App Inventor

PID_00253722

Jordi Villar Colomé

Universitat Oberta de Catalunya

Jordi Villar Colomé

Jordi Villar Colomé (Badalona, 1969) es doctor ingeniero industrial por la Universidad Politécnica de Cataluña. Posgrado en Comercio Electrónico por la Universidad Ramon Llull. En el ámbito de la ingeniería, ha desarrollado su actividad profesional en centros internacionales de investigación, y posteriormente como consultor de proyectos TIC. Ejerce la docencia desde 1999 en centros de secundaria de ámbito público y concertado. Tiene experiencia en la implementación en el aula de proyectos de innovación educativa STEAM basados en MIT App Inventor. Actualmente, en el Instituto Badalona VII, participa como formador de formadores en programas STEAM.

Primera edición: febrero 2019

Av. Tibidabo, 39-43, 08035 Barcelona
Diseño: Manel Andreu
Realización editorial: Oberta UOC Publishing, SL

Índice

Introducción

El teléfono celular o móvil ha dejado de ser una simple herramienta de comunicación, para convertirse en todo un fenómeno social y cultural que traspasa fronteras y franjas de edad. Día tras día, aumenta el interés entre el público en general por la programación de aplicaciones para estos dispositivos, ya sea con fines educativos, profesionales, empresariales o simplemente por diversión.

El curso «Programación con MIT App Inventor de aplicaciones para dispositivos Android» se dirige a formadores que pretendan iniciar en este campo a personas de cualquier edad, con poca o ninguna experiencia previa en la programación de aplicaciones. El entorno de programación está expresamente desarrollado por un equipo del MIT (Massachusetts Institute of Technology) para un perfil de alumno de educación secundaria. Usamos únicamente programas informáticos de uso gratuito o *freeware*. Tan solo es necesario disponer de un ordenador con conexión a internet y un teléfono inteligente o tableta con sistema operativo Android.

El aprendizaje está basado en pequeños retos que habrá que ir superando, haciendo uso del dispositivo móvil desde el primer momento.

Glosario/abreviaciones

Android: sistema operativo diseñado principalmente para dispositivos móviles con pantalla táctil, como teléfonos inteligentes y tabletas, entre otros.

Aplicación: en informática, una aplicación es un programa informático diseñado como herramienta con la que un usuario puede interactuar para llevar a cabo una tarea o actividad concreta, en la mayoría de los casos relacionada con la obtención, el tratamiento y la representación de datos. Actualmente, con el uso de dispositivos móviles, se ha extendido el término *app*, que es un acortamiento de la palabra inglesa *application*.

Programar: proceso de diseño, redacción, prueba, corrección, mantenimiento y mejora del conjunto ordenado de instrucciones que definen el funcionamiento de un programa informático o aplicación.

Entorno de programación: aplicación informática que proporciona todas las herramientas necesarias para la programación de aplicaciones. Normalmente, consta de un editor de código fuente, herramientas de simulación y ejecución y un detector de errores.

Objetivos

1. Dotar a nuestro alumnado de los conocimientos necesarios para que sean capaces de desarrollar sus propias aplicaciones.

2. Familiarizarse y sacar el máximo partido posible de las diferentes posibilidades y recursos que ofrecen los dispositivos móviles.

3. Capacitar a nuestros alumnos para que puedan continuar por cuenta propia su proceso de investigación y aprendizaje, en función de los intereses y expectativas de cada uno de ellos.

4. Iniciar al alumnado en los conceptos, estructuras y procedimientos básicos, comunes a la programación informática en general.

1. El entorno MIT App Inventor

1.1. Introducción

MIT App Inventor es un entorno de programación intuitivo y visual que permite crear aplicaciones totalmente funcionales para dispositivos móviles Android. Facilita la creación de estas aplicaciones en mucho menos tiempo que los entornos de programación tradicionales.

MIT App Inventor funciona en línea de manera gratuita, y permite crear aplicaciones directamente en un navegador web. Además de un ordenador con conexión a internet y una versión actualizada del navegador, también es necesario disponer de una cuenta propia de Google Gmail a la que asociar nuestra sesión de usuario.

Con todos estos ingredientes, ya estaremos listos para empezar a desarrollar nuestras propias aplicaciones móviles.

1.2. Resumen

En este módulo, comprobaremos que disponemos de todos los elementos de hardware y software necesarios para poder llevar a cabo nuestros proyectos con MIT App Inventor, y los pondremos a punto.

Crearemos nuestro propio perfil de usuario de MIT App Inventor a partir de nuestra cuenta de Google Gmail.

Aprenderemos a crear un proyecto desde cero, y daremos nuestros primeros pasos en el entorno de programación.

Desarrollaremos una aplicación muy sencilla y la probaremos en un dispositivo móvil.

Verificaremos, en definitiva, que todos los elementos con los que vamos a trabajar funcionan correctamente.

1.3. Requisitos técnicos

Para poder empezar a dar nuestros primeros pasos con MIT App Inventor, debemos comprobar que disponemos del hardware y software mínimos imprescindibles, tal y como se especifica en su sitio web:

1) Ordenador y sistema operativo

- Macintosh (con procesador Intel): Mac OS X 10.5 o superior.
- Windows: Windows XP, Windows Vista, Windows 7 o superior.
- GNU/Linux: Ubuntu 8 o superior, Debian 5 o superior.

2) Navegador

- Mozilla Firefox 3.6 o superior.
- Apple Safari 5.0 o superior.
- Google Chrome 4.0 o superior.
- Microsoft Internet Explorer **no es compatible**.

3) Dispositivo móvil

Teléfono inteligente o tableta con sistema operativo Android 2.3 (Gingerbread) o superior.

4) Cuenta propia de Google Gmail

En caso de no tenerla, es necesario crear una: https://accounts.google.com/SignUp.

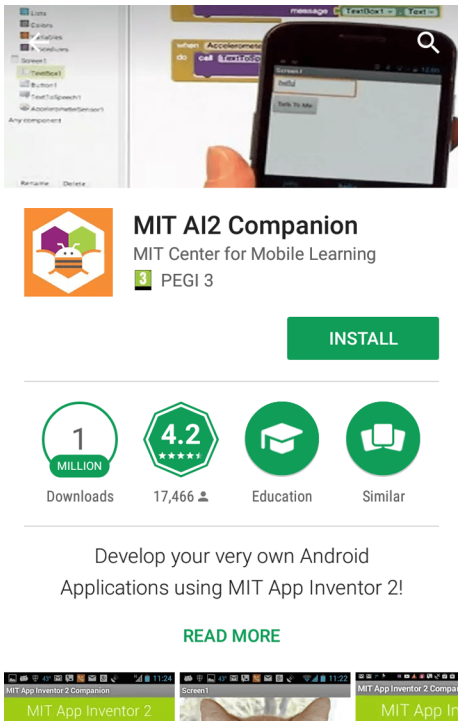
5) Conexión a internet y MIT AI2 Companion

Será aconsejable que el ordenador y el dispositivo móvil estén conectados a internet a través de la misma red Wi-Fi. En la mayoría de las redes domésticas sencillas, el sistema también suele funcionar si la conexión del ordenador a internet se hace a través de cable de red.

En estos casos, tan solo será necesario **instalar en el teléfono inteligente o tableta** la aplicación llamada **MIT AI2 Companion**, que podemos encontrar en Google Play (figura 1). Esta aplicación permite simular directamente en nuestro dispositivo móvil el funcionamiento de las aplicaciones que estamos desarrollando.

En caso contrario existen otras alternativas, aunque algo más complejas. Podéis consultar el apartado de problemas técnicos frecuentes al final de la unidad.

Figura 1. Página de instalación de **MIT AI2 Companion** en Google Play



1.4. Crear nuestra propia sesión

Una vez preparado el software y hardware necesarios, vamos a crear nuestra propia sesión de usuario en el entorno MIT App Inventor, siguiendo estos pasos:

1) Desde el ordenador, y con alguno de los navegadores compatibles (Mozilla, Safari, Chrome), accedemos al sitio web de MIT App Inventor: http://appinventor.mit.edu/explore/.

2) Hacemos clic en la parte superior derecha, en la etiqueta «**Create apps!**».

3) El sistema exige la identificación usando usuario y contraseña de nuestra propia cuenta Google Gmail. Es posible que hayamos iniciado una sesión Gmail previamente en el mismo navegador. En este caso, accederemos directamente al paso siguiente.

4) El sistema pedirá que demos autorización expresa para usar la cuenta Google Gmail con la que nos hemos identificado para crear nuestra sesión en MIT App Inventor. Hacemos clic en «**Permitir**».

5) A continuación, deberemos leer y aceptar las condiciones de servicio y la política de privacidad de datos. Hacemos clic en «**I accept the terms of use**».

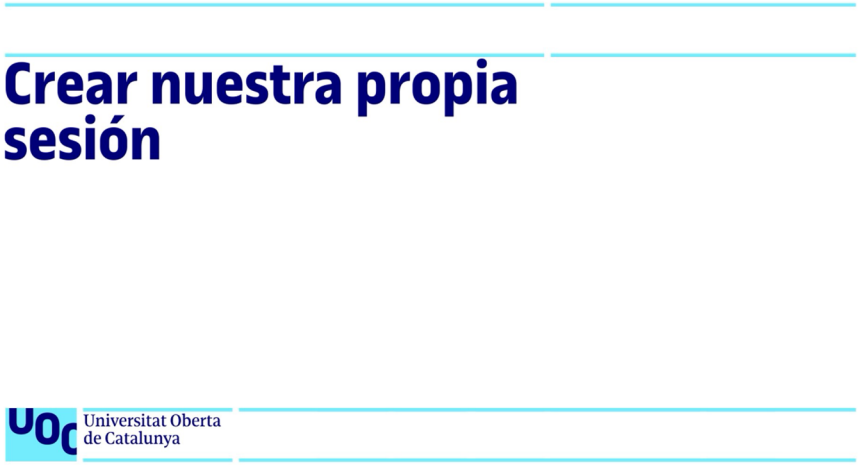
6) Ya estamos dentro del entorno MIT App Inventor. Aparecerá una ventana en la que nos piden responder una encuesta para mejorar la calidad del servicio. Podemos dejarlo para más tarde, haciendo clic en «**Take Survey Later**».

7) Dado que es la primera vez que accedemos a nuestra sesión, a continuación aparece una ventana de bienvenida al sistema. Hacemos clic en «**Do Not Show Again**» y luego en «**Continue**».

8) Una segunda ventana de bienvenida desaparecerá en cuanto hagamos clic fuera de ella.

Podéis ver el proceso anterior en este videotutorial.

Vídeo 1. Crear nuestra propia sesión

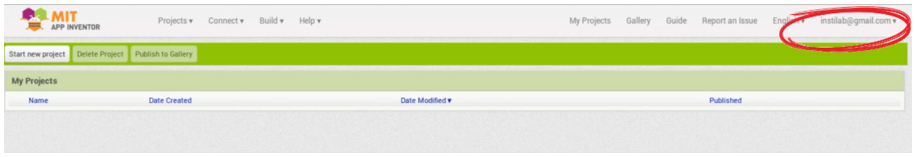


Simulación realizada en el entorno MIT App Inventor

En la figura 2, se muestra el entorno MIT App Inventor la primera vez que accedemos a nuestra sesión. Observamos lo siguiente:

- La lista de **proyectos** («My Projects») aún está vacía.
- Nuestra dirección de Gmail aparece en la esquina superior derecha.
- Al lado de nuestra dirección de Gmail, un menú desplegable nos permite cambiar el idioma del entorno.

Figura 2. Aspecto del entorno MIT App Inventor la primera vez que accedemos a nuestra sesión



Nuestra dirección de Gmail deberá aparecer en la esquina superior derecha.

1.5. Crear el primer proyecto

Ya estamos preparados para crear nuestro primer proyecto, paso previo a la creación de aplicaciones. Una vez creada nuestra sesión, seguiremos estos pasos:

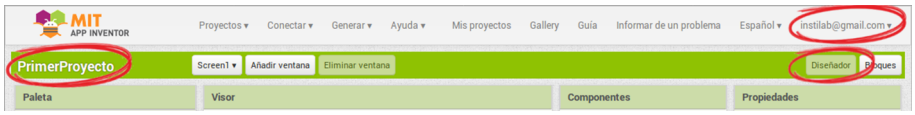
1) Hacemos clic sobre el botón «Comenzar un proyecto nuevo» («Start new project»), situado en la parte superior izquierda.

2) A continuación, aparece una ventana que nos invita a dar nombre a nuestro nuevo proyecto. Utilizamos el nombre «PrimerProyecto».

3) Ya aparece el nuevo proyecto en la lista de «**Proyectos**». Al cabo de pocos segundos, el sistema nos redirige al entorno donde podremos empezar a programar nuestra aplicación. En caso contrario, hacemos clic sobre el nombre del proyecto.

4) Aparecemos en el entorno de «**Diseñador**». En la parte superior izquierda, observamos el nombre de nuestro proyecto; y en la parte superior derecha, nuestra dirección de Gmail, tal y como se aprecia en la figura 3.

Figura 3. Aspecto del entorno MIT App Inventor, una vez creado el proyecto «PrimerProyecto»



Nuestra dirección de Gmail deberá aparecer en la esquina superior derecha, y estaremos en modo «Diseñador».

Podéis ver el proceso en este videotutorial.

Vídeo 2. Crear el primer proyecto

Simulación realizada en el entorno MIT App Inventor

1.6. Diseñar y programar una aplicación de prueba

Ya estamos preparados para construir una sencilla aplicación de prueba. Consistirá en un simple botón en pantalla que, al hacer clic sobre el mismo, cambia de color. Para ello, una vez situados en el entorno de «**Diseñador**» de nuestra recién creada sesión, seguiremos estos pasos:

1) En el menú «**Paleta-Interfaz de usuario**», situado a la izquierda de la ventana, hacemos clic sobre el elemento «**Botón**» y lo arrastramos y lo soltamos dentro del espacio «**Screen1**», situado en la columna «**Visor**».

2) Observamos que aparece fijado en la parte superior izquierda de «**Screen1**», con el texto «Texto para Botón1».

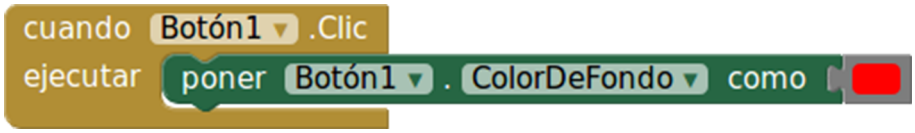
3) A continuación, hacemos clic sobre el botón «**Bloques**» en la parte superior derecha, justo debajo de nuestra dirección de Gmail. Esto nos lleva al entorno de programación propiamente dicho.

4) Dentro de menú «**Bloques**» de la izquierda, hacemos clic sobre «**Botón1**», para abrir un segundo menú donde, entre otros muchos, aparece en primer lugar el bloque «cuando Botón1.Clic ejecutar», que arrastramos con el ratón hacia la derecha, fuera del menú gris, y lo soltamos dentro del espacio «**Visor**».

5) Repetimos el paso anterior, pero esta vez con el bloque de color verde «poner Botón1.ColorDeFondo como», que soltamos dentro del bloque anterior para que encajen como dos piezas de puzle, tal y como se ve en la figura 4.

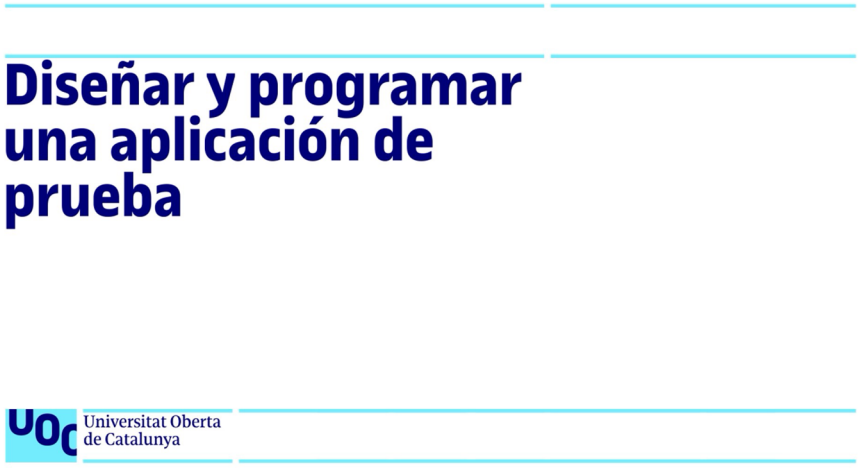
6) Para finalizar, vamos a asignar un color yendo de nuevo al menú «**Bloques**» de la izquierda, pero esta vez hacemos clic sobre «**Colores**», y arrastramos el bloque con el color rojo (o cualquier otro) para encajarlo en el hueco libre que queda del anterior bloque «poner Botón1.ColorDeFondo como», tal y como se ve en la figura 4.

Figura 4. Disposición de bloques resultado de la programación de la aplicación de prueba en «PrimerProyecto»



Podéis ver el proceso en este videotutorial.

Vídeo 3. Diseñar y programar una aplicación de prueba



Simulación realizada en el entorno MIT App Inventor

1.7. Probar una aplicación

Llega el momento de comprobar que el programa funciona correctamente, para lo cual vamos a recurrir a la aplicación **MIT AI2 Companion**, que instalamos en nuestro dispositivo móvil al principio del módulo.

1) En la barra superior de la aplicación web, desplegamos el menú «**Conectar**» y elegimos la opción «**AI Companion**». A continuación, aparece una ventana emergente con un código QR junto con un código de 6 letras.

2) Es el momento de abrir la aplicación **MIT AI2 Companion** instalada en nuestro dispositivo móvil, y que nos ofrece dos opciones para conectarnos con el ordenador:

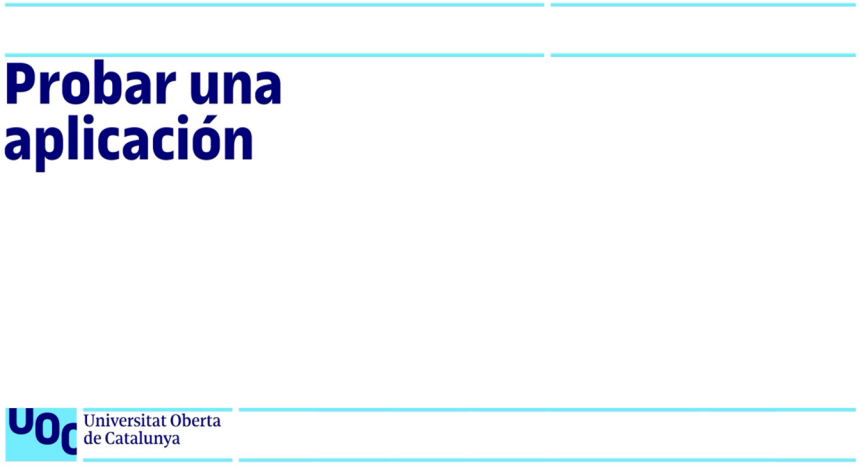
- Teclear el código de 6 letras y hacer clic en «**connect with code**».

- O bien escanear el código QR, para lo cual hay que hacer clic en «**scan QR code**».

3) Si todo está correcto, al cabo de unos segundos vemos aparecer en la pantalla del dispositivo móvil el botón de la aplicación de prueba que, al hacer clic sobre el mismo, debería cambiar de color.

Podéis ver el proceso en este videotutorial.

Vídeo 4. Probar una aplicación



Simulación realizada en el entorno MIT App Inventor

Hay que recordar que se trata de una simulación, y que la aplicación no se va a instalar en ningún momento. En cuanto se interrumpa la conexión entre el dispositivo móvil y el ordenador, la aplicación de prueba dejará de funcionar y no dejará rastro. En próximos módulos, aprenderemos cómo instalar una aplicación de manera permanente.

En caso de que no se estableciera la conexión entre el ordenador y el dispositivo móvil, hay que consultar alguna de las soluciones que se proponen en el apartado de pro blemas técnicos frecuentes, al final de la unidad.

1.8. Actividades

Actividad 1

Para demostrar que se ha creado con éxito la sesión de usuario, tenéis que obtener una captura de pantalla similar a la figura 3, guardarla como documento de imagen en formato JPG, GIF, PNG o similar y enviar este documento.

Como en la figura 3, es necesario que en la imagen se vea claramente el nombre de vuestro proyecto en la parte superior izquierda, y vuestra dirección de Gmail en la parte superior derecha.

Para más información de cómo obtener y manejar una imagen de la pantalla, podéis consultar:

- Si tenéis Windows 10, la mejor opción es el programa Recortes. https://support.microsoft.com/es-es/help/13776/windows-use-snipping-tool-to-capture-screenshots.

- En otras plataformas de Windows: https://techlandia.com/convertir-imagen-captura-pantalla-archivo-jpg-como_267951/.

Información sobre cómo hacer una captura de pantalla en Windows y Mac OS X: https://www.java.com/es/download/faq/screenshot.xml

Para más información sobre cómo enviar el documento creado, podéis preguntar en el foro.

Actividad 2

Una vez creada la aplicación de prueba que se propone en este módulo, hay que obtener el fichero de exportación correspondiente, de la siguiente manera:

1) En la barra superior, desplegamos el menú «**Proyectos**» y elegimos la opción «**Exportar a mi ordenador el proyecto (.aia) seleccionado**».

2) El sistema genera un fichero, «PrimerProyecto.aia», que se descargará y se guardará en la carpeta de descargas del navegador.

Podéis ver el proceso en este videotutorial.

Vídeo 5. Exportar proyecto

Simulación realizada en el entorno MIT App Inventor

Enviad vuestro fichero «PrimerProyecto.aia» para su evaluación.

Para más información sobre cómo enviar el fichero creado, podéis preguntar en el foro.

1.9. Bibliografía y recursos en línea

Sitio web de MIT App Inventor: http://appinventor.mit.edu/

Requisitos técnicos para la instalación y uso de MIT App Inventor: http://appinventor.mit.edu/explore/ai2/setup.html

Creación de cuenta Google Gmail: https://accounts.google.com/SignUp

MIT AI Companion en Google Play: https://play.google.com/store/apps/details?id=edu.mit.appinventor.aicompanion3

Tutorial para la instalación de App Inventor Companion: http://appinventor.mit.edu/explore/ai2/setup-device-wifi.html

Videotutoriales del MIT para seguir practicando: https://www.youtube.com/watch?v=Vdo8UdkgDD8; https://www.youtube.com/watch?v=0hikoCvM3oc

1.10. Problemas técnicos frecuentes

1) No dispongo de un dispositivo móvil para probar las aplicaciones.

Será conveniente instalar un emulador Android en el ordenador. Podéis ver enlace.

2) Dispongo de dispositivo móvil, pero no de red Wi-Fi para probar las aplicaciones.

Será conveniente preparar una conexión a través de cable USB. Podéis ver enlace

3) MIT AI Companion no se conecta con el ordenador

- Comprobar que ordenador y dispositivo móvil están conectados a la misma red Wi-Fi. Podéis ver enlace

- La configuración de algunas redes privadas complejas, como escuelas o empresas, puede impedir la conexión. Debéis consultar con el administrador del sistema.

- Reiniciar la conexión desde MIT App Inventor: menú «Conectar» → «**Reiniciar conexión**» (o «**Reiniciar completamente**»).

- Optar por instalar un emulador Android en el ordenador: podéis ver «Problema A».

- Optar por preparar una conexión a través de cable USB: podéis ver «Problema B».

- Optar por instalar la aplicación creada directamente en el dispositivo móvil:

Podéis ver enlace (*APK file*).

2. Texto, números, imagen y sonido

2.1. Introducción

Una vez preparado el entorno de programación, ya estamos listos para generar nuestras primeras aplicaciones. Empezamos con una aplicación sencilla basada en el manejo de texto y números, lo cual nos permite aprender conceptos básicos de programación. Mejoraremos nuestra aplicación con la incorporación de imagen y sonido. Finalmente, aprendemos a instalar nuestra aplicación de manera permanente en cualquier dispositivo móvil.

2.2. Resumen

Distinguiremos las zonas de trabajo de los espacios «Diseñador» y «Bloques» del entorno de programación.

Introduciremos los conceptos de componente, propiedad y evento como elementos básicos de una aplicación móvil.

Nos iniciaremos en el uso de métodos, funciones y estructuras como herramientas de programación.

Comprobaremos la importancia del algoritmo condicional para el control del flujo de un programa.

Aprenderemos a usar variables como elementos imprescindibles en programación.

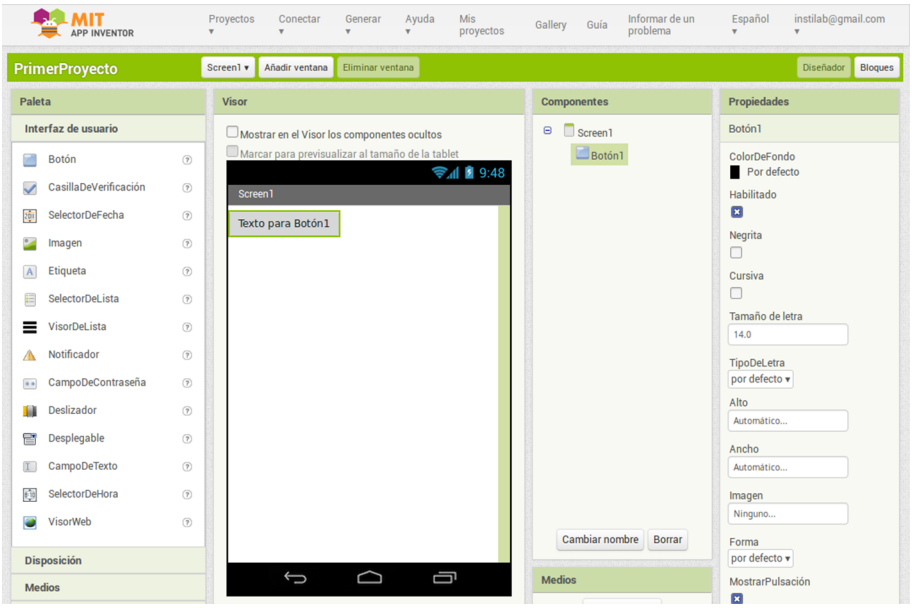
Incorporaremos imagen y sonido a las aplicaciones.

Veremos el procedimiento para instalar nuestra aplicación de manera permanente en cualquier dispositivo móvil.

2.3. Espacio «Diseñador». Componentes y propiedades

El entorno MIT App Inventor se divide en dos espacios o pantallas: el espacio «**Diseñador**» y el espacio «**Bloques**». El espacio «Diseñador», tal y como se puede apreciar en la figura 5, se divide en cuatro zonas o columnas, de izquierda a derecha: «**Paleta**», «**Visor**», «**Componentes**» y «**Propiedades**».

Figura 5. Las 4 columnas del espacio «Diseñador»: «Paleta», «Visor», «Componentes» y «Propiedades»



1) Paleta

En la columna «**Paleta**», encontramos los componentes.

Los componentes son elementos que se pueden incorporar a nuestra aplicación y que intervienen en su aspecto y funcionamiento. Se dividen en visibles (si son visibles en pantalla) u ocultos (si su funcionamiento es externo a la pantalla). En función de su naturaleza, se agrupan en diferentes menús desplegables.

En la figura, 1 se ve desplegado el menú «**Interfaz de usuario**». El componente «**Botón**», por ejemplo, es un componente visible de este menú. Para incorporar un componente a la aplicación, basta con arrastrarlo con el ratón (hacer clic + arrastrar) a la pantalla, en la columna «**Visor**».

2) Visor

Consta de una pantalla de móvil en la que se representa la disposición de los componentes que vamos incorporando a nuestra aplicación. Si deseamos hacer una simulación del funcionamiento de la aplicación, no basta con el «**Visor**». Será necesario usar MIT AI Companion.

3) Componentes

A medida que vamos incorporando elementos visibles al «**Visor**», estos aparecen en la columna «Componentes», en una lista jerárquica. Desde esta columna, podemos eliminar y cambiar el nombre de un componente. Para ello, hay que seleccionarlo y hacer clic en los botones «Cambiar nombre» o «Borrar».

Al seleccionar un componente, sus características aparecen en la columna «**Propiedades**».

4) Propiedades

Todos y cada uno de los componentes tienen una serie de **propiedades**. Las propiedades de un componente determinan su aspecto, la información que contiene o su estado de funcionamiento. Estas propiedades se pueden modificar y consultar antes o durante la ejecución del programa.

Recuperamos el «PrimerProyecto», tal y como hemos visto en el vídeo 3, o arrastramos un componente «**Botón**» desde la «**Paleta**» al «**Visor**». Si seleccionamos el botón en la columna «**Componentes**», como en la figura 1, vemos que aparecen las siguientes propiedades:

ColorDeFondo	Por defecto	Color de fondo para el botón, por defecto gris.
Habilitado	x	Funcionamiento del botón. Puede deshabilitarse.
Negrita		Tipo de letra para el texto del botón. Desactivada.
Cursiva		Tipo de letra para el texto del botón. Desactivada.
Tamaño de la letra	14.0	Tamaño de letra para el texto del botón.

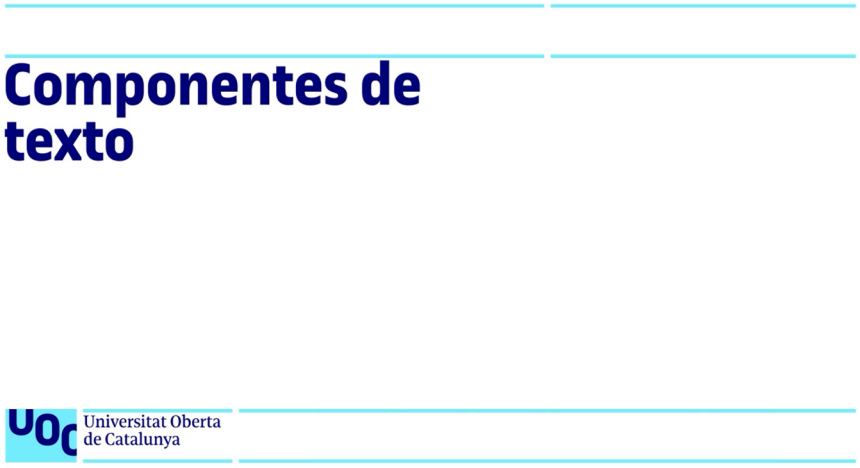
...

Y así, hasta 15 propiedades distintas para el componente «Botón».

2.4. Componentes de texto. Espacio Bloques. Eventos

Para comprender el funcionamiento de estos componentes, vamos a crear una aplicación a la que llamaremos «ProyectoHola», en la que el dispositivo móvil pregunte el nombre del usuario y le salude educadamente, tal y como se ve en este vídeo.

Vídeo 6. Componentes de texto



Simulación realizada en el entorno MIT App Inventor

Primero hacemos una lista de los **componentes** que necesitamos:

- Una **etiqueta** para que el dispositivo se comunique con nosotros y escriba en ella sus mensajes.
- Un **campo de texto** para podernos comunicar con el móvil y escribir en él nuestro nombre.
- Un **botón** para enviar el texto que escribamos.

Luego hacemos una lista ordenada de las **acciones** que tenemos que hacer el dispositivo y el usuario:

```
1. Aplicación: escribe un mensaje preguntando el nombre del usuario.
2. Usuario:    escribe su nombre.
3. Usuario:    envía el nombre escrito.
4. Aplicación: construye la frase.
5. Aplicación: escribe la frase.
```

Esta planificación de componentes y acciones que debe tener la aplicación es esencial en el proceso de programación. Es aconsejable hacer esta previsión antes de empezar a construir el código propiamente dicho.

En el siguiente vídeo, detallamos el proceso de programación con MIT App Inventor:

- Creamos el proyecto, que en el vídeo llamamos «ProyectoHola».
- Incluimos la etiqueta, el campo de texto y el botón en el «**Visor**». Les cambiamos el nombre por «Mensaje», «Nombre» y «Enviar». Este hecho no

afecta al funcionamiento, pero ayuda a la comprensión del código. También cambiaremos el texto del botón a «Enviar».

Vídeo 7. Proceso de programación con MIT App Inventor

Simulación realizada en el entorno MIT App Inventor

Una vez acabado el diseño de pantalla, pasamos al espacio «**Bloques**». En este espacio, aparece un menú de **bloques** y un **visor** donde vamos a ir situando y encajando estos bloques. Será el espacio de programación propiamente dicho.

Para programar la lista de acciones, vamos a trabajar con **eventos**. Los **eventos** son hitos o instantes muy concretos del transcurso del programa, que marcan el inicio y el final de cada una de las acciones de nuestra lista.

Así, por ejemplo, nuestra primera acción es «escribe la pregunta del nombre del usuario». Esta acción se debe desencadenar tan pronto como se inicia la pantalla «Screen1» de la aplicación. En el menú «**Bloques**», si hacemos clic en el componente «Screen1», los eventos asociados se muestran como bloques de color marrón. El bloque que debemos desplazar al **visor** es el que nos muestra la figura 6.

Figura 6. Bloque para ejecutar acciones al inicializarse la pantalla «Screen1»

El otro evento necesario en nuestra aplicación es el que desencadena el usuario al hacer clic en el botón para enviar el texto con su nombre. Hay que buscar el bloque de la figura 7 en el componente «Enviar» del menú «**Bloques**».

Figura 7. Bloque para ejecutar acciones al hacer clic en el botón «Enviar»

Acción 1: buscamos en el menú «Mensaje» el bloque «**poner Mensaje.Texto como**», y lo completamos con un bloque de texto con la frase «¿Cuál es tu nombre?». El conjunto lo desplazamos dentro del bloque de la figura 2. A los bloques como «**poner Mensaje.Texto como**» los llamaremos ***métodos***. Un **método** se utiliza para cambiar las propiedades de un componente.

Acción 4: con el bloque «**Unir**» del menú «Texto», construimos la frase concatenando un bloque de «Texto» que contenga la palabra «Hola» con la propiedad «**Nombre.Texto**» del menú «Nombre». El bloque «**Unir**» es una **función**. Una función devuelve un resultado a partir de unos datos iniciales.

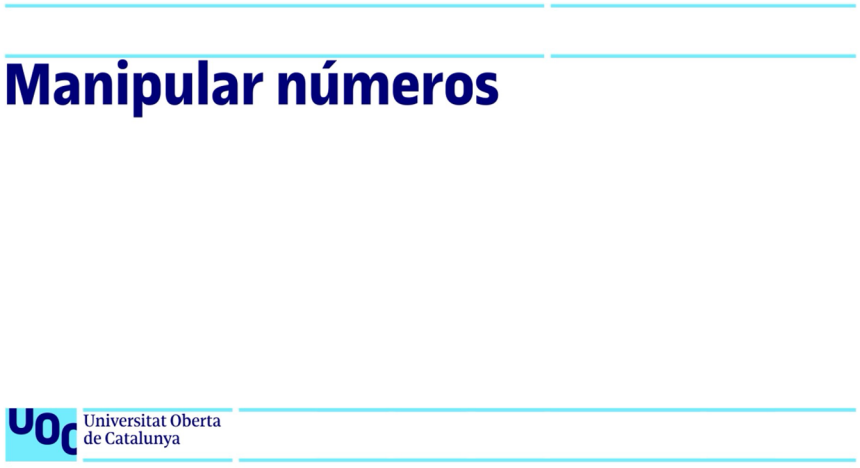
Acción 5: buscamos en el menú «Mensaje» el bloque «**poner Mensaje.Texto como**», y lo completamos con el bloque que hemos construido en la acción 4. Finalmente, hay que desplazar el bloque resultante dentro del bloque de la figura 3.

La aplicación ya está lista para ser probada con MIT AI2 Companion. Ahora, probad la aplicación en vuestro móvil o tableta. ¿Entendéis lo que está haciendo la aplicación?

2.5. Manipular números. Variables. Estructuras condicionales

Seguimos con un nuevo proyecto, al que llamaremos «NúmeroSecreto1», en el que el usuario intentará adivinar un número secreto del 1 al 100 con las pistas que irán apareciendo en pantalla, tal y como se ve en este vídeo.

Vídeo 8. Manipular números



Simulación realizada en el entorno MIT App Inventor

1) Lista de componentes

Serán los mismos de la aplicación anterior: una etiqueta «Mensaje», un campo de texto «Numero» y un botón «Enviar». El texto inicial para la etiqueta es «Adivina un número del 1 al 100». El texto inicial para el campo de texto lo dejamos en blanco. El texto para el botón, «Enviar».

2) Lista de acciones

```
1. Aplicación: establece el número secreto.
2. Usuario:    escribe un intento.
3. Usuario:    envía el intento.
4. Aplicación: según el valor del intento informa "¡Acertaste!" o "El
               número secreto es mayor" o "El número secreto es menor".
```

Como veremos, en el espacio «**Bloques**» será necesario usar tres nuevos tipos de bloques:

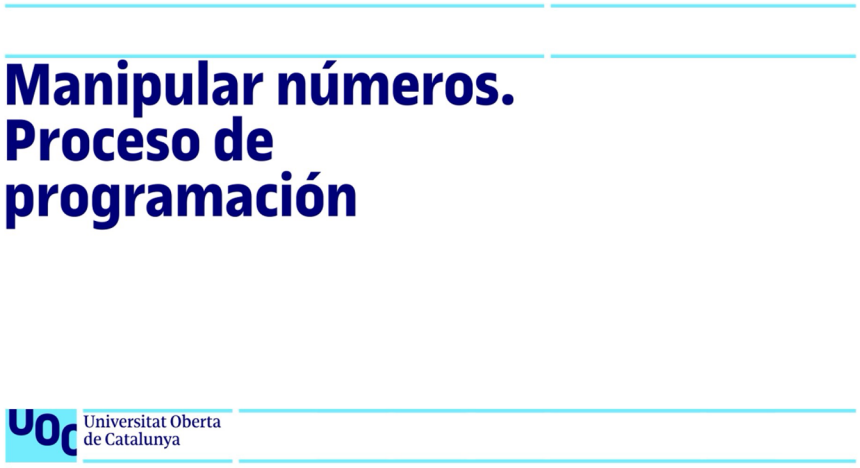
1) Variable: menú «**Variable**». Una variable es un espacio de memoria, identificado por un nombre arbitrario, donde almacenamos un valor de texto o numérico que puede ser consultado y modificado a voluntad desde la aplicación. En nuestro caso, en la **acción 1**, sirve para almacenar el número secreto generado y en la **acción 4**, para consultar su valor. Adoptará el nombre «NumSecreto».

2) Entero aleatorio: menú «**Matemáticas**». Esta función permite generar un entero aleatorio que se encuentre dentro de un intervalo definido por otros dos enteros. En nuestro caso, se usa en la **acción 1** para generar el número secreto.

3) Si_entonces_: menú «**Control**». Estructura condicional que permite controlar las acciones de la aplicación en función de una condición previa que debe resultar verdadera o falsa. En nuestro caso, en la **acción 4**, permite decidir el mensaje que hay que escribir en pantalla después de cada intento. Una **estructura** nos permite controlar el comportamiento del programa a partir de los valores de determinadas variables.

En este vídeo, se muestra el proceso de programación de la aplicación «NúmeroSecreto1».

Vídeo 9. Manipular números. Proceso de programación



Simulación realizada en el entorno MIT App Inventor

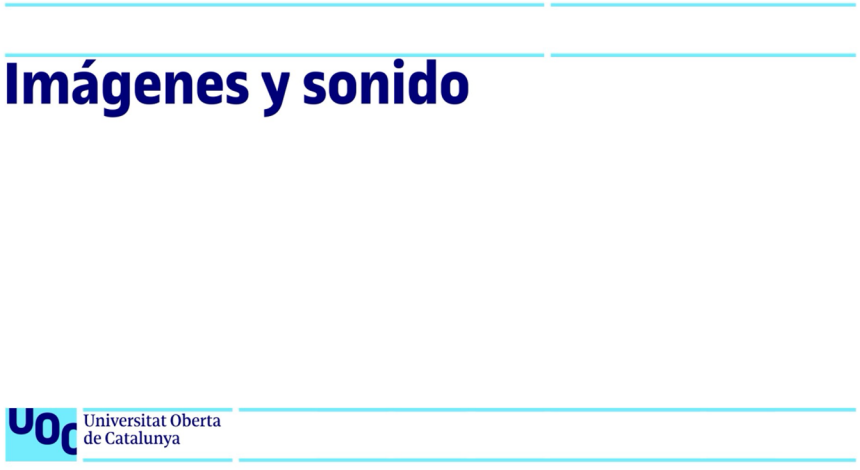
2.6. Imágenes y sonido

Vamos a mejorar la aplicación «NúmeroSecreto1» con tres modificaciones:

- Justificando al centro los elementos de la pantalla.
- Incorporando dos emoticonos: 😃😞, que aparecerán en pantalla, según si el intento del usuario es o no correcto, respectivamente. Descargad los archivos de imagen contento.png y triste.png en vuestro ordenador.
- Haciendo sonar dos efectos de sonido distintos según si el intento del usuario es o no correcto. Descargad los archivos de sonido aplauso.mp3 y fallo.mp3 en vuestro ordenador.

La nueva aplicación, de nombre «NúmeroSecreto2», debe funcionar como en este vídeo.

Vídeo 10. Imágenes y sonido



Simulación realizada en el entorno MIT App Inventor

1) Lista de componentes

Habrá que ampliar la lista de componentes con:

- Una **disposición vertical** (menú «**Disposición**»), donde incluir todos los componentes visibles para que queden centrados verticalmente.

- Una **imagen** (menú «**Interfaz de usuario**») que corresponda a uno u otro de los emoticonos. Emplearemos el nombre «Emoticono».

- Un **sonido** (menú «**Medios**») para reproducir los efectos de sonido. Emplearemos el nombre «Efecto».

2) Lista de acciones

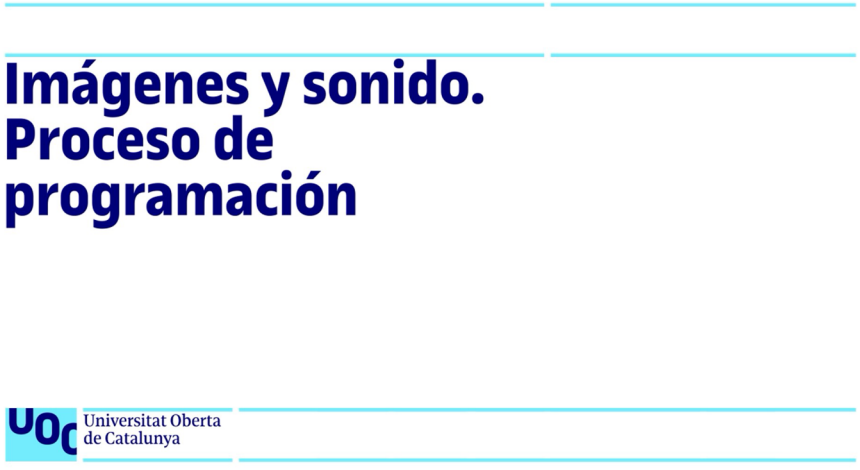
Las mismas del proyecto anterior, ampliando la acción 5:

```
5. Aplicación: si el intento es correcto entonces
                    escribe "¡Acertaste!"
                    muestra el icono contento.png
                    asigna el audio aplauso.mp3 como efecto de sonido
               si no
                    muestra el icono triste.png
                    asigna el audio fallo.mp3 como efecto de sonido
                    si el número secreto es mayor que el intento
                    entonces
                         escribe "El número secreto es mayor"
                    si no
                         escribe "El número secreto es menor"

               haz sonar el efecto de sonido asignado.
```

En este vídeo se muestra el proceso de programación de la aplicación «NúmeroSecreto2», a partir de la aplicación «NúmeroSecreto1».

Vídeo 11. Imágenes y sonido. Proceso de programación



Simulación realizada en el entorno MIT App Inventor

2.7. Instalación permanente de la aplicación

Hasta ahora, hemos verificado el funcionamiento de nuestras aplicaciones a través de una conexión Wi-Fi con MIT AI Companion. Al salir del entorno de programación, la aplicación deja de funcionar sin dejar rastro. A continuación, vamos a compilar e instalar una aplicación para que quede de manera permanente entre las demás del dispositivo móvil.

De manera previa al proceso, hay que autorizar al dispositivo móvil la instalación de aplicaciones descargadas desde un sitio distinto a Google Play Store (y MIT App Inventor lo es). Este es un criterio de seguridad establecido por defecto en todos los dispositivos. En el menú Android de ajustes, hay que hacer lo siguiente:

Ajustes -> Seguridad -> Fuentes desconocidas -> Permitir la instalación de aplicaciones desde fuentes desconocidas.

También debemos asegurarnos de que contamos con una aplicación específica para leer códigos QR (no con MIT AI Companion). Por ejemplo, cualquiera de las que aparecen aquí.

Ahora, desde el menú superior del entorno vamos a Generar -> Aplicación (generar código QR para el archivo APK), y vemos que, tras compilar durante algún tiempo, aparece un código QR en pantalla. Escaneando este código, descargamos un archivo en formato APK que, al ser abierto, lleva a cabo la instalación de la aplicación.

Si queremos conservar el archivo APK para su posterior distribución, optamos por la segunda opción: Generar -> Aplicación (guardar archivo APK en nuestro ordenador).

2.8. Actividades

Las actividades que hay que entregar en este módulo son:

1) Enviar los archivos «ProyectoHola.aia» y «ProyectoHola.apk», resultado del apartado 4.

2) Enviar los archivos «NúmeroSecreto1.aia» y «NúmeroSecreto1.apk», resultado del apartado 5.

3) Enviar los archivos «NúmeroSecreto2.aia» y «NúmeroSecreto2.apk», resultado del apartado 6.

2.9. Glosario

Compilar: es el proceso de traducción de un programa del lenguaje en el que ha sido escrito al lenguaje propio del dispositivo que lo va a ejecutar.

Componente: los componentes, también llamados *objetos*, son elementos que se pueden incorporar a nuestro programa y que intervienen en su aspecto y funcionamiento.

Estructura: una estructura nos permite controlar el comportamiento del programa a partir de los valores de determinadas variables.

Evento: los eventos son hitos o instantes muy concretos del transcurso del programa, que marcan el inicio y/o el final de las acciones en las que este se divide.

Función: una función produce un resultado a partir de unos datos iniciales.

Método: un método es un procedimiento que se utiliza para cambiar las propiedades de un componente en el transcurso de un programa.

Propiedad: las propiedades de un componente, también llamadas *atributos*, determinan su aspecto, la información que contiene o su estado de funcionamiento. Estas propiedades se pueden modificar y consultar antes o durante la ejecución del programa.

Variable: una variable es un espacio de memoria, identificado por un nombre arbitrario, donde almacenamos un valor de texto o numérico que puede ser consultado y modificado a voluntad desde el programa.

3. Animación y creación de juegos

3.1. Introducción

Según datos del 2017, el portal de estadísticas sensortower.com sitúa en el *top ten* de corporaciones con más descargas de aplicaciones del mercado a seis firmas dedicadas a los juegos para plataforma móvil, compitiendo directamente con Google y Facebook-WhatsApp. En el segmento de usuarios entre 10 y 18 años, esta tendencia se acentúa aún más.

Es evidente que la animación y creación de juegos es un estímulo muy interesante para seguir avanzando en el aprendizaje de la programación de aplicaciones para dispositivos Android.

Una vez sentados los fundamentos básicos de algorítmica y programación, estamos preparados para afrontar el siguiente paso.

3.2. Resumen

Identificaremos los componentes del menú «**Dibujo y animación**» de MIT App Inventor.

Aprenderemos a manejar los componentes «**Lienzo**» y «**Pelota**» como base para juegos y animaciones en 2D.

Usaremos algunos de los **sensores** del dispositivo móvil como mandos de control para juegos y animaciones.

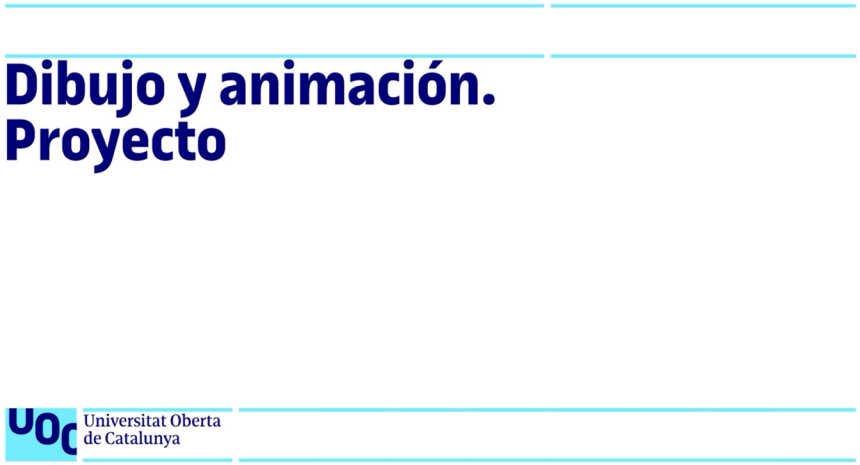
Nos iniciaremos en la inteligencia artificial para poder detectar y reaccionar ante las diferentes situaciones que pueden surgir durante el transcurso del juego.

3.3. Menú «Dibujo y animación». Sistema de coordenadas

Para jugar, siempre necesitamos una cancha de juego y un personaje. En AppInventor, el equivalente a la cancha será el componente «**Lienzo**», y como personaje optaremos por el componente «**Pelota**», aunque también se puede emplear «**SpriteImagen**». Todos estos elementos pertenecen al menú «**Dibujo y animación**».

Vamos a crear una aplicación, a la que llamaremos «ProyectoJuego». Será un juego en el que podremos mover una pelota sobre la pantalla del dispositivo móvil, mediante dos botones: uno hacia arriba y otro hacia abajo, tal y como se aprecia en este vídeo.

Vídeo 12. Dibujo y animación. Proyecto

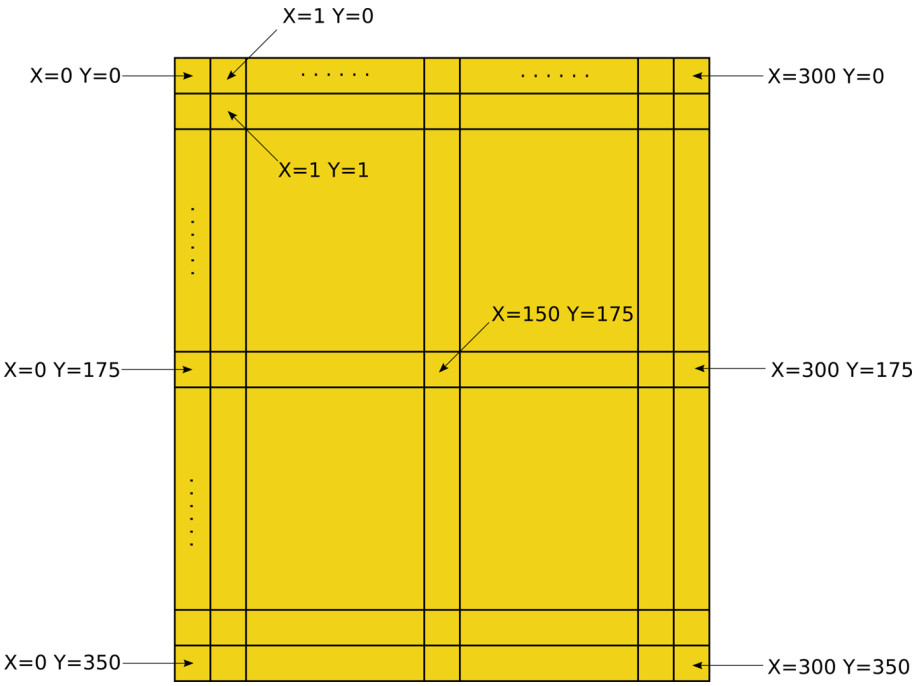


Simulación realizada en el entorno MIT App Inventor

Antes de empezar, debemos saber lo que es el **sistema de coordenadas** del lienzo, para identificar la posición de la pelota sobre el mismo. Un espacio plano, como el lienzo, se dice que tiene dos dimensiones (2D). Esto es así porque se necesitan al menos dos variables, a las que llamaremos **X** e **Y**, para localizar un punto dentro de su espacio. En un tablero de ajedrez, por ejemplo, para identificar una casilla concreta hace falta saber el número **X** de la columna y el número **Y** de la fila que ocupa.

En nuestro juego, la medida de nuestro lienzo será de 300 × 350 píxeles, lo cual equivaldría a un tablero de 300 casillas de ancho por 350 casillas de alto. Si situamos la pelota en la casilla superior izquierda, diremos que sus coordenadas son **X** = 0, **Y** = 0. Tal y como se aprecia en la figura 8, si la desplazamos a la casilla situada inmediatamente a su derecha, entonces pasa a ocupar la posición **X** = 1, **Y** = 0, y la casilla **X** = 1, **Y** = 1 es la que queda justo debajo. Si queremos situar la pelota en el centro exacto del lienzo, debemos mandarla a las coordenadas **X** = 150, **Y** = 175, justo la mitad de sus dos dimensiones.

Figura 8. Coordenadas de algunas de las posiciones de un lienzo de 300 × 350 píxeles



Para empezar, planificamos la lista de componentes y de acciones necesarias:

1) Lista de componentes

a) Un **lienzo** (Paleta / Dibujo y animación) dentro del cual se pueda desplazar el personaje (la pelota).

Propiedades

Nombre	Color	Ancho	Alto
cancha	amarillo	300 píxeles	350 píxeles

b) Una **pelota** (Paleta / Dibujo y animación) protagonista del juego, cuyos movimientos podamos controlar. Se sitúa inicialmente en el centro del lienzo.

Propiedades

Nombre	Color	X	Y
bola	negro	150	175

c) Dos **botones** (Paleta / Interfaz de usuario) para controlar el movimiento de la pelota.

Propiedades

Nombres	Texto	ColorDeFondo	ColorDeTexto
arriba	arriba	Azul	Blanco
abajo	abajo	Azul	Blanco

2) Lista de acciones:

```
1. Usuario:    pulsa uno de los dos botones (arriba o abajo).
2. Aplicación: sitúa la pelota en la nueva posición.
```

Para la programación de las acciones en el espacio «**Bloques**», la figura 9 muestra cómo conseguir el desplazamiento de la **bola** modificando el valor de su coordenada **Y** con el evento «clic» de cada uno de los botones. Para mover la **bola** hacia arriba o hacia abajo, se resta o se suma respectivamente un píxel al valor vigente de **Y**.

Figura 9. Control de la posición de la **bola** en función del botón en el que se ha hecho clic

Ahora probad la aplicación en vuestro móvil o tableta. ¿Funciona como en el vídeo 12 de ejemplo? ¿Entendéis lo que está haciendo la aplicación? ¿Qué ocurre cuando la **bola** alcanza los límites de la **cancha**?

A continuación, vamos a introducir una serie de mejoras a «ProyectoJuego», para lo cual crearemos una nueva versión, llamada «ProyectoJuegoMejorado», de la siguiente manera:

Proyectos -> Guardar proyecto como... -> Nuevo nombre: ProyectoJuegoMejorado.

Las mejoras son:

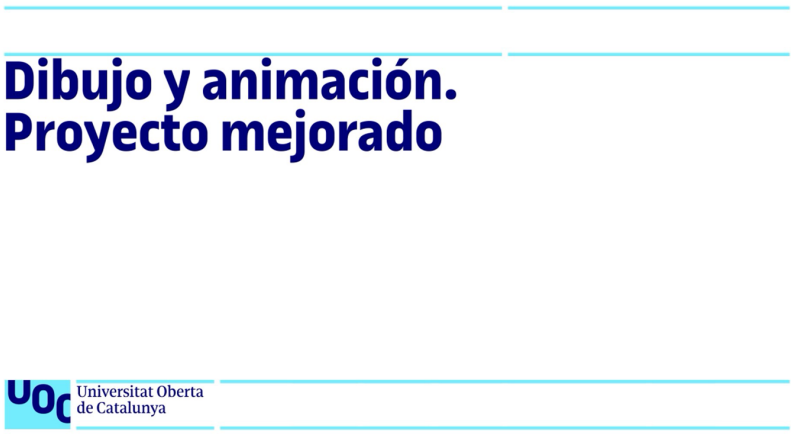
- Aumentar la velocidad de desplazamiento de la **bola**.
- Impedir que la **bola** sobrepase los límites de la cancha.
- Mover la bola también a derecha e izquierda.
- Dar continuidad al movimiento de la **bola** hasta que se modifique haciendo clic en un botón.

1) Aumentar la velocidad de desplazamiento de la bola

En los bloques de la figura 9, en lugar de sumar o restar un solo píxel a la coordenada **Y** de bola, podemos hacerlo con 2, 3 o más píxeles, de manera que el desplazamiento con cada clic del botón será el doble, el triple o más.

Probad la aplicación en vuestro móvil o tableta. ¿Funciona como en este vídeo de ejemplo? En el vídeo, hemos sumado 4.

Vídeo 13. Dibujo y animación. Proyecto mejorado

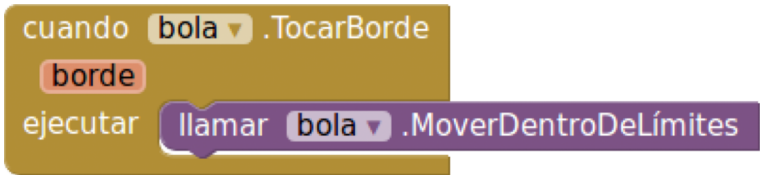


Simulación realizada en el entorno MIT App Inventor

2) Impedir que la bola sobrepase los límites de la cancha

Se recomienda usar el evento y el método de «**bola**» que se propone en la figura 10.

Figura 10. Evento y método del elemento «**bola**» para mantenerla dentro de los límites del lienzo



Probad la aplicación en vuestro móvil o tableta. ¿Funciona correctamente?

3) Mover la bola también a derecha e izquierda

1) Vamos al espacio «**Diseñador**».

2) En la paleta, en el apartado «Disposición», seleccionaremos «Disposición horizontal» y lo moveremos a la parte inferior de la pantalla.

3) Arrastraremos los dos botones que ya tenemos dentro del nuevo elemento, tal y como se muestra en este vídeo.

Vídeo 14. Temporización y detección de colisiones

Simulación realizada en el entorno MIT App Inventor

4) Añadimos los dos nuevos botones a los ya existentes, en el espacio «Disposición horizontal». El resultado debe quedar como se ve en la figura 11.

Propiedades

Nombres	Texto	ColorDeFondo	ColorDeTexto
derecha	derecha	Verde	Negro
izquierda	izquierda	Verde	Negro

5) Incorporar los dos respectivos eventos o **bloques**, como en la figura 9, pero esta vez modificando la coordenada **X** de «**bola**».

4) Dar continuidad al movimiento de la bola hasta que se modifique haciendo clic en un botón

1) En el espacio «**Diseñador**», hay que modificar la **propiedad** «velocidad» de «**bola**»:

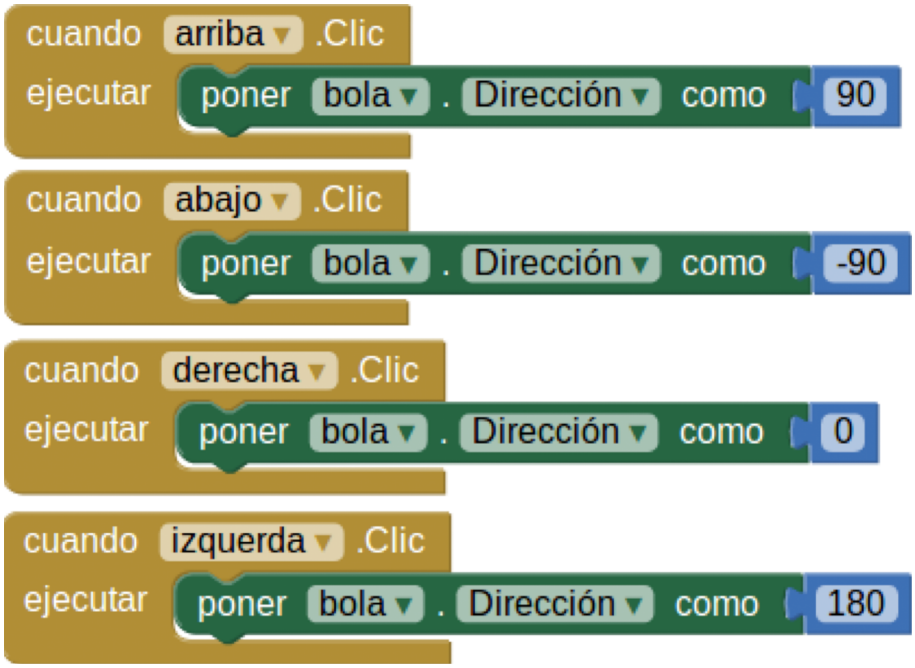
Propiedades

Velocidad
5

2) En el espacio «Bloques», hay que cambiar los 4 métodos asociados a los cuatro eventos de los botones por los que se indican en la figura 11. Se asigna como dirección el ángulo de esta respecto al eje horizontal derecho.

Probad la aplicación en vuestro móvil o tableta con estas nuevas modificaciones. ¿Funciona tal y como esperabais? ¿Entendéis lo que ha cambiado respecto a la versión anterior? ¿Qué otras mejoras se os ocurren?

Figura 11. Cambio en la dirección de la bola hacia arriba: bola orientación = 90. Los valores para abajo, derecha, izquierda-derecha son, respectivamente: -90, 0, 180



3.4. Control del juego con sensores del móvil

Para este apartado, vamos a seguir evolucionando nuestro juego y crearemos una nueva versión a partir de «ProyectoJuegoMejorado», a la que llamaremos «ProyectoJuegoSensores», de la siguiente manera:

> Proyectos -> Guardar proyecto como… -> Nuevo nombre: ProyectoJuegoSensores

Los objetivos son dos:

- «Lanzar» la **bola** rozándola con un dedo, en la dirección y con la velocidad deseadas. Usaremos los sensores de la pantalla táctil.

- Inclinar la pantalla para hacer que la **bola** «ruede» sobre la **cancha**, en la dirección y con la velocidad con las que lo haría sobre un plano real. Usaremos los sensores de inclinación.

1) Lanzar

Sin eliminar los **bloques** de los botones de la figura 11, incorporaremos un evento «**bola.Lanzado**» dentro del menú «**bola**», como en la figura 12. Los bloques «**tomar dirección**» y «**tomar velocidad**» se obtienen haciendo clic sobre las etiquetas «**velocidad**» y «**dirección**» del bloque «**bola.Lanzado**».

Figura 12. Bloque evento «**bola.lanzado**»

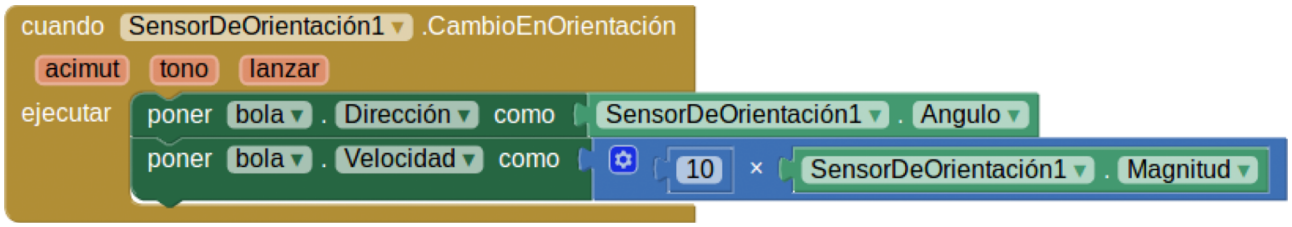
Probad la aplicación en vuestro móvil o tableta. Arrastrad la bola con vuestro dedo. ¿Cómo podéis aumentar la velocidad de la bola desde el programa? ¿Cómo podéis aumentar el tamaño de la bola?

2) Inclinar y rodar

En modo «**Diseñador**», incorporad a la pantalla del **visor** el componente «**Sensor de orientación**», desde el menú «**Sensores**» de la **paleta**. Se trata de un componente no visible, con lo que acabará situado fuera del visor.

A continuación, pasamos al modo «**Bloques**» para incorporar un bloque evento «**SensordeOrientación1.CambioEnOrientación**», de manera que cada vez que cambiemos la inclinación del dispositivo móvil respecto a su posición horizontal, la bola se desplazará en esa dirección y con una velocidad proporcional a la magnitud de la inclinación (véase la figura 13). Multiplicamos por 10 la velocidad para poder apreciar mejor el movimiento.

Figura 13. Bloque evento «SensordeOrientación1.CambioEnOrientación»

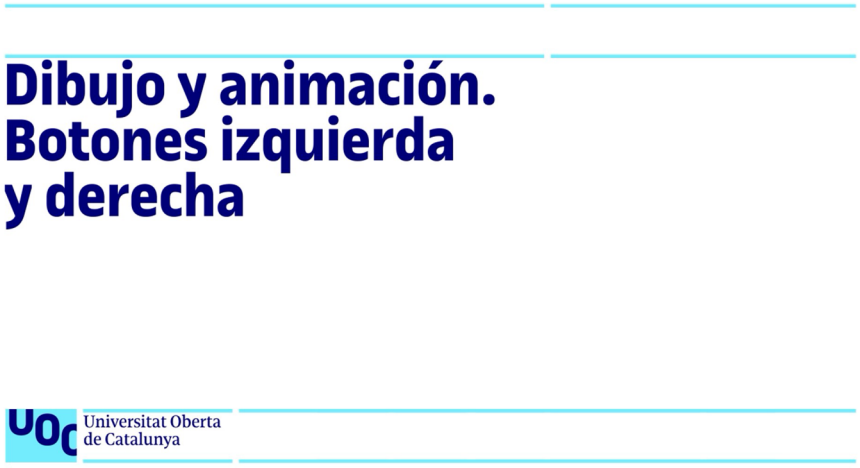


Antes de probar la aplicación en vuestro móvil o tableta, deberéis desactivar la opción de autorrotación de la pantalla. En caso contrario, al inclinar el dispositivo, se alteraría la disposición del lienzo. Esta última versión deberá incluir las tres maneras de controlar la bola: botones, sensor de pantalla y sensor de inclinación. ¿Funciona tal y como se esperaba? ¿Entendéis lo que ha cambiado respecto a la versión anterior? ¿Qué otras mejoras se os ocurren?

3.5. Temporización y detección de colisiones

Vamos a dar los últimos retoques al juego, añadiendo a un adversario que aparecerá en pantalla durante unos segundos y que deberemos alcanzar antes de que cambie de posición, tal y como se puede ver en este vídeo.

Vídeo 15. Dibujo y animación. Botones izquierda y derecha



Simulación realizada en el entorno MIT App Inventor

Volvemos a crear una nueva versión a partir de «ProyectoJuegoSensores», a la que llamaremos «ProyectoJuegoFinal», de la siguiente manera:

> Proyectos -> Guardar proyecto como… -> Nuevo nombre: ProyectoJuegoFinal

1) Lista de componentes (adicionales)

a) Una nueva **pelota** adversario en el juego. De color y tamaño diferente a la original, que se situará sobre el lienzo, en posiciones aleatorias.

Propiedades

Nombre	Color	Radio
bola2	rojo	10

b) Un **reloj** del menú «**Sensores**». Se trata de un cronómetro que, tras un tiempo determinado, desencadena un evento. En nuestro caso, el cambio de posición de «**bola2**» es cada 5 segundos (5.000 milisegundos). Se trata de un componente no visible, con lo cual acabará situado fuera del visor.

Propiedades

Nombre	IntervaloDelTemporizador
crono	5000

2) Lista de acciones que hay que añadir a las originales

```
Acción 1: cada 5 segundos mover bola2 a una nueva posición aleatoria en la
          cancha.
Acción 2: en caso de colisión desaparecen bola y bola2.
```

Analicémoslas por separado:

Acción 1: cada 5 segundos, mover «**bola2**» a una nueva posición aleatoria en la cancha.

En modo «**Bloques**», incorporar el evento «**crono.Temporizador**» para que desencadene un cambio en valor de las coordenadas **X** e **Y** de «**bola2**», dentro del rango de dimensiones de la **cancha** (como en la figura 14).

Acción 2: en caso de colisión, desaparecen «**bola**» y «**bola2**».

Añadir el evento «**bola2.EnColisiónCon**», de manera que cuando «**bola**» alcance «**bola2**», las dos desaparezcan y concluya el juego (como en la figura 15).

Figura 14. Bloque evento «**crono.Temporizador**»

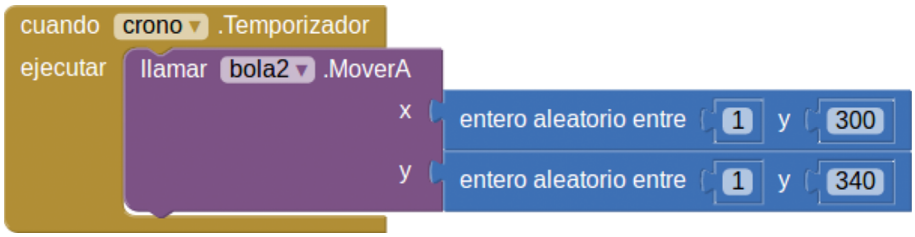
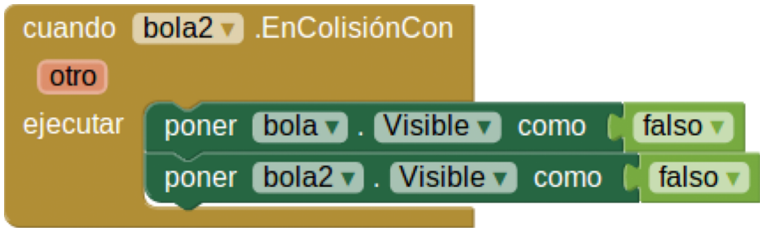


Figura 15. Bloque evento «**bola2.EnColisiónCon**»



Probad la aplicación en vuestro móvil o tableta. ¿Funciona como estaba previsto? ¿Qué mejoras se os ocurren? ¿Os atrevéis a usar el componente «**SpriteImage**» (Paleta / Dibujo y animación) para sustituir las pelotas por imágenes para los protagonistas del juego?

3.6. Actividades

Las actividades que hay que entregar en este módulo son:

1) Enviar los archivos «ProyectoJuego.aia» y «ProyectoJuego.apk», del apartado 3.

2) Enviar los archivos «ProyectoJuegoMejorado.aia» y «ProyectoJuegoMejorado.apk», resultado del apartado 3.

3) Enviar los archivos «ProyectoJuegoSensores.aia» y «ProyectoJuegoSensores.apk», resultado del apartado 4.

4) Enviar los archivos «ProyectoJuegoFinal.aia» y «ProyectoJuegoFinal.apk», resultado del apartado 5.

4. Distribución de componentes en pantalla

4.1. Introducción

Una parte muy importante en el éxito de una aplicación es un correcto diseño, distribución y aspecto de los componentes en pantalla. La disposición de la información y de los elementos interactivos es esencial para una comprensión intuitiva del funcionamiento y uso de la aplicación.

También tiene impacto sobre la experiencia del usuario y en la valoración de la aplicación, y es un factor clave para su más o menos rápida recomendación y difusión.

4.2. Resumen

Haremos el diseño del aspecto en pantalla de una aplicación, priorizando la facilidad y la comprensión de uso.

Aprenderemos a relacionar las principales características de un diseño de pantalla en cuanto a la disposición y distribución de sus componentes.

Utilizaremos los distintos componentes de distribución de MIT App Inventor, para ir incorporando progresivamente las líneas maestras de nuestro diseño.

Destacaremos los valores más apropiados para las propiedades de cada componente que para el diseño previsto.

4.3. Diseño previo de componentes en pantalla

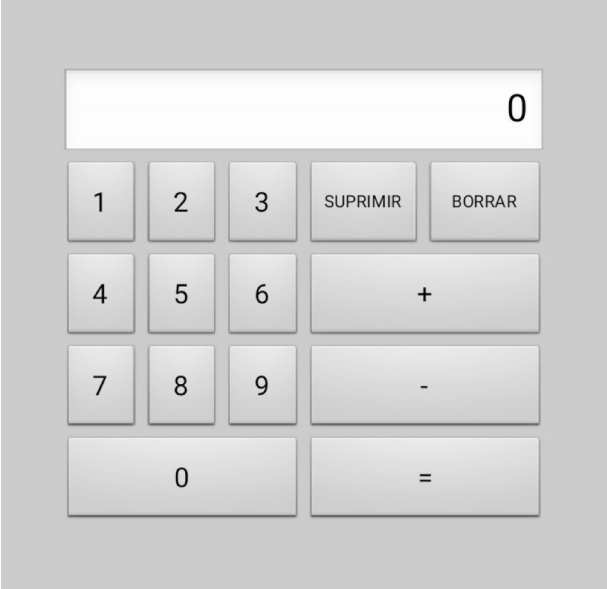
En el módulo anterior, nos dimos cuenta de que una misma aplicación, una vez en funcionamiento, puede verse de muchas maneras distintas en pantalla. Más allá de los componentes y acciones que dotan de funcionalidad nuestra aplicación, debemos prestar especial atención al diseño y la distribución visual de sus componentes.

Para ello, será necesario imaginar de antemano el aspecto que creamos más adecuado para el usuario, desde el punto de vista estético y de usabilidad. Es muy aconsejable hacer un esquema o dibujo, aunque sea a mano alzada, para tener una referencia concreta de lo que se pretende conseguir.

Para practicar la distribución de componentes, vamos a crear una aplicación llamada «ProyectoCalculadora», que consistirá en la programación de una calculadora clásica para hacer sumas y restas de números enteros.

De las muchas opciones de diseño posibles para esta calculadora, vamos a marcarnos como objetivo el esquema que se recoge en la figura 16.

Figura 16. Diseño previo de la distribución de componentes para el «ProyectoCalculadora»



Del análisis atento del esquema de la figura 16, observamos los siguientes puntos que hay que tener en cuenta, ordenados de más general a más particular:

- **P1**. La pantalla tiene el gris como color de fondo.
- **P2**. La calculadora ocupa la parte central de la pantalla.
- **P3**. En la calculadora, se distinguen dos zonas dispuestas verticalmente: **ventana** y **teclado**.
- **P4**. La calculadora ocupa aproximadamente el 80 % del ancho de la pantalla.
- **P5**. La **ventana** ocupa todo el ancho de la calculadora.
- **P6**. El número dentro de la ventana tiene justificación a la **derecha**.
- **P7**. El **teclado** se divide horizontalmente en dos zonas independientes de igual anchura: **numérico** y **operaciones**.
- **P8**. El teclado **numérico** y el teclado de **operaciones** se dividen en cuatro niveles verticales.

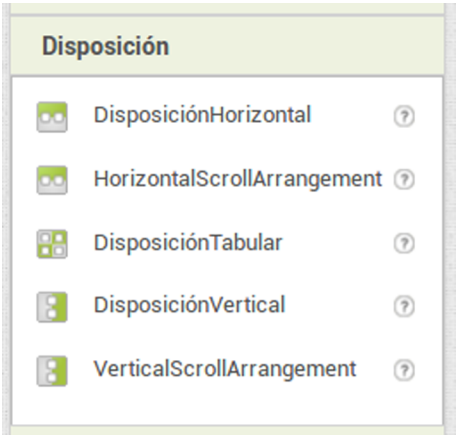
- **P9**. El teclado **numérico** tiene tres niveles, con tres botones iguales, y el cuarto nivel tiene un único botón (0) que ocupa todo el ancho.

- **P10**. El primer nivel del teclado **operaciones** lo comparten dos teclas iguales, y los tres niveles restantes contienen un solo botón ocupando todo el ancho.

4.4. Componentes de disposición

Algunos de los 10 puntos anteriores, deducidos del diseño de la figura 16, se podrán solucionar ajustando al valor apropiado las propiedades correspondientes de cada componente. Ahora bien, la posición relativa de los componentes y de los grupos de componentes será fijada con las herramientas que nos proporciona el menú «Paleta/Disposición» del modo «Diseñador», y que se muestran en la figura 17.

Figura 17. Componentes del menú «Paleta/Disposición»



Estos componentes permiten la **disposición** horizontal o vertical como posición relativa entre dos o más componentes. También permiten la disposición en forma de tabla y el efecto *scroll* o ascensor, en el caso de que los componentes ocupen más espacio que el fijado.

Los componentes del menú «Paleta/Disposición» se arrastran hasta la pantalla del **visor** igual que el resto de los componentes, tal y como hemos visto en módulos anteriores.

Procederemos analizando punto por punto la lista de diez del apartado anterior, y deduciendo cómo combinar los componentes de **disposición** para obtener el resultado previsto para «ProyectoCalculadora» en la figura 18.

P1. La pantalla tiene el gris como color de fondo.

P2. La calculadora ocupa la parte central de la pantalla.

P1 y **P2** se van a obtener modificando «DispHorizontal», «DispVertical» y «ColorDeFondo» en las **propiedades** de «Screen1», tal y como se aprecia en la figura 18.

Figura 18. **Propiedades** de «Screen1» para conseguir **P1** y **P2**

P3. En la calculadora, se distinguen dos zonas dispuestas verticalmente: **ventana** y **teclado**.

P4. La calculadora ocupa aproximadamente el 80 % del ancho de la pantalla.

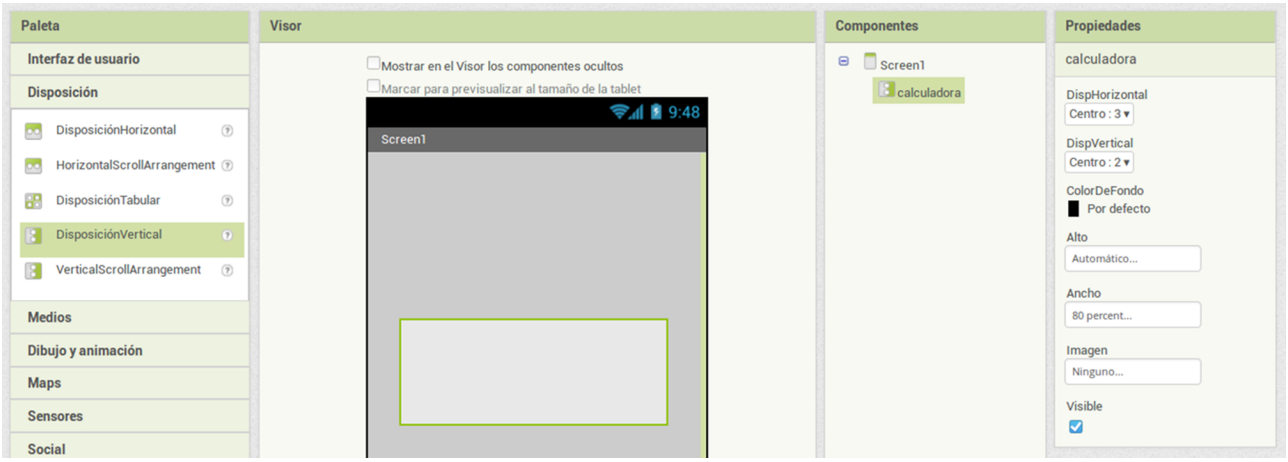
Como es lógico, la disposición vertical de dos componentes se obtiene insertando un componente «DisposiciónVertical» con las siguientes propiedades:

Propiedades

Nombre	DispHorizontal	DispVertical
calculadora	Centro	Centro
Alto	**Ancho**	
Automático	80 %	

Para obtener el resultado que se representa en la figura 19.

Figura 19. Aspecto del modo «Diseño» tras implementar el componente **calculadora**



P5. La **ventana** ocupa todo el ancho de la calculadora.

P6. El número dentro de la ventana tiene justificación a la **derecha**.

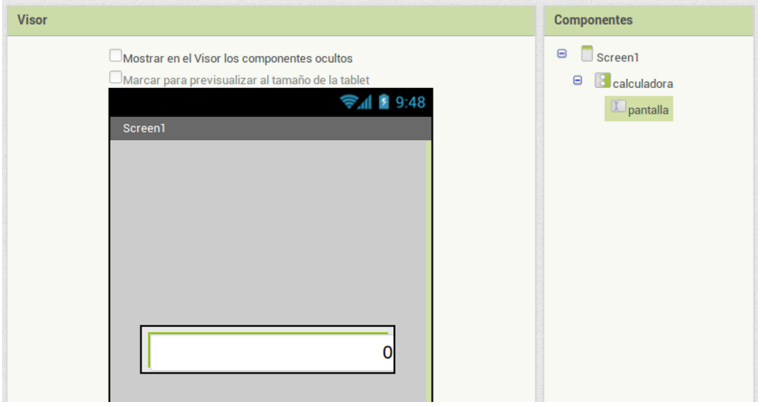
Para la pantalla de la calculadora, a continuación insertamos un componente «CampoDeTexto» del menú «Paleta/Disposición» dentro de la **calculadora**, con las siguientes propiedades:

Propiedades

Nombre	Tamaño de letra	Ancho
pantalla	20	Ajustar al contenedor
PosiciónDelTexto	**Pista**	Texto
derecha	0	0

Obteniendo así el resultado reflejado en la figura 20.

Figura 20. Aspecto del modo «Diseño» tras implementar el componente **pantalla**



P7. El **teclado** se divide horizontalmente en dos zonas independientes de igual anchura: **numérico** y **operaciones**.

P8. El teclado **numérico** y el teclado de **operaciones** se dividen en cuatro niveles verticales.

Para **P7**, hay que hacer dos pasos:

1) Insertar un componente «DisposiciónHorizontal», al que llamaremos **teclado**, de manera que quede justo debajo de **pantalla** pero dentro de **calculadora**, con las siguientes propiedades:

Propiedades

Nombre	DispHorizontal	Ancho
teclado	centro	Ajustar al contenedor

2) Insertar, dentro del recién creado espacio horizontal **teclado**, dos componentes «DisposiciónVertical» adyacentes, a los que llamaremos, respectivamente, **numérico** y **operaciones**, con las siguientes propiedades:

Propiedades

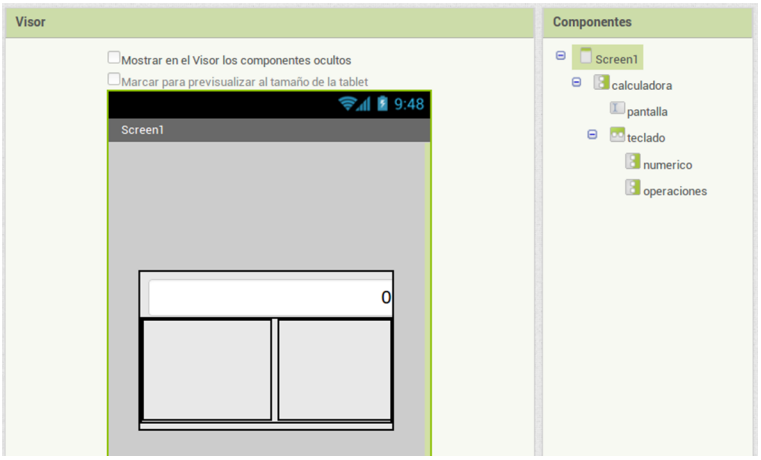
Nombre	DispHorizontal	Ancho
numérico	centro	40 %

Propiedades

Nombre	DispHorizontal	Ancho
operaciones	centro	40 %

Tras esto, las ventanas «**Visor**» y «**Componentes**» del modo de diseño deben tener el siguiente aspecto:

Figura 21. Aspecto del modo «Diseño» tras implementar los componentes de **teclado numérico y operaciones**



Cabe destacar que los dos componentes **numérico** y **operaciones** son del tipo «DisposiciónVertical», porque tendrán que alojar los cuatro niveles en los que se van a disponer las teclas de la calculadora, tal y como está previsto en la figura 16. El ancho de 40 % para cada una de ellas se explica para repartir en dos zonas iguales el 80 % de ancho de pantalla que debe tener la calculadora diseñada en la figura 21.

P9. El teclado **numérico** tiene tres niveles con tres botones iguales, y el cuarto nivel tiene un único botón (0) que ocupa todo el ancho.

En cada uno de los tres primeros niveles de **numérico**, hay que disponer horizontalmente 3 botones consecutivos de la misma medida. Parece evidente que habrá que insertar dentro del espacio **numérico** tres componentes del tipo «DisposiciónHorizontal» para alojar, respectivamente los botones (1, 2, 3); (4, 5, 6) y (7, 8, 9), con las siguientes propiedades:

Propiedades

Nombre	DispHorizontal	Ancho
B123	centro	Ajustar al contenedor

Propiedades

Nombre	DispHorizontal	Ancho
B456	centro	Ajustar al contenedor

Propiedades

Nombre	DispHorizontal	Ancho
B789	centro	Ajustar al contenedor

A continuación, insertaremos tres botones en cada uno de los espacios anteriores, cada uno para el dígito # correspondiente del 1 al 9, en el orden estipulado en la figura 16, y todos ellos con las mismas propiedades:

Propiedades

Nombre	PosiciónDelTexto	Ancho	Texto
B#	centro	Ajustar al contenedor	#

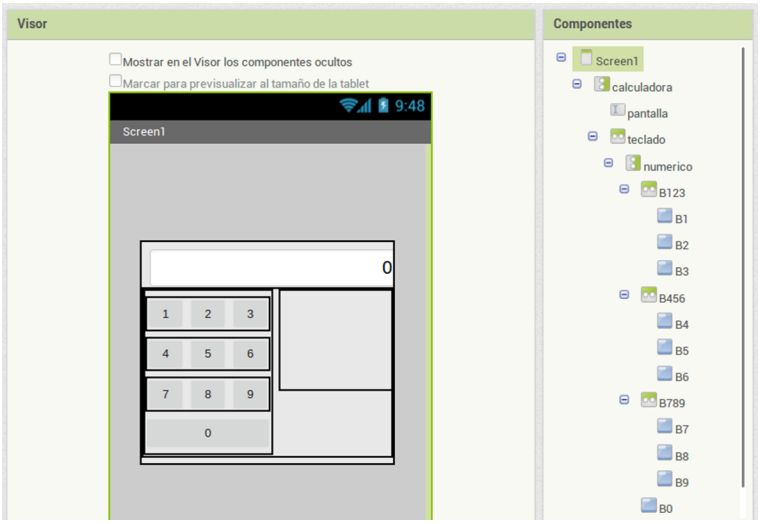
Para acabar, solo falta en el último nivel una única tecla con el dígito 0, que ocupe todo el ancho del espacio:

Propiedades

Nombre	PosiciónDelTexto	Ancho	Texto
B0	centro	Ajustar al contenedor	0

La figura 22 ilustra el aspecto del modo «Diseño» tras incluir todos los componentes del espacio **teclado**.

Figura 22. Aspecto del modo «Diseño» tras incluir todos los componentes del espacio **teclado**



P10. El primer nivel del teclado **operaciones** lo comparten dos teclas iguales, y los tres niveles restantes contienen un único botón, que ocupa todo el ancho.

Finalmente, para **P10** parece bastante claro que hay que combinar un componente del tipo «DisposiciónHorizontal» y dos botones «Suprimir/Borrrar» en el primer nivel, con sendos botones para la suma, resta e igual en los siguientes niveles.

Propiedades para «DisposiciónHorizontal»:

Propiedades

Nombre	DispHorizontal	Ancho
SUPBOR	centro	Ajustar al contenedor

Propiedades para los botones «Suprimir/Borrar»:

Propiedades

Nombre	Tamaño de letra	Ancho
Suprimir/Borrar	9	Ajustar al contenedor
PosiciónDelTexto	**Texto**	
centro	Suprimir/Borrar	

Propiedades para los botones de suma, resta e igual:

Propiedades

Nombre	PosiciónDelTexto	Ancho	Texto
BSUM	centro	Ajustar al contenedor	+

Propiedades

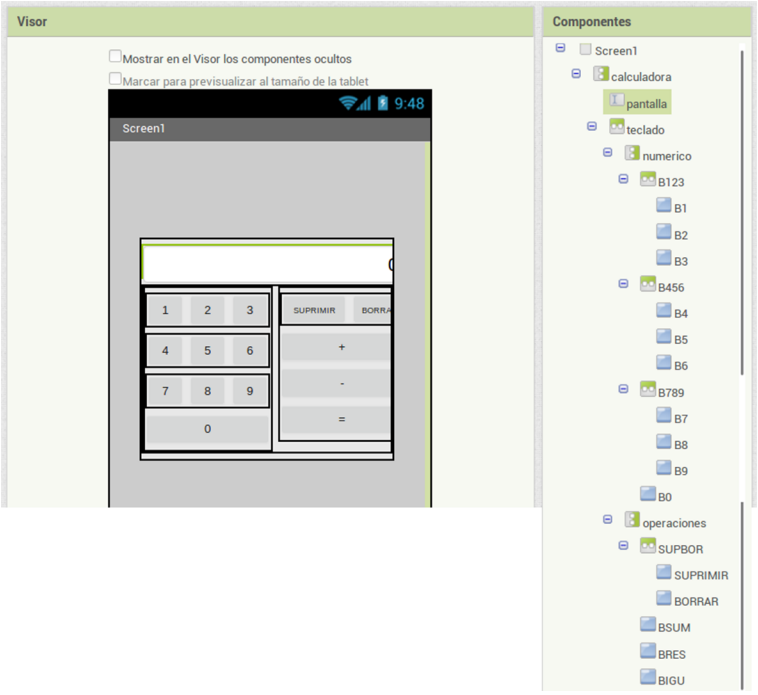
Nombre	PosiciónDelTexto	Ancho	Texto
BRES	centro	Ajustar al contenedor	-

Propiedades

Nombre	PosiciónDelTexto	Ancho	Texto
BIGU	centro	Ajustar al contenedor	=

La figura 23 refleja el resultado final del proceso en las ventanas «**Visor**» y «**Propiedades**» del modo «Diseño».

Figura 23. Aspecto final del modo «Diseño» correspondiente a la disposición de la figura 16



Probad la aplicación en vuestro móvil o tableta. ¿Tiene el aspecto previsto? ¿Qué parte no es exactamente como esperabais? ¿Cómo podríais solucionarlo? ¿Qué partes os ha costado más entender?

De momento, de la calculadora solo se ha hecho el diseño y no está operativa. Próximamente, pasaremos a la programación de los bloques. ¿Cómo imagináis que deberá ser esta programación?

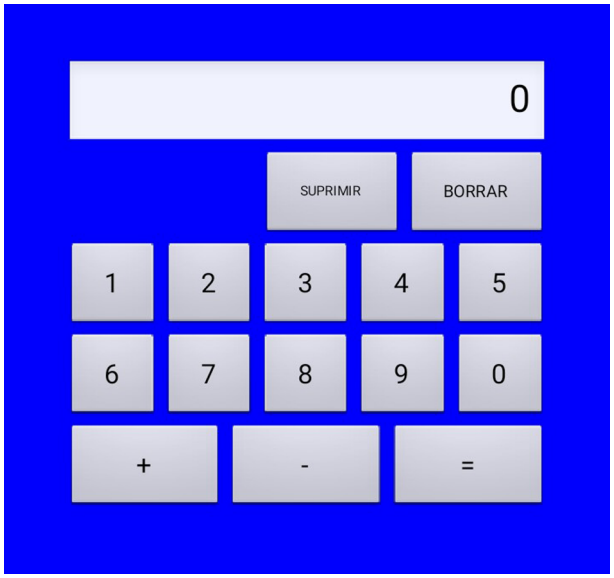
4.5. Actividades

Las actividades que hay que entregar en este módulo son:

1) Enviar los archivos «ProyectoCalculadora.aia» *y* «ProyectoCalculadora.apk», resultado final de implementar el diseño de la figura 16. No es necesaria la programación de bloques.

2) Enviar los archivos «ProyectoCalculadora1.aia» y «ProyectoCalculadora1.apk», resultado de implementar el diseño indicado en la figura 24. No es necesaria la programación de bloques.

Figura 24. Diseño previo de distribución de componentes para la actividad número 2, «ProyectoCalculadora1»



5. Procedimientos y funciones

5.1. Introducción

Una vez consolidadas las competencias referentes al lenguaje computacional y a las estructuras algorítmicas básicas, ya estamos preparados para conocer nuevas técnicas avanzadas de programación, como el uso de procedimientos y funciones.

La programación procedimental o funcional consiste en simplificar el programa en unas pocas expresiones preparadas para ser invocadas o llamadas tantas veces como sea necesario, lo que produce un resultado correcto para cualquier escenario posible.

Cada caso o escenario viene descrito por una serie de variables y/o constantes previamente definidas y conocidas por el procedimiento, como si de una función matemática se tratase.

Esta técnica de programación ofrece muy buenas prestaciones en términos de velocidad de ejecución y tamaño de los programas, con el consiguiente ahorro de tiempo de ejecución y programación.

5.2. Resumen

Dotaremos de funcionalidad a la calculadora, para la que ya hemos diseñado previamente una interfaz de usuario.

Optimizaremos las acciones asociadas a teclas con función similar mediante la programación de un solo procedimiento para todas ellas.

Aprenderemos cómo invocar estos procedimientos y a establecer los parámetros mínimos necesarios para su correcto funcionamiento.

5.3. Recordatorio de «ProyectoCalculadora»

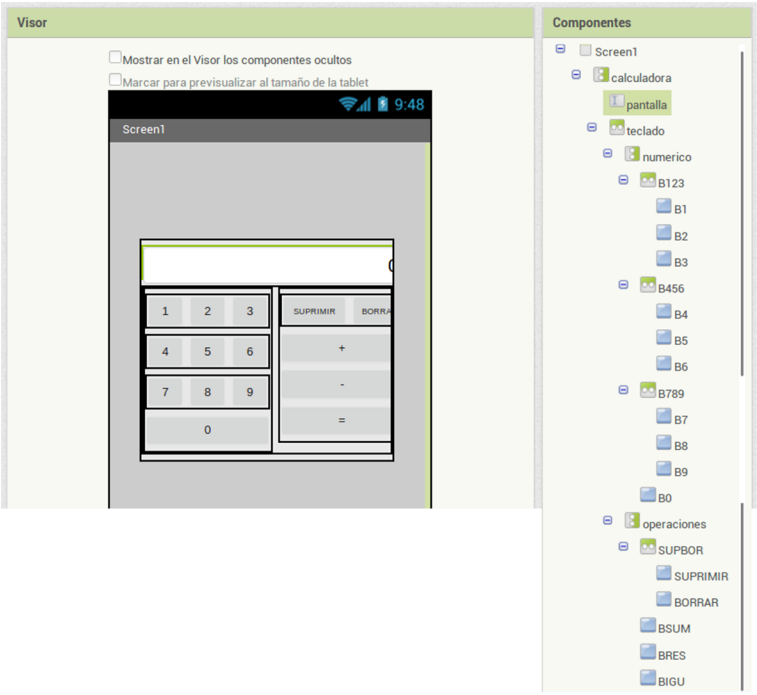
Vamos a recuperar la aplicación «ProyectoCalculadora», llevada a cabo en el módulo anterior. Recordemos que se limitaba a ser un ejercicio de diseño y distribución de componentes, cuyo resultado se aprecia en la figura 25. En este módulo, vamos a añadir funcionalidad a las teclas. Para ello, una vez dentro de nuestra sesión en MIT App Inventor, vamos a crear una copia de esta aplicación, que llamaremos «ProyectoNuevaCalculadora»:

Proyectos -> Guardar proyecto como... -> Nuevo nombre: ProyectoNuevaCalculadora

Figura 25. Aspecto final del ProyectoCalculadora, aún sin funcionalidad.

A modo de recordatorio, en la figura 26 podemos ver la lista de todos los componentes utilizados, con sus respectivos nombres.

Figura 26. Distribución de componentes para «ProyectoNuevaCalculadora», con sus respectivos nombres.



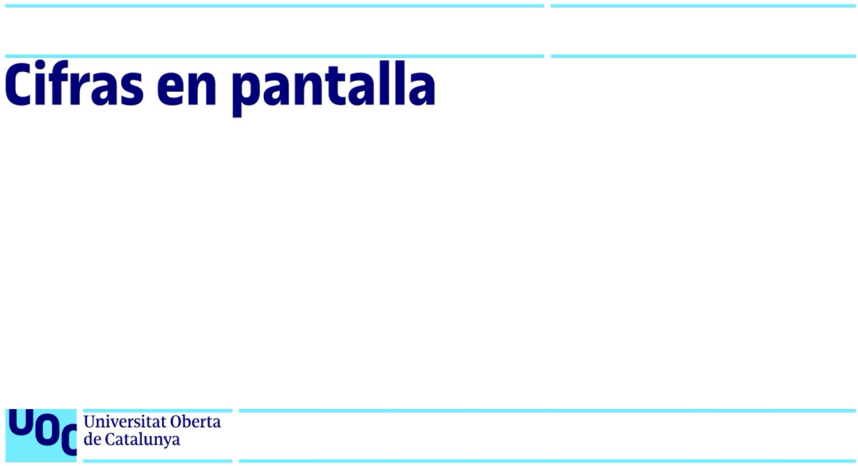
5.4. Cifras en pantalla

El primer paso para conseguir que la nueva calculadora funcione consiste en hacer aparecer en pantalla el dígito correspondiente al botón o tecla pulsada, tal y como se muestra en la figura 27.

Figura 27. Evento para escribir en la pantalla de la calculadora el dígito 1 cuando se pulsa el botón B1

Incluyendo en el entorno «**Bloques**» un evento como el de la figura 27 para cada uno de los 9 botones restantes (B2, B3, B4, B5, ..., B0), podremos escribir cualquiera de los 10 dígitos en pantalla, tal y como se aprecia en este vídeo.

Vídeo 16. Cifras en pantalla



Simulación realizada en el entorno MIT App Inventor

Habréis notado que, tal y como está ahora la aplicación, la calculadora solo nos deja escribir números de un solo dígito. De momento, vamos a trabajar con números menores de 10. Más adelante, conseguiremos que acepte valores mayores.

Para borrar el contenido de la pantalla, combinaremos un bloque de evento del botón «Borrar» con un bloque de control de texto de **pantalla**, como se muestra en la figura 28.

Figura 28. Evento para borrar el contenido de la pantalla

5.5. Suma de enteros de un dígito

Llegados a este punto, vamos a decidir que para que la aplicación sume dos números enteros, habrá que proceder como en este vídeo, es decir:

Vídeo 17. Suma de enteros de un dígito

Simulación realizada en el entorno MIT App Inventor

1) Introducimos el primer número pulsando el botón correspondiente. El número pulsado aparece en pantalla.

2) Pulsamos el botón de la operación suma (+). El número permanece en pantalla.

3) Introducimos el segundo número, pulsando el botón correspondiente. El nuevo número pulsado aparece en pantalla.

4) Pulsamos el botón para obtener el resultado (=). El resultado de la operación aparece en pantalla.

Es importante destacar que el evento para obtener el resultado (cuando «**BIGU**.Clic») se produce varias acciones después de haber establecido el primer sumando. Este hecho obliga a memorizar el valor de este primer sumando, en una variable a la que llamaremos ***término***.

Teniendo en cuenta lo anterior, podríamos proponer la siguiente **lista de acciones**:

1. Aplicación: crea la variable **término**.
2. Usuario: pulsa uno de los diez botones (**B1**,**B2**, **B3**, **B4**, **B5**,..., **B0**)para elegir el primer término de la suma.
3. Aplicación: escribe el dígito anterior en «CampoDeTexto **pantalla**».
4. Usuario: pulsa el botón **BSUM**(+)correspondiente a la operación suma.
5. Aplicación: guarda el valor visible en **pantalla** dentro de la variable **término**.
6. Usuario: pulsa uno de los diez botones (**B1**,**B2**, **B3**, **B4**, **B5**,..., **B0**)para elegir el segundo término de la suma.
7. Aplicación: escribe el dígito anterior en «CampoDeTexto **pantalla**».
8. Usuario: pulsa el botón **BIGU** (=) para obtener el resultado.
9. Aplicación: suma el valor del dígito en **pantalla** con el valor de la variable **término**, y escribe el resultado en **pantalla**.

Las acciones 3 y 7 se cubren con una sola implementación: la ya comentada en la figura 27 para cada uno de los diez dígitos distintos. La acción 3 se implementa en la ventana de **bloques**, según se ilustra en la figura 29.

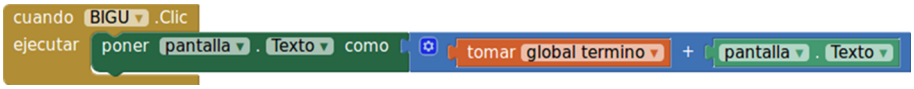
Figura 29. Declaración de la variable **término** (acción 1)

En cambio, el evento que desencadena el botón **BSUM**, correspondiente a la acción 5, es el de la figura 30.

Figura 30. La aplicación guarda el valor visible en pantalla dentro de la variable término (acción 5)

Para completar esta primera fase de la nueva aplicación, solo queda implementar la acción 9 para poder calcular y escribir en pantalla el resultado de la suma. Se desencadena mediante el botón **BIGU**, y la figura 31 propone una posible construcción.

Figura 31. Al pulsar el botón **BIGU** (=), la aplicación suma el valor del dígito en **pantalla** con el valor de la variable **término**, y reescribe el resultado en **pantalla** (acción 9)



Probad ahora la aplicación en vuestro móvil o tableta. ¿Funciona como en el vídeo 17 de ejemplo? ¿Entendéis lo que está haciendo la aplicación? ¿Habéis tenido que corregir algún bloque? ¿Qué ocurre si seguimos sumando números al resultado?

5.6. Suma o resta de enteros de un dígito

Vamos a incorporar a nuestra calculadora la posibilidad de elegir entre la operaciones suma (+) o resta (-), como en este vídeo. Seguimos trabajando aún con números enteros entre 0 y 9. Os recomendamos crear una nueva versión del proyecto, a la que llamaremos «ProyectoNuevaCalculadora01»:

> Proyectos -> Guardar proyecto como… -> Nuevo nombre: ProyectoNuevaCalculadora01

Vídeo 18. Suma o resta de enteros de un dígito

Simulación realizada en el entorno MIT App Inventor

La posibilidad de seleccionar entre dos operaciones no cambia sustancialmente la lista de acciones de la versión anterior. No obstante, al igual que ocurría con la variable **término** para recordar el primer dato del cálculo, habrá que guardar constancia de la operación que el usuario elige en cada caso, mediante una variable llamada **operación**. De esta manera, las acciones de la anterior lista que se verán afectadas son la acción 1, la acción 4 y la acción 9:

1. Aplicación: crea las variables **término** y **operación**.
2. Usuario: pulsa uno de los diez botones (**B1**, **B2**, **B3**, **B4**, **B5**, ..., **B0**) para elegir el primer término de la suma o de la resta.
3. Aplicación: escribe el dígito anterior en «CampoDeTexto **pantalla**».
4. Usuario: pulsa el botón **BSUM**(+)o el botón **BRES**(-)de la operación escogida.
5. Aplicación: si pulsa el botón **BSUM** (+)
   guarda el valor "+" en la variable **operación**
   si pulsa el botón **BRES** (-)
   guarda el valor "-" en la variable **operación**

   guarda el valor visible en **pantalla** dentro de la variable **término**.
6. Usuario: pulsa uno de los diez botones (**B1**, **B2**, **B3**, **B4**, **B5**, ..., **B0**) para elegir el segundo término de la suma o de la resta.
7. Aplicación: escribe el dígito anterior en «CampoDeTexto **pantalla**».
8. Usuario: pulsa el botón **BIGU** (=) para obtener el resultado.
9. Aplicación: si el valor de la variable **operación** es "+"
   suma el valor de la variable **termino** con el valor del dígito en **pantalla**
   si el valor de la variable **operación** es "-"
   resta el valor de la variable **término** del dígito en **pantalla**

   escribe el resultado en **pantalla**.

A partir de estas consideraciones, vemos que es necesario desdoblar los bloques de las figuras 29, 30 y 31; tal y como se muestra en las figuras 32, 33 y 34, respectivamente.

Figura 32. Declaración de las variables **término** y **operación** (acción 1)

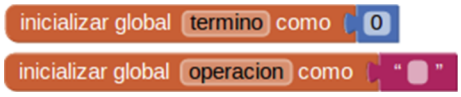


Figura 33. La aplicación guarda el valor visible en **pantalla** dentro de la variable **término**, y el valor correspondiente en la variable **operación** (acción 5)

Figura 34. Al pulsar el botón **BIGU** (=), la aplicación suma o resta los valores del dígito en **pantalla** y de la variable **término**, y reescribe el resultado en **pantalla** (acción 9).

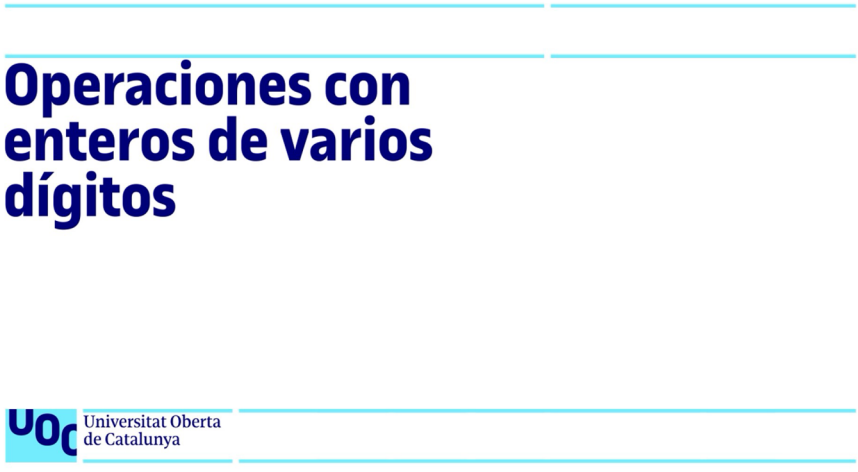
Ya podéis probar la aplicación en vuestro móvil o tableta. ¿Funciona como estaba previsto? ¿Entendéis por qué se programa así la aplicación? ¿Qué dificultades han ido surgiendo en el proceso de programación?

5.7. Operaciones con enteros de varios dígitos

Vamos a completar el proyecto de la calculadora consiguiendo que opere con números enteros de varios dígitos, como en el vídeo. Os recomendamos crear una nueva versión del proyecto, a la que llamaremos «ProyectoNuevaCalculadora02»:

> Proyectos -> Guardar proyecto como… -> Nuevo nombre: ProyectoNuevaCalculadora02

Vídeo 19. Operaciones con enteros de varios dígitos



Simulación realizada en el entorno MIT App Inventor

En esencia, la nueva versión se diferencia de la anterior únicamente en la posibilidad de escribir enteros de más de una cifra. El resto del procedimiento, de las acciones y, por tanto, de los bloques no va a cambiar.

Para entender cómo debemos programar ahora la introducción de los dígitos, analicemos un caso concreto. Imaginemos que queremos escribir el número 625. En una calculadora convencional, basta con pulsar de manera sucesiva las teclas 6, 2 y 5. Podemos resumir el proceso en la tabla 16.

Tabla 16. Proceso de escritura del número 625 en la pantalla de la calculadora

Tecla pulsada	Pantalla
	0
6	6
2	62
5	625

Lo primero que debe llamar nuestra atención es que:

6 = 0 + 6

62 = 60 + 2

625 = 620 + 5

Es decir,

6 = (0 x 10) + 6

62 = (6 x 10) + 2

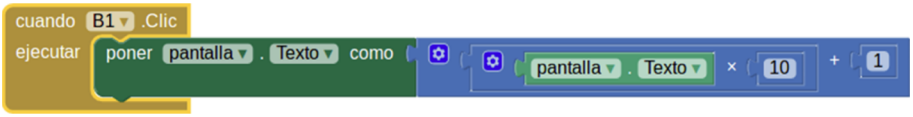
625 = (62 x 10) + 5

A la vista de esto, podemos deducir que la fórmula o función matemática que nos va a permitir calcular el nuevo valor que debe aparecer en pantalla tras pulsar una de las teclas de la calculadora podría ser:

Nuevo valor en pantalla = (actual valor en pantalla × 10) + valor de la tecla pulsada

Por tanto, siguiendo el método habitual, el bloque de programación asociado al evento de pulsar la tecla 1 (botón **B1**), por ejemplo, sería ahora el que se recoge en la figura 35, y habría que crear un bloque similar para cada una de las distintas teclas del 0 al 9.

Figura 35. Al pulsar el botón **B1** (1), en cualquier momento, la aplicación recoge el valor actual en **pantalla**, lo multiplica por 10, suma 1 y muestra el nuevo valor así calculado en **pantalla**, sustituyendo el existente



Sin embargo, hay una manera más ágil y sencilla de programar: mediante la utilización de **procedimientos**. Si programamos un solo bloque con la función definida, y la invocamos cada vez que pulsamos una tecla con su valor correspondiente, obtenemos un programa más optimizado en términos de complejidad, rapidez de programación y consumo de recursos del sistema. En el caso que nos ocupa puede parecer poco, pero es una buena práctica cuando nos enfrentemos a programas más grandes y complejos, en los que el tiempo de cálculo y la memoria ocupada son factores críticos.

Un **procedimiento** es un programa dentro de un programa mayor, que ejecuta una acción determinada cada vez que es invocado por su nombre, previamente definido. Un procedimiento puede recibir datos (llamados *argumentos* o *parámetros*) en el momento de ser invocado. El valor de estos argumentos es tenido en cuenta e influye en el resultado del procedimiento. Una **función** es un caso particular de procedimiento que, como efecto final de su ejecución, devuelve un único resultado en forma de dato.

Para crear un procedimiento en MIT App Inventor, hay que ir directamente a la pantalla «**Bloques**» y, en el menú lateral, localizar en «**Procedimientos**» el bloque «**Como procedimiento ejecutar**». Para invocar el procedimiento en cada tecla numérica de la calculadora, el bloque es «**Llamar procedimiento x**», donde x es el argumento que se envía al procedimiento. En nuestro caso, x es el valor numérico del botón pulsado, tal y como se indica en la figura 36, para los botones **B1**, **B2** y **B3**. El procedimiento, al ser invocado y recibir el argumento **x**, actualiza el entero en pantalla, según la programación que se muestra en la figura 37, bloque que solo hay que componer una vez.

La figura 38 muestra cómo obtener el bloque «**Tomar x**».

Figura 36. Cambio en la programación del evento clic, para invocar **procedimiento** de la figura 37, en la escritura de un entero de varios dígitos en pantalla

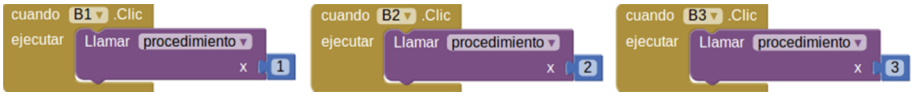
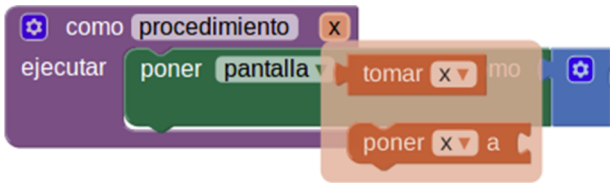


Figura 37. Estructura de **procedimiento**, empleando el bloque de la figura 35 una sola vez para cualquier tecla pulsada. El valor de la tecla es recibido a través del parámetro **x**

Figura 38. Para obtener el bloque «**Tomar x**», hay que situar el cursor durante dos segundos sobre la x de la cabecera del bloque «**Procedimiento**»



Por último, es necesario solucionar un pequeño detalle: poner a cero el «CampoDeTexto **pantalla**» cada vez que acabemos de escribir el primer término de la operación, es decir, cada vez que pulsemos «+» o «-». En caso contrario, el segundo término se acumularía al anterior. Hay que añadir el bloque «poner **pantalla. Texto** como 0» a continuación de los dos bloques de los eventos **BSUM** y **BRES** de la figura 33.

Con todo lo anterior, intentad programar el funcionamiento de las diez teclas numéricas de la calculadora empleando el procedimiento propuesto en la figura 37.

¿Funciona como esperabais? ¿Con qué dificultades os habéis enfrentado? ¿Cómo las habéis solucionado? ¿Podéis mejorar el funcionamiento de la calculadora? ¿Cómo lo conseguiréis?

5.8. Actividades

Las actividades que hay que entregar en este módulo son:

1) Enviar los archivos «ProyectoNuevaCalculadora.aia» y «ProyectoNuevaCalculadora.apk», correspondientes a la aplicación que permite hacer **sumas** de enteros de **un solo dígito**.

2) Enviar los archivos «ProyectoNuevaCalculadora01.aia» *y* «ProyectoNuevaCalculadora01.apk», correspondientes a la aplicación que permite hacer **sumas y restas** de enteros de **un solo** dígito.

3) Enviar los archivos «ProyectoNuevaCalculadora02.aia» *y* «ProyectoNuevaCalculadora02.apk», correspondientes a la aplicación que permite hacer **sumas y restas** de enteros de **varios dígitos**.

4) Enviar los archivos «ProyectoNuevaCalculadora03.aia» *y* «ProyectoNuevaCalculadora023apk», correspondientes a una nueva aplicación que permita hacer **multiplicaciones**, **sumas y restas** de enteros de **varios dígitos**.

6. Listas y bases de datos

6.1. Introducción

En el actual paradigma de la industria 3.0, con la irrupción de conceptos como *ciberseguridad*, *Big Data* o *internet de las cosas*, cada vez se cumple más aquello de que la información es poder.

Como su nombre indica, la informática es la tecnología que tiene como objetivo la obtención y el tratamiento automático de la información y de los datos. Las estructuras llamadas *bases de datos* nos permiten ordenar, manipular y consultar gran cantidad de información en formato digital. Una base de datos no deja de ser la versión sofisticada de una lista o conjunto de listas interrelacionadas entre sí.

Vamos a zambullirnos en las herramientas que App Inventor nos proporciona para sacar el máximo partido a nuestras propias bases de datos, desde la más simple a la más compleja.

6.2. Resumen

Crearemos listas de datos para aprender su funcionamiento.

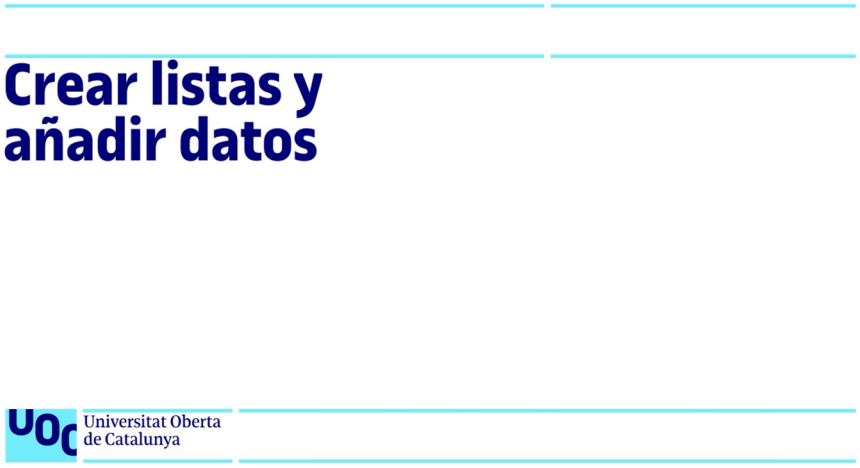
Exploraremos técnicas de búsqueda cruzada entre dos listas interrelacionadas entre sí.

Manipularemos las listas creadas para añadir, borrar y modificar datos.

6.3. Crear listas y añadir datos

Para la primera actividad de este módulo, vamos a crear una aplicación que nos ayude a hacer la lista de la compra. Tendrá por nombre «ProyectoListaCompra». Al abrir la aplicación, se nos propondrá que elijamos de entre un catálogo general de productos aquellos que queramos comprar, y los irá añadiendo a nuestra lista de la compra, tal y como se puede en este vídeo.

Vídeo 20. Crear listas y añadir datos



Simulación realizada en el entorno MIT App Inventor

De hecho, vamos a necesitar dos listas. A una la vamos a llamar «**catálogo**», y habrá que crearla con todos los productos disponibles (que vamos a limitar a 6, para abreviar). La otra, inicialmente vacía, llevará por nombre «**compra**».

Antes de empezar, vamos a imaginar el aspecto que nuestra aplicación deberá tener en pantalla. Proponemos un componente llamado «**SelectorDeLista**» para la lista «**catálogo**», y un «**VisordeLista**» para la lista «**compra**», de modo que, al seleccionar un producto del **catálogo**, se incorpore a la lista «**compra**».

Ya estamos en condiciones de planificar los componentes y las acciones necesarias:

1) Lista de componentes

a) Un «**SelectorDeLista**» (Paleta / Interfaz de usuario) para el catálogo.

Propiedades

Nombre	Ancho	Texto
catálogo	50 %	Catálogo

b) Un «**VisordeLista**» (Paleta / Interfaz de usuario) para la lista de la compra.

Propiedades

Nombre	Alto	TextSize
compra	200 píxeles	44

2) Lista de acciones

1. Aplicación: crear lista «**catálogo**», inicialmente vacía.
2. Aplicación: crear lista «**compra**», inicialmente vacía.
3. Aplicación: llenar la lista «**catálogo**» con los productos (pan, leche, huevos, azúcar, aceite, sal).
4. Usuario: seleccionar un producto de la lista «**catálogo**».
5. Aplicación: añadir el producto seleccionado a «**compra**».

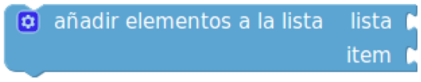
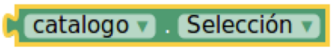
Ya en modo «**Bloques**», el sistema trata una lista como si de una variable se tratara, así que habrá que combinar los bloques «**Inicializar global**» (menú «Variables») con «**Crear una lista vacía**» (menú «Listas») (figura 39).

Figura 39. Creación de las listas «**compra**» y «**catálogo**»

Hay que llenar la lista «**catálogo**» tan pronto como arranque la aplicación, así que combinaremos un bloque de evento «**cuando Screen1.Inicializar**» (menú «Control») con «**añadir elementos a la lista**» (menú «Listas»). Una vez hecho esto, asignamos la lista al componente «catálogo» con «**poner catálogo.Elementos como**» (menú «Catálogo») más «**tomar global catálogo**» (menú «Variables») (figura 40).

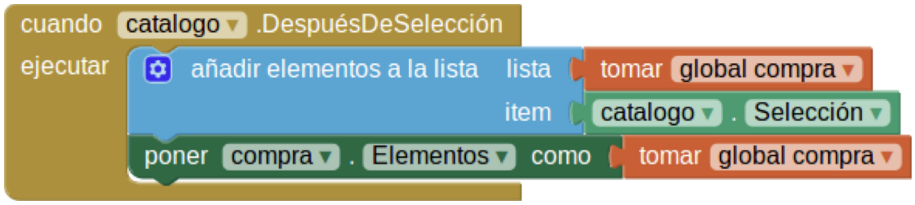
Figura 40. Rellenado de la lista «catálogo» con los productos y asignación a su componente

En la acción 4, se detecta la interacción del usuario con el bloque evento «**cuando catálogo.DespuésDeSelección**» (menú «Catálogo»). En la acción 5, hay que añadir el producto seleccionado a la lista «**Compra**», para lo cual emplearemos los siguientes bloques, combinándolos tal y como se muestra en la figura 41.

Acción «**añadir elementos a la lista**» (menú «Listas»)	
Lista a la que va a ser añadido, en «Lista» (menú «Variables»)	
Producto seleccionado del catálogo, en «Ítem» (menú «Catálogo»)	

Para acabar, hay que actualizar el «VisordeLista **compra**» con los actuales elementos de la «lista **Compra**».

Figura 42. Bloque para añadir productos de «**catálogo**» a la lista «**compra**»



Probad la aplicación en vuestro móvil o tableta. ¿Funciona como esperabais? ¿Qué aspectos son susceptibles de mejora? ¿Cómo lo lograríais? ¿Qué partes os han costado más?

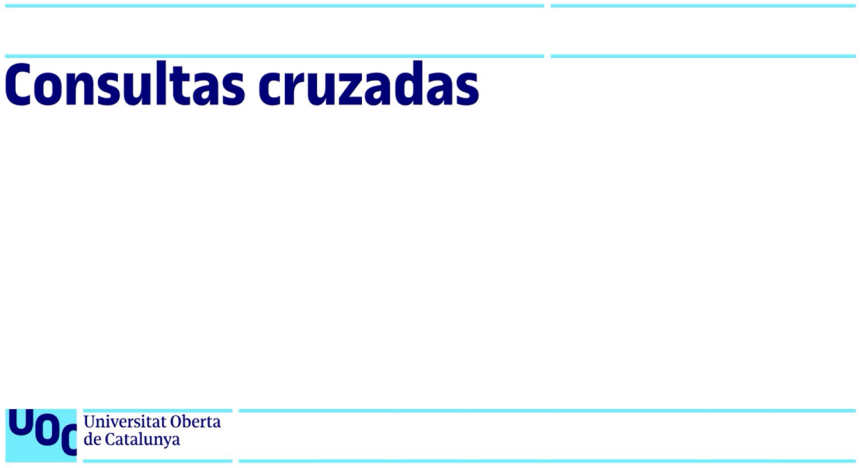
6.4. Consultas cruzadas

Vamos a dar un paso más allá, e incorporaremos nuevas funcionalidades a nuestra aplicación, de manera que se va a pasar a llamar «ProyectoPrecioCompra». Antes de seguir, procedemos a hacer una copia de seguridad de la aplicación anterior:

> Proyectos -> Guardar proyecto como... -> Nuevo nombre: ProyectoPrecioCompra

El objetivo será que, al mismo tiempo que confeccionamos nuestra lista, la aplicación nos vaya calculando el importe total que nos costará la compra. Por supuesto, necesitaremos una lista adicional con los precios de cada uno de los productos del catálogo, entre otros. El resultado final se puede ver en este vídeo.

Vídeo 21. Consultas cruzadas



Simulación realizada en el entorno MIT App Inventor

1) Lista de componentes (adicionales)

a) Una **etiqueta** (Paleta / Interfaz de usuario) para poder visualizar el importe en pantalla.

Propiedades

Nombre	Texto
importe	Importe total de la compra:

2) Lista de acciones que hay que añadir a las que ya teníamos

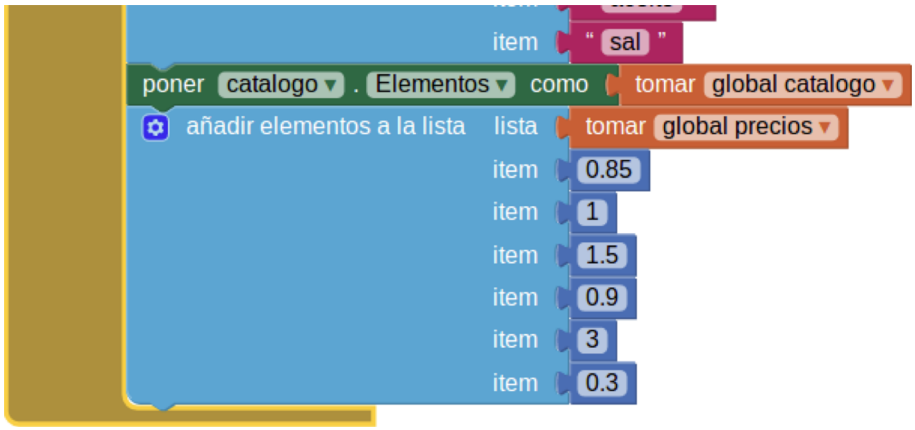
```
    ...
1bis. Aplicación: crear una variable importe, inicialmente igual a
                  cero.
    ...
2bis. Aplicación: crear lista «precios», inicialmente vacía.
    ...
3bis. Aplicación:  llenar la lista «precios» con los respectivos precios
                   de los productos (pan, leche, huevos, azúcar, aceite,
                   sal).
    ...
4bis. Aplicación: buscar el precio del producto seleccionado en la lista
                  «precios»
                     sumar a la variable importe el valor del precio
                     encontrado
                     actualizar la etiqueta importe con el nuevo valor
                     de la variable importe.
```

Así pues, para las acciones 1 bis y 2 bis, procedemos igual que en la acción 1, añadiendo dos grupos de bloques más (figura 43).

Figura 43. Declaración de la variable **importe** y creación de la lista «**precios**»

Para la acción 3 bis, procedemos de manera similar a la acción 3, pero teniendo en cuenta que esta vez se trata de una lista de números y no de cadenas de texto. Hay que escribir la separación decimal con un punto (.) y no con una coma (,), siguiendo la notación anglosajona. Por último, debemos tener la precaución de introducir los precios en el mismo orden que siguen en la lista «**Catálogo**» los productos a los que corresponden. El bloque así construido se debe incrustar dentro del bloque de evento «**cuando Screen1.Inicializar**», y a continuación de los bloques ya existentes de la aplicación anterior (figura 44).

Figura 44. Rellenado de la lista «**Precios**» con los precios correspondientes a los productos de la lista «**Catálogo**»



Para hacer una consulta cruzada entre dos listas ordenadas, hay que averiguar el **índice** de un elemento en la tabla. El índice señala el orden que ocupa este elemento en la lista (suponiendo que no esté repetido). En nuestro caso, hemos preparado las listas para que los índices de un producto en la lista «**catálogo**» coincidan con el índice de su precio en la lista «**precios**». Una vez averiguado el primero, daremos directamente con el segundo.

La consulta, pues, tiene dos fases. Para averiguar el **índice** del producto seleccionado en la lista «**catálogo**», empleamos el bloque «**catálogo.ÍndiceSeleccionado**» (menú «catálogo»). Una vez que tenemos el índice, lo insertamos en el bloque «**seleccionar elemento de la lista**» (menú «Listas»), junto con la lista «**tomar global precios**» (menú «Variables»). En la figura 45, vemos cómo se obtiene el precio del producto seleccionado.

Figura 45. Obtención del producto seleccionado en la lista «**catálogo**» a partir de su índice y su correspondencia cruzada en la lista «precios»

Ahora habrá que actualizar el valor de la variable del **importe** total de la compra, sumando el precio del producto acabado de obtener con el importe acumulado, si lo hubiera, combinando un bloque «**poner global importe a**» (me-

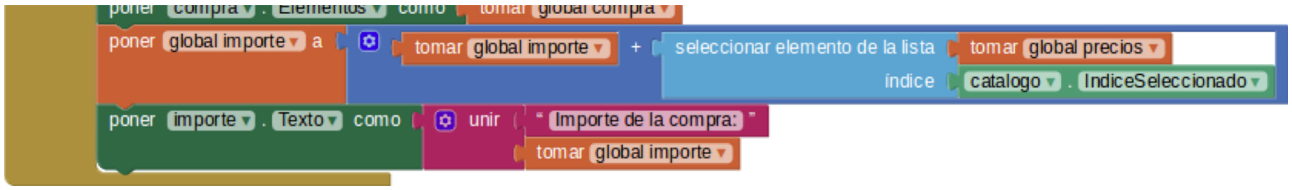
nú «Variables»), un bloque «**tomar global importe**» (menú «Variables»), una bloque «suma» (menú «Matemáticas») y el bloque de la figura 45, tal y como se muestra en la figura 46.

Figura 46. Actualización de la variable «**Importe**» de la compra tras selección de un nuevo producto

Para terminar, hay que poner en pantalla el valor del importe a través de la etiqueta «**importe**» con un bloque «**poner importe.Texto como**» (menú «Importe»). Todo el conjunto así obtenido hay que incrustarlo dentro del bloque de evento «**cuando catálogo.DespuésDeSelección**», ya creado en la anterior versión de la aplicación. El resultado final se muestra en la figura 47.

Probad la aplicación en vuestro móvil o tableta. ¿Entendéis el funcionamiento? ¿Qué dificultades habéis encontrado? ¿Qué mejoraríais de la interfaz de usuario?

Figura 47. Actualización de la variable «**Importe**» y de la etiqueta «**Importe**» tras la selección de un nuevo producto con «**cuando catálogo.DespuésDeSelección**»



6.5. Actividades

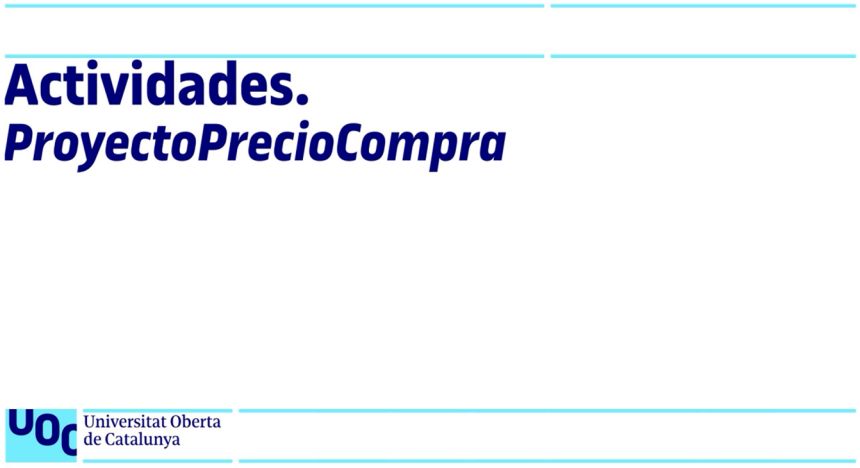
Las actividades que hay que entregar en este módulo son:

1) Enviar los archivos «ProyectoListaCompra.aia» y «ProyectoListaCompra.apk», del apartado 3.

2) Enviar los archivos «ProyectoPrecioCompra.aia» y «ProyectoPrecioCompra.apk», resultado del apartado 4.

3) Además, os pediremos que, a partir de «ProyectoPrecioCompra», desarrolléis una nueva aplicación, a la que vamos a llamar «ProyectoPrecioCompraFinal», que permita no solo añadir productos a la lista de la compra, sino también borrarlos, a la vez que va actualizando consecuentemente el importe total de la misma. En este vídeo podéis ver un ejemplo de funcionamiento.

Vídeo 22. Actividades. *ProyectoPrecioCompra*



Simulación realizada en el entorno MIT App Inventor

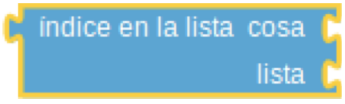
Os daremos algunas pistas para que podáis empezar por vuestra cuenta:

a) No olvidéis planificar las nuevas acciones que tendrá vuestra aplicación. Revisad la lista de acciones que os hemos propuesto durante el módulo, e incorporad las nuevas.

b) Deberéis añadir un módulo muy similar al bloque ya existente «**catálogo.DespuésDeSelección**» que ya habéis hecho pero, esta vez, para cuando seleccionemos elementos de la lista «**compra**».

c) Recordad que para buscar un precio, tenéis que averiguar el índice que el producto que queréis eliminar tiene en la lista «**catálogo**». Os será muy útil el módulo «**índice en la lista**» (menú «Listas») de la figura 48, donde la **cosa** es el producto seleccionado para borrar, y la **lista** es la del **catálogo**.

Figura 48. Bloque «**índice en la lista**». Menú «Listas»



d) Finalmente, será necesario actualizar la lista «**Compra**». Os sugerimos que insertéis un bloque «**ponercompra.Elementos como (compra.Elementos)**» tras el bloque «**eliminar elemento de la lista**».

Como evidencias de esta tercera actividad, tendréis que enviar los archivos «ProyectoPrecioCompraFinal.aia» y «ProyectoPrecioCompraFinal.apk», además de un documento de texto «ListaAccionesCompraFinal» (DOC, ODF, TXT, PDF, etc.), con la lista completa de las acciones en las que habéis descompuesto la aplicación.

7. Proyecto final

7.1. Introducción

En este último apartado, desplegaremos las competencias adquiridas durante el curso. No se tratará de una simple prueba de contenidos, sino que también entrarán en juego la creatividad para imaginar soluciones, la habilidad para documentarse y buscar información, la perseverancia para ir superando las dificultades a medida que vayan surgiendo y la capacidad para identificar, de entre las múltiples estrategias de programación, la más adecuada en cada caso.

Se trata de un proyecto totalmente abierto en el que habrá que respetar, eso sí, unos requerimientos mínimos comunes a partir de los cuales podemos dar rienda suelta a nuestra creatividad y nuestro talento.

Tal y como ya va siendo costumbre durante el curso, se exigirá una fase previa de diseño de la aplicación, en la que la reflexión y planificación del proyecto tienen que dar lugar a una exhaustiva lista de componentes y acciones lo más detallada posible.

Será totalmente legítimo buscar inspiración en la infinidad de recursos que sobre App Inventor se puede encontrar hoy día en la Red. De hecho, el mérito reside en tener el criterio suficiente como para saber encontrar, elegir y adaptar aquellas soluciones que se adecuen mejor a cada caso particular.

También será necesario generar una documentación que describa la aplicación y su modo de uso, así como propuestas de mejora para sucesivas versiones.

En todo el proceso, contaremos con el apoyo, el consejo y la orientación del tutor del curso.

7.2. Resumen

En este apartado, programaremos una aplicación para dispositivos móviles Android utilizando la plataforma App Inventor. El aspecto y la funcionalidad son de libre elección, pero están sujetos a una serie de requerimientos que se detallan en el apartado siguiente.

Con anterioridad y posterioridad a la programación de la aplicación, habrá que hacer el correspondiente análisis y documentación.

7.3. Requerimientos

Es necesario que la aplicación:

1) Incorpore un control de identidad previo a la misma, mediante una única combinación usuario + contraseña válida que deberá introducir el usuario. Los valores deberán ser:

usuario: appinventor

contraseña: cursouoc

Una vez superado este control de identidad, la aplicación permitirá el acceso a algún tipo de funcionalidad, juego o utilidad que debe cumplir todos y cada uno de los siguientes requisitos generales:

2) El usuario debe interactuar con la aplicación para influir sobre su funcionamiento y resultado final, más allá de la función marcha/paro (*on*/*off*).

3) Tiene que hacer uso de **al menos uno** de los componentes del menú «Paleta/Disposición».

4) Ha de permitir la interacción del usuario mediante **al menos uno** de los componentes del menú «Paleta / Dibujo y animación».

5) Tiene que hacer uso de **al menos un** archivo imagen (máximo 100 Kb).

6) Debe hacer uso de **al menos un** archivo de sonido (máximo 200 Kb).

7) Debe utilizar **al menos uno** de los componentes del menú «Paleta/Sensores».

8) Tiene que hacer uso de **al menos un** bloque **de cada uno** de los siguientes tipos: «Control, Matemáticas, Texto, Variables».

No es imprescindible que la aplicación:

1) Haga uso de más de una pantalla.

2) Haga uso de bloques del tipo «Lógica, Listas, Colores, Procedimientos».

7.4. Planificación

Durante los anteriores proyectos del módulo, se ha llevado a cabo un proceso de diseño y planificación de la aplicación, previo a la fase de programación propiamente dicha.

De la misma manera, os pediremos que antes de programar dediquéis un tiempo a reflexionar y documentarse para decidir:

- ¿Qué quiero hacer?
- ¿Qué aspecto va a tener?
- ¿Cómo lo voy a hacer?
- ¿Qué necesito para hacerlo?

Recomendamos que apuntéis vuestras ideas y dibujéis los esquemas producto de este proceso de análisis. Os tenéis que volver a hacer las preguntas anteriores varias veces, hasta que intuyáis que se ha dado con una buena solución, sin perder nunca de vista los requerimientos.

Llegado este momento, siguiendo el modelo que hemos venido usando en los módulos anteriores, redactad la **lista de componentes** y la **lista de acciones** de la aplicación. Deben ser lo más exhaustivas, detalladas y descriptivas posible. Estas listas, junto con la aplicación y la documentación, formarán parte de los contenidos que os pediremos como evidencia de vuestro trabajo. Esta es la plantilla del archivo «**planificacion.doc**» para la redacción de las listas anteriores.

7.5. Documentación

Documentar una aplicación es casi más importante que su programación. La documentación es, en buena medida, la referencia que utilizarán los futuros usuarios de la aplicación para decidir si merece la pena descargarla e instalarla en su móvil.

En nuestro caso, la documentación constará de los siguientes apartados:

1) Nombre de la aplicación

2) Descripción

Texto de entre 100 y 150 palabras, en el que se explica en qué consiste la aplicación. De qué tipo es, qué hace, qué necesita para su funcionamiento. Qué consigue el usuario o qué problema se soluciona con su uso. Qué ventajas aporta respecto a aplicaciones similares, si las hubiera.

3) Modo de empleo

Texto de entre 150 y 200 palabras en el que se instruye y orienta a los usuarios sobre la puesta en marcha, el funcionamiento y el uso de la aplicación. Debe incluir consejos para sacarle el máximo partido y advertir sobre posibles problemas o eventualidades y cómo evitarlos. El texto puede venir acompañado de capturas de pantalla para facilitar su comprensión.

4) Conclusiones y propuestas de mejora

Texto de entre 100 y 150 palabras en el que se incluyen valoraciones de tipo personal sobre qué ha representado la realización del proyecto. Hay que destacar los problemas técnicos que han surgido y cómo se han solucionado. También, y a la vista del resultado final, es interesante proponer posibles mejoras de cara a una nueva versión de la aplicación.

5) Recursos web

Lista lo más completa posible de las páginas web consultadas en busca de información y recursos para la elaboración del proyecto.

Esta es la plantilla del archivo «**documentacion.doc**» para la redacción de los apartados anteriores.

7.6. Actividades

Los archivos que hay que entregar para la evaluación del proyecto final son cuatro:

documentacion.doc	Documentación de la aplicación elaborada, según se explica en el apartado 5.
planificacion.doc	Listas de componentes y lista de acciones de la aplicación elaborada, según se explica en el apartado 4.
ProyectoFinal.aia	Fichero .aia de la aplicación efectuada.
ProyectoFinal.apk	Fichero .apk de la aplicación efectuada.