1-10-2015

|  |
| --- |
| PROYECTO FINAL DE CARRERA | Abel GARCIA SANCHEZ |



|  |  |
| --- | --- |
| UNIVERSIDAD SAN JORGE | ANALISIS Y DESARROLLO CON APP INVENTOR |

|  |  |
| --- | --- |
| Análisis App Inventor | |
| 2 de OCTUBRE de 2015 | |
|  |  |
| Índice  1 Propósito del documento  2 Antecedentes (Origen de app inventor, etc)  3 Tutorial de desarrollo  3.1 Instalación y configuración  3.2 Mi primer programa  3.3 Componentes de app inventor  3.4 Desarrollo paso a paso del juego mole  4 Conclusiones  4.1 Ventajas e inconvenientes.  4.2 Integración de App inventor en la enseñanza  Anexos  Enlaces de interés. | |
|  | |

Hoja de control del documento

Control de plantilla

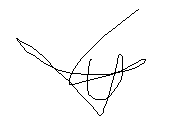
|  |  |  |
| --- | --- | --- |
| Versión | Fecha de publicación | Descripción de los cambios |
| 1.0 | 01/10/2015 | Inicio del documento. |

Declaración del alumno

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido. Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma



Fecha

10 de Enero de 2016

1. Resumen

El presente proyecto, tiene como objetivo el análisis y utilización de la herramienta ***App Inventor*** para el desarrollo de aplicaciones móviles para sistemas operativos Android. App Inventor es una herramienta originalmente desarrollada por Google. Su desarrollo estuvo basado en la informática educativa, huyendo del desarrollo de programas a través de comandos por línea, Google desarrollo este entorno de trabajo basado en un entorno visual en el que se desarrolla.

El proyecto abordará los siguientes puntos:

* Breve reseña histórica sobre el origen de App Inventor, desde sus inicios en Google y su actual mantenimiento en el MIT, además de revisar sus funcionalidades.
* También se incluirá un breve manual, el cual puede usarse de guía para empezar a utilizar la herramienta App Inventor e iniciarse en la programación con esta herramienta..
* Para finalizar, se abordará el desarrollo de un videojuego para móviles con plataforma cuyo sistema operativo sea Android. La metodología que se utilizará para este desarrollo, es la metodología clásica en cascada.

1. Introducción

🡨

Desde hace muchos años estamos viviendo en la llamada ‘edad digital’, y sin duda alguna los dispositivos móviles han logrado desplazar a otros dispositivos para colocarse como un dispositivo indispensable para la mayoría de las personas.

Si echamos la vista atrás, podemos ver aquellos bloques que pesaban medio kilo y que su única función era la de realizar llamadas telefónicas, que en definitiva era para lo que se habían inventado.

Desde esos inicios donde cada operador creaba sus propios menús de navegación para gestionar las funcionalidades del móvil hasta la actualidad, donde sistemas operativos están instalados en los equipos móviles, han pasado 20 años.

Hoy en día aunque la función principal del teléfono móvil sigue siendo la de hablar por teléfono, tiene una infinidad más de funciones incorporadas (realizar fotos, escribir texto, hojas de cálculo, etc), lo que hace que tengamos en un dispositivo con muchas funciones incorporadas sin la necesidad de llevar otros aparatos.

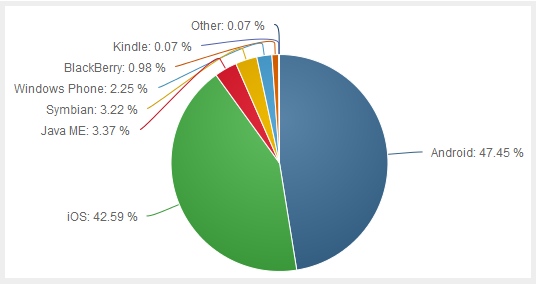
Por estos motivos es de gran importancia tener el conocimiento para poder desarrollar y explotar la infinidad de posiblidades que ofrecen los dispositivos móviles, para poder ofrecer nuevas funciones o funciones ya existentes pero con ciertas mejoras.

Dentro de toda la variedad de tipos de dispositivos móviles, dentro de marcas, modelos. Tamaños, etc, existe una divisón clara, que es el sistema operativo de soporte. En la actualidad el mercado smartphone se lo reparten 6 compañías con sus respectivos sistemas operativos. Estos sistemas operativos son Android, iOS, BlackBerry OS, Microsoft Windows Phone, Symbian, y Bada:

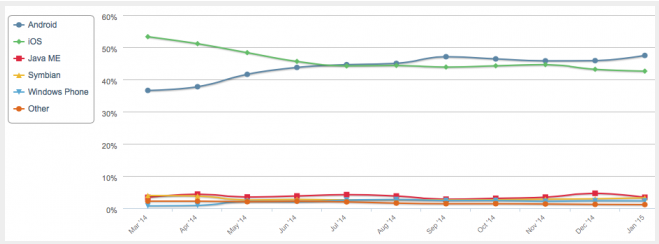
* Android: SO basado en Linux que se ejecuta en Java, desarrollado por Google. En 2007 se lanzó el primer kit para desarrolladores y desde esa fecha hasta la actualidad ha ido escalando posiciones hasta colocarse en lo más altos cómo sistema operativo en dispositivos de móviles y tablets.
* iOS: Sistema operativo desarrollado por Apple. En su origen, ffue diseñados solamente para el disponsitivo iPhone, pero sus uso se ha extendido a otros dispositivos de la marca Apple como iPod e iPad.
* BlackBerry OS: Desarrollado por Research In Motion (RIM) para dispositivos BlackBerry. Este sistema operativo está basado en Java y emplea MIDP sobre CLDC. Soporta multitarea y diferentes métodos de entrada como trackwheel, trackball, touchpad y pantallas táctiles.
* Microsoft Windows Phone: Sistema operativo propietario de Microsoft. Como puntos fuertes, podemos decir que soporta pantallas táctiles de alta resolución, integración con redes sociales y multiescritorio. Pero también tiene puntos débiles, como los siguientes: Mala sincronización de datos con el PC, transferencia por Bluetooth e intercambio de tarjetas de memoria externas.
* Symbian OS: Desarrollado por Symbian Ltd. fue fundada por la unión de varias empresas de telefonía móvil principalmente Nokia, Motorola o Ericsson entre otras. En sus principios, el SO más utilizado en los dispositivos móviles, ya que gestionaba de forma muy eficiente la memoria y energía del dispositivo, soportando en tiempo real los protocolos de comunicación y telefonía. Además soporta múltiples lenguajes de programación como Symbian C++, Java ME, Open C, etc.

En la actualidad este sistema operativo apenas se utiliza, pero debe estar en esta lista de SO principales ya que fue una gran revolución y el pionero en el cambio de paradigma que se dio al uso de los dispositivos móviles.

* Bada: Sistema operativo desarrollado por Samsung que permite cualquiera de los kernel de Linux. Sus aplicaciones son desarrolladas en C++ y ofrece varios controles de interfaz de usuario a los desarrolladores. También soporta diversos sensores, GPS y detección de rostros.



Tendencia de los sistemas operativos para móviles en los últimos años



Como se puede observar en la gráfica anterior el SO iOS va perdiendo terreno con respecto a Android

Datos estadístivos obtenidos de

<https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/>

**2.1 Motivación**

La elección de utilizar como soporte un dispositivo móvil, viene motivada por el actual escenario  tecnológico volcado en esta tecnología y en la revolución que ha existido en los últimos años y la previsión futura existente.

Los dispositivos móviles y en concreto los SMARTPHONE, han tenido una aceptación en los últimos años entre los usuarios que ha superado todas las previsiones realizadas y hoy en día el 99.99% de las personas adultas disponen de uno de estos dispositivos o incluso de varios.

Además el uso y funcionalidad que se le da al móvil, no es solamente el de su utilización para telefonía móvil, sino que se utilizan para otro tipo de funcionalidades para las que en sus inicios no fueron ni planteadas, cómo pueden ser las siguientes:

* Cámara fotográfica.
* Realización de video.
* Operaciones ofimáticas de cálculo, procesamiento de texto, etc.
* Navegación por Internet.
* Correo electrónico.
* Redes sociales
* Utilización cómo consulta de videojuegos
* Etc etc

En este trabajo no se va a profundizar en la evolución de los dispositivos móviles y de los sistemas operativos utilizados, pero si se ofrecerá una pequeña pincelada para entender la gran evolución que han tenido. En la siguiente tabla se puede observar esta evolución:

(tabla con disponsitivos móviles, sistemas operativos, capacidades, usuarios finales, etc)

Otro tema a tener en cuenta, es el sistema operativo del dispositivo móvil. Los SO disponibles son los siguientes:

(tabla con los dispositivos SO)

En la actualidad, los más utilizados son xxx y android. La selección de Android, viene dada por ser un SO de Software libre y por la cantidad de herramientas gratuitas que se pueden utilizar para su desarrollo. El SO de MAC está más encapsulado y solamente está destinado a los dispositivos MAC.

Sobre la plataforma de desarrollo, existen las siguientes opciones:

(tabla con plataformas, eclipse, android estudio, app inventor)

La elección de App inventor es por su sencillez para realizar pequeñas aplicaciones, prototipos y la forma tan peculiar e intuitiva de realizar programas, ya que es a través de bloques.

Para finalizar con este apartado de motivaciones, la elección de desarrollar un juego, es por el hecho de hacer una aplicación diferente a todas las que he podido desarrollar a lo largo de toda mi vida laboral, ya que siempre he desarrollado aplicaciones departamentales para soluciones empresariales en distintos entornos cómo PYMES y especialmente en Banca, en el sector de fiscalidad, por lo que hacer algo completamente diferente me llamaba la atención y me proporciona una motivación extra.

---

La selección de realizar un juego con esta herramienta, ya es una elección personal, ya que durante todos los años que llevo dedicado al desarrollo de Software, siempre la orientación de dicho Software, ha tenido una dirección para aplicaciones departamentales, cuyas funcionalidades se han apartado bastante de la mero entretenimiento, por lo que me parece una muy buena opción dedicar este proyecto al desarrollo de un tipo de software como es un Juego, cuyo único fin que quiere lograr es el de entretener al usuario final.

Y la elección del sistema operativo android, es evidente, ha que para la realización de software libre es la mejor opción, además de poder utilizar herramientas de software libre para el desarrollo.

**2.2. Medios utilizados**

Los medios utilizados para el desarrollo del juego, han sido los siguientes:

Elementos de hardware:

* Hp Intel inside CORE i5 con 8gb de RAM, 400Gb HD, etc etc.
* Smartphone Móvil Samsung S2 con sistema operativo android versión xxx
* Móvil Samsung Note 2 con SO Android version xx
* Móvil nisu con sistema operac

Elementos de software.

* SO Windows 8 o Ubuntun . Independientes
* App Inventor.
* Simulador de App Inventor.
* Procesador de textos.
* Proyect. Para planificar el proyecto.

1. Antecedentes/Estudio del arte

App Inventor es una herramienta originalmente desarrollada por Google. Su desarrollo estuvo basado en la informática educativa, huyendo del desarrollo de programas a través de comandos por línea, Google desarrollo este entorno de trabajo basado en un entorno visual en el que se desarrolla a través de una serie de bloques para poder crear distintos programas para dispositivos móviles.

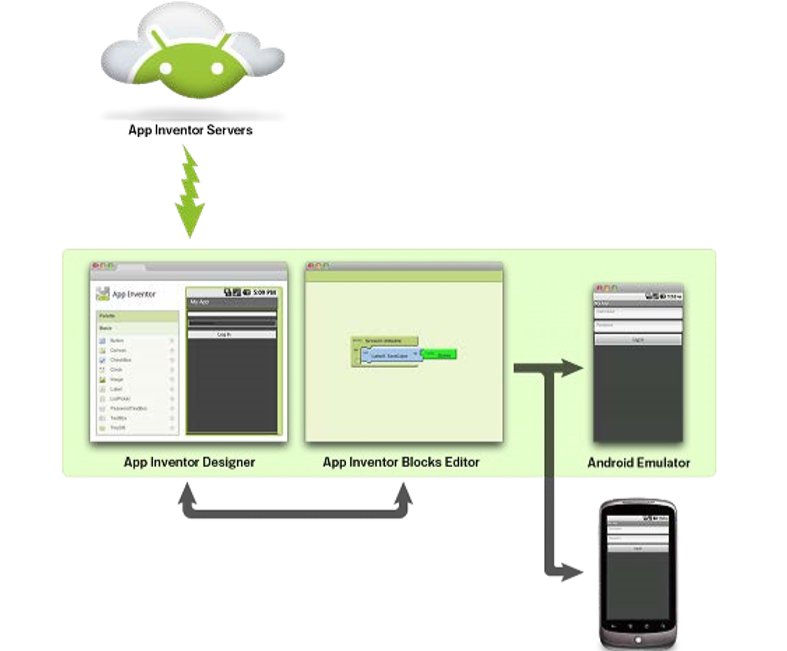
El objetivo que se buscaba con App Inventor es que cualquier persona no familiarizada con la programación y SDK de Android, fuera capaz de crear aplicaciones que corrieran bajo Android.

App Inventor está basada en una interfaz gráfica similar a Scratch y StarLogo en la que se permite al usuario ir creando una aplicación a base de mover, arrastrar, dibujar distintos bloques y una vez ‘dibujado’ el entorno gráfico, permite enlazar los distintos componentes entre ellos como si fueran piezas de puzle, permitiendo la interacción entre todos ellos.

Además de la forma peculiar de desarrollo (que parece un puzle), otra de las características de esta herramienta es que se ejecuta vía web, sólo es necesario ejecutar una pequeña aplicación para mostrar el Editor de Bloques ( Block Editor ) y el emulador, incluso las aplicaciones se guardarán en la web. Por lo cual se puede acceder a nuestros proyectos desde cualquier equipo conectado a Internet. Permite conectarse directamente al equipo Android, aunque en algunos casos no será posible, por ello incluye un emulador para probar las aplicaciones que se desarrollen. Permite guardar el proyecto en PC, para enviarlo a otros usuarios o para hacer copia de seguridad y permite también descargar la aplicación .apk compilada para ser instalada en el dispositivo Android. Por lo cual no se necesita ningún tipo de compilador ni nada por el estilo, todo se realiza en el entorno de App Inventor.

El 12 de julio de 2010 Google puso a disposición al público la herramienta de manera gratuita y en la actualidad App Inventor es mantenida por el MIT (Massachusetts Instusachusetts Institute of Technology ) bajo la licencia de software libre, ya que el 31 de diciembre de 2011 Google puso fin a su desarrollo y cedió el código al MIT.

La arquitectura básica de App inventor es la siguiente:



Para poder desarrollar un programa para la plataforma Android, también es conveniente dar un vistazo a la arquitectura básica de Android, a sus componentes y a sus procesos:

1. bjetivos

El presente proyecto, tiene como objetivo el desarrollo de un juego destinado a dispositivos móviles con sistema operativo Android.

Para poder realizar el desarrollo, se utilizará la herramienta App Inventor. Dado que no estamos familiarizados con dicha herramienta de desarrollo, inicialmente se incluirá un análisis y un pequeño manual de App Inventor, para poder dar los primeros pasos en la configuración y desarrollo y de esta forma poder iniciar el desarrollo del juego o de otro tipo de aplicaciones.

En líneas generales, los objetivos del proyecto estarán definidos en los siguientes puntos:

* Origen e historia de la herramienta AppInventor.
* Tutorial de AppInventor
  + Instalación
  + Configuración
  + Componentes de AppInventor
  + Desarrollo
  + Simulación de juegos
  + Importación a móviles
* Desarrollo de un juego completo con las siguientes requisitos:
  + Juego con distintos niveles de dificultad
  + Distintas pantallas
  + Puntuación
  + Tiempo de juego
  + Vidas
  + Registro de máximas puntuaciones
* Analizar los siguientes puntos (conclusiones):
  + Introducción AppInventor en la enseñanza pública
  + Desarrollo de prototipos con AppInventor
  + Introducción cómo herramienta para hacer prototipos en empresas de desarrollo

1. Metodología

Para el apartado desarrollo de juego, se va a elegir una metodología de desarrollo clásica por fases o en cascada, es decir que una fase no se iniciará hasta que no termine la anterior.

Las fases del proyecto serán las siguientes:

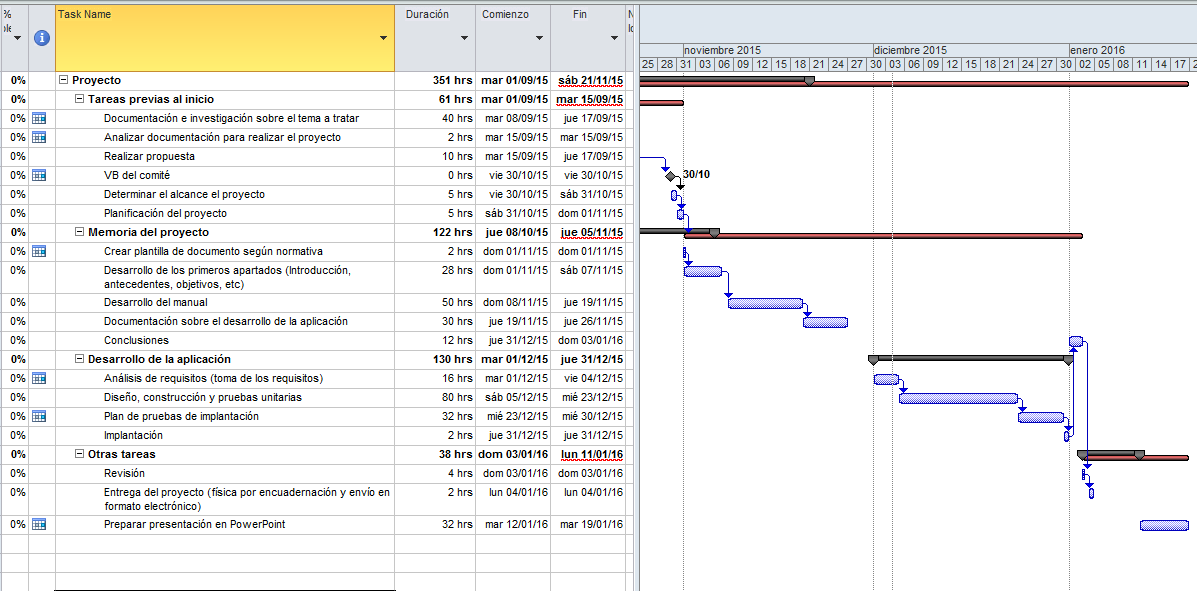
* **Planificación del proyecto.** Inventariar todas las tareas que se deben realizar y planificarlas utilizando como herramienta el Proyect.
* **Análisis de requisitos.** Toma de los requisitos y funcionalidades que debe tener el juego.
* **Diseño, construcción y pruebas unitarias.** Fase que aborda el diseño y la construcción del juego según los requisitos y funcionalidades aportadas, con la incorporación de las pruebas unitarias de cada elemento que se vaya desarrollando.
* **Pruebas de implantación.** Pruebas de caja negra, caja blanca, de implantación, etc. Crear un plan de pruebas para poder dar el VB. Revisar el análisis de pruebas desarrollado en calidad del software.
* **Implantación.** Pasos que se deben seguir para implantar el juego en los dispositivos móviles, publicación del juego en appstore, etc.

<incluir gráfico>

1. Planificación del proyecto

Para el correcto desarrollo del proyecto y su seguimiento, se han definido las siguientes tareas, con sus esfuerzos estimados y sus fechas estimadas de inicio y de fin.

|  |  |  |  |
| --- | --- | --- | --- |
| **PLANIFICACIÓN DE TAREAS** | | **FECHA PREVISTA** | |
| **Tareas previas al inicio** | **Horas** | **INICIO** | **FIN** |
| Documentación e investigación sobre el tema a tratar | 40 | 08/09/2015 | 17/09/2015 |
| Analizar documentación para realizar el proyecto | 2 | 15/09/2015 | 15/09/2015 |
| Realizar propuesta del proyecto | 10 | 16/09/2015 | 10/10/2015 |
| Visto Bueno del comité | 0 | 30/10/2015 | |
| Determinar el alcance el proyecto | 4 | 31/10/2015 | 31/10/2015 |
| Planificación del proyecto | 5 | 31/10/2015 | 05/11/2015 |
| **TOTAL** | 61 |  |  |
| **Memoria del proyecto** | **Horas** |  |  |
| Crear plantilla de documento según normativa | 2 | 01/11/2015 | 01/11/2015 |
| Desarrollo de los primeros apartados (Introducción, antecedentes, objetivos, etc) | 28 | 01/11/2015 | 20/11/2015 |
| Desarrollo del manual | 50 | 08/11/2015 | 19/11/2015 |
| Documentación sobre el desarrollo de la aplicación | 30 | 01/12/2015 | 31/12/2015 |
| Conclusiones | 12 | 01/12/2015 | 03/01/2016 |
| **TOTAL** | 122 |  |  |
| **Desarrollo de la aplicación** | **Horas** |  |  |
| Análisis de requisitos (toma de los requisitos) | 16 | 01/12/2015 | 04/12/2015 |
| Diseño, construcción y pruebas unitarias | 80 | 05/12/2015 | 23/12/2015 |
| Plan de pruebas de implantación | 32 | 23/12/2015 | 30/12/2015 |
| Implantación | 2 | 31/12/2015 | 31/12/2015 |
| **TOTAL** | 130 |  |  |
| **Otras tareas** | **Horas** |  |  |
| Entrega primer borrador | 1 | 10/12/2015 | 14/12/2015 |
| Entrega segundo borrador (si hiciera falta) | 1 | 22/12/2015 | 23/12/2015 |
| Revisión | 4 | 04/01/2016 | 04/01/2016 |
| Entrega del proyecto (física por encuadernación y envío en formato electrónico) | 2 | 07/01/2016 | 07/01/2016 |
| Preparar presentación en PowerPoint y defensa | 32 | 12/01/2016 | 19/01/2016 |
| **TOTAL** | 40 |  |  |
| **TOTAL DE HORAS** | 353 |  |  |



1. Tutorial de desarrollo (implementación estudio desarrollo)

En este apartado, se presenta un pequeño tutorial de guía para empezar a utilizar la herramienta App Inventor.

**7.1 Instalación y configuración.**

Como se ha dicho en el anterior apartado, App Inventor se ejecuta vía WEB por lo que no es necesaria su instalación en el PC donde se quieran desarrollar las aplicaciones simplemente necesitaremos una cuenta de Google para empezar a utilizarlo.

Pero dependiendo de cómo queramos realizar las pruebas de las aplicaciones que vayamos realizando, existen las siguientes opciones:

1.- Con un dispositivo Android con conexión WIFI. No sería necesaria la descarga de ningún Software en su PC, solamente tendrá que instalar la aplicación de Companion App Inventor en su dipositivo.

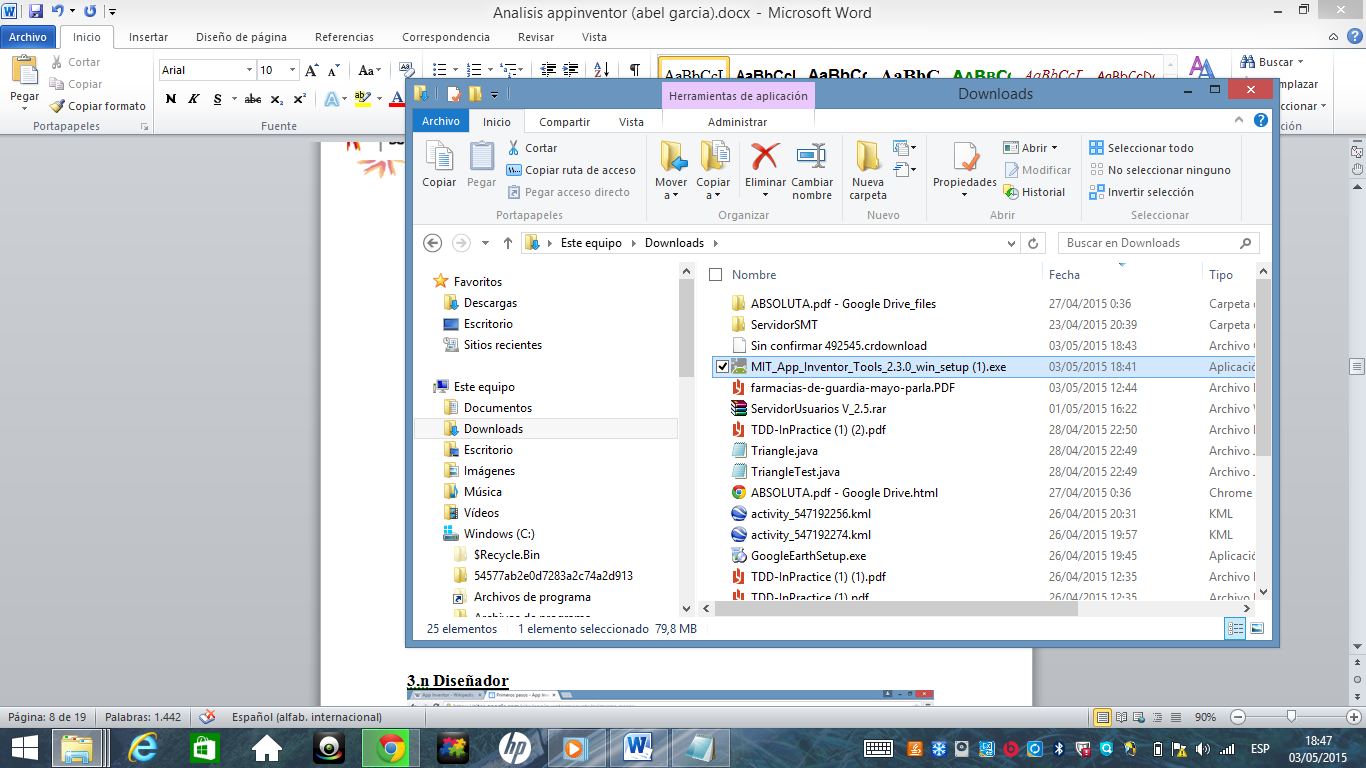
2.- Si tienes dispositivo Android pero no dispones de WIFI, podrás utilizar tu dispositivo realizando la conexión mediante un cable USB, pero necesitas tener instalado el software (drivers) necesario para la comunicación entre su PC y su dispositivo android.

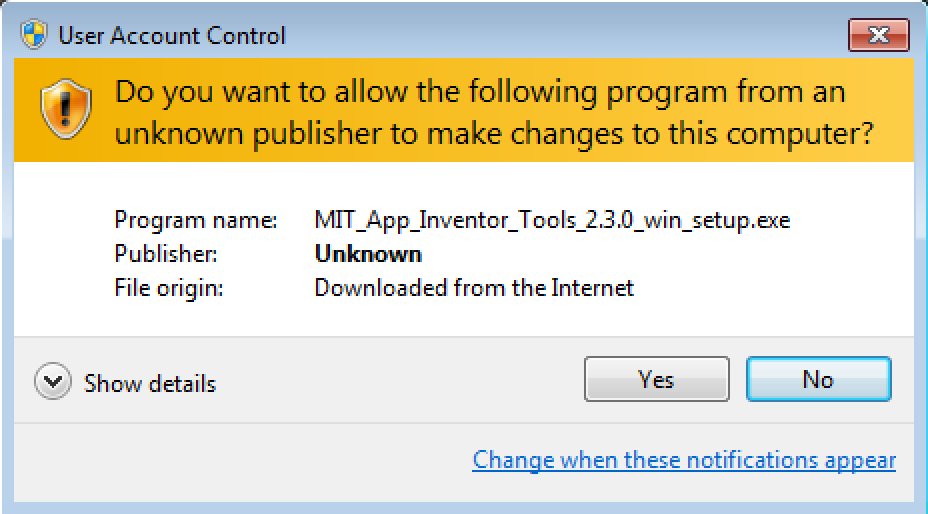
3.- Si no quiere utilizar un dipositivo Android para realizar las pruebas, podrá descargar un emulador. Si ha elegido esta última opción, deberá seguir los siguientes pasos 🡪

Paso 1. Descargar el emulador llamado **aiStarter** e instalarlo. En el siguiente enlace se puede descargar (a fecha 3 de mayo de 2015) 🡪

<http://appinv.us/aisetup_windows>

Paso 2. Ejecutar el archivo descargado. **MIT\_Appinventor\_Tools\_2.3.0 (~80 MB)**





Paso 3. Ejecutar aiStarter.

Desde Windows y Linux se debe ejecutar manualmente para después poder utilizar el emulador.



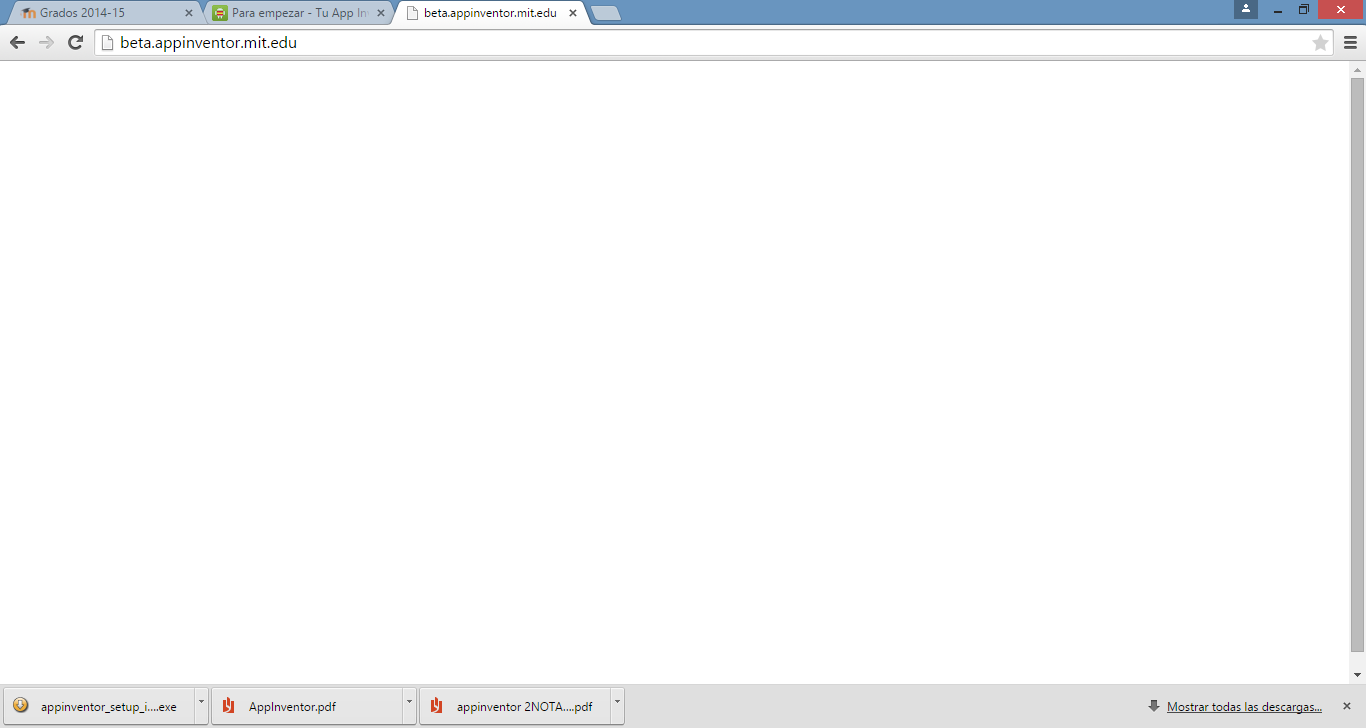
En los usuarios de Mac **aiStarter** se iniciará automáticamente en su Mac, y se ejecutará de forma transparente en segundo plano.

**Primera ejecución**

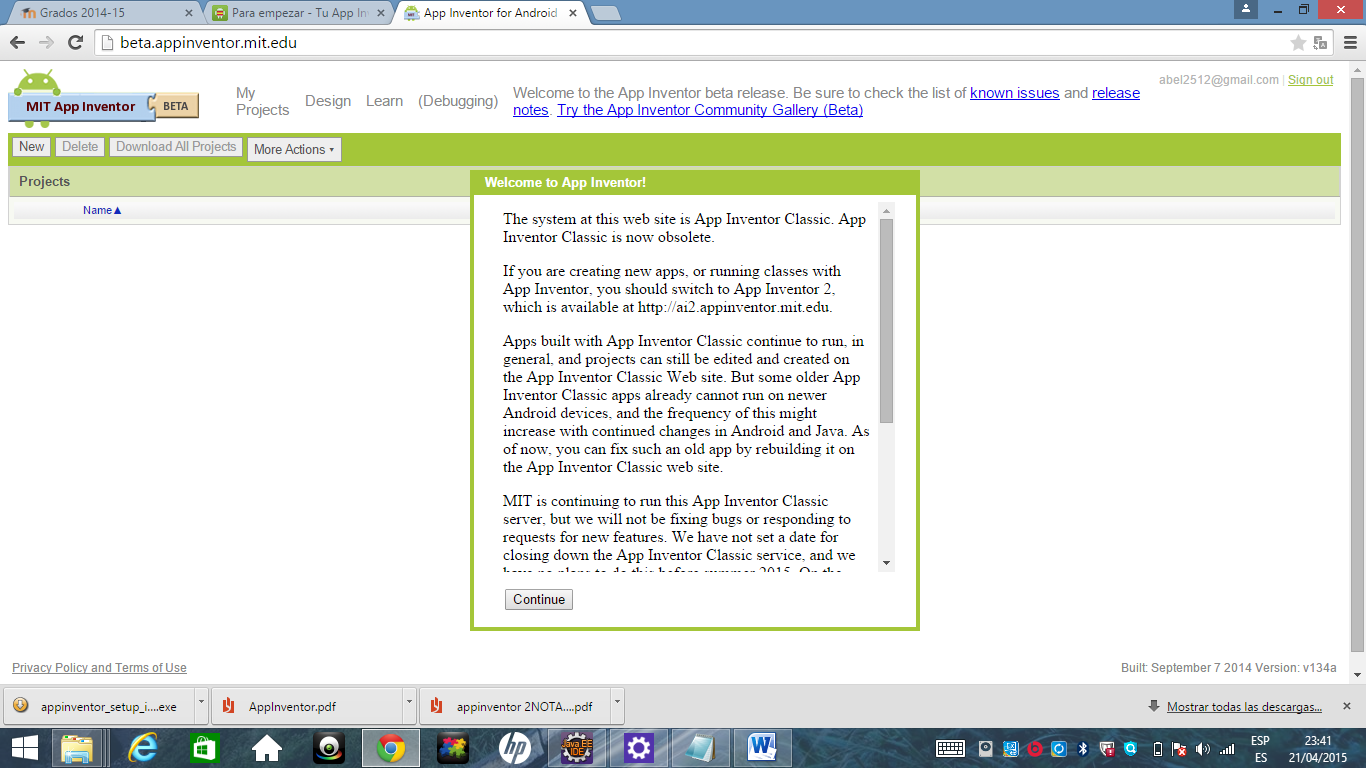
Lo primero que vamos a necesitar es una cuenta en google para poder acceder a App Inventor

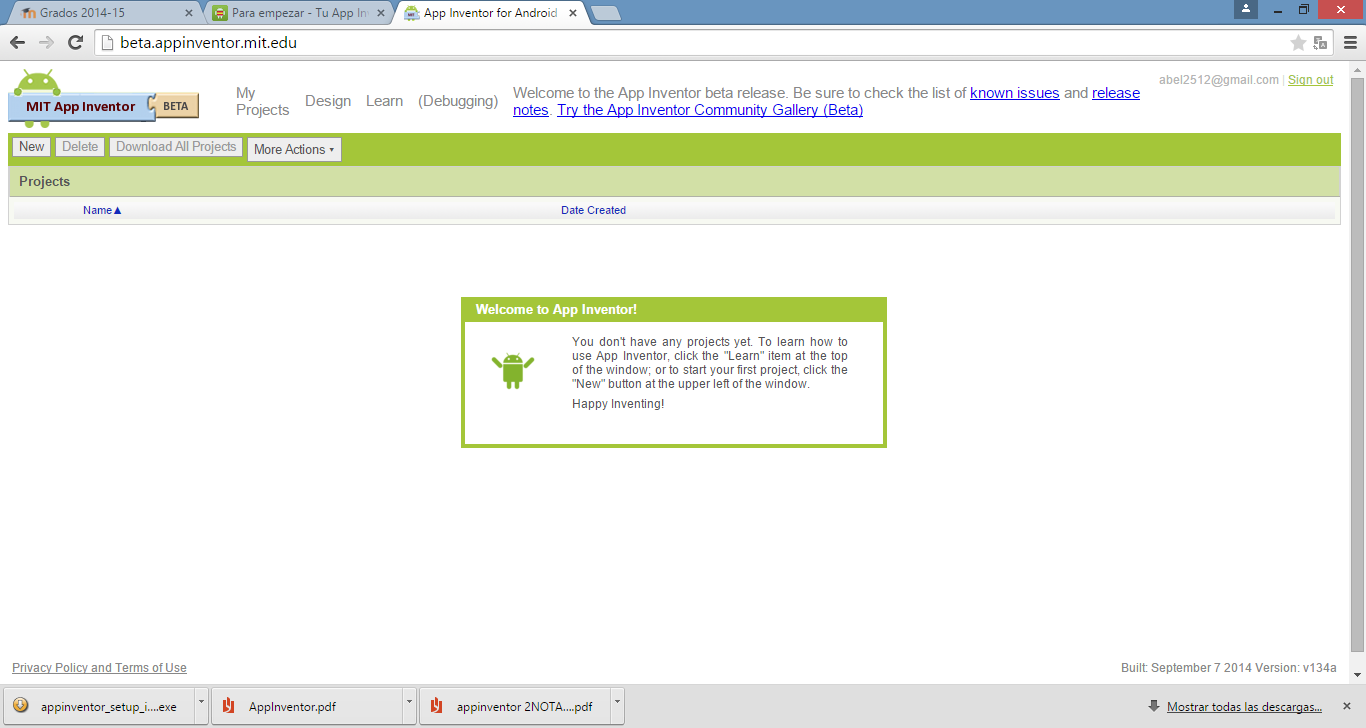


Pulsamos ‘permitir’ para compartir la dirección electrónica con el MIT

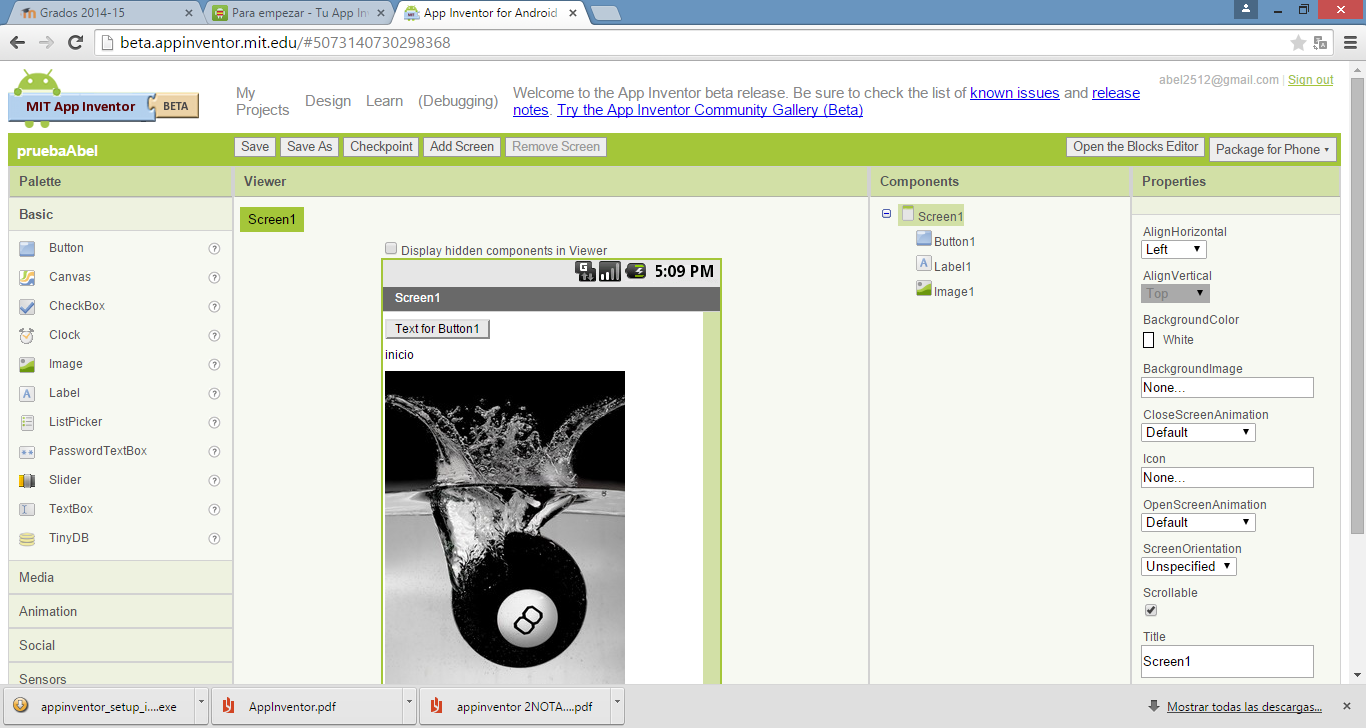


Nos da la bienvenida y aparecen unos mensajes informativos para indicarnos que App Inventor ha quedado obsoleto y existen una nueva versión de App Inventor 2

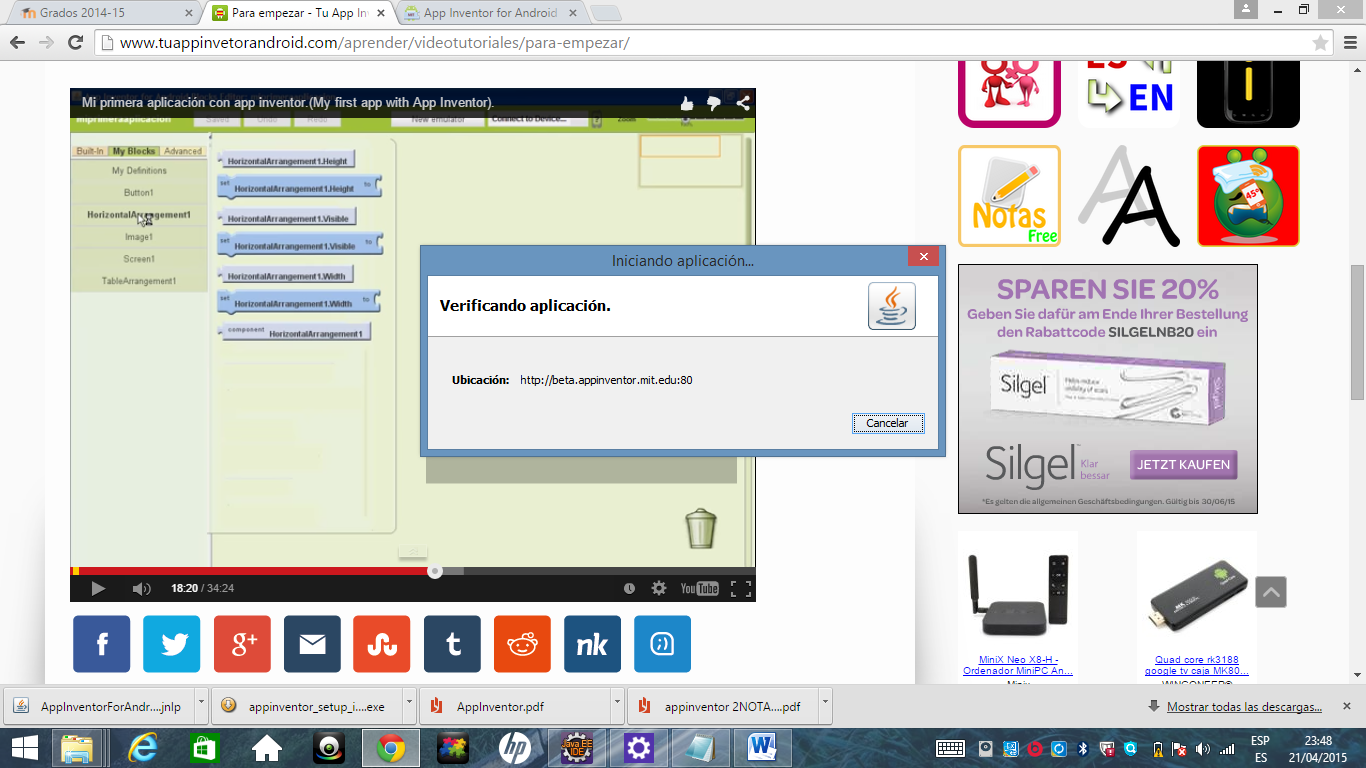


Aun así accedemos para realizar una primera prueba

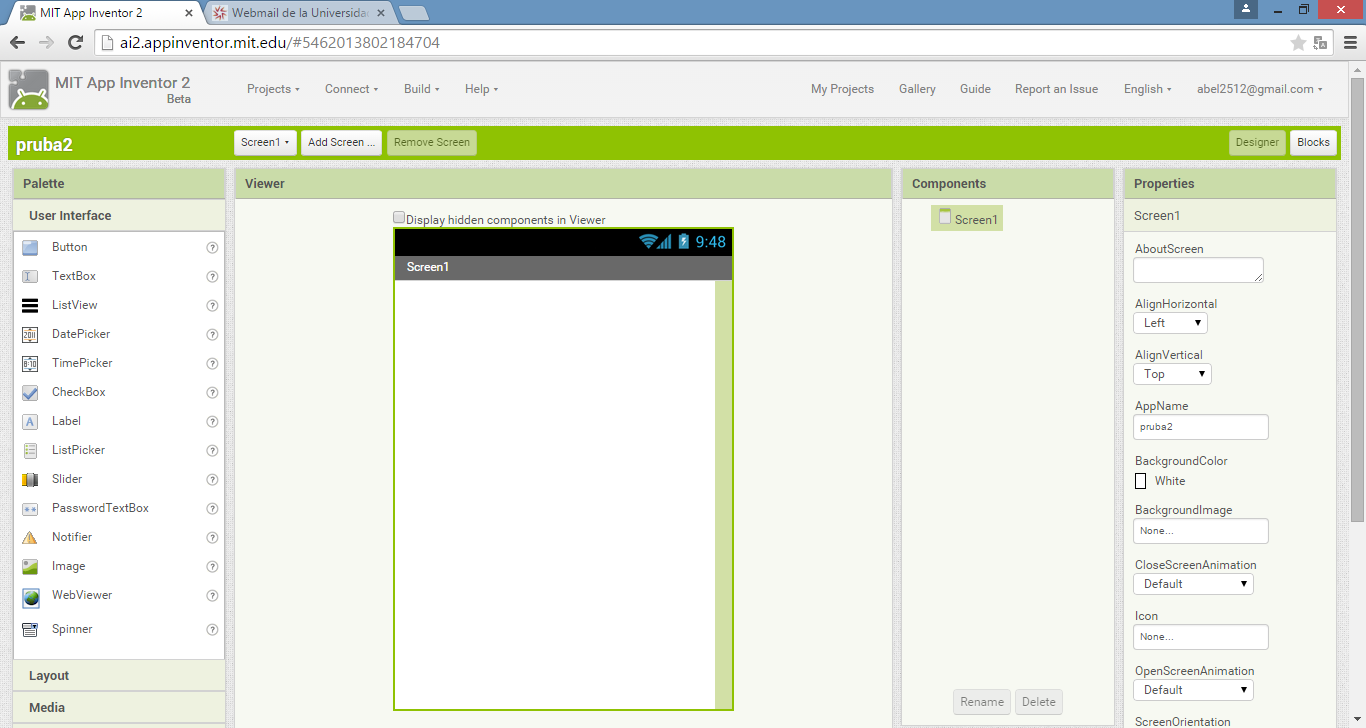
Incluimos una fotografía



Y accedemos a los bloques



Hacemos caso a una de las recomendaciones anteriores, y accedemos a la página de app inventor2 y cómo se puede observar en la siguiente imagen, el aspecto es muy parecido (por no decir idéntico) que la versión anterior, utilizando la misma división en 4 partes del escritorio de trabajo. Uno de los principales cambios de la versión 2 es que se puede seleccionar el español como idioma.



**7.2 Mi primer programa.**

Este primer programa va a ser desarrollado con App Inventor 2 ya que es la última versión. Será un programa simple para aprender a utilizar algunas herramientas básicas.

Las funcionalidades de este primer programa serán las siguientes:

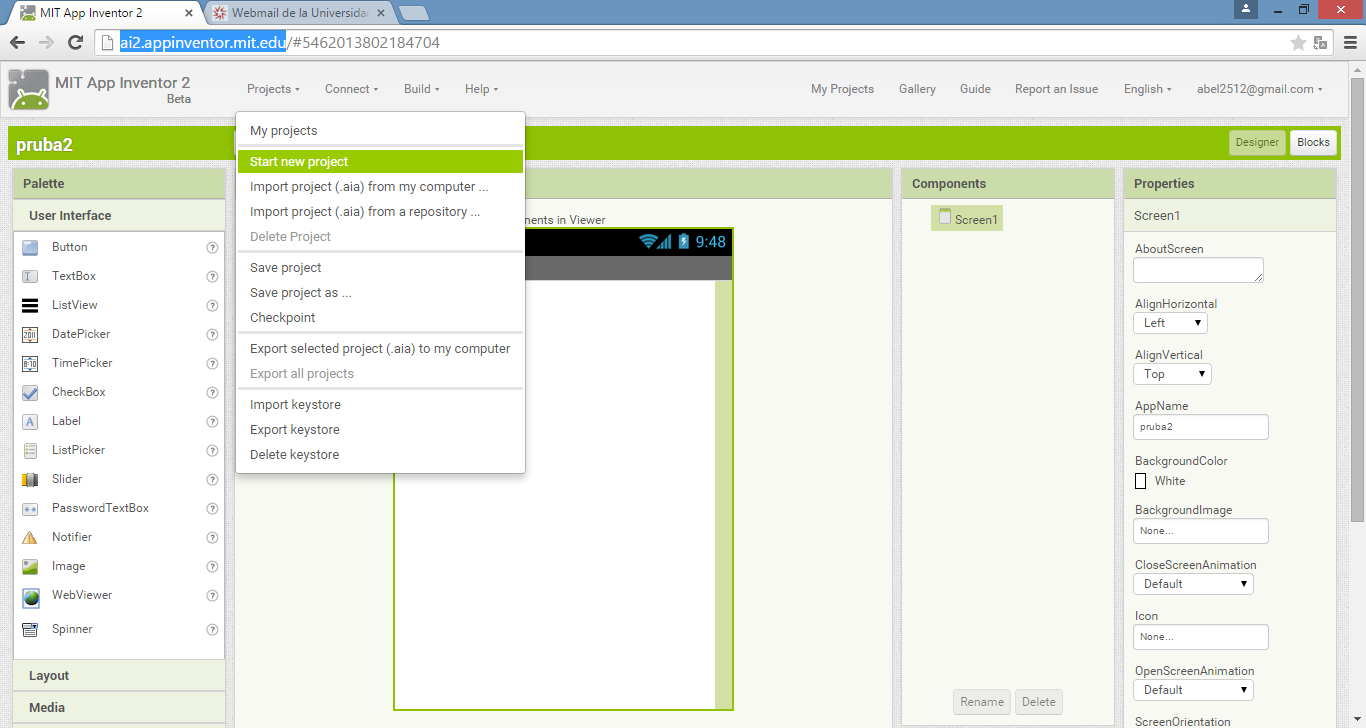
* Viajar entre ventanas
* Pulsar un botón para que cambie de color.
* Incluir un botón para que aparezca un selector de lista
* Acceso a la agenda de teléfonos del teléfono móvil.
  + Recuperar un nombre de contacto y visualizar por pantalla.
  + Recuperar un teléfono y visualizar por pantalla

Lo primero que debemos hacer es acceder al entorno de desarrollo de app inventor2

<http://ai2.appinventor.mit.edu/>

Si no hemos iniciado sesión en Google, nos obligará.

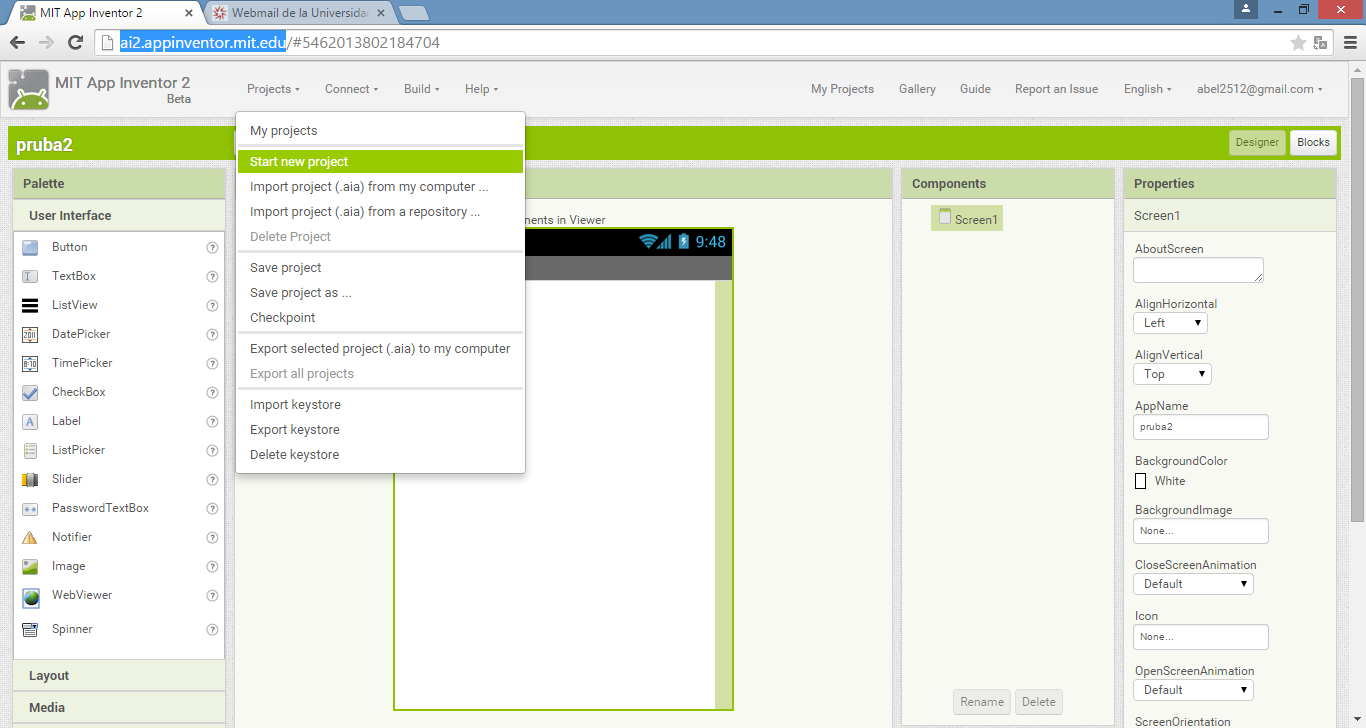
Nos identificamos y ya accedemos al entorno de trabajo. En este entorno pulsamos ‘start new proyect’ y lo llamaremos ‘prueba1’



Existen 3 pasos principales para desarrollar aplicaciones con App inventor una vez iniciado el proyecto. Estos pasos son los siguientes:

1. Definir los formatos de las ventanas del programa. Podrá haber una o varias.
2. Una vez definidas las ventanas, se deben añadir las funciones que queremos que realice el programa, para ello debemos acceder a la parte de bloques.
3. Una vez definidos los pasos 1 y 2, quedaría probar la aplicación.

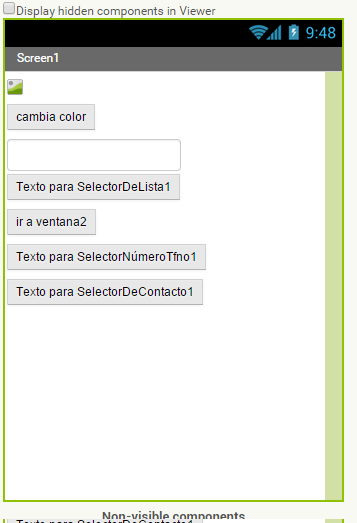
**Paso 1. Definir formatos.**



Hasta el visor de componentes

Arrastrar de aquí

Arrastramos componentes al visor de componentes para componer la primera ventana:



Selector de teléfono

Selector de contacto

Botón

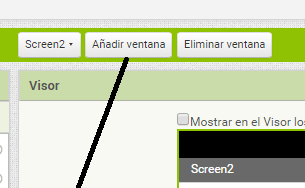
Caja de texto

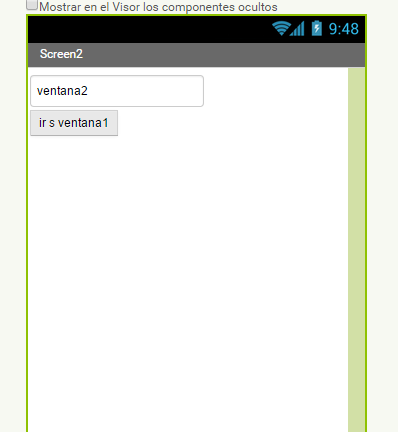
Componente botón

COMPONENTES

Selector lista.

Añadimos nueva ventana



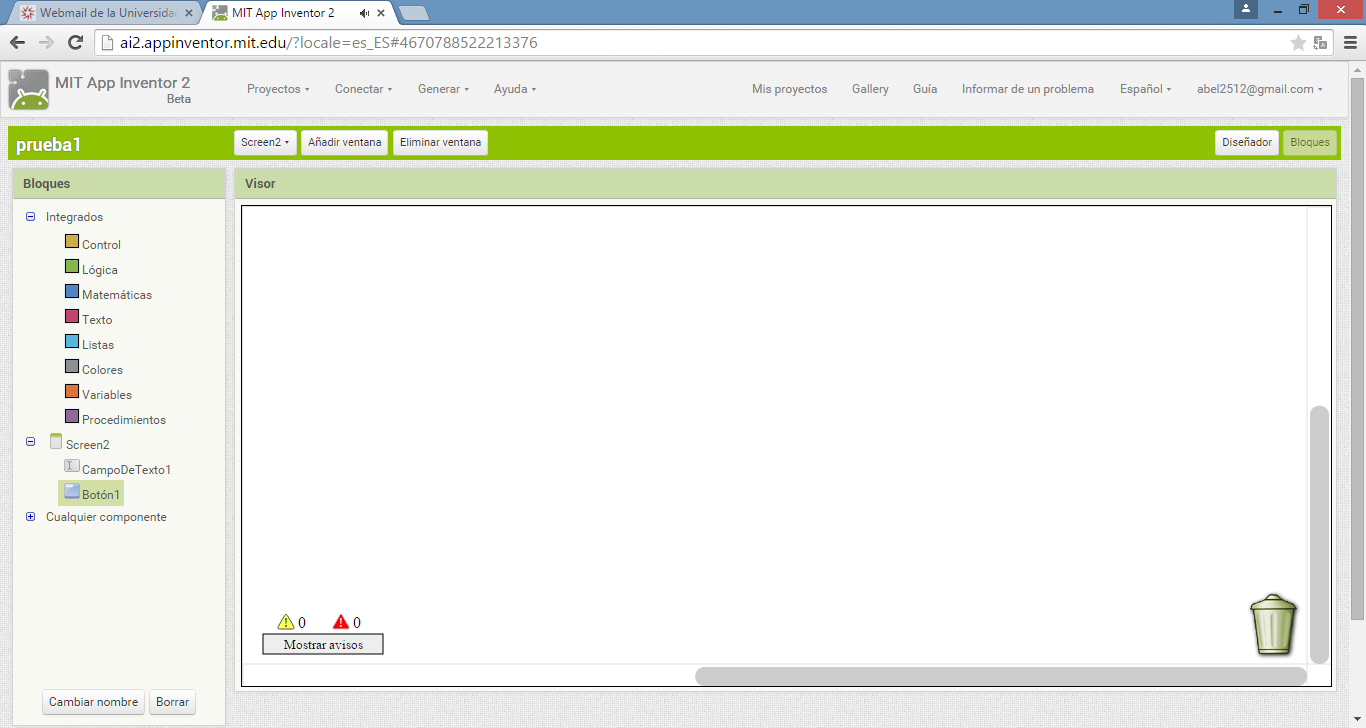


Esta segunda ventana es muy sencilla, solamente definida para poder probar como se viaja entre dos pantallas distintas, por lo que solamente se ha añadido un texto y un botón para volver a la ventana1.

**Paso 2. Bloques**

Una vez definido el formato de las ventanas que van a ser usadas en esta primera aplicación, debemos asociar a los distintos componentes algún tipo de funcionalidad, para ello utilizamos este apartado de bloques, que es parecido a un juego de puzle, donde tenemos que ir encajando las distintas piezas para poder crear la aplicación con las funciones que deseemos.

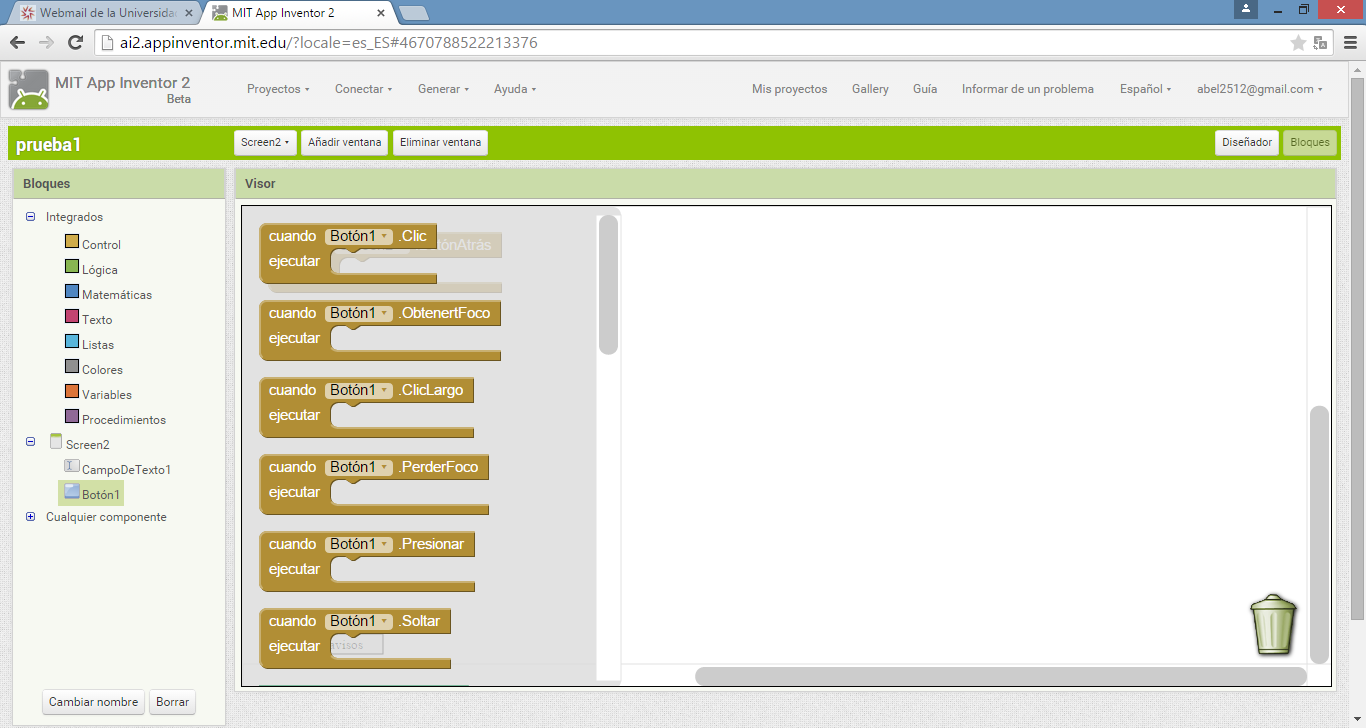
El aspecto que tiene es el siguiente:



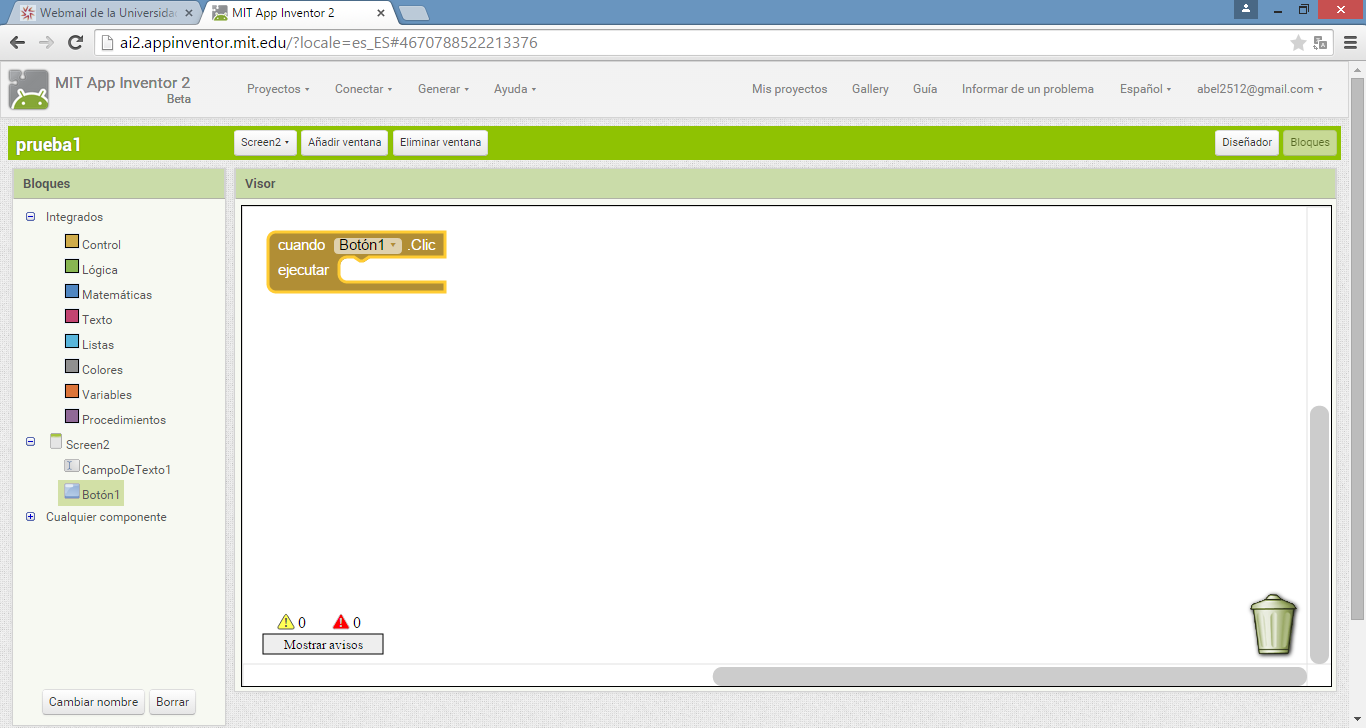
Zona reservada para construir el puzle (la aplicación)

Zona donde están ubicadas las piezas del puzle que se pueden utilizar (los componentes

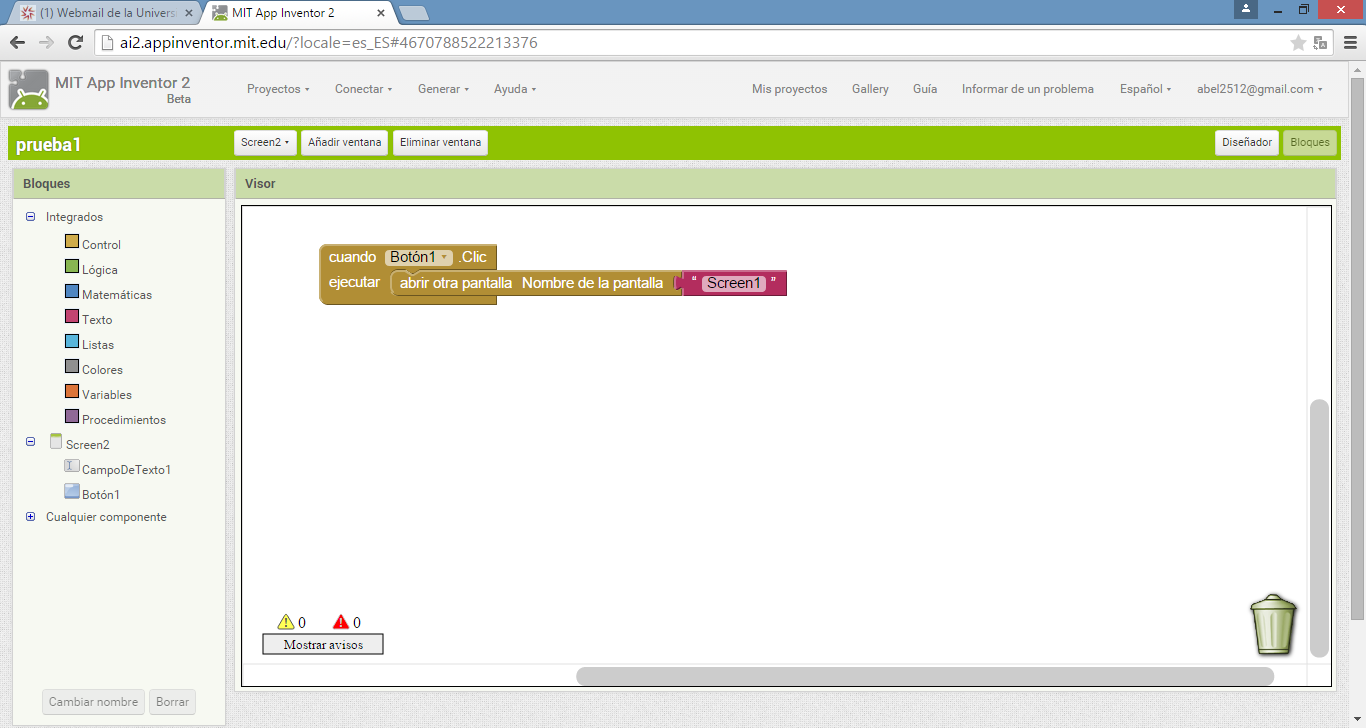
Pulsando el componente de la parte izquierda, nos aparecerán las distintas piezas que podremos utilizar (las opciones), por ejemplo pulsamos el componente boton1:



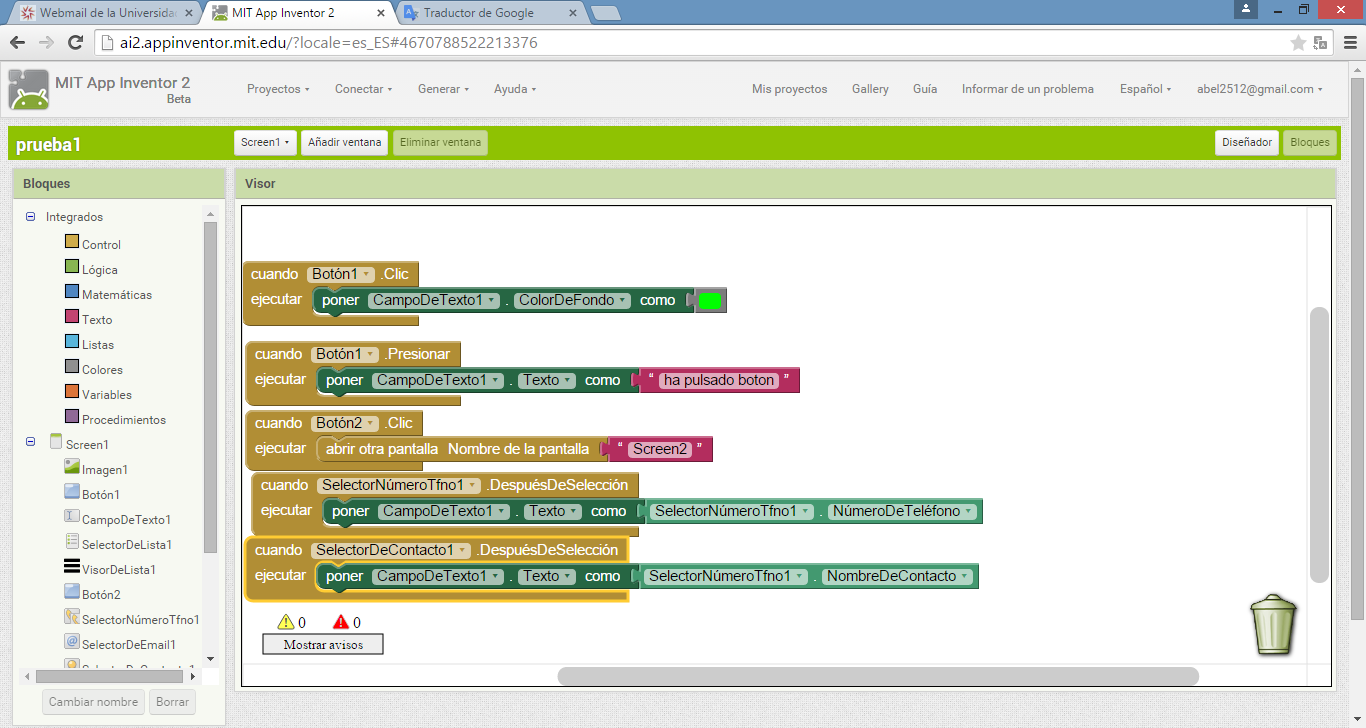
Y seleccionamos y seleccionamos la pieza ‘cuando boton1 clic’



Incluimos un control asociado al pulsar botón, que es iniciar una nueva ventana llamada Screen1



Se definen las funciones de la Screen1:



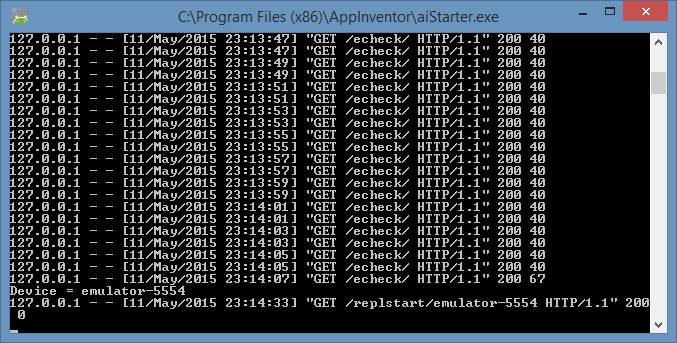
**Paso 3. Probar aplicación.**

Existen 3 opciones

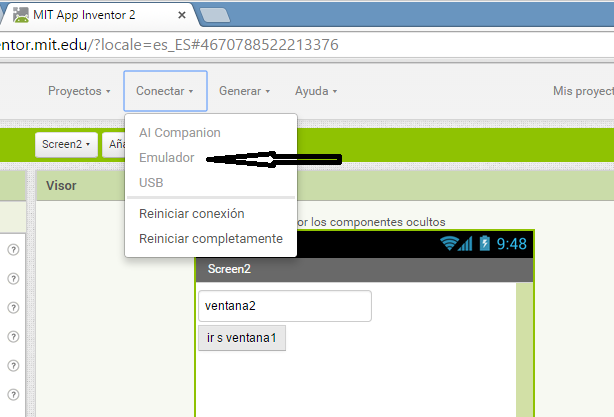
* Emulador.
* Utilizar dispositivo físico (móvil)
* Generar archivo apk

Emulador.

Previamente se debe ejecutar el programa del emulador aiStarter



Después desde el escritorio de desarrollo en conectar, pulsamos Emulador





Esperamos un pequeño ‘delay’ dependiendo del ordenador donde corra el emulador y aparecerá la siguiente pantalla:



Vamos a probar las funcionalidades descritas en al inicio del apartado, las recordamos:

* Viajar entre ventanas
* Pulsar un botón para que cambie de color.
* Incluir un botón para que aparezca un selector de lista
* Acceso a la agenda de teléfonos del teléfono móvil.
  + Recuperar un nombre de contacto y visualizar por pantalla.
  + Recuperar un teléfono y visualizar por pantalla

Funcionalidad, viajar entre ventanas:

Pulsamos botón ‘Ir a ventana1’ y nos aparecerá la ventana1



Y si volvemos a pulsar ir a ventana2, volveremos a la ventana 2



Funcionalidad Cambiar de color. Pulsamos el botón de ‘cambia de color’



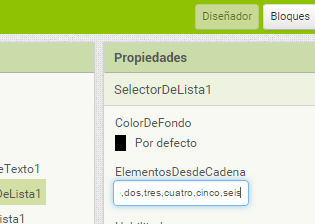
Funcionalidad selector lista.



La lista que aparece al pulsar este botón, son los elementos que se han definido en la parte de características de componentes ‘ElementosDesdeCadena’.

Probar, cambiar sobre los bloques y volver a probar es muy interactivo, no se necesita recompilar nada, por ejemplo añado un nuevo elemento en la lista

Vuelvo a acceder a la lista y ya me aparece el nuevo elemento







Para probar el resto de funcionalidades, incluyo varios contactos en el emulador



Seleccionamos Abel y al volver a la pantalla principal el número de teléfono se mueve al campo de texto



Aparece el número de teléfono seleccionado

Ahora Pulsamos el selector de contactos y seleccionamos el contacto de Marta



Aparece el número de contacto seleccionado

Funcionalidad. Acceso a los números de teléfono del móvil. Pulsamos botón de texto para seleccionar número de contacto

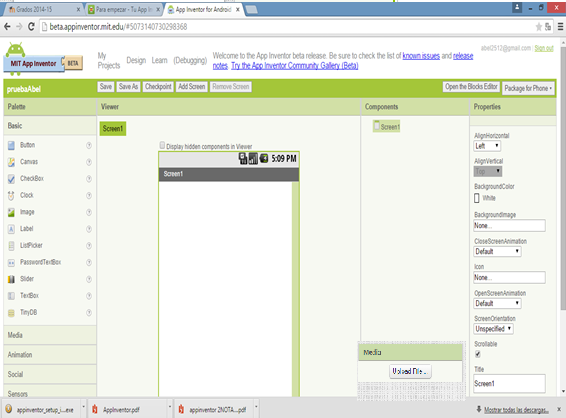




**7.3 Componentes de App inventor.**

**Escritorio de desarrollo.**

El escritorio de desarrollo se divide en las siguientes partes:



Propiedades de componentes

Multimedia

Árboles de componentes

Visor de componentes insertados

Paleta de componentes

Como se puede observar en la anterior imagen, el escritorio de trabajo se divide en 5 zonas:

* Paleta de componentes. Contiene todos los elementos que podemos insertar en nuestra aplicación. Existen 8 categorías de elementos y dentro de cada una de ellas se agrupan los distintos componentes que podemos utilizar, estos componentes pueden ser, cuadro de texto, botones, etc. Las 8 categorías son las siguientes:
  + 1. Interfaz de usuarios.
    2. Disposición.
    3. Medios
    4. Dibujo y animación
    5. Sensores
    6. Almacenamiento
    7. Conectividad
    8. Lego
* Visor de componentes insertados. En este contenedor, se simula la apariencia que tendrá la aplicación en el la pantalla del dispositivo móvil. Para añadir un elemento a la pantalla hay que arrastrarlo desde la paleta de componentes y soltarlo en el visor y dependiendo si el elemento tenga o no visibilidad, aparecerá o no aparecerá.
* Arboles de componentes. Lista de componentes que se han ido insertando en la aplicación de desarrollo. Cualquier componente insertado en el visor aparecerá en este apartado. *Importante*, para borrar un componente, se deberá borrar desde este apartado.
* Multimedia. Se muestran los elementos de tipo multimedia (imágenes y sonidos), que estarán disponibles en el proyecto. Para poder usar este tipo de medios, los archivos de imágenes y/o sonidos deberán subirse previamente al proyecto.
* Propiedades de los componentes. Cada componente insertado tiene unas propiedades específicas, las cuales pueden visualizarse y algunas de ellas modificarse en este apartado. Algunas de estas propiedades pueden ser por ejemplo el tamaño del texto, color, visible o no visible, etc.

**Editor de bloques.**

A través de los bloques o piezas de puzle, es como conseguiremos crear una aplicación para que un dispositivo móvil cumpla con las funcionalidades deseadas.

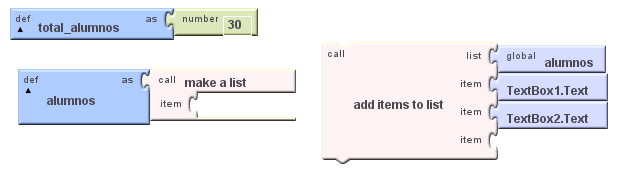
Como en cualquier lenguaje de programación en línea, en el editor de bloques nos podremos encontrar los distintos elementos para poder crear nuestros programas, pero en formato de bloques. Estos elementos básicos son: variables, condiciones, bucles, sentencias y eventos.

A continuación se describe cada uno de ellos:

Variables

Un elemento indispensable en cualquier lenguaje de programación son las variables, ya que sin ellas no podríamos guardar valores, asignar valores a texto, visualizar variables, entre otras muchas cosas.

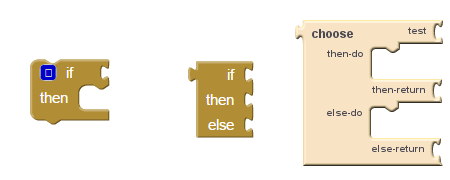
Los tipos podrán ser numéricos o de texto y se declaran al asignarles valor la primera vez.



Condiciones

Utilizando las condiciones, podremos crear programas que bifurquen por un camino u otro dependiendo de ciertas condiciones.

App inventor incluye las estructuras condicionales de if-then, if-then-else las cuales son utilizadas en cualquier lenguaje de programación, pero además incluye choose que es prácticamente igual a *if-then-else* con la diferencia de que permite devolver una variable en función de que se cumpla o no el test.



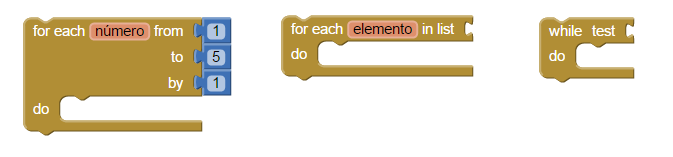
El color de los bloques de las condiciones es marrón-naranja.

Bucles

Los bucles se utilizan para poder repetir una parte del código un número determinado de veces o mientras se cumplan ciertas condiciones.

En App inventor se disponen de los siguientes bucles:

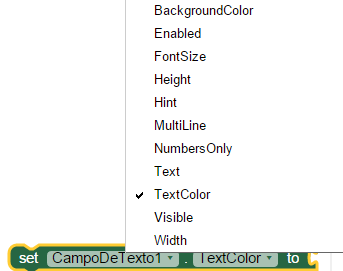
* While. Se ejecutan las sentencias que se encuentran dentro del bloque (do) mientras que se cumpla la condición que se defina en el test.
* For range. Es el for que se suele utilizar en otros lenguajes de programación, como en Java o en C. Se debe indicar el nombre de la variable, donde comienza el valor, donde debe terminar y la unidad de incremento. Se repetirán las sentencias incluidas dentro del ‘do’.
* Foreach. Es un iterador sobre elementos de una lista. Al igual que los anteriores, las sentencias que se encuentren dentro del ‘do’ serán las que se repitan.



El color de los bucles es marrón-naranja.

Sentencias

Con las sentencias, podremos indicar al dispositivo móvil que realice una acción. Existen muchos tipos de acciones; visualizar texto por pantalla, visualizar imágenes, cambiar color al texto, eliminar elementos, etc, etc.

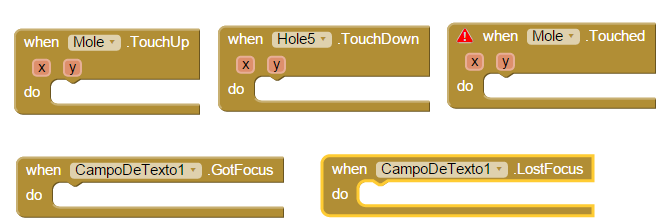
Este tipo de bloques está dividido en dos colores, los verdes se utilizan para modificar variables o propiedades de un objeto y los violetas que llaman a funciones. 

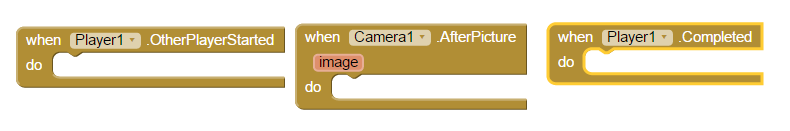


Eventos

Los eventos son los elementos con los cuales los programas pueden responder a la interacción con los usuarios y la interfaz, de tal manera que ante una acción del usuario, se pueda dar una respuesta.

Existen gran cantidad de eventos, algunos de ellos son los siguientes:





App Inventor no sólo dispone de objetos que se puedan colocar en la Screen, como puede ser un botón o una imagen, sino que también dispone de otro tipo de objetos que nos darán acceso a las funcionalidades que tienen los teléfonos y tablets, como acceso a Internet, Bluetooth, realizar llamadas, enviar y recibir mensajes, coordenadas de posición GPS, etc.

**7.4 Desarrollo paso a paso del juego Mole.**

A continuación se describen los pasos para desarrollar un juego basado en el típico juego ‘Mole’, pero con una serie de variantes.

Las reglas del juego Mole son las siguientes:

* El juego ‘Mole’ consiste en que un topo va apareciendo en una serie de agujeros y el usuario debe tocar el topo con el dedo antes de que se esconda en el agujero. Si lo consigue se sumará un punto a su marcador y sino no aparecerán puntos.

En el juego desarrollado en este manual, además de las funciones básicas de Mole, se añadirá una nueva ventana con una versión mejorada del juego, en la que se incluirán las siguientes funcionalidades que no existen en el original:

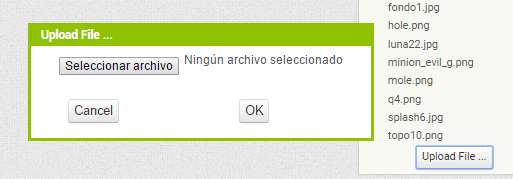
* Nueva ventana para incluir la versión mejorada. Esta ventana dispondrá de más agujeros, y de un entorno gráfico más agradable visualmente.
* Añadir nuevo topo para poder jugar. El nuevo topo se moverá más rápidamente, pero la puntuación si se toca será de 2 puntos.
* Un botón de reset para iniciar el marcador a 0.
* Botón para poder cambiar entre ambas versiones.
* Inclusión de sonidos tipo ‘beep’ para que suene y vibre cada vez que se tocan los topos.
* Aumentar la dificultad. Según se van realizando puntos, los topos se irán moviendo más rápidamente para complicar el juego. Cuando se reseteen los puntos, los topos volverán a moverse como al inicio.

**Desarrollo de la versión original.**

Entramos a la App Inventor y creamos nuevo proyecto llamado JuegoAppinvento\_Abel.

* Subimos al proyecto los archivos multimedia que vayamos a utilizar, en este caso las imágenes .png de mole y hole:



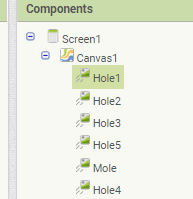
* Insertamos elemento Canvas de dibujo y le ponemos fondo verde.



* Insertamos 5 elementos ‘ImageSprite’ que se encuentran en la categoría de dibujo y animación:



* Pulsamos encima de cada uno de estos elementos y les cambiamos el nombre en características. Les nombramos con hole1…hole6 para después poderlos identificar más fácilmente.



* Seguimos insertando elementos en nuestra pantalla, hasta que quede más o menos como la siguiente imagen:

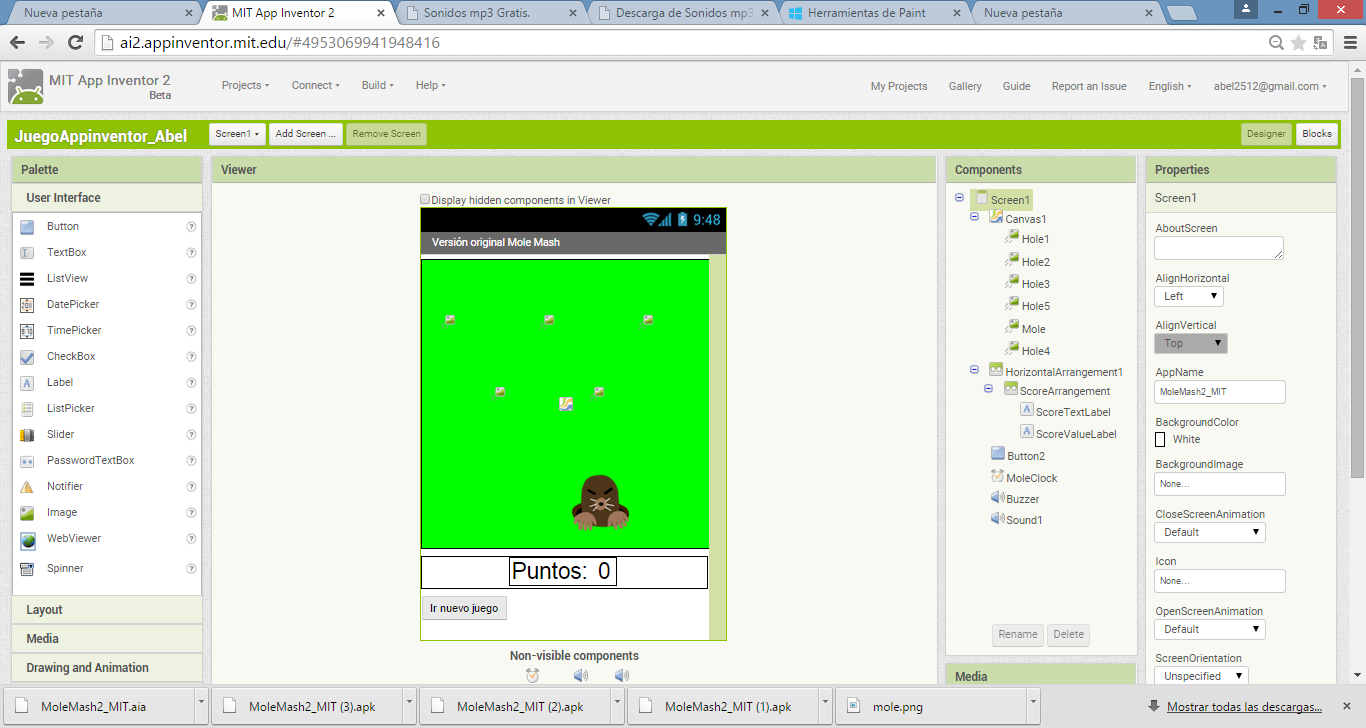


Imagen de animación ImageSprite para que se pueda ir moviendo y en características le asignamos en picture la imagen de mole.png para que aparezca la figura del topo

Elemento Caja de texto (textbox)

Etiqueta

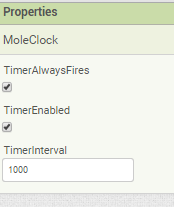
Elemento botón.



* Asignamos 3 elementos más que nos serán de utilidad para poder definir el tiempo de espera para que se vaya moviendo el topo. Estos componentes pertenecen a la categoría de multimedia y no son visibles:

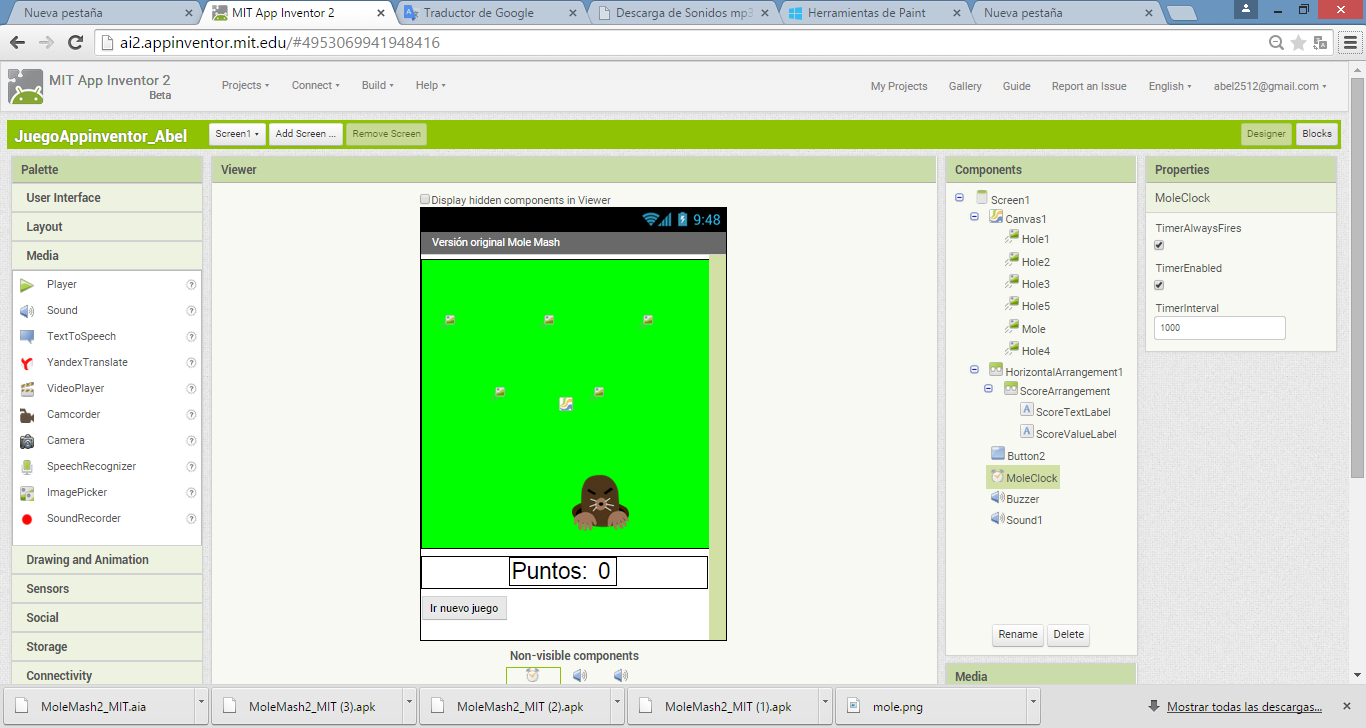


El Clock es un temporizador al que podremos definirle un tiempo de espera en milisegundos.



Y el sound lo podremos utilizar para asociar sonidos o para vibraciones.

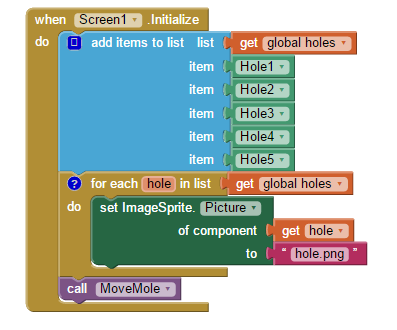
* Una vez que están definidos dos los componentes de nuestro juego, debemos realizar la programación a través de bloques para que los distintos elementos iteraciones. Para esto pulsamos el botón de blocks que se encuentra en la parte superior derecha de la pantalla (ya explicado en el primer programa ejemplo)



* Lo primero que haremos será definir una variable global tipo lista llamada holes, que nos será de utilidad para el tratamiento de los hoyos.



* A continuación dentro del evento de inicio de pantalla, incluiremos todas las acciones que necesitamos en el momento de iniciar la pantalla 1.

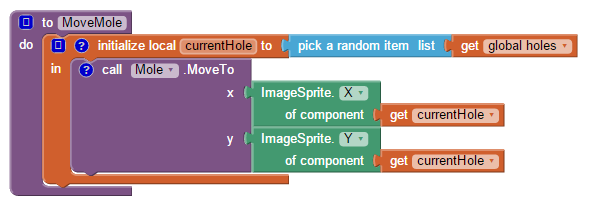


Se define una tabla y cada item se le asocia un hoyo.

Se incluye for each para asociar la imagen hole.png en los hoyos vacios.

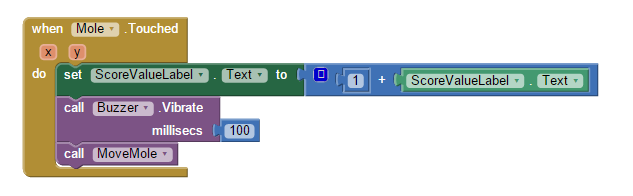
Y para finalizar se llama a un procedimiento ‘MoveMole’ que se definirá en el siguiente punto y servirá para ir moviendo el topo

* Procedimiento ‘MoveMole’, moviendo el topo.



Definimos el procedimiento y dentro del bloque insertamos el componente initialize local y pick a randoom ítem list, para que aleatoriamente nos situemos en un elemento de la lista.

* Evento para detectar que se ha tocado el topo.

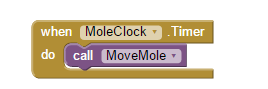


Cuando se toca el topo, lo primero es sumar 1 al marcador (piezas verdes y azules)

Después se llama al componente Buzzer que es de tipo Sound y lo que hace es vibrar durante 100 milisegundo el móvil.

Y para finalizar, se vuelve a llamar al procedimiento de MoveMole para cambiar de lugar el topo.

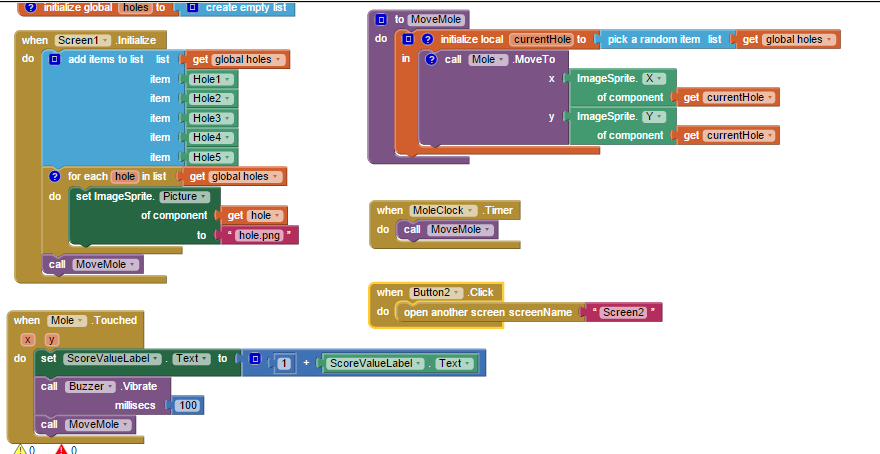
* El siguiente evento se utiliza para mover el topo cada cierto tiempo. Este tiempo son los milisegundos definidos cuando creamos el componente MoleClock.



* Para finalizar, se incluye el siguiente evento, para que cuando se pulse el Button2 se inicie la Screen2 que pertenece a la variante del juego.



Una vez creado el ‘puzle’ se debería probar el juego. El puzle final debería tener el siguiente aspecto:



Ejecutamos el emulador (previamente se debe lanzar aiStarter.exe ya explicado en el apartado del primer programa) y nos debería aparecer la siguiente ventana dentro del móvil virtual:



Si pulsamos logramos tocar el topo, los puntos irán aumentando de 1 en 1.

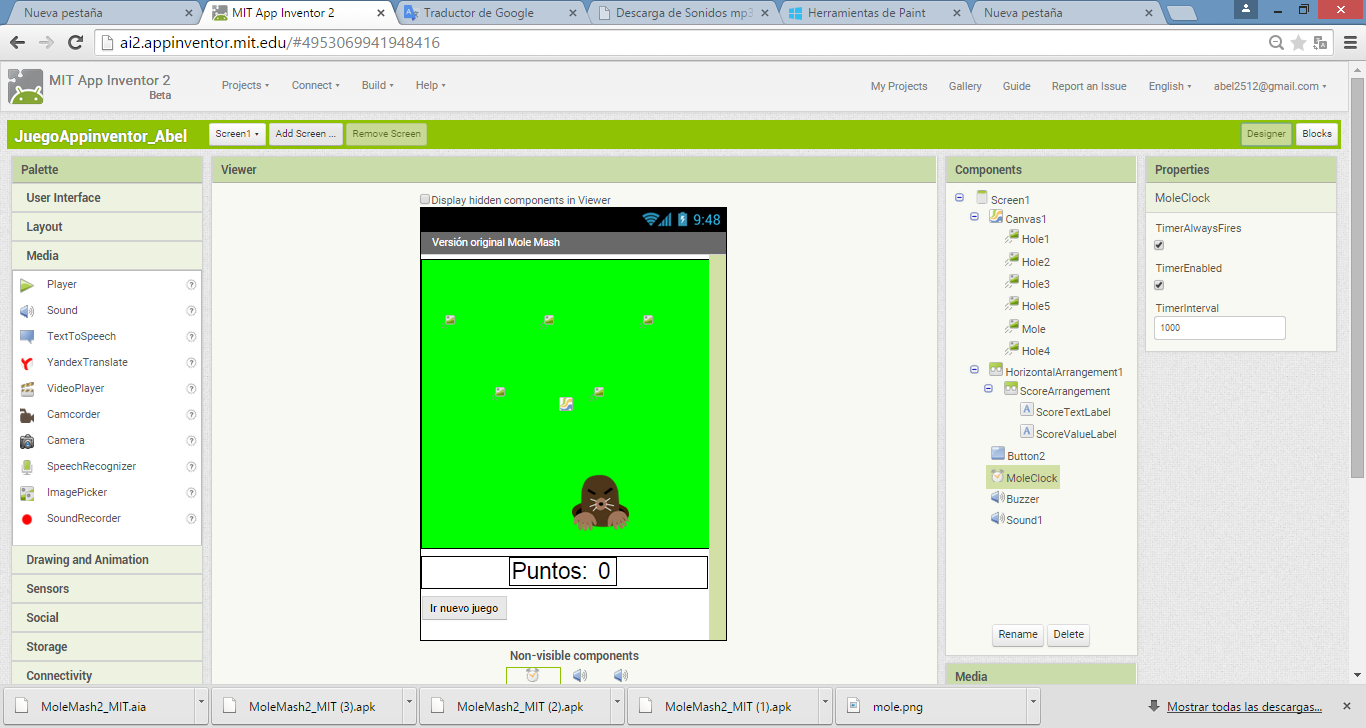
**Desarrollo de la versión mejorada.**

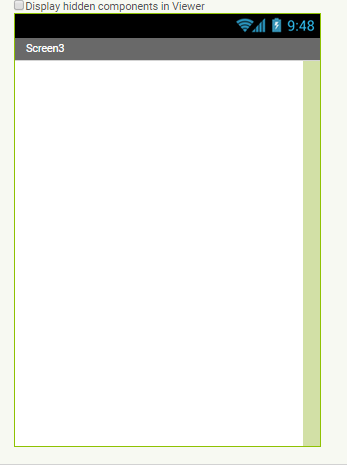
Como se ha dicho al inicio de este apartado, la nueva versión del juego tendrá las siguientes funcionalidades o características adicionales:

* Nuevo entorno gráfico.
* Nuevo topo más rápido y con más puntuación.
* Un botón de reset para iniciar el marcador a 0.
* Botón para poder cambiar entre ambas versiones.
* Inclusión de sonidos tipo ‘beep’ para que suene y vibre cada vez que se tocan los topos.
* Más dificultad. Según se aumentan los puntos los topos se moverán más rápido.

Los pasos para el desarrollo son los siguientes:

* Insertar una nueva ventana en el proyecto.

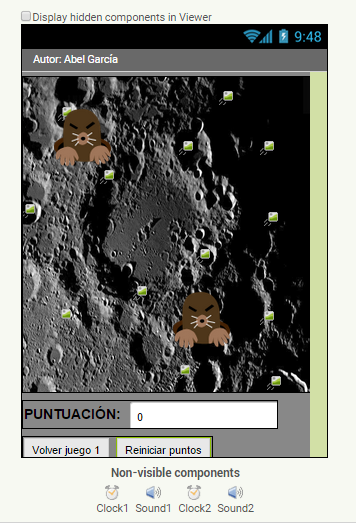




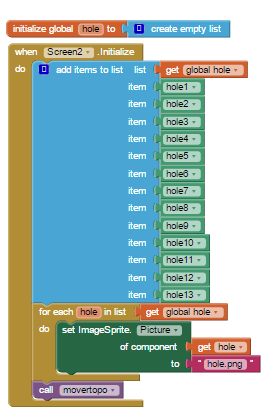
* Subimos al proyecto los archivos multimedia que vayamos a utilizar, serán los mismos que en la versión anterior más los siguientes elementos:

|  |  |
| --- | --- |
| Nombre | Descripción |
| Luna22.jpg.  Nos servirá para que haga de fondo |  |
|  | Se inserta otro temporizador para el segundo topo |
| Sonidos beep1 y beep2 mp3 | Al tocar los topos suene el sonido asociado |
| Un nuevo botón | para iniciar el marcador a 0 |
| Nuevos elementos de animación ImageSprite | Para disponer de más agujeros de donde puedan aparecer los topos. Se incluyen 7 nuevos agujeros, hasta un total de 13 |
| Nuevo elemento ImageSprite | A este elemento se le asociará el segundo topo |
| Elementos de tipo layaut, ‘HorizontalArragement’ | Para poder incluir los nuevos botones en una misma línea |

La pantalla debería quedar:

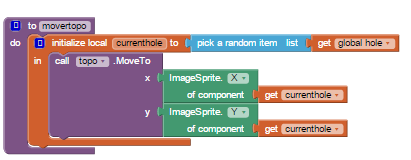


* Una vez diseñada la pantalla, volvemos a los bloques.
* Al igual que en el anterior juego, lo primero que haremos será definir una variable global tipo lista llamada holes, que nos será de utilidad para el tratamiento de los hoyos.

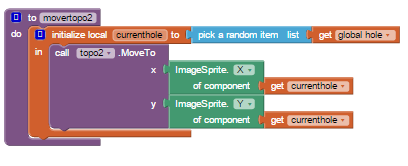


Y a continuación inicializaremos todos los elementos de la pantalla dentro del evento de inicio de pantalla. Igual que la anterior versión pero con más elementos en la tabla.

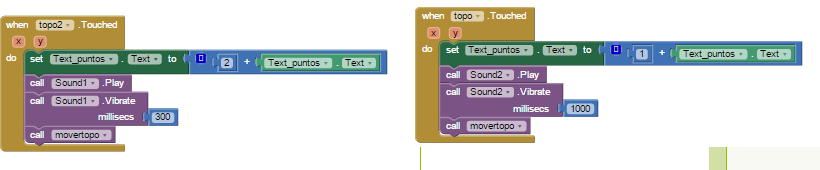
* Procedimiento ‘MoveMole’, no cambiar



Pero se define otro procedimiento igual pero con el nombre movertopo2



* Los eventos de tocar topo tendrán la variante para que asociar un sonido cada vez que se toque el topo.



En el bloque del topo 2, se puede observar que la puntuación se incrementará en 2 si el topo es tocado.

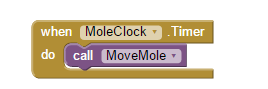
* Un nuevo evento sería el de iniciar la puntuación, que se puede observar en el siguiente bloque:



* Y en el siguiente bloque, cumpliría con la funcionalidad de volver al juego anterior:



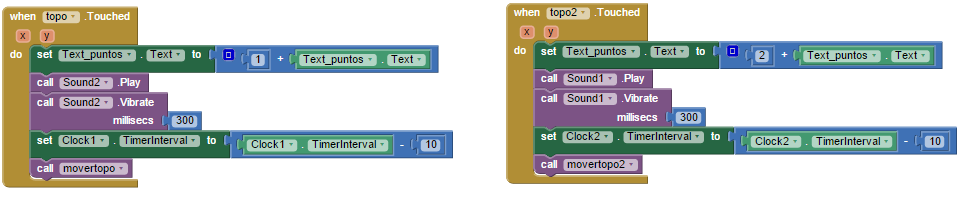
* El siguiente evento se utiliza para mover el topo cada cierto tiempo. Este tiempo son los milisegundos definidos cuando creamos el componente MoleClock.



* Incluir más velocidad en el movimiento de los topos cada vez que se aumentan los puntos.

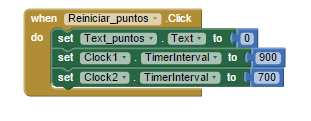
Insertaremos el siguiente bloque en los bloques generales de toque de los topos:ç





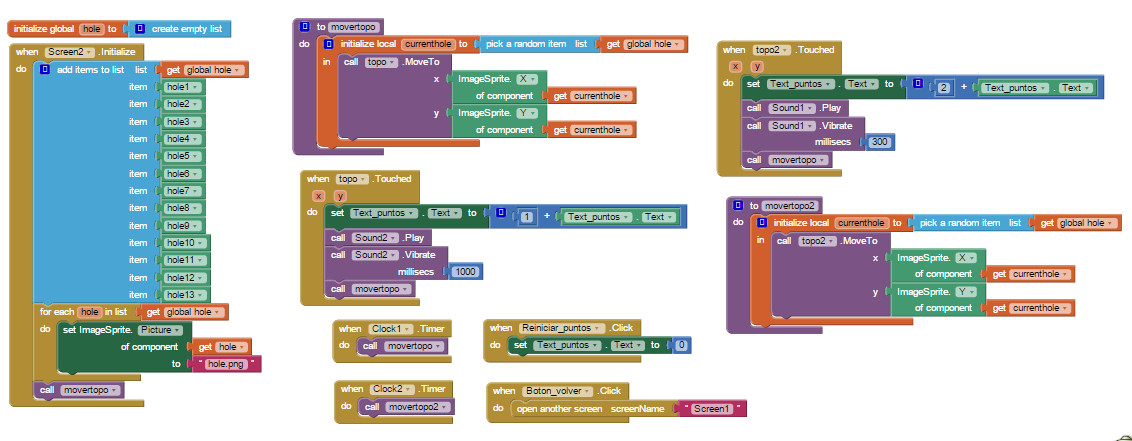
De esta forma, cada vez que se toque un topo, el tiempo de espera se irá reduciendo lo que provocará que el topo se mueva más rápidamente.

* Iniciar el marcador y la velocidad de los topos.



Ponemos el tiempo de retardo del inicio

El aspecto final del puzle sería el siguiente (en la siguiente imagen faltaría el bloque de actualización de la velocidad):



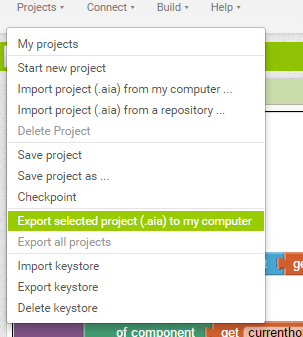
Ejecutamos el emulador y nos debería aparecer la siguiente ventana dentro del móvil virtual:



Se realizan pruebas de navegación entre ambos juegos, toque de topos, aumento de puntuación, sonidos, etc y todo funciona correctamente.

Para finalizar:

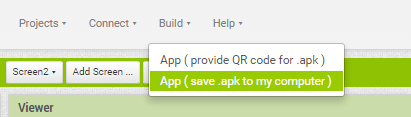
* Exportamos el proyecto para tener una copia en local.

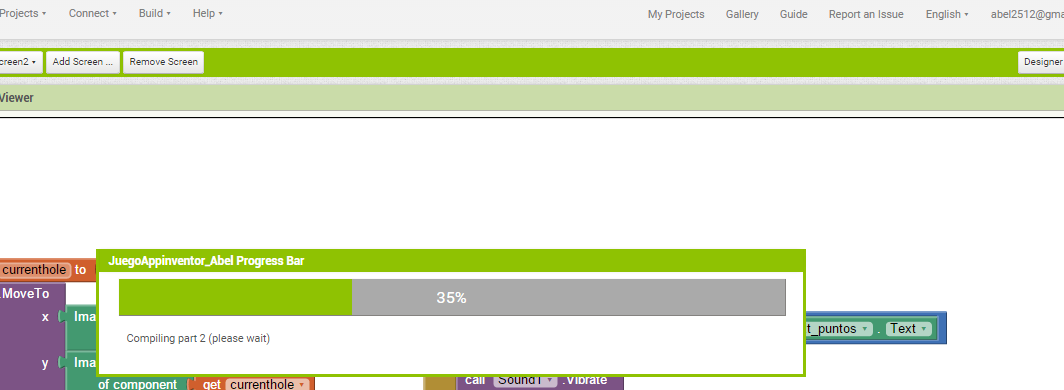


Descargará un archivo con formato nombre del proyecto punto aia:



* Instalar la aplicación en un dispositivo móvil. Para este paso, nos crearemos un archivo .apk el cual después lo copiaremos al dispositivo móvil lo ejecutaremos para su instalación.





1. Desarrollo del juego TOPIN

**8.1 Introducción.**

El juego TOPIN, está basado en un juego clásico llamado MOLE que consiste en un Topo que va apareciendo aleatoriamente por unos escenarios donde existen agujeros y el jugador debe tocar el muñeco del Topo antes de que desaparezca. Si logra tocar el Topo antes de que desaparezca, se incrementará su puntuación.

En este juego se incluirán una serie de mejoras al juego oriinal para aumentar su jugabilidad y su atractivo para el usuario final. Las mejoras más relevantes son las siguientes:

* Menú principal con distintas opciones de navegación.
* Inclusión de escenarios de juego diferentes.
* Nivel de dificultad progresivo.
* Añadir más personales en el juego.
* Incluir puntuación.
* Inclusión de vidas.
* Tiempo máximo para pasar de niveles.
* Registro de puntuaciones máximas.
* Inclusión de sonidos tipo ‘beep’ para que suene y vibre cada vez que se tocan los topos.
* Entorno visual mejorado

Para el desarrollo del juego, se va a seguir una metodología clásica de desarrollo en cascada o por niveles, en las que se van a establecer los siguientes niveles:

* Incluyo gestión de riesgos??? O no hace falta
* Análisis de requisitos.
* Diseño, construcción y pruebas unitarias.
* Plan de pruebas.
* Implantación

Al usar una metodología en cascada, no se puede iniciar una fase hasta que no finalice la anterior.

**8.2 Análisis de requisitos.**

El presente apartado contiene los requisitos definidos para el desarrollo del juego y los deducidos a partir de los mismos tras un análisis realizado para establecer una solución de desarrollo.

|  |  |
| --- | --- |
| Identificación  Requerimiento | Descripción del requerimiento |
| REQ-0001 | Menú principal con distintas opciones de navegación |
| REQ-0002 | Inclusión de escenarios de juego diferentes. |
| REQ-0003 | Nivel de dificultad progresivo |
| REQ-0004 | Añadir más personales en el juego. |
| REQ-0005 | Incluir puntuación. |
| REQ-0006 | Inclusión de vidas |
| REQ-0007 | Tiempo máximo para pasar de niveles. |
| REQ-0008 | Registro de puntuaciones máximas. |
| REQ-0009 | Inclusión de sonidos tipo ‘beep’ para que suene y vibre cada vez que se tocan los topos. |
| REQ-0010 | Soporte SO android distintas versiones. |
| REQ-0011 | Compatibilidad con distintos dispositivos |
| REQ-0012 | Entorno visual |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0001 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Menú principal con distintas opciones de navegación | | | |
| **Descripción detallada** | Incluir un menú principal al iniciarse el juego para qué el jugador pueda elegir entre una de las siguientes opciones:   * Juego * Puntuaciones máximas * Ajustes * Ayuda * Enlaces de interés | | | |
| **Criterio de aceptación** | Revisión de pruebas unitarias | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0002 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Inclusión de escenarios de juego diferentes. | | | |
| **Descripción detallada** | El juego debe de disponer de distintos escenarios o niveles de juego, a los cuales se podrá ir accediendo si se superan los escenarios anteriores. A esto se le denomina ’Pasar de nivel’ o ‘Pasar de pantalla’.  Para pasar de nivel, se establecerá un tiempo máximo y una puntuación mínima. Si no se logra la puntuación mínima en el tiempo máximo, no se logrará progresar de nivel y se le quitará una vida al jugador.  Si al jugador le quedan vidas, dispondrá de otro intento para superar el nivel y si se queda sin vidas, el juego finalizará ‘GAME OVER’ | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0003 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Nivel de dificultad progresivo. | | | |
| **Descripción detallada** | El juego cada vez debe ser más complicado, para que al jugador le sea más difícil pasar al siguiente nivel. Para lograr este objetivo, los topos que se deben tocar, cada vez se moverán más rápido y el tiempo para conseguir la puntuación necesaria se irá reduciendo. | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0004 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Añadir más personales en el juego. | | | |
| **Descripción detallada** | Según vaya aumentando la dificultad y los niveles, se incrementarán el número de personajes en el juego, para dificultar más el obtener puntuación. | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0005 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Incluir puntuación. | | | |
| **Descripción detallada** | Se incluye un sistema de puntuación. Cuando se toque un topo, se incrementará esta puntuación, si se toca un pincho se decrementará la puntuación. | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0006 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Inclusión de vidas. | | | |
| **Descripción detallada** | El jugador dispondrá de 3 vidas para poder ir superando los distintos niveles. Cuando se le terminen las vidas, el jugo finalizará. | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0007 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Tiempo máximo para pasar de niveles. | | | |
| **Descripción detallada** | Para cada nivel, se establecerá un tiempo máximo. Definido en el REQ-0002 ya que está relacionado con este requisito. | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0008 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Registro de puntuaciones máximas. | | | |
| **Descripción detallada** | Los mejores resultados, se quedarán registrados y podrán ser consultados.. | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0009 | | **Tipo:** Requisito funcional | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Inclusión de sonidos tipo ‘beep’ para que suene y vibre cada vez que se tocan los topos | | | |
| **Descripción detallada** | Inclusión de sonidos tipo ‘beep’ para que suene y vibre cada vez que se tocan los topos | | | |
| **Criterio de aceptación** | Revisión de pruebas | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0010 | | **Tipo:** Requisito técnico | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Soporte SO android distintas versiones. | | | |
| **Descripción detallada** | El juego será desarrollado para dispositivos móviles con SO Android con amplio abanico de versiones. | | | |
| **Criterio de aceptación** | Realizar pruebas en distintos dispositivos móviles con versiones de Android diferentes. | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0011 | | **Tipo:** Requisito técnico | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Compatibilidad con distintos dispositivos. | | | |
| **Descripción detallada** | Compatibilidad con tablets y móviles | | | |
| **Criterio de aceptación** | Instalar y probar el juego en dispositivos móviles y en tablets | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0012 | | **Tipo:** Requisito técnico | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Compatibilidad con tablets y móviles | | | |
| **Descripción detallada** | Compatibilidad con tablets y móviles | | | |
| **Criterio de aceptación** | Instalar y probar el juego en dispositivos móviles y en tablets | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID Requisito:** REQ-0012 | | **Tipo:** Requisito técnico | | |
| **Estado** | Nuevo | **Fecha de alta** | **Última revisión** | |
| 01-12-2015 |  | |
| **Breve descripción** | Entorno visual | | | |
| **Descripción detallada** | Crear un entorno visual llamativo | | | |
| **Criterio de aceptación** | Solicitar opinión de usuarios | | | |
| **Origen e Historia** | Descripción | | | Fecha |
| Añadido | | | 01-12-2015 |
| Aceptada | | |  |

**8.3 Diseño, construcción y pruebas unitarias.**

**DISEÑO**

Diseño de navegación del juego

La navegación entre las distintas pantallas del juego será la siguiente;

MENÚ PRINCIPAL

ENLACES

AJUSTES

AYUDA

JUEGO

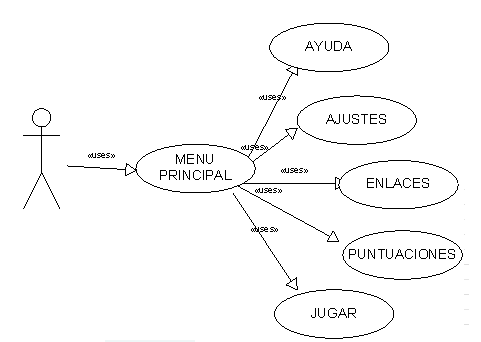
PUNTUACIÓN

NIVEL n

NIVEL 1

. . .

Casos de uso



Descripción de los casos de uso

Caso de Uso 1: Op. Menú principal.

|  |  |
| --- | --- |
| Características | |
| Resumen | Pantalla principal del juego. Desde está pantalla, se podrá acceder a todas las funciones del juego. |
| Pre-Condiciones | Acceso al juego |
| Post-Condiciones | Ninguna |
| Actor Principal | Jugador |
| Actores Secundarios | Jugador |
| **Diseño** |  |
| Elementos | Imagen jpg,, |

|  |  |  |
| --- | --- | --- |
| Curso normal | | |
| **Paso** | **Actor** | Descripción |
| Ejecución del juego | Jugador | Se pulsa en el icono del juego y se accederá a está pantalla |
| Salir del juego | Jugador | Al pulsar la tecla de ‘volver atrás’ del dispositivo, se finalizará el juego |
| Opciones | Jugador | Al pulsar sobre alguna de las opciones de la pantalla, se accederá a esa opción. |

Caso de Uso 2: Op. Ayuda.

|  |  |
| --- | --- |
| Características | |
| Resumen | Breve descripción de las distintas opciones del juego |
| Pre-Condiciones | Pulsar opción de ‘Ayuda’ dentro del menú principal |
| Post-Condiciones | Ninguna |
| Actor Principal | Jugador |
| Actores Secundarios | Jugador |
| **Diseño** |  |
| Elementos | Imagen |

|  |  |  |
| --- | --- | --- |
| Curso normal | | |
| **Paso** | **Actor** | Descripción |
| Selección | Jugador | Se pulsa la opción del menú principal correspondiente. |
| Volver | Jugador | Volver al menú principal. Pulsando la tecla de volver atrás |

Caso de Uso 3: Op. Enlaces.

|  |  |
| --- | --- |
| Características | |
| Resumen | Inclusión de distintos enlaces |
| Pre-Condiciones | Pulsar opción de ‘Enlaces’ dentro del menú principal |
| Post-Condiciones | Ninguna |
| Actor Principal | Jugador |
| Actores Secundarios | Jugador |
| **Diseño** |  |
| Elementos | Imagen |

|  |  |  |
| --- | --- | --- |
| Curso normal | | |
| **Paso** | **Actor** | Descripción |
| Selección | Jugador | Se pulsa la opción del menú principal correspondiente. |
| Volver | Jugador | Volver al menú principal. Pulsando la tecla de volver atrás |

Caso de Uso 4: Op. Puntuaciones.

|  |  |
| --- | --- |
| Características | |
| Resumen | Visualización de las puntuaciones máximas |
| Pre-Condiciones | Pulsar opción de ‘Puntuaciones’ dentro del menú principal |
| Post-Condiciones | Ninguna |
| Actor Principal | Jugador |
| Actores Secundarios | Jugador |
| **Diseño** |  |
| Elementos | Imagen |

|  |  |  |
| --- | --- | --- |
| Curso normal | | |
| **Paso** | **Actor** | Descripción |
| Selección | Jugador | Se pulsa la opción del menú principal correspondiente. |
| Volver | Jugador | Volver al menú principal. Pulsando la tecla de volver atrás |

Caso de Uso 5: Op. Juego.

|  |  |
| --- | --- |
| Características | |
| Resumen | Función principal del juego. En está pantalla el jugador podrá iniciar un nuevo juego |
| Pre-Condiciones | Pulsar opción de ‘Juego’ dentro del menú principal |
| Post-Condiciones | Ninguna |
| Actor Principal | Jugador |
| Actores Secundarios | Jugador |
| **Diseño** |  |
| Elementos | Imagen |

|  |  |  |
| --- | --- | --- |
| Curso normal | | |
| **Paso** | **Actor** | Descripción |
| Selección | Jugador | Se pulsa la opción del menú principal correspondiente. |
| **Volver** | **Jugador** | Volver al menú principal. Pulsando la tecla de volver atrás |
| Jugar | Jugador | Interactuar con la pantalla para jugar. |

CONSTRUCCIÓN

PRUEBAS UNITARIAS

Definición de las distintas pantallas

**8.4 Plan de pruebas.**

**Introducción a la Planificación de las pruebas**

En este apartado, se va a definir una estrategia para definir los casos de prueba necesarios para validar el software desarrollado. Para conseguir dicho propósito se va a seguir la siguiente metodología de definición de pruebas, cuyas etapas son las siguientes:

**Planificación y preparación 🡪 Ejecución 🡪 Análisis**

Medidas de seguridad

Planeación y Preparación de las prueba:

Establecer objetivos

Modelo construcción

Casos de prueba

Procedimientos

En*trada*

Casos de prueba

Ejecución

Retroalimentación y ajustes

Si

Defectos encontrados

Análisis de resultados ok

Análisis y seguimiento

* Planificación y preparación. Fase en la sé que tomarán las decisiones más importantes de la prueba. Objetivos y metas generales, La planificación estará descrita en el apartado 3 y la preparación de las pruebas en el apartado 4.
* Ejecución. Actividades (definido en el apartado 4):
  + La asignación de tiempo y recursos de prueba.
  + La invocación y la ejecución de pruebas, y la recolección de información de ejecución y medidas.
  + Comprobación de resultados de las pruebas y la identificación de los fallos del sistema.
* Análisis. Una vez realizadas las pruebas, se deberá realizar un análisis de los resultados para determinar si son correctas o no.

**Escenarios de las pruebas.**

Podemos imaginar varios escenarios, los cuales determinarán los tipos de prueba que debemos utilizar:

* Un escenario donde se vaya a implantar este software que se debe validar y que sustituye a otro, de esta manera nos puede dar ‘juego’ para poder realizar pruebas alpha, beta y de aceptación en el cliente, que serán los SMARTPHONE de los usuarios.
* Otro escenario en el que no queramos que los usuarios realicen ningún tipo de prueba. En este caso se utilizarán pruebas de cobertura. Para este caso se utilizará el simulador de App Inventor.

Pruebas beta. Iteraciones continuas Liberación controlada de software de manera que estos clientes beta ayudan desarrollo de software y organizaciones a mejorar su calidad de software (aceptación de los usuarios).

Las pruebas alfa son pruebas operativas simuladas o reales de los posibles usuarios / clientes o un equipo de prueba independiente en el sitio de los desarrolladores. Las pruebas alfa se emplea a menudo para el software off-the-shelf como una forma de pruebas de aceptación interna, antes de que el software va a la prueba beta (aceptación interna).

**Planificación de las pruebas.**

Para poder definir los casos de prueba, previamente habría que definir las requisitos/funcionalidades que deben ser objeto de la prueba. Estos requisitos deben ser los mismos definidos en el apartado de ‘análisis de requisitos’ y los recordamos:

|  |  |
| --- | --- |
| Identificación  Requerimiento | Descripción del requerimiento |
| REQ-0001 | Menú principal con distintas opciones de navegación |
| REQ-0002 | Inclusión de escenarios de juego diferentes. |
| REQ-0003 | Nivel de dificultad progresivo |
| REQ-0004 | Añadir más personales en el juego. |
| REQ-0005 | Incluir puntuación. |
| REQ-0006 | Inclusión de vidas |
| REQ-0007 | Tiempo máximo para pasar de niveles. |
| REQ-0008 | Registro de puntuaciones máximas. |
| REQ-0009 | Inclusión de sonidos tipo ‘beep’ para que suene y vibre cada vez que se tocan los topos. |
| REQ-0010 | Soporte SO android distintas versiones. |
| REQ-0011 | Compatibilidad con distintos dispositivos |
| REQ-0012 | Entorno visual |

Para poder realizar las pruebas funcionales, se deben localizar los elementos que van a intervenir en las pruebas, los puntos de entrada y los de salida, para este caso tenemos los siguientes, con la siguiente estructura:

DATOS DE ENTRADA 🡪 PROCESAMIENTO 🡪 SALIDA

Posibles casos 🡪

Posibles casos :

* Caso 1 FUNC-001. Número romano 🡪 procesamiento 🡪 número decimal.
* Caso 2 FUNC-002 . Número decimal de entrada 🡪 procesamiento 🡪 número romano

Una vez analizados todos los elementos del sistema, la siguiente decisión, es determinar cuántas pruebas se deben realizar para poder obtener la aceptación y poder implantar el software en producción.

Para esta decisión es importante no tener limitaciones de tiempo ni limitaciones de presupuesto, ya que si las tuviéramos, la calidad de las pruebas iría en decremento según estas limitaciones.

Pero aunque tengamos tiempo ilimitado y presupuesto ilimitado, muy pocas veces se pueden probar todas las variantes de un software. En este ejemplo, no podríamos probar todos los casos, ya que de entrada el conjunto de números positivos o romanos son infinitos.

Para poder resolver los problemas expuestos en los párrafos anteriores y determinar el juego de pruebas adecuados para dar el VB al software, se suelen utilizar dos tipos de prueba:

* Medir la calidad directamente a través de la fiabilidad del software. La cuestión es que este enfoque requiere el uso real de los clientes. (pruebas de uso). En este caso se podrían utilizar los tipos de prueba beta y alfa, pero no afectan a nuestro estudio.
* Medir la calidad indirectamente a través de la cobertura de la prueba. La mayoría de las técnicas tradicionales de prueba, ya sea estructurales o funcionales, utilizan diversas formas de cobertura de la prueba como los criterios de detención. Esto significa que el proceso de prueba se detiene cuando un conjunto de pruebas se ejecutan con éxito en el software. En este caso, el aspecto clave es identificar cual es la cobertura de la prueba requerida.

Recapitulando, las pruebas que se van a utilizar son pruebas de caja negra o funcionales (la pruebas de caja blanca se han realizado en las pruebas unitaria) y para dar el OK a las pruebas, se van a utilizar pruebas de tipo de cobertura.

Para determinar los siguientes dominios y subdominios, hemos tenido en cuenta los valores de entrada y la salida esperada:

* Validación de Requisitos. Validación de los requisitos definidos. Definidos con CASO-V-nnn
* Fallas. Casos de prueba con identificación CASO-F-nnn.
  + Reglas (fallas las mismas pero que fallen).
  + Enteros (incorrectos pero fuera de limintes).
  + Subdominos romanos Baltasar.
* Límites. Casos de prueba con identificación CASO-L-nnn.
  + Enteros.
  + Romanos

**Casos de prueba.**

Para cada caso de prueba, se debería definir una especie de tabla para identificar la prueba, los resultados esperados, la duración en preparar la prueba, etc. Podría ser algo así:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** | Identificación del caso de prueba | | **Responsable:** | Responsable o responsables de realizar la prueba |
| **Requisito/**  **Funcionalidad** | Funcionalidad que se va a validar | | | |
| **Descripción** | Breve descripción de la prueba | | | |
| **Datos de entrada** | Datos de entrada para el caso de prueba | | | |
| **Datos de salida esperados** | Resultados esperados | | | |
| **Análisis de los resultados** |  | | | |
| **Historia** | Fecha | Estado y duración en horas | | |
|  | Definición de la prueba | | |
|  | Preparación de la prueba | | |
|  | Finalizada la prueba | | |

**Pruebas de validación de requisitos**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** | CASO-V-001 | | **Responsable:** | Abel García |
| **Requisito/ funcionalidad** | REQ-0001 **Menú principal con distintas opciones de navegación** | | | |
| **Descripción** | Probar que la primera pantalla que aparece cuando se inicia el juego es la del Menú principal y que las opciones disponibles son accesibles cuando se pulsa en ellas.  Probar que cuando se pulsa la tecla de volver, se finaliza el juego | | | |
| **Datos de entrada** | Pulso en la pantalla | | | |
| **Datos de salida esperados** | Si se pulsa volver o salir, se finalizará el juego. Si pulsamos en una opción, se accederá a esa opción. | | | |
| **Análisis de los resultados** | Revisión de la navegación correcta entre las pantallas | | | |
| **Historia** | Fecha | Estado y duración en horas | | |
| 03/12/2015 | Definición de la prueba 1h | | |
| 20/12/2015 | Preparación de la prueba 1h | | |
| pendiente | Finalizada la prueba | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** | CASO-V-002 | | **Responsable:** | Abel García |
| **Requisito/ funcionalidad** | REQ-0002 **Inclusión de escenarios de juego diferentes.** | | | |
| **Descripción** | Comprobar que pasa de nivel cuando cumple con los requisitos. | | | |
| **Datos de entrada** | Jugar en el juego hasta conseguir los objetivos del nivel | | | |
| **Datos de salida esperados** | Pasar de nivel | | | |
| **Análisis de los resultados** | Revisión de la navegación correcta entre las pantallas | | | |
| **Historia** | Fecha | Estado y duración en horas | | |
| 03/12/2015 | Definición de la prueba 1h | | |
| 20/12/2015 | Preparación de la prueba 1h | | |
| pendiente | Finalizada la prueba | | |

ETC ETC <DEFINIR RESTO DE CASOS DE REQUERIMIENTOS>

**Pruebas de fallas.**

Definir casos de pruebas en los que el software debe emitir un error, ya que los datos introducidos no son correctos.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** | CASO-F-0001 | | **Responsable:** | Abel |
| **Funcionalidad/**  **Requisitos** | Probar la navegación entre todas las pantallas | | | |
| **Descripción** | Probar la navegación entre todas las pantallas intentando tocar por todas las zonas buscando vulnerabilidades | | | |
| **Datos de entrada** | Pulsar por todas las zonas | | | |
| **Datos de salida esperados** | No causar reacciones no controladas | | | |
| **Análisis de los resultados** | Revisar navegación correcta | | | |
| **Historia** | Fecha | Estado y duración en horas | | |
| 03/12/2015 | Definición de la prueba 1h | | |
| 20/12/2015 | Preparación de la prueba 1h | | |
| pendiente | Finalizada la prueba | | |

ETC ETC <DEFINIR RESTO DE CASOS PARA BUSCAR FALLOS

**Pruebas de estrés.**

Incluir unas pruebas en las que se intente buscar errores en el sistema, como por ejemplo incluir datos cuando aún se están procesando otros, etc.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID:** | CASO-F-0001 | | **Responsable:** | Abel |
| **Funcionalidad/**  **Requisitos** | Probar la navegación entre todas las pantallas | | | |
| **Descripción** | Probar la navegación entre todas las pantallas intentando tocar por todas las zonas buscando vulnerabilidades | | | |
| **Datos de entrada** | Pulsar por todas las zonas | | | |
| **Datos de salida esperados** | No causar reacciones no controladas | | | |
| **Análisis de los resultados** | Revisar navegación correcta | | | |
| **Historia** | Fecha | Estado y duración en horas | | |
| 03/12/2015 | Definición de la prueba 1h | | |
| 20/12/2015 | Preparación de la prueba 1h | | |
| pendiente | Finalizada la prueba | | |

ETC ETC <DEFINIR MAS PRUEBAS DE ESTRÉS>

**8.5 Implantación**

<Incluir los pasos que se deben seguir para realizar la siguientes implantaciones>

* Simulación del juego
* Crear ejecutable del juego para instalar en dispositivos móviles
* Subir a la plataforma de market para que se pueda descargar el juego.
* Instalar en dispositivo móvil, como Smartphone, con distintas versiones
* Instalar en tablet.

1. *Resultados*

“¿Qué datos/productos se han obtenido al

finalizar el proyecto?”

Resultados de simulaciones y procesos del

Proyecto. Se deben reflejar las pruebas y

tests realizados, errores detectados…

Artefactos producidos.

Desviaciones de la metodología, acciones

correctivas, riesgos eliminados o mitigados a

lo largo de la vida del proyecto.

1. Conclusiones.

**10.1 Ventajas y desventajas.**

App inventor es una herramienta de desarrollo de aplicaciones para dispositivos móviles que tiene las siguientes ventajas:

* Una de las principales ventajas es: Aplicación gratuita (Software libre).
* No es necesario instalación IDE en el PC donde se quiera utilizar. La ejecución es directamente en página WEB.
* Es necesario tener conexión a internet, pero si no disponemos de conexión a internet, también existe una versión para instalar en el equipo.
* La programación está basa en bloques y es bastante intuitiva y se requieren conocimientos mínimos (inclusive ninguno) para empezar a desarrollar aplicaciones sencillas.
* El almacenamiento se realiza en la nube, por lo que podremos acceder a nuestros desarrollos desde cualquier lugar.
* Dispone de un simulador muy sencillo para poder probar las aplicaciones en el pc de desarrollo.
* Exportar la aplicación para poder instalar en el dispositivo móvil es una tarea sencilla.
* Se puede utilizar en las empresas de desarrollo para realizar prototipos de una forma ágil y rápida.

También existen una serie de desventajas, las cuales se describen en los siguientes puntos:

* Las aplicaciones no se pueden subir a Android Market.
* El tamaño de los archivos APk es elevado.
* No permite aplicaciones muy complejas aunque si completas. Según hemos comentado en los beneficios, los desarrollos realizados con App Inventor están más enfocados a pequeñas aplicaciones o a prototipos.
* El editor del entorno de bloques tiene muy pocas posibilidades de iteración, como copiar y pegar en otras ventanas con las que se está trabajando.

**10.2 Integración de App Inventor en la enseñanza.**

Con la entrada de la nueva ley ‘LOMCE’, en el curso 2015-2016, se incluye un nuevo temario de ‘Tecnologías de la Información’ en Bachillerato, en la cual además de analizar otros puntos de la informática (internet. Bases de datos, etc), existe un apartado de programación gráfica por bloques, en la que se utilizará Scratch como herramienta para su aprendizaje. Scratch es un lenguaje de programación basado en bloques desarrollado por el MIT y orientado al desarrollo de programas para niños.

Existen otros documentos para futuras actualizaciones de la materia de ‘Tecnologías de la información’, donde se profundiza más en la programación basada en bloques y donde se proponen los siguientes puntos de aprendizaje, los cuales podrían ser implementados en un futuro:

* Utilizar con destreza un entorno de programación gráfica por bloques
  + Describe el proceso de desarrollo de una animación o un juego y enumera las fases principales de su desarrollo.
  + Emplea, con facilidad, las diferentes herramientas básicas del entorno de programación.
  + Sitúa y mueve objetos en una dirección dada.
  + Inicia y detiene la ejecución de un programa.
  + Modifica, mediante la edición, la apariencia de objetos. Crea nuevos objetos: actores, fondos y sonidos.
  + Maneja, con soltura, los principales grupos de bloques del entorno.
  + Utiliza, con facilidad, los comandos de control de ejecución: condicionales y bucles.
  + Emplea de manera adecuada variables y listas.
  + Usa, con soltura, la interacción entre los elementos de un programa.
  + Analiza el funcionamiento de un programa a partir de sus bloques.
  + Identifica y considera las implicaciones del “diseño para todos” para los programas que realiza.
* Desarrollar y programar aplicaciones móviles sencillas en entornos de programación por bloques
  + Describe el proceso de diseño de una aplicación para móviles y las fases principales de su desarrollo.
  + Utiliza con precisión las diferentes herramientas del entorno de desarrollo.
  + Distingue los diferentes tipos de datos y sus formas de presentación y almacenamiento.
  + Clasifica los objetos disponibles, sus métodos y eventos.
  + Identifica las posibilidades de interacción con los sensores de los que dispone un terminal móvil.
  + Reconoce y evalúa las implicaciones del “diseño para todos” para los programas que realiza.
  + Desarrolla aplicaciones informáticas para su ejecución en dispositivos móviles utilizando diferentes sensores y elementos de interfaz.
  + Describe las características y normas de publicación de diferentes plataformas para la publicación de aplicaciones móviles.

Como se puede observar en los puntos anteriores, la herramienta de App Inventor por sus características, cumple con todos los requisitos y podría ser una clara candidata para ser usada en la asignatura de ‘Tecnologías’ y sustituir a Scratch ya que esta última no cumpliría con todos los requerimientos descritos en los puntos anteriores.

ANEXOS

**ENLACES DE INTERES.**

* Página programa ejemplos

<http://www.iesromerovargas.es/android/>

* Videos appinventor1 y appinventor2 (primeros pasos, tutoriales ejemplos, está muy bien esta web)

<https://sites.google.com/site/appinventormegusta/novedades-en-app-inventor-2>

<http://www.tuappinvetorandroid.com/2014/06/30/tutorial-componente-sharing-de-app-inventor-2/>

* Descarga del emulador de móvil para windows **aiStarter.exe:**

<http://appinv.us/aisetup_windows>

* Acceso a la herramienta App inventor:

<http://ai2.appinventor.mit.edu/>

ANEXOS