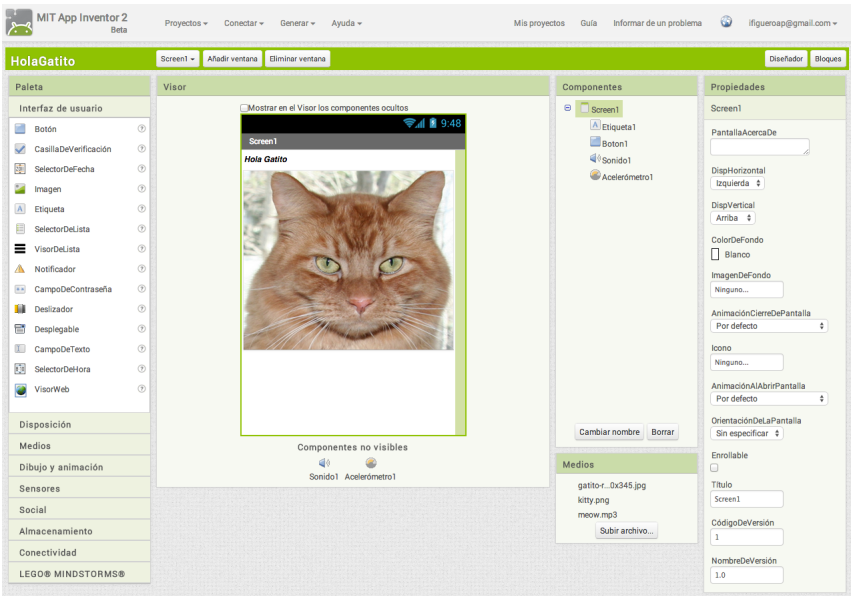

Figura 1.2: Diseñador de Componentes

[1] Necesitas una cuenta de Google para utilizar AppInventor

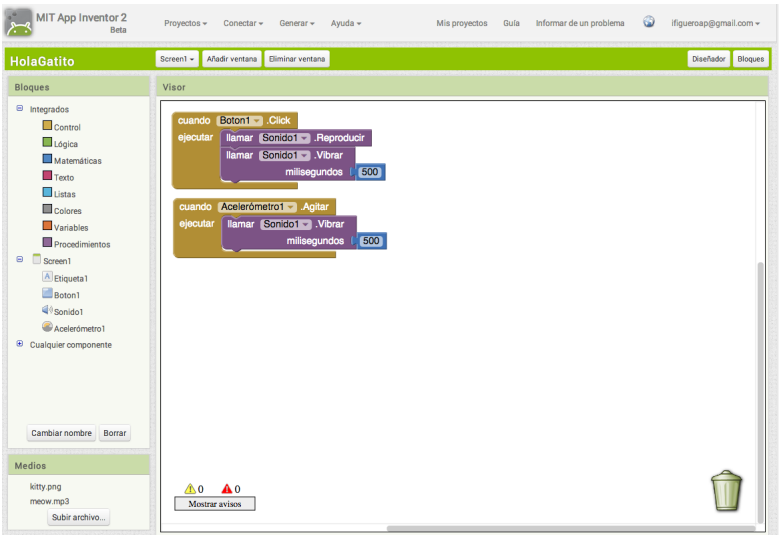

Figura 1.3: Editor de Bloques

La primera vez que ingreses a `http://ai2.appinventor.mit.edu`, verás la página de Proyectos, la que seguramente estará en blanco porque aún no has creado ningún proyecto. Para crear un proyecto, presiona el botón "Comenzar un proyecto nuevo" en la esquina superior izquierda de la página, ingresa el nombre "HolaGatito" (los nombres de proyecto son sin espacios), y luego presiona "Aceptar". La Figura 1.4 muestra la creación del proyecto ***Hola Gatito***.

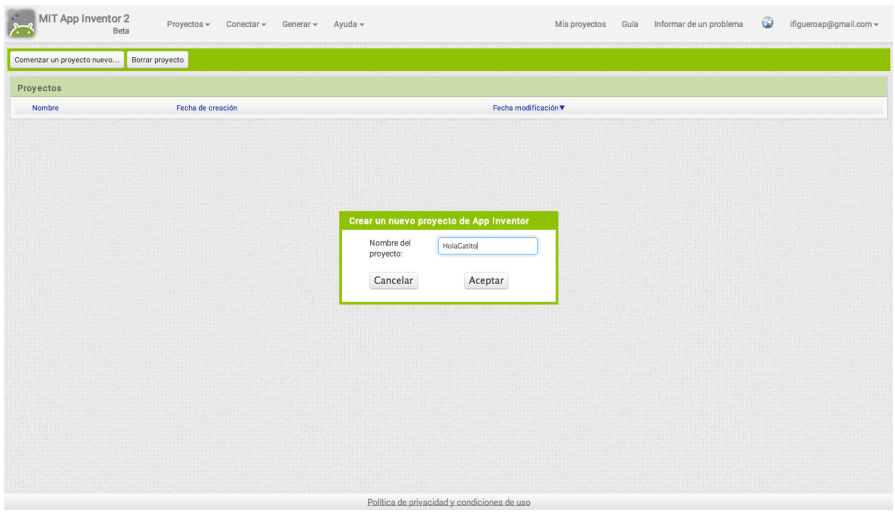

Figura 1.4: Crear proyecto ***Hola Gatito***

La primera ventana que se abre es el Diseñador de Componentes. El Editor de Bloques está disponible al hacer click en el botón "Bloques" en la esquina superior derecha de la ventana.

AppInventor es una herramienta computacional *en la nube*, lo que significa que tu aplicación está almacenada en un servidor en línea mientras tú trabajas. Por lo tanto si cierras AppInventor, tu aplicación estará ahí cuando regreses; no necesitas guardar nada en tu computador, como cuando trabajas con archivos de Word.

Diseñando los Componentes

La primera herramienta que utilizarás es el Diseñador de Componentes (o simplemente Diseñador). Los componentes son los elementos que combinas para crear aplicaciones, parecido a los ingredientes de una receta. Algunos componentes son muy simples, como una **Etiqueta**, la cual muestra texto en la pantalla, o un **Botón**, el cual presionas para iniciar una acción. Otros componentes son más elaborados: un **Lienzo** para dibujar, que puede contener imágenes estáticas (sin movimiento) y también animaciones; un **Acelerómetro**, que es un sensor de movimiento que funciona de manera parecida a un Wiimote y detecta cuando mueves o agitas el dispositivo. Otros componentes permiten enviar y recibir mensajes de texto (SMS), reproducir música, sonidos, videos, obtener información desde sitios web, etc.

Cuando abras el Diseñador, éste aparecerá como se muestra en la Figura 1.5

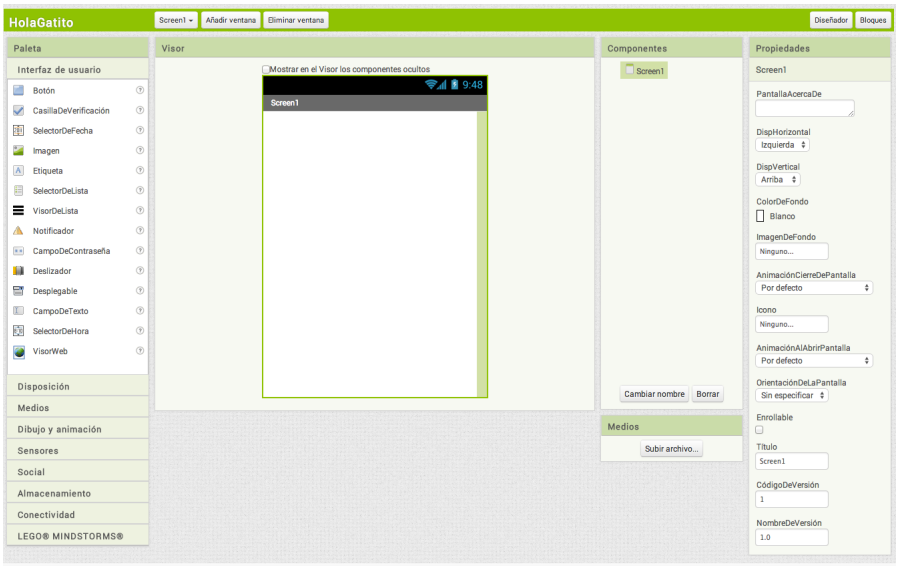

Figura 1.5: El Diseñador de Componentes

El Diseñador está dividido en varias áreas:

- En el centro hay un área que se llama Visor. Aquí es donde tú colocas los componentes y los ordenas para especificar cómo quieres que se vea tu aplicación. El Visor muestra una aproximación o borrador de cómo se verá la aplicación, por lo que, por ejemplo, una línea de texto podría cortarse en una posición distinta según el dispositivo que uses. Para ver *realmente* cómo se verá una aplicación es necesario probarla directamente en un dispositivo.
- A la izquierda del Visor está la Paleta, que es una lista de todos los componentes que puedes usar en tu aplicación. La Paleta está dividida en secciones. Inicialmente sólo los componentes de la Interfaz de Usuario están visibles, pero basta con hacer click en los nombres de las otras secciones para ver los componentes de cada una de ellas (por ejemplo, Medios, Sensores, etc.).
- A la derecha del Visor está la lista de Componentes, que muestra los componentes usados en tu proyecto. Cualquier componente que arrastres en el Visor también aparecerá en esta lista. Actualmente, el proyecto sólo tiene un componente: **Screen1**, que representa la pantalla del dispositivo.
- Abajo del área Componentes se muestran los Medios (imágenes y sonidos) en el proyecto. Este proyecto todavía no tiene ningún archivo multimedia, pero pronto agregarás algunos.

Al extremo derecho de la pantalla hay una sección que muestra las Propiedades de los componentes; cuando seleccionas un componente en el Visor, verás su lista de Propiedades en esta sección. Las Propiedades

son detalles sobre cada componente que tú puedes cambiar. Por ejemplo, al seleccionar una **Etiqueta**, puedes ver propiedades relaconadas al color, texto, tipo de letra, etc. En este momento está mostrando las propiedades de la pantalla, de nombre **Screen1**, las que incluyen el color de fondo, imagen de fondo y título, entre otras.

Para la aplicación ***Hola Gatito*** necesitarás dos componentes *visibles* (puedes pensar sobre este tipo de componentes como aquellos que se pueden ver en la aplicación): la **Etiqueta** que muestra el texto "Hola Gatito" y un **Botón** con una imagen de un gato. También necesitarás un componente de **Sonido**, que es *no-visible*, y que sabe cómo reproducir sonidos, tales como el maullido del gato. Además, necesitarás otro componente no-visible, el **Acelerómetro**, para detectar cuándo el dispositivo está siendo agitado. Ahora veremos paso a paso cómo construir la aplicación con cada uno de estos componentes.

Agregar la Etiqueta

El primer componente por agregar es una **Etiqueta**:

1. Selecciona la sección "Interfaz de Usuario" en la Paleta (si es que no está ya abierta), y arrastra una **Etiqueta** hacia el Visor. Luego de hacerlo, verás una forma rectangular en el Visor con el texto "Texto para Etiqueta1".

2. En el área de Propiedades de la etiqueta, busca la propiedad "Texto". Cambía el valor de esta propiedad por el texto "Hola Gatito" y luego presiona Enter. Verás que el texto cambia en el Visor.

3. Cambia el **ColorDeFondo** de la etiqueta haciendo click en la caja, que actualmente dice "Ninguno", para seleccionar un color de la lista que aparecerá. Selecciona Azul. También cambia el **ColorDeTexto** de la etiqueta a Amarillo. Finalmente, cambia el **Tamaño de letra** a 20.

Luego de seguir estos pasos, el Diseñador debería verse como en la Figura 1.6.

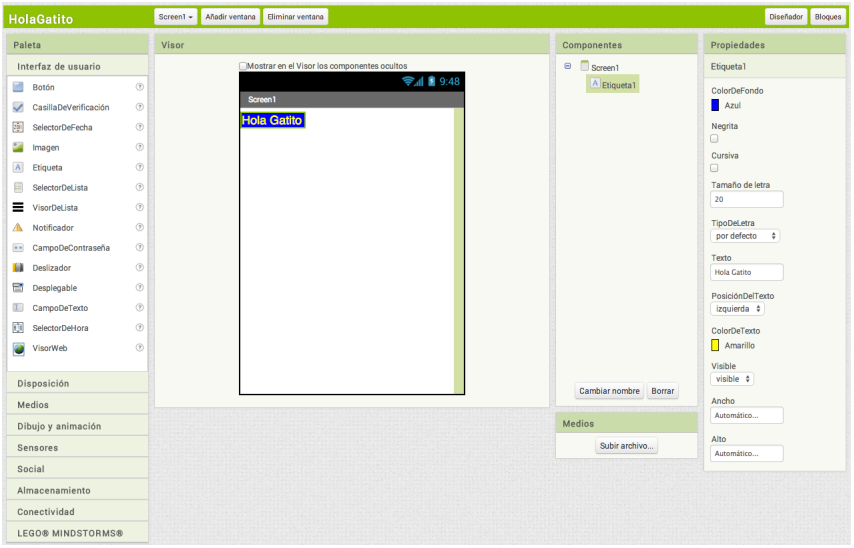

Figura 1.6: La aplicación ahora tiene una etiqueta

Agregar el Bóton

El gatito para ***Hola Gatito*** está implementado como un **Botón**—creas un botón normal, y luego cambias su imagen de fondo a la del gatito. Para primero agregar un botón normal, debes buscar el componente **Botón** en la Paleta. Arrastra el botón hacia el Visor, ubicándolo debajo de la etiqueta. Verás que un botón rectangular aparece en el Visor.

Ahora tenemos un botón que usaremos para gatillar el efecto de sonido cuando alguien lo presiona, pero lo que realmente queremos es ver la foto de un gatito, y no un simple rectángulo. Para hacer que el botón se vea como la foto del gatito debes hacer lo siguiente:

1. Primero, necesitas descargar una imagen del gatito y guardarla en tu computador. Puedes descargar la imagen desde la dirección `TODO`. *.png* es una extensión para un formato de archivo estándar, similar a *.jpg* y *.gif*; todos estos formatos pueden ser usados por AppInventor, así como la mayoría de los formatos de sonido estándar, como *.mpg* o *.mp3*. También puedes descargar el sonido del maullido desde `TODO`. Si quieres, puedes usar tus propias imágenes y sonidos.

2. El área de Propiedades debería mostrar las propiedades del botón. Si no es así, selecciona el botón en el Visor para que así sea. En las propiedades del botón presiona el área bajo el teto "Imagen" (que actualmente dice "Ninguno").

3. Presiona "Subir archivo", y luego presiona "Seleccionar archivo" para buscar en tu computador el archivo con la foto del gatito (si no usas tu propia imagen, el nombre es *gatito.png*). Luego presiona "Aceptar".

4. Luego que la imagen se suba, el nombre de archivo debería estar disponible como una opción para la propiedad **Imagen** del botón. Presiona "Aceptar" para seleccionarla. También verás el archivo listado en el área de Medios en la ventana del Diseñador. Además, en el Diseñador verás que el botón ahora se ve como la foto del gatito que acabas de seleccionar.

5. Quizás te diste cuenta que la foto del gatito todavía tiene las palabras "Texto para Botón1". Probablemente no quieras eso en tu aplicación, por lo tanto borra el texto de la propiedad **Texto** para el componente **Botón1**.

Ahora el Diseñador debería verse como en la Figura 1.7.

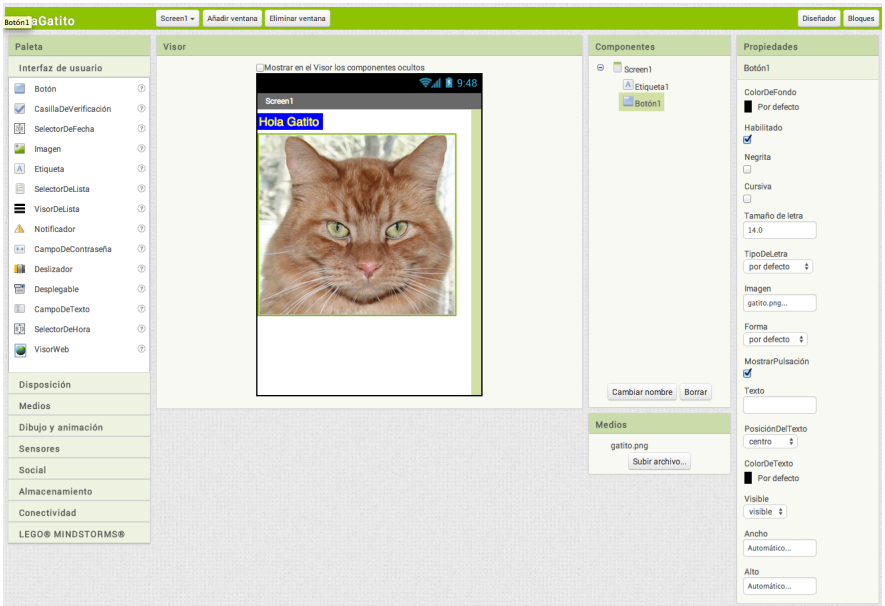

Figura 1.7: La aplicación con una etiqueta y un botón con una imagen de un gatito

Agregar el Maullido

En tu aplicación quieres que el gatito maulle cuando el botón sea presionado. Para esto, necesitarás agregar el sonido del maullido, y programar el comportamiento del botón para reproducir ese sonido cuando el botón es presionado:

1. Si aún no has descargado el sonido del maullido, descárgalo ahora desde `TODO`.

2. Ve a la Paleta, y selecciona la sección Medios. Arrastra un componente **Sonido** y cólocalo en el Visor. Sin importar el lugar donde lo arrastraste, este componente aparecerá en un área abajo del Visor, llamada "Componentes no visibles". Los componentes no visibles son objetos que hacen cosas para la aplicación pero que no aparecen en la interfaz visual del usuario de la aplicación.

3. Selecciona el componente **Sonido1** para mostrar sus propiedades. Selecciona su propiedad **Origen** y sigue los pasos para subir el archivo *miau.mp3* que descargaste anteriormente. Una vez que finalices este paso, deberías ver los archivos *gatito.png* y *miau.mp3* en el área de Medios en el Diseñador.

La tabla FOO muestra los componentes que has agregado hasta ahora a la aplicación ***Hola Gatito***.

Tipo de Componente	Sección en la Paleta	Nombre	Propósito
Botón	Interfaz de usuario	Botón1	Presionar para que el gato maulle.
Etiqueta	Interfaz de usuario	Etiqueta1	Muestra el texto "Hola Gatito".
Sonido	Medios	Sonido1	Reproduce el sonido del maullido.

Tabla 1.1: Componentes que has agregado a la aplicación ***Hola Gatito***

Probar la Aplicación

Con AppInventor, puedes ver y probar tu aplicación en un dispositivo Android a medida que la vas creando. Probar tu aplicación de manera incremental, paso a paso mientras la desarrollas, es una práctica usada por muchos desarrolladores de software, y que te ahorrará muchas horas de trabajo!

Conexión por puerto USB Para conectar las aplicaciones de AppInventor con tus dispositivos Android usaremos una conexión por cable USB. Esto requiere la instalación de un software especial en tu computador—pero que ya está preinstalado![2]

Para probar tu aplicación, conecta tu dispositivo al computador usando un cable USB, y luego selecciona la opción "Conectar" y específicamente la opción "USB", tal como se muestra en la Figura 1.8

Figura 1.8: Conexión AppInventor por USB

Si todo funciona correctamente, deberías ver la aplicación ***Hola Gatito*** ejecutándose en tu dispositivo, incluyendo todos los componentes que agregaste. A medida que hagas cambios en el Diseñador de Componentes o en el Editor de Bloques, estos cambios también aparecerán en el dispositivo. Consulta a tu tutor ante cualquier problema.

Si tu aplicación aparece en el dispositivo, presiona la imagen del gato. ¿Crees que algo pasará? En realidad no pasará nada, porque tu aplicación todavía no le ha dicho al botón qué es lo que debe hacer al ser presionado. Este es el primer punto importante para comprender sobre AppInventor: para cada componente que agregas en el Diseñador, tienes que ir hacia el Editor de Bloques y crear el código para que algo pase con ese componente.

[2]Las instrucciones de instalación, en inglés, están disponibles en `http://appinventor.mit.edu/explore/ai2/setup.html`

Agregando Comportamiento a los Componentes

Acabas de agregar componentes de tipo **Botón**, **Etiqueta** y **Sonido** como los bloques con los que construyes tu primera aplicación. Ahora hagamos que el gatito maulle cuando presionas el botón. Esto tienes que hacerlo con el Editor de Bloques. Presiona el botón "Bloques" en la esquina superior derecha del Diseñador de Componentes.

Observa bien la ventana del Editor de Bloques. Aquí es donde le dices a los componentes qué hacer y cómo hacerlo. Ahora le dirás al botón del gatito que reproduzca un sonido cuando el usuario lo presiona. Si los componentes son los ingredientes en una receta, puedes pensar que los bloques son las instrucciones de cocina.

Haciendo Maullar al Gatito

En la esquina superior izquierda de la ventana, debajo del encabezado "Bloques", puedes ver una columna que incluye una sección "Integrados", y una sección para cada componente de los que creaste en el Diseñador: **Botón1**, **Etiqueta1**, y **Sonido1**. Cuando haces click en uno de los componentes, aparecen un montón de opciones (bloques) para ese componente. Presiona el componente **Botón1**. Al hacerlo, se muestra una selección de los bloques que puedes usar para decirle al botón qué debe hacer; esta lista comienza con el bloque **Botón1.Click**, como se muestra en la Figura 1.9.

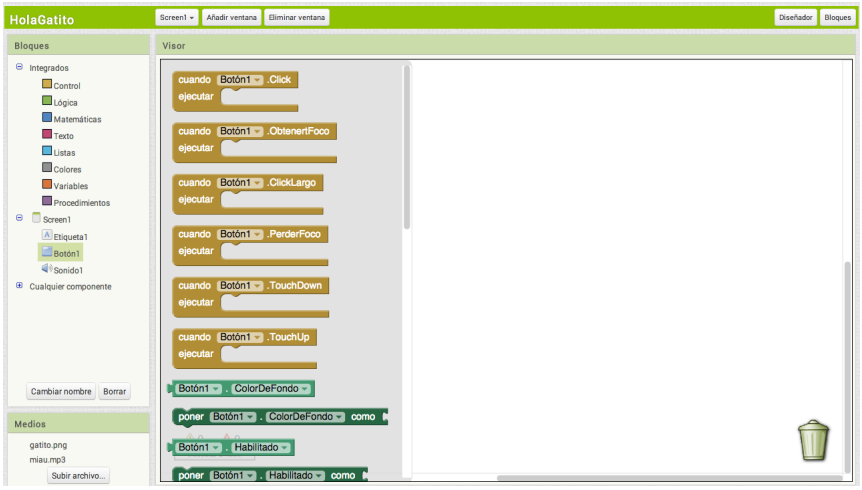

Figura 1.9: Al seleccionar el componente **Botón1** se muestran sus bloques

Arrastra el bloque **Botón1.Click** y arrástralo en el Visor (el espacio en blanco para trabajar con los bloques). Puedes darte cuenta que la palabra *cuando* está incluida en el bloque. Los bloques que incluyen la palabra *cuando* se llaman *controladores de eventos*. Ellos especifican lo que los componentes deberían hacer *cuando* algun evento particular ocurre. En este caso, estamos interesados en el evento de que un usuario de la aplicación presione el gatito (que en realidad es un botón), tal como se muestra en la Figura 1.10. Luego, agregaremos algunos bloques para programar lo que pasará en respuesta a ese evento.

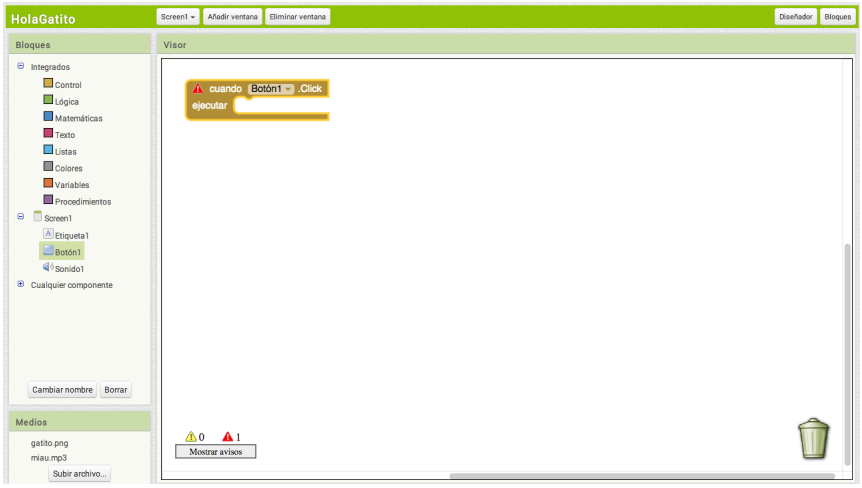

Figura 1.10: Especificarás una respuesta al usuario que presiona el botón usando el bloque **Boton.Click**

Ahora selecciona el componente **Sound1** y luego arrastra el bloque **llamar Sonido1.Reproducir**. Recuerda que anteriormente configuramos el **Origen** del componente con el archivo *miau.mp3*. Observa ahora que el bloque **llamar Sonido1.Reproducir** tiene una forma tal que es posible ensamblarlo con el espacio marcado como "ejecutar" en el bloque **Botón1.Click**. AppInventor está diseñado de manera que sólo ciertos bloques pueden ser ensamblados juntos; de esta manera tu siempre sabrás que estás conectando bloques que en realidad trabajan juntos. En este caso, los bloques con la palabra *llamar* hacen que los componentes hagan cosas. Los dos bloques deben conectarse para formar una sola unidad, como se muestra en la Figura 1.11. Escucharás un sonido cuando los bloques se conecten correctamente.

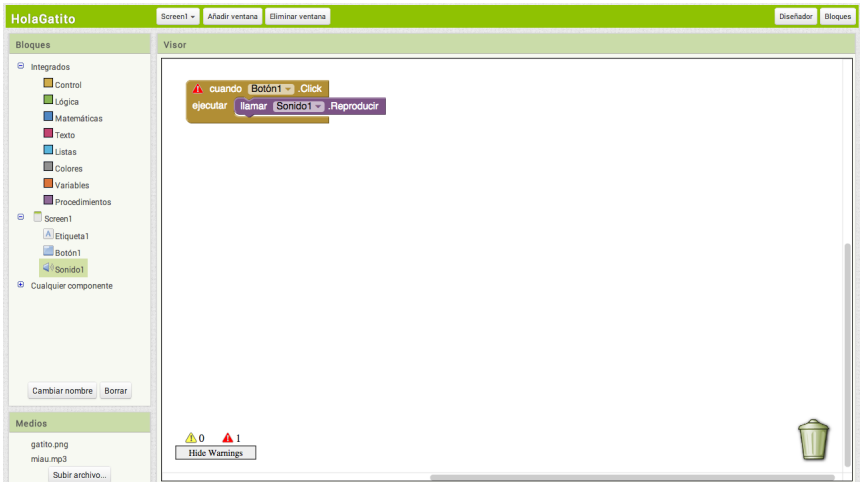

Figura 1.11: Ahora cuando alguien presione el botón, se escuchará el maullido

A diferencia del código programado de manera tradicional (el que a menudo se ve como un revoltijo de "jerigonza"), los bloques de eventos-respuestas en AppInventor describen los comportamientos que estás intentando crear. En este caso, estamos esencialmente diciendo "Oye AppInventor, cuando alguien presione el botón del gatito, reproduce el sonido del maullido".

Prueba tu Aplicación Asegúrate que todo esté funcionando correctamente—es importante que pruebes tu aplicación cada vez que agregas algo nuevo. Presiona el botón en el dispositivo. Deberías escuchar el

maullido. Felicitaciones, tu primera aplicación se está ejecutando!

Agregar el Ronrroneo

Ahora vamos a hacer que el gato ronrronee y maulle cuando presionas el botón. Simularemos el ronrroneo haciendo vibrar el dispositivo. Eso puede sonar difícil, pero en realidad es muy fácil porque el componente **Sonido** que usamos para reproducir el maullido también puede hacer vibrar el dispositivo. AppInventor te ayuda a aprovechar la funcionalidad esencial de los dispositivos sin tener que preocuparse de *cómo* el dispositivo vibra en la práctica. No necesitas hacer nada nuevo en el Diseñador, simplemente puedes agregar un nuevo comportamiento al botón en el Editor de Bloques.

1. Ve al Editor de Bloques y selecciona el componente **Sonido1**.

2. Selecciona el bloque **llamar Sonido1.Vibrar** y arrastralo hacia abajo del bloque **llamar Sonido1.Reproducir**. El bloque debería ajustarse en su lugar, como se muestra en la Figura 1.12.

Figura 1.12: Reproduciendo el sonido y vibrando en el evento Click

3. Observa ahora que el bloque **llamar Sonido1.Vibrar** incluye el texto "milisegundos". Un espacio abierto en un bloque significa que necesitas conectar algo ahí para especificar en detalle el comportamiento del bloque. En este caso, debes decirle al bloque por cuánto tiempo debería vibrar. Necesitas agregar esta información en milésimas de segundo (milisegundos), lo que es bastante común en muchos lenguajes de programación. Por lo tanto, para hacer que el dispositivo vibre por medio segundo, tienes que poner un valor de 500 milisegundos. Para poner un valor de 500 necesitas arrastrar un bloque numérico. Selecciona el componente integrado "Matemáticas", como se muestra en la **??**. Deberías ver un bloque con un cero como primer elemento. Puedes arrastrar este bloque y cambiar su valor por cualquier otro número.

Figura 1.13: Agregando un bloque numérico para especificar la duración de la vibración

4. Arrastra el bloque numérico y verás un bloque azul con el número cero, como se muestra en la Figura 1.14.

Figura 1.14: Agregando un bloque numérico (0 es el valor por defecto).

5. Cambia el 0 a 500 haciendo click en el bloque y escribiendo el nuevo valor, como se muestra en la Figura 1.15.

Figura 1.15: Cambiando el valor del bloque numérico a 500.

6. Conecta el bloque numérico 500 en el espacio del bloque **llamar Sonido1.Vibrar**, como se muestra en la Figura 1.16

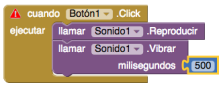

Figura 1.16: Conectando el bloque numérico 500 en el espacio para configurar los milisegundos.

Prueba tu aplicacion!

Agitando el Dispositivo

Ahora agreguemos un elemento final que aprovecha otra característica de Android: hacer que el gatito maulle cuando agitas el dispositivo. Para hacer esto, usarás un componente llamado **Acelerómetro** que puede sentir cuando agitas o mueves el dispositivo.

1. En el Diseñador, expande la sección Sensores en la Paleta y arrastra un **Acelerómetro** hacia el Visor. No te preocupes sobre el lugar donde lo arrastrarás, ya que es un componente no visible que aparecerá en la sección justo abajo del Visor.

2. Vas a querer manejar el que alguien agite el dispositivo como un evento diferente y separado de cuando se presiona el botón. Eso significa que necesitas un nuevo controlador de eventos. Ve al Editor de Bloques, donde debería haber un nuevo componente **Acelerómetro1**. Seleccionalo y arrastra el bloque **Acelerómetro1.Agitar**.

3. De la misma forma que lo hiciste con el botón cuando es presionado, arrastra un bloque **llamar Sonido1.Reproducir** y conéctalo en el espacio del bloque **Acelerómetro1.Agitar**. Los bloques de la aplicación deben quedar como los que se muestran en la Figura 1.17. Prueba tu aplicación agitando el dispositivo!

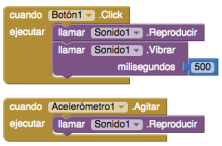

Figura 1.17: Los bloques para la aplicación ***Hola Gatito***

Compartir tu Aplicación

Puedes compartir tu aplicación de varias maneras. Para compartir la aplicación ejecutable (el archivo .apk que se instala directamente en un dispositivo), primero presiona "Generar" y escoge "App (guardar archivo .apk en mi ordenador". Esto creará un archivo con una extensión *.apk* en tu computador. Puedes compartir este archivo con otros envíandolo como archivo adjunto en un correo, el cual abrirán con su cliente de correo en el dispositivo donde instalarán la aplicación. También puedes subir el archivo .apk en la web (por ejemplo en DropBox o en tu portafolio). Sólo debes asegurarte que las personas que quieran instalar tu aplicación deben permitir las "fuentes desconocidas" en la configuración del dispositivo, para permitir la instalación de aplicaciones que no provienen de la tienda de aplicaciones de Android.

También puedes crear un código QR para tu aplicación de manera que las personas puedan escanear el código en sus dispositivos, desde la web o incluso desde algún poster. Existen numerosas herramientas para crear un código QR desde una URL (por ejemplo, `http://qrcode.kaywa.com/`). Una vez que tengas el código QR puedes insertarlo en una página web u otros documentos.

Además, también puedes compartir el *código fuente* (los bloques) de tu aplicación con otro desarrollador que use AppInventor. Para hacer esto, selecciona "Mis Proyectos", elige la aplicación que deseas compartir (en este caso ***HolaGatito***), selecciona "Proyecto", y luego selecciona "Exportar a mi ordenador el proyecto (.aia) seleccionado". El archivo creado en tu computador tendrá extensión *.aia*. Esto le dará a otra persona una copia completa de tu aplicación, la que podrán usar para editar y personalizar sin afectar tu propia versión.

Personalizaciones

Después que desarrolles las aplicaciones de este taller, seguramente pensarás muchas maneras de mejorarlas. A medida que avancemos con las aplicaciones, también te sugeriremos ideas para que intentes implementarlas. El personalizar las aplicaciones te llevará a explorar los componentes y bloques disponibles, y a aprender a programar por ti mismo sin las instrucciones detalladas que son dadas en los tutoriales.

Aquí hay algunas ideas para mejorar la aplicación ***Hola Gatito***:

- Mientras agitas el dispositivo, los maullidos sonarán de forma extraña, como si hubiera eco. Esto pasa porque el acelerómetro está gatillando el evento agitar muchas veces por segundo, por lo que los maullidos se solapan. Si te fijas en el componente **Sonido** en el Diseñador, verás una propiedad que se llama **IntervaloMinimo**. Esta propiedad determina el tiempo mínimo que hay que esperar para reproducir dos sonidos de forma consecutiva. Actualmente tiene un valor de 400 milisegundos (casi medio segundo), lo que es menor que la duración de un maullido. Jugando con el valor de esta propiedad podrás cambiar la manera en que los maullidos se solapan entre sí.
- Si exportas la aplicación, la ejecutas, y luego caminas con tu dispositivo en el bolsillo, tu dispositivo maullará cada vez que te muevas bruscamente—algo que quizás pueda ser vergonzoso! Las aplicaciones Android típicamente están diseñadas para seguir ejecutándose incluso cuando no las estas mirando; por lo que tu aplicación sigue comunicándose con el acelerómetro y los maullidos continúan. Para salir realmente de la aplicación, debes mantener presionado el botón de menu en la aplicación ***Hola Gatito***. Se te mostrará una opción para cerrar la aplicación, al seleccionarla la aplicación estará completamente cerrada.

Resumen

A continuación repasamos los principales conceptos cubiertos en este tutorial:

- Construyes aplicaciones seleccionando componentes en el Diseñador y diciéndoles qué hacer y cuándo hacerlo en el Editor de Bloques.
- Algunos componentes son visibles y otros no lo son. Los visibles aparecen en la interfaz de usuario de la aplicación. Los no visibles hacen cosas como reproducir sonidos.

- Defines el comportamiento de los componentes juntando bloques en el Editor de Bloques. Primero arrastras un controlador de eventos como **Boton1.Click**, y luego pones bloques de comandos como **Sonido1.Reproducir** en su interior. Cualquier bloque contenido dentro de **Boton1.Click** será realizado cuando el usuario presione el botón.
- Algunos comandos necesitan información extra para hacerlos funcionar. Un ejemplo es **Sonido1.Vibrar**, que necesita saber cuántos milisegundos debe vibrar. Estos valores se llaman *argumentos* o *parámetros*.
- Los números se representan como bloques numéricos. Puedes conectar estos bloques en comandos que toman números como argumentos.
- AppInventor tiene componentes que representan los sensores del dispositivo. El **Acelerómetro** puede detectar cuándo el dispositivo se mueve.
- Puedes empaquetar las aplicaciones y descargarlas al teléfono, donde se ejecutan de forma independiente a AppInventor.

1.2. Discusión y Personalización

Para ayudarte a consolidar tus conocimientos sobre AppInventor te invitamos a responder las siguientes preguntas, y a personalizar tu aplicación siguiendo las sugerencias que se presentan a continuación.

Preguntas

1. AppInventor tiene dos ventanas principales. ¿Cuáles son y qué haces en ellas?
2. Probando e Instalando una Aplicación
   - ¿Cómo pruebas una aplicación a medida que la vas desarrollando?
   - ¿Cómo puedes instalar una aplicación, que tú construiste, en tu dispositivo Android?
   - ¿Qué pasaría si no tuvieras un teléfono o tablet, pero quisieras programar algunas aplicaciones? ¿Podrías hacerlo? ¿Cómo lo harías?
3. En la aplicación ***Hola Gatito*** menciona un(a):
   - componente visible
   - componente no visible
   - propiedad
   - evento
   - controlador de eventos
   - llamada a función
4. ¿Qué es un controlador de eventos? ¿De qué está compuesto?
5. Una aplicación consiste de su interfaz de usuario y su comportamiento. ¿En qué consiste el comportamiento de la aplicación?

Ejercicios de Personalización

1. Agrega un componente **CasillaDeVerificación** que indica si el gatito está durmiendo o no. Si la casilla está chequeada, el gatito duerme, y si no, está despierto.
2. Modifica el comportamiento de la aplicación para tomar en cuenta si el gatito duerme o no. Mientras el gatito duerme, el presionar el botón no emite ningún sonido ni hace vibrar el dispositivo. Si el gatito duerme y se agita el dispositivo, entonces se despierta.

1.3. Proyecto: Botonera de Sonidos

La actividad final del primer día de este taller consiste en que realices tu propio proyecto, con un poquito de ayuda inicial. La meta es que desarrolles una aplicación estilo "botonera de sonidos", que consiste en múltiples botones que al ser presionados emiten distintos sonidos. Para ayudarte un poco, hemos desarrollado una versión preliminar de la aplicación, la que puedes descargar desde IF ►*TODO*◄ , y luego importar directamente en AppInventor.

Interfaz de Usuario La Figura 1.18 muestra la interfaz de usuario de la aplicación, que consiste en dos botones. A diferencia de ***Hola Gatito***, estamos usando un componente de **Disposición** para ordenar los botones en la pantalla. Específicamente usamos una **DisposiciónTabular** que crea una rejilla donde pueden ponerse otros componentes.

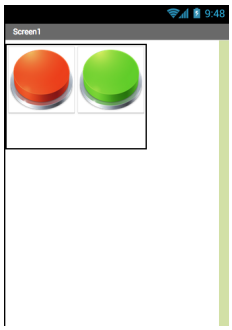

Figura 1.18: Interfaz de usuario de la plantilla para la botonera de sonidos

Comportamiento El código de la aplicación se muestra en la Figura 1.19. El código es muy similar al de ***Hola Gatito***, pero tiene una diferencia fundamental. La aplicación tiene sólo 1 componente **Sonido**, que se usa para reproducir los sonidos de cada botón. Esto se logra cambiando la propiedad **Origen** de forma *dinámica*, dependiendo del botón que es presionado. Para cambiar el origen del sonido, se especifica el nombre del archivo a utilizar (que debe estar subido con anterioridad).

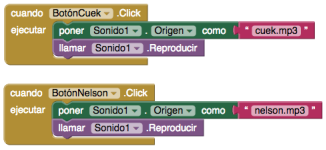

Figura 1.19: Código de la plantilla para la botonera de sonidos

Requerimientos del Proyecto Puedes crear la aplicación que quieras basándote en la idea y plantilla originales. Sin embargo tu aplicación debiera cumplir al menos los siguientes requerimientos:

- Debe tener una interfaz de usuario compleja, usando los componentes de **Disposición**.
- Debe tener al menos cuatro sonidos que se reproduzcan en respuesta a distintos eventos (por ejemplo, presionar botones, agitar dispositivo, recibir mensajes de texto, etc.)
- Debe tener algun comportamiento condicional, usando bloques **si, sino**.

Ideas Algunas ideas para ayudarte con tu proyecto:

- Puedes encontrar y descargar sonidos desde internet, por ejemplo en `http://www.myinstants.com/`.
- Reproducir notas de tus canciones favoritas, discursos, o charlas.
- Software educacional para niños, por ejemplo una aplicación con los sonidos de una granja.
- Un juego donde hay que descubrir el nombre de la canción. Al presionar el botón se escucha una parte de la canción, y al presionar otro botón se muestra el nombre de la canción.
- Una aplicación que hace cosas diferentes cuando recibe mensajes de texto desde otro teléfono.
- Una aplicación que te permite presionar las fotos de tus compañeros para ver sus nombres y escuchar sus voces.

1.4. Material de Apoyo

Entendiendo la Arquitectura de una Aplicación

Muchas personas pueden decir qué es lo que es una aplicación, desde la perspectiva del usuario, pero entender qué es una aplicación desde la perspectiva de un ***programador*** es más complicado. Las aplicaciones tienen una estructura interna, lo que se conoce como la *arquitectura de la aplicación*, que se debe entender para poder crear aplicaciones de manera efectiva.

Una manera de describir el interior de una aplicación es separarla en dos partes: sus *componentes*, y sus *comportamientos*. En general, estos conceptos corresponden a las dos ventanas principales de AppInventor. Por un lado, se usa el Diseñador de Componentes para especificar los componentes (u objetos) de la aplicación, y por otro lado se usa el Editor de Bloques para programar cómo la aplicación responde al usuario y a otros eventos externos. O sea, el Editor de Bloques se usa para programar el comportamiento de la aplicación. La Figura 1.20 muestra una vista general de la arquitectura de una aplicación.

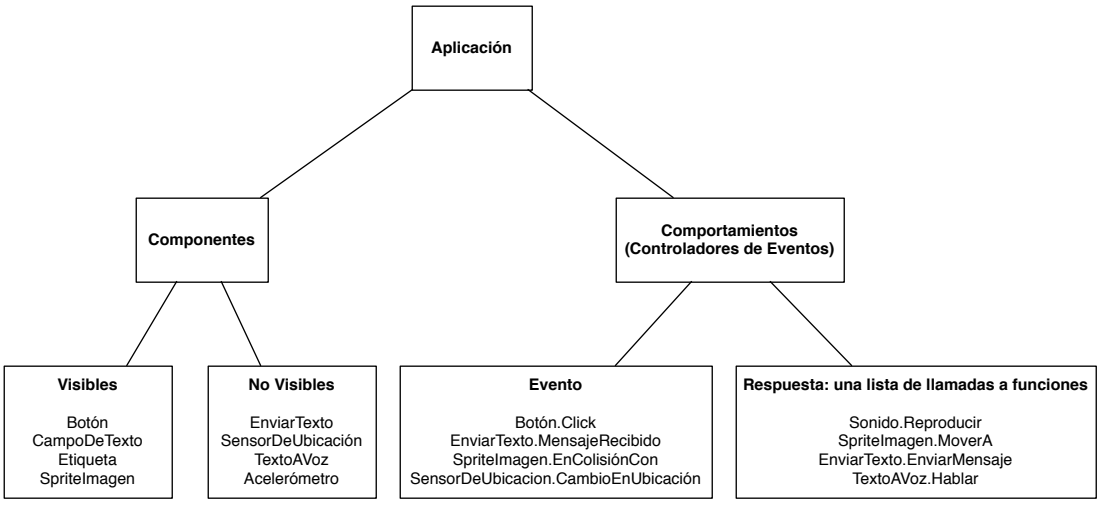

Figura 1.20: Arquitectura de una Aplicación: Componentes y Comportamiento

Componentes

Existen dos tipos principales de componentes en una aplicación: *visibles* y *no-visibles*. Los componentes visibles de una aplicación son aquellos que se pueden ver cuando la aplicación se ejecuta, por ejemplo los botones, cajas de texto y etiquetas. Al conjunto de componentes visibles de una aplicación también se le conoce como la *interfaz de usuario*.

Los componentes no-visibles son aquellos que no se pueden ver, y que por lo tanto no son parte de la interfaz de usuario. Su propósito es proveer acceso a las funcionalidad preexistentes de un dispositivo. Por ejemplo, el componente **EnviarTexto** envía y procesa los mensajes de texto (SMS), el componente **SensorDeUbicación** determina la ubicación del dispositivo, y el componente **TextoAVoz** habla un mensaje escrito como texto. Los componentes no-visibles representan la tecnología del dispositivo que está a disposición del programador.

Tanto los componentes visibles como no-visibles se definen por un conjunto de *propiedades*. Las propiedades son espacios de memoria para almacenar información sobre el componente. Los componentes visibles, tales como botones o etiquetas, tienen propiedades como su anchura, altura y alineamiento, los que en conjunto definen cómo luce el componente. Las propiedades de un componente son como celdas de una hoja de cálculo. El programador las modifica en el Diseñador de Componentes para definir la apariencia *inicial* del componente. También es posible utilizar bloques para cambiar estos valores durante la ejecución de la aplicación.

Comportamiento

Los componentes de una aplicación son generalmente sencillos de comprender, por ejemplo un campo de texto se usa para ingresar información o un botón se usa para ser presionado. En cambio, el comportamiento de una aplicación es conceptualmente difícil y a menudo complejo. El comportamiento define cómo la aplicación debiera responder a eventos, tanto eventos iniciados por el usuario (por ejemplo, se presiona un botón) como eventos externos (por ejemplo, se recibió un mensaje de texto). La dificultad de especificar ese comportamiento interactivo es el por qué la programación es un desafío.

Afortunadamente, AppInventor provee un lenguaje de *bloques* para especificar estos comportamientos. Los bloques hacen que programar el comportamiento sea similar a juntar las piezas de un puzzle, en contraste a recordar y escribir código como en lenguajes de programación tradicionales. Además, AppInventor está diseñado para especificar comportamientos en respuesta a eventos de una manera sencilla y directa.

Una Aplicación es Como una Receta de Cocina

Tradicionalmente se ha comparado al software (programas, aplicaciones) con una receta de cocina. Como en una receta, una aplicación tradicional sigue una secuencia lineal de instrucciones, como las de la Figura 1.21, que el computador debiera ejecutar.

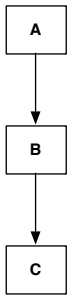

Figura 1.21: Una aplicación tradicional sigue una serie de pasos secuenciales, como una receta de cocina.

Si consideramos como ejemplo una aplicación de un cajero automático, una primera operación (A) sería

iniciar una transacción bancaria, luego (B) especificar el monto que se desea retirar, y finalmente (C) modificar la cuenta del cliente, entregar el dinero y luego imprimir el saldo por pantalla.

Una Aplicación Como un Conjunto de Controladores de Eventos

La visión de una aplicación como una receta de cocina calza bien con las aplicaciones o programas que se hacían en los inicios de la computación, pero no es una gran idea para la programación de dispositivos móviles, ni en la Web, ni en la mayoría de las aplicaciones y plataformas actuales en computación. La mayor parte del software moderno no realiza un puñado de instrucciones en un orden predeterminado. Lo que se hace es que el software *reaccione* ante distintos *eventos*—la mayoría iniciados por la interacción entre el usuario y la aplicación (por ejemplo, abrir un video en Youtube).

En el caso de las aplicaciones móviles tenemos diversos eventos gatillados por el usuario. Por ejemplo, al presionar el botón "Enviar", la aplicación responde enviando un mensaje de texto. El deslizar el dedo por la pantalla táctil también es otro evento. La aplicación podría responder dibujando una línea entre el punto donde se comenzó a deslizar el dedo y el punto donde se levantó.

Considerando lo anterior, las aplicaciones modernas se pueden entender mejor como máquinas de eventos-respuestas. Ocurre que las aplicaciones igual incluyen "recetas"—secuencias de instrucciones—pero la diferencia es que cada receta es realizada sólo en respuesta a algún evento en particular. La Figura 1.22 ilustra esta idea.

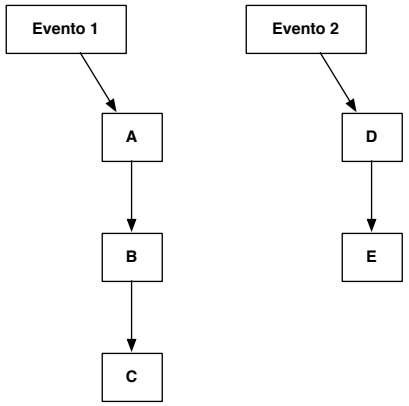

Figura 1.22: Una aplicación consiste en un conjunto de controladores de eventos.

A medida que los eventos ocurren, la aplicación reacciona ejecutando una secuencia de funciones. Las funciones son cosas que se pueden hacer con un componente (o hacia un componente), como mandar un mensaje de texto o cambiar la propiedad de algún componente (por ejemplo, cambiar el texto de un botón en la interfaz de usuario). *Llamar* a una función significa invocar a esa función, o sea hacer que ocurra lo que esa función hace. Usaremos el término *controlador de eventos* para referirnos tanto a un evento como al conjunto de funciones que se ejecutan como respuesta.

Muchos eventos son iniciados por el usuario, pero algunos no lo son. Una aplicación puede reaccionar a eventos que ocurren al *interior del teléfono*, tales como cambios en su sensor de orientación o su reloj (o sea, respecto al paso del tiempo), y también a eventos creados por cosas *externas al teléfono*, tales como la recepción de un mensaje de texto o una llamada telefónica, o la llegada de datos desde la Web. Esto se muestra en la Figura 1.23.

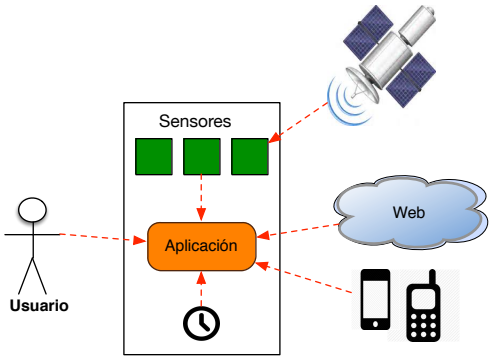

Figura 1.23: Eventos internos y externos al teléfono.

Una razón por la cual la programación en AppInventor es intuitiva es porque se basa directamente en este modelo evento-respuesta. Una aplicación se comienza a programar arrastrando un bloque de evento, el cual tiene la forma "*Cuando ... ejecutar ...*". Por ejemplo, consideremos una aplicación que responde al evento de presionar un botón leyendo el texto que el usuario a ingresado en una caja de texto, se programa con un único controlador de eventos, como se muestra en la Figura 1.24.

Figura 1.24: Código para hablar texto ingresado por el usuario.

Estos bloques especifican que cuando el usuario presiona el **Botón1**, el componente **TextoAVoz** debiera hablar las palabras que el usuario ha ingresado en el **CampoDeTexto1**. La respuesta al evento **Botón1.Click** es la llamada a la función **TextoAVoz.Hablar**. El controlador del evento son todos los bloques de la figura.

En AppInventor, todas las actividades ocurren en respuesta a algún evento. Por lo tanto una aplicación no debiera contener bloques que estén afuera de un bloque "Cuando-ejecutar". Por ejemplo, no tiene sentido que los bloques de la Figura 1.25 estén flotando solos en el editor de bloques.

Figura 1.25: Las respuestas deben ir asociadas a un controlador de eventos.

Tipos de Eventos

La **??** resume los tipos de eventos existentes en las aplicaciones AppInventor. A continuación describiremos cada uno de ellos.

Tipo de Evento	Ejemplo
Evento iniciado por el usuario	*Cuando* el usuario presiona Botón1, *realizar* ...
Evento de inicialización	*Cuando* la aplicación se empieza a ejecutar, *realizar* ...
Evento de temporizador	*Cuando* pasen 20 milisegundos, *realizar* ...
Evento de animación	*Cuando* dos objetos colisionen, *realizar* ...
Evento externo	*Cuando* el teléfono recibe un mensaje de texto, *realizar* ...

Tabla 1.2: Tipos de Eventos

Eventos iniciados por el usuario Los eventos iniciados por el usuario son el tipo de evento más común. Con aplicaciones tipo "trivia", el presionar botones es la manera usual de gatillar respuestas de la aplicación. Otras aplicaciones más gráficas responden a toques en la pantalla y a arrastrar elementos por la misma.

Eventos de inicialización Algunas veces una aplicación necesita realizar ciertas funciones justo cuando la aplicación comienza, y no en respuesta a alguna actividad del usuario final. Para este propósito, AppInventor considera el "lanzar" la aplicación como un evento. Por lo tanto, si el programador lo requiere, se pueden especificar funciones para ser ejecutadas inmediatamente una vez que la aplicación se abre. Para ello se utiliza el bloque **Screen1.Inicializar**. En el juego *Atrapa el Topo*, que se realizará el segundo día se utilizará este evento para asignar una posición inicial al azar a un elemento del juego, de forma similar al código de la Figura 1.26.

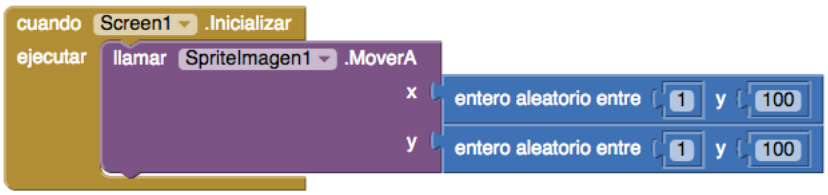

Figura 1.26: Asignando una posición al azar al inicio de la aplicación.

Eventos de temporizador Algunas actividades de una aplicación son gatilladas por el paso del tiempo. Por ejemplo, uno puede considerar que una animación es un objeto que se mueve gatillado por un evento de temporizador. AppInventor tiene un componente **Reloj** que se usa para gatillar eventos de temporizador. Por ejemplo, el código para mover una pelota por la pantalla 10 pixeles horizontalmente cada cierto intervalo de tiempo es similar al de la Figura 1.27.

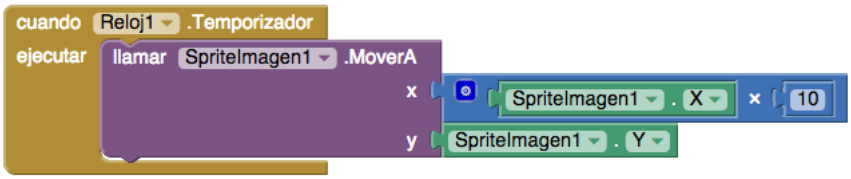

Figura 1.27: Animación gatillada por el reloj.

Eventos de animación Las actividades que involucran objetos gráficos (sprites) dentro de lienzos gatillarán eventos. Por lo tanto es posible programar juegos y otras aplicaciones interactivas especificando qué debiera ocurrir en eventos tales como un objeto alcanzando el borde del lienzo, o la colisión de dos objetos. Un código de ejemplo de colisión se muestra en la Figura 1.29.

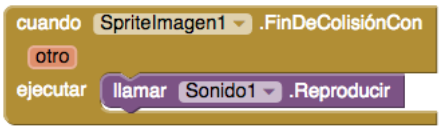

Figura 1.28: Controlador de eventos para la colisión entre dos objetos de un juego.

Eventos externos Cuando el teléfono recibe información GPS sobre su ubicación también se gatilla un evento. Lo mismo pasa cuando el teléfono recibe un mensaje de texto, por ejemplo, el código para una respuesta automática a un mensaje de texto recibido se muestra en la **??**.

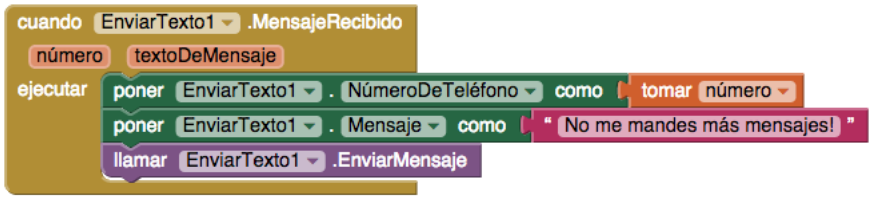

Figura 1.29: Respuesta automática a un mensaje de texto.

Todos los tipos de eventos se consideran de la misma forma que cualquier otro. En conclusión, cada aplicación creada usando AppInventor consiste en un conjunto de controladores de eventos: algunos para inicializar cosas, otros para responder al usuario, otros gatillados por el tiempo, y otros gatillados por eventos externos. Tu trabajo como programador es conceptualizar tu aplicación de esta manera, y diseñar la respuesta de cada controlador de eventos relevante para tu aplicación.

Los Controladores de Eventos pueden Hacer Preguntas

Las respuestas a los eventos no siempre siguen recetas de cocina secuenciales, sino que es posible hacer preguntas y repetir operaciones. "Hacer preguntas" significa consultar los datos almacenados en la aplicación y determinar el curso de acción (o *rama* de acción) dependiendo de las respuestas a estas consultas. En estos casos decimos que la aplicación tiene *ramas condicionales*, como se ilustra en la Figura 1.30.

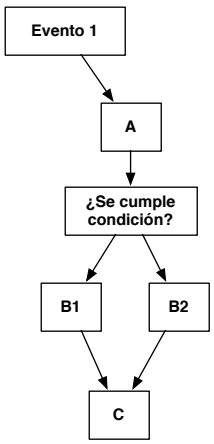

Figura 1.30: Aplicación con ramas condicionales de ejecución.

Las pruebas condicionales son preguntase tales como "¿Llegó el puntaje a 100?", o "¿El mensaje de texto recién recibido fue enviado por Juan?". Las preguntas pueden incluir fórmulas complejas que incluyen comparadores algebraicos (menor que, mayor que, igual a) y lógicos ("y", "o", "no"). Los comportamientos condicionales en AppInventor se especifican usando bloques **si, sino**. Por ejemplo, los bloques de la Figura 1.31 reportarán el mensaje "Ganaste!" si el jugador tiene más de 100 puntos.

Figura 1.31: Preguntar si el puntaje es superior a 100.

Los Controladores de Eventos pueden Repetir Operaciones

Además de hacer preguntas y ejecutar distintas recetas según cada caso, una respuesta a un evento puede también incluir repeticiones de operaciones múltiples veces. AppInventor provee diversos bloques para especificar tales operaciones, tales como **por cada** y **mientras-comprobar-ejecutar**, los cuales encierran una secuencia de bloques cuya ejecución se repetirá. Todos los bloques dentro de un **por cada** se ejecutan una vez para cada elemento en una *lista*. Por ejemplo, el código enviar un mensaje de texto a una lista de números de teléfono es similar al de la Figura 1.32.

Figura 1.32: Usar repetición para enviar un mensaje de texto a cada teléfono en una lista.

Los Controladores de Eventos pueden Recordar Cosas

A menudo un controlador de eventos necesita hacer un seguimiento de la información a medida que ejecuta sus bloques. Esta información puede almacenarse en ubicaciones de memoria conocidas como *variables*, las cuales se definen en el Editor de Bloques. De hecho, en las secciones anteriores ya hemos visto el uso de variables para almacenar el puntaje de un juego y para definir una lista de números de teléfono.

Las variables son como las propiedades, pero no están asociadas a ningún componente en particular. En un juego, por ejemplo, se puede definir una variable puntaje la cual será modificada por los controladores de eventos según la lógica propia de cada juego. Las variables almacenan los datos de manera temporal, mientras la aplicación se está ejecutando; pero cuando se cierra la aplicación los datos dejan de estar disponibles.

A veces una aplicación necesita recordar cosas no sólo mientras se ejecuta, sino incluso cuando es cerrada y vuelta a abrir. Por ejemplo para mantener un historial de los mayores puntajes, es necesario almacenar esta información a largo plazo, de manera que esté disponible la próxima vez que alguien juegue a este juego. Los datos que se mantienen incluso después de que una aplicación se cierra se llaman *datos persistentes*, y se almacenan en algún tipo de base de datos. En los siguientes días del taller profundizaremos más sobre el uso de variables y de datos persistentes.

1.5. Resumen