



UNIVERSITAT  
POLITÈCNICA  
DE VALÈNCIA

**International ICT Week**  
**July 8-12, 2024**

# **Mobile Apps for Android**

## **with MIT App Inventor 2**

**Fast visual development with  
media, GPS and databases**

# Objectives

## Specific Objectives

---

Knowing a tool for development of mobile applications, as well as the basic techniques that close the system cycle

---

MIT App Inventor: Very fast learning curve, but robust enough for real projects

---

Ideal for rapid application development without worrying so much about code syntax (similar to Scratch)

---

Possibility of carrying out a professional appearance app in the (limited) time that we have for this one week course

---



Possibility of distributing our apps through Google Play

# Distribution of sessions

## MIT App Inventor



### Session 1

Platforms and development environments. Justification of platform and environment chosen. Presentation projects. Server presenter. Introduction MIT App Inventor



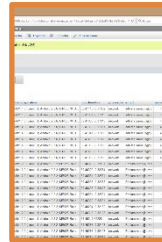
### Session 2

Programming with the Block Editor. Working with several screens. Components Images, audio and video. Camera. Sensors: GPS, accelerometer ...



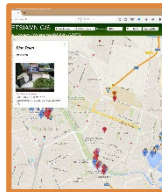
### Session 3

Storage. Local databases Connectivity. Going further: Extensions.



### Session 4

Working with servers: Webservices. Technology needed to interact with server. Collecting and storing information. Collection of basic scripts

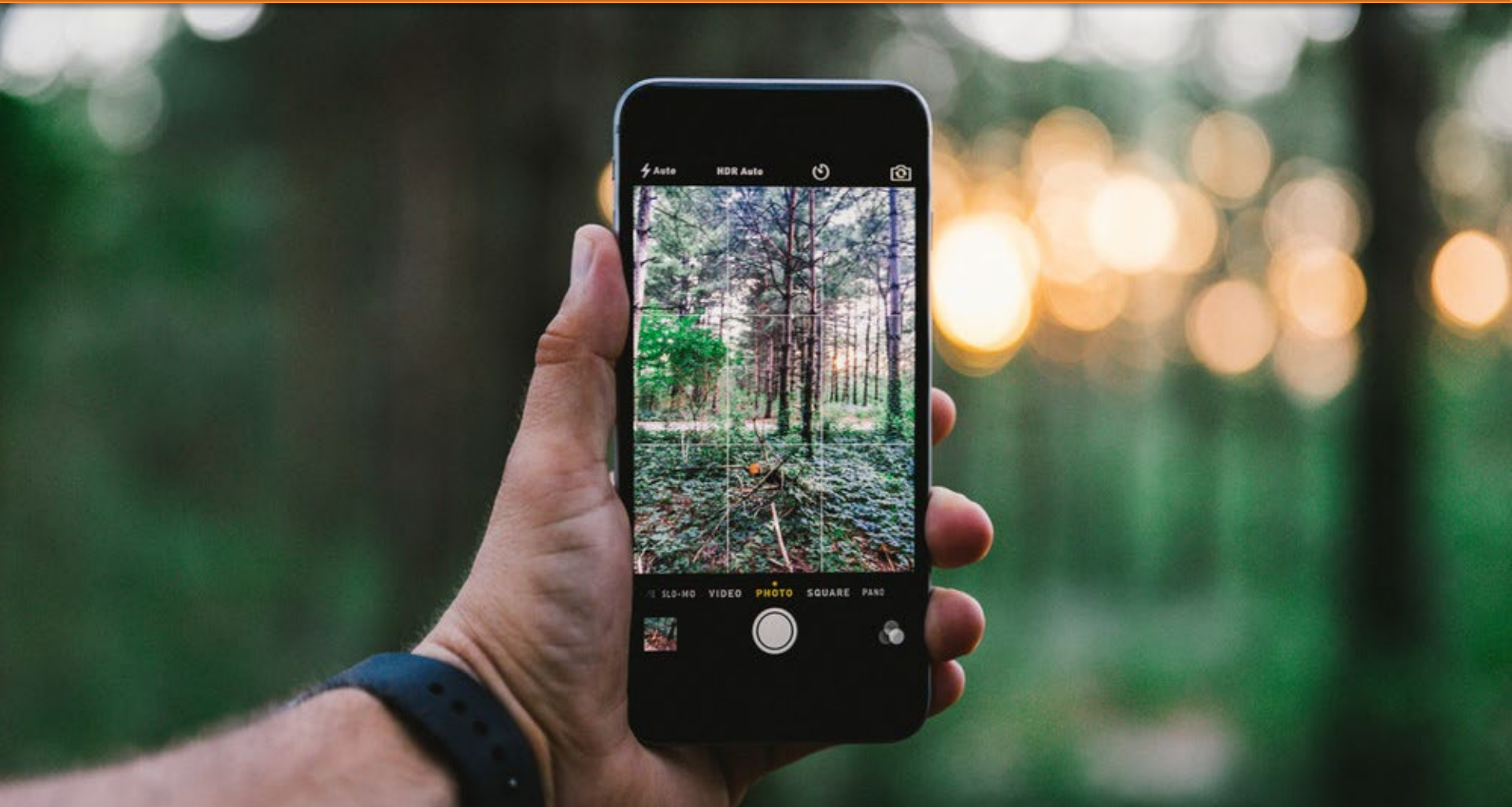


### Session 5

Representation of server information. Sharing data. Sharing on social networks. Distribute our App. Publish our App on Google Play.

# Session 1

## Introduction



# The fantastic potential of today's Smartphones

- Powerful multi-core processors
- Large amount of RAM memory
- Huge storage capacity
- Photo camera (more than one)
- Always connected to internet
- Bluetooth connectivity, NFC, GSM, SMS
- Positioning: GPS, Galileo, Glonass, BeiDou
- Proximity sensors, inclination, gyroscope, steps, barometric, temperature, light intensity, microphone, magnetic fields, fingerprint reader, barcode reader ...
- Speech recognition, flashlight ...

And always in continuous improvement



...and we carry with them 24h with us!



# How far can we go?

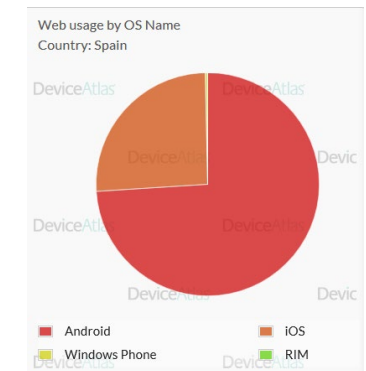
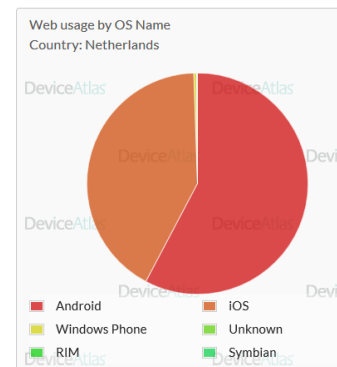
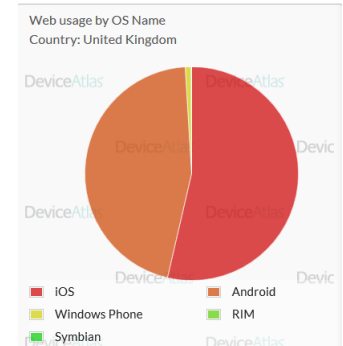
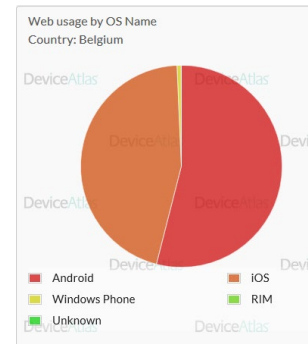
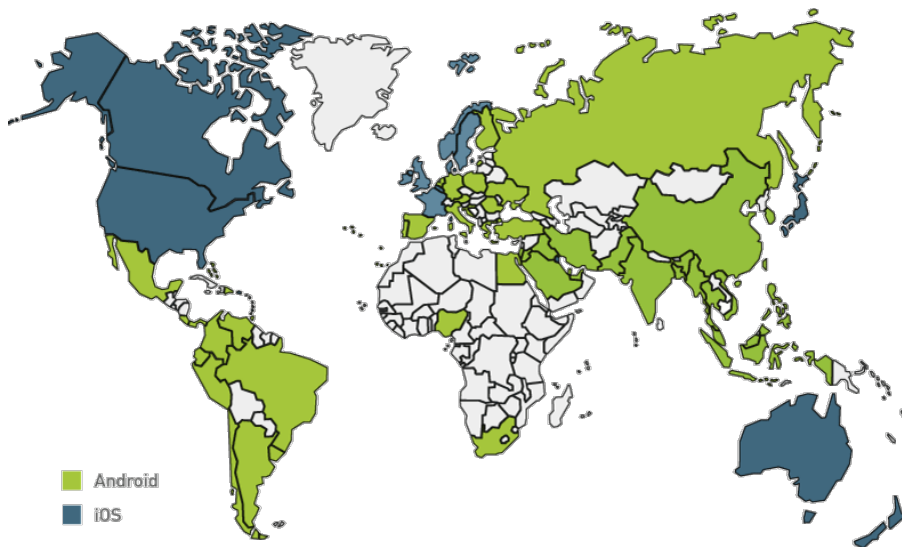
The image is a composite showing the CultiApp web interface, a smartphone displaying the app, and a map with colored markers. The web interface at the top features the CultiApp 1.0 logo, filters for date (2014-09-01 to 2015-12-01), crop type (Todos los cultivos), infrastructure (Todas las infraestructuras), ailment (Todas las afecciones), and sample type (Todos los muestreos). It shows 614 results. The smartphone in the center displays the app's main form with fields for Cultivo (Naranja), Infraestructura (Infr. no identificada), Afección (Afección no identificada), Auxiliar (Aux. no identificado), and Método (Difusor confusión sexual). It also includes a photo upload section and buttons for sending data online or offline. The background map on the right shows a grid of agricultural fields with numerous colored circular markers (yellow, orange, red, green) indicating data points. Labels on the map include 'Camino Calvario', 'Camino Vialesa', and 'CV-304'.

With the support  
of good web  
services ...  
As far as our  
imagination goes!

# Smartphones and Tablets Platforms



- Google Android
- Apple iOS



# Mobile development environments

- **Native languages and tools** from each platform:

1. Objective-C, Swift and XCode in iOS
2. Java and Android Studio in Android

- **Multiplatform tools that can compile to native code.**

Like Xamarin Studio or Flutter (from Google, Dart language)

- **Multiplatform tools based on HTML5.**

Like PhoneGap, React Native (RN), Ionic or Apache Cordova.





# MIT App Inventor

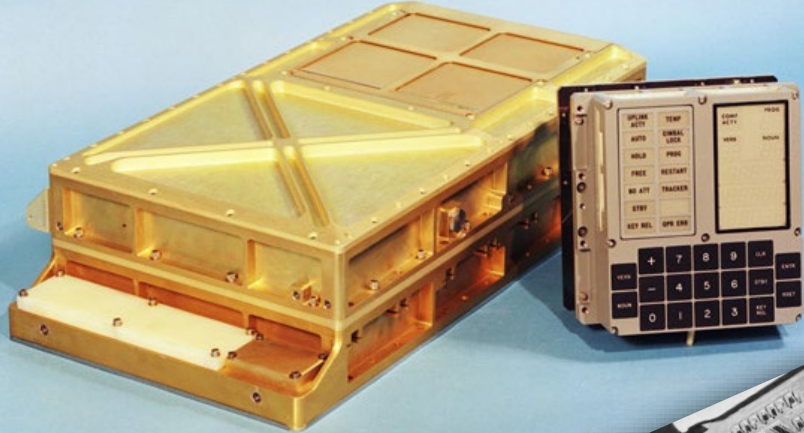


**Tool with which we will try to squeeze  
all that potential**

**Let's see why that tool...**

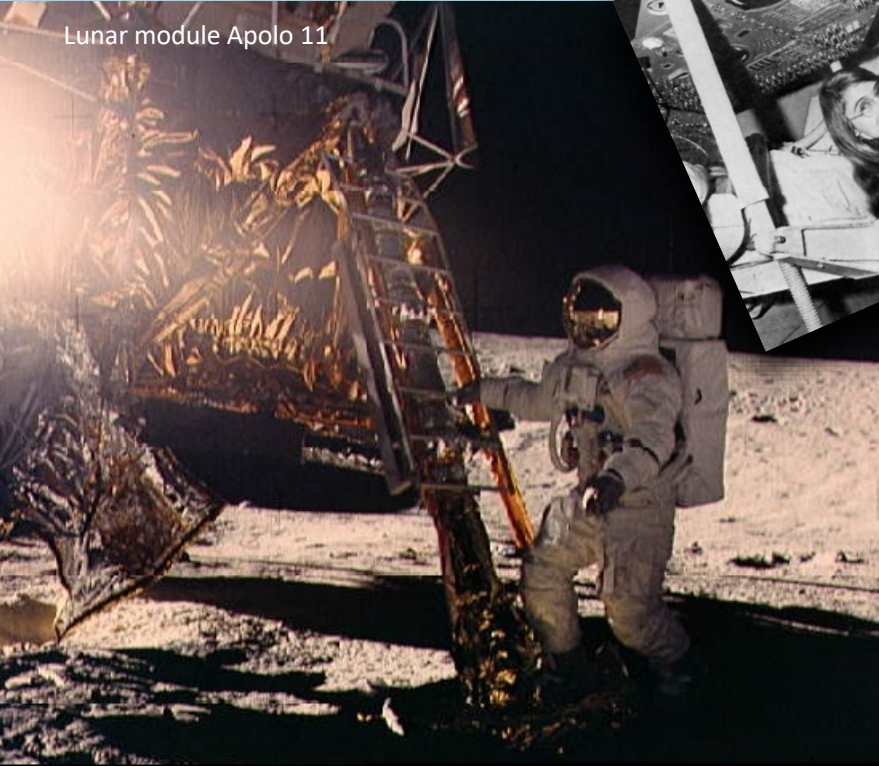
# MIT: Massachusetts Institute of Technology

Apollo Guided Computer and the DSKY



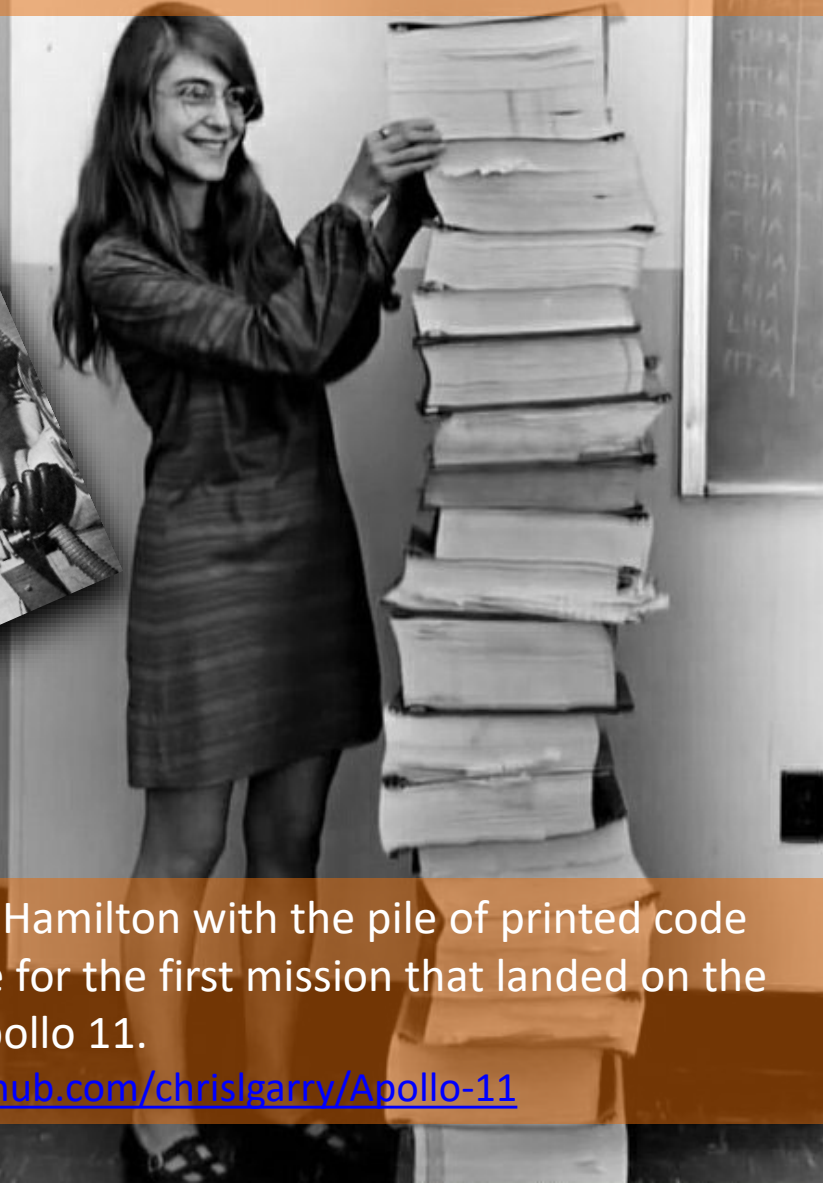
60's years

Lunar module Apollo 11



Margaret Hamilton with the pile of printed code she wrote for the first mission that landed on the Moon, Apollo 11.

<https://github.com/chrislgarry/Apollo-11>



# MIT: Massachusetts Institute of Technology

The same place, 50 years after

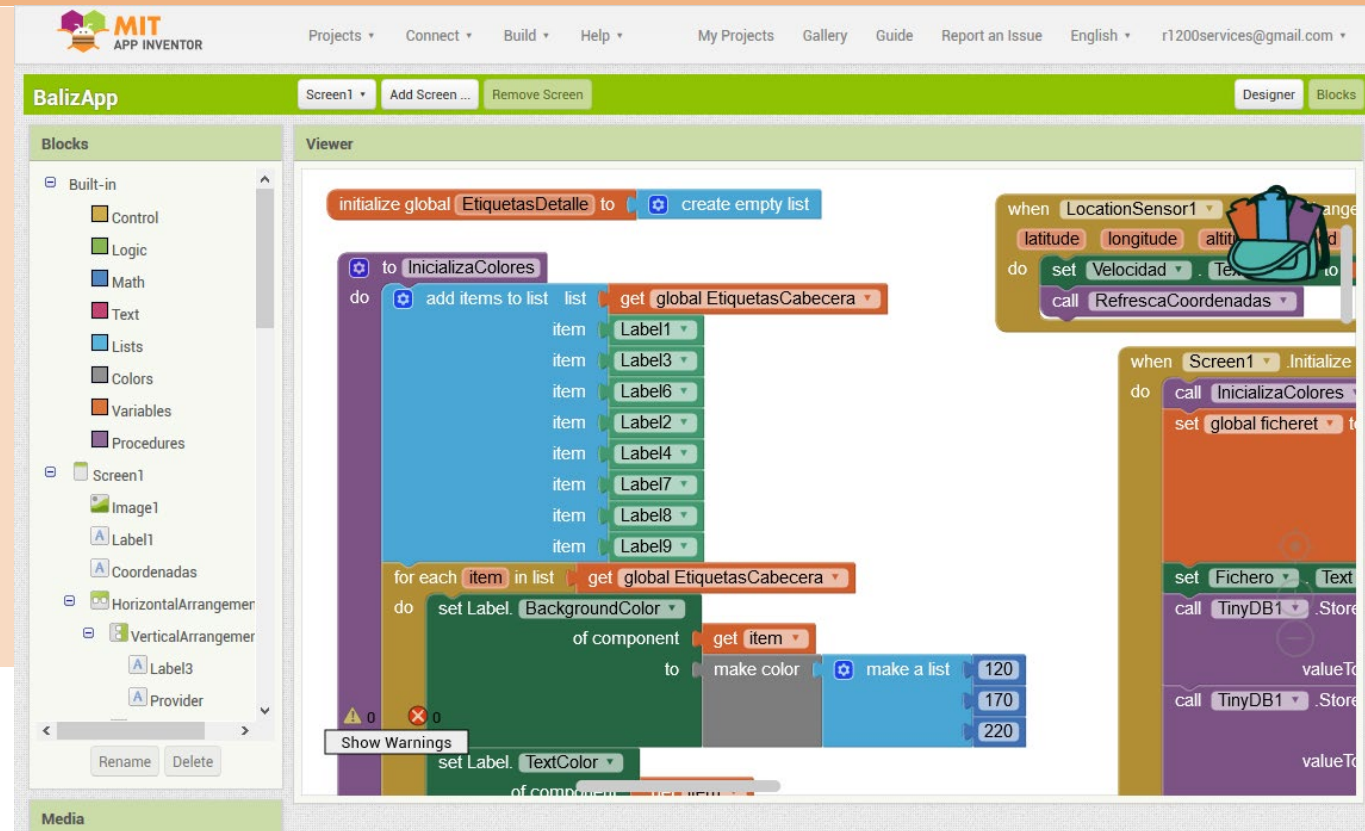
Initially developed by Professor Hal Abelson and a Google Education team, App Inventor runs as a web service managed by staff from the MIT Center for mobile learning, since 2010

MIT App Inventor's source code:

<https://github.com/mit-cml/appinventor-sources>

*Official Google Research Blog (2009)*

<https://ai.googleblog.com/2009/07/app-inventor-for-android.html>



The processor of the on-board computer of Apollo 11 was 2MHz, with less than 40kB.

Compare it to your 2GHz onboard smartphone and 32GB, on average, and think what you can not do with it ...

# Justification platform and environment chosen



- **Easy learning**
  - Blocky Concept
  - Without having to type code in the usual way in programming
- **Good Access to the Smartphone resources**
  - Almost every gadget at your fingertips
- **The resulting applications are robust.** It seems like a children's game, but it is not.
- **It is a tool for programming smartphones and tablets on the cloud**, just like Gmail is email on the cloud or Spotify is music on the cloud.



# Justification platform and environment chosen



- **Free and open-source\*** (<https://github.com/mit-cml/appinventor-sources> - <http://appinventor.mit.edu/appinventor-sources/>) **and with the possibility of uploading our Apps to Google Play**
- **It's a tool with the seal of Google and MIT**
- **All you need is a web browser and a Google account, and for debugging, a physical mobile or a virtual emulator.**
- **It will allow us to be addressed in the short duration of our course**

# Justification platform and environment chosen



- **Other advantages:**
  - Globally chosen as a basic learning tool for student programming
  - Great community of users
  - Usual in "Hackathons" or "Appathons" (this course could be a kind of)
  - Many "social" uses rewarded (citizen science)
  - Finally, since 2021: Available for Apple iOS on the App Store (<http://appinventor.mit.edu/blogs/evan/2021/03/04-mit-app-inventor-ios-app-store>)

	Live test apps while coding	Test apps with an Android emulator	Share app source code	Use extensions in your code	Install apps on phone or tablet
iOS	✓	✓	✓		
Android	✓	✓	✓	✓	✓

## Future Work

We are currently working to implement the following:

- Packaging apps (.ipa files) for permanent installation in an iOS device or for distribution on the App Store. This functionality is currently in a closed beta.

# What includes and what not, MIT App Inventor?

- **Includes:**

- App Inventor is a tool that includes (almost) all the sensors / elements / components / calls to processes available in smartphones, and if it does not include them, they can be included through the **Extensions**
  - For example: to work with WiFi and networking (SSIDs, IPs, etc.)

- **Exclude:**

- To work in background, that is, when the App is not running in the foreground (but can be running while screen is off)
- Push notifications (although it can be emulated with Firebase notifications)

# What are students expected to complete in this course?

- **Creation of an operational project in group:**
  - To choose among those offered by the professors: one of them of immediate practical application
  - Or one free to the taste of the student, with the ingredients that will be indicated, using as many smartphone resources as necessary



# MIT App Inventor:

## The complete ecosystem (1/2)

1) **Web browser** (recommended Chrome, IE not supported) opening <http://ai2.appinventor.mit.edu> and using a Google account + install the **SW aiStarter** on the PC

2) To test and debug we will need:

- 1) Or an **Android physical device** (Smartphone/Tablet)
- 2) Or an **Android emulator** on the PC



3) **Browser-Android connection** for debugging, using the tool **MIT AI2 Companion (App)**



- 1) With a USB Cable  
(don't forget DEBUB MODE)



- 2) With a WiFi connection (the best option, not always available in UPV)



# MIT App Inventor:

## The complete ecosystem (2/2)

### 4) If we are going to work online:

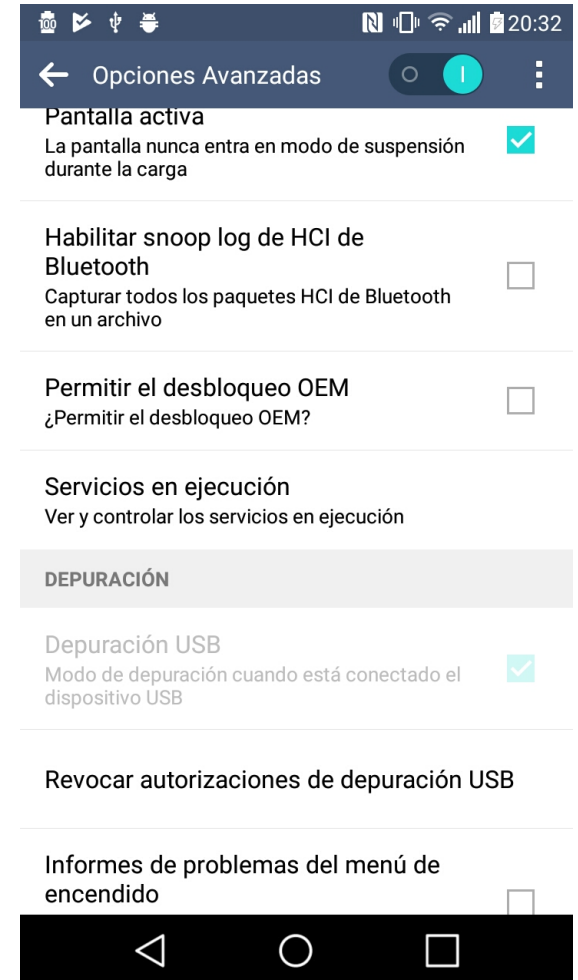
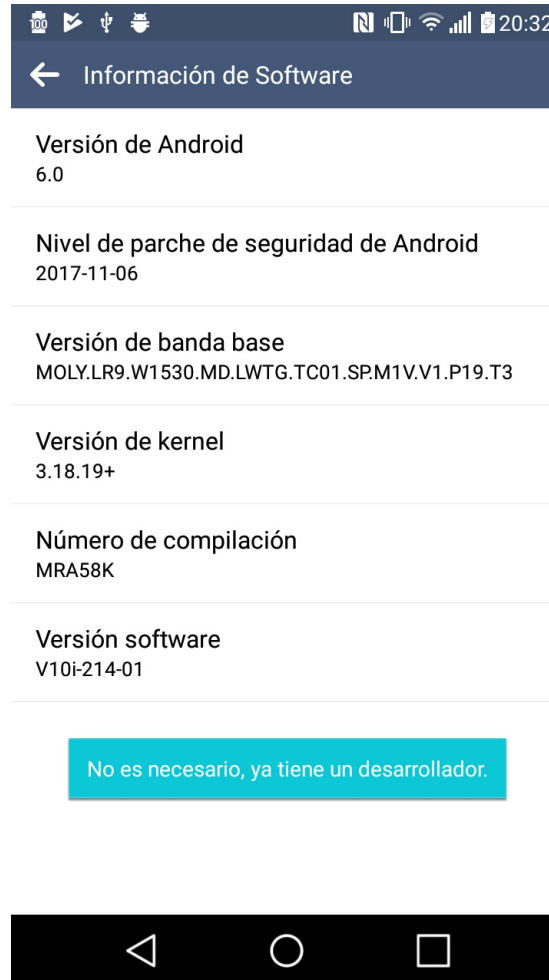
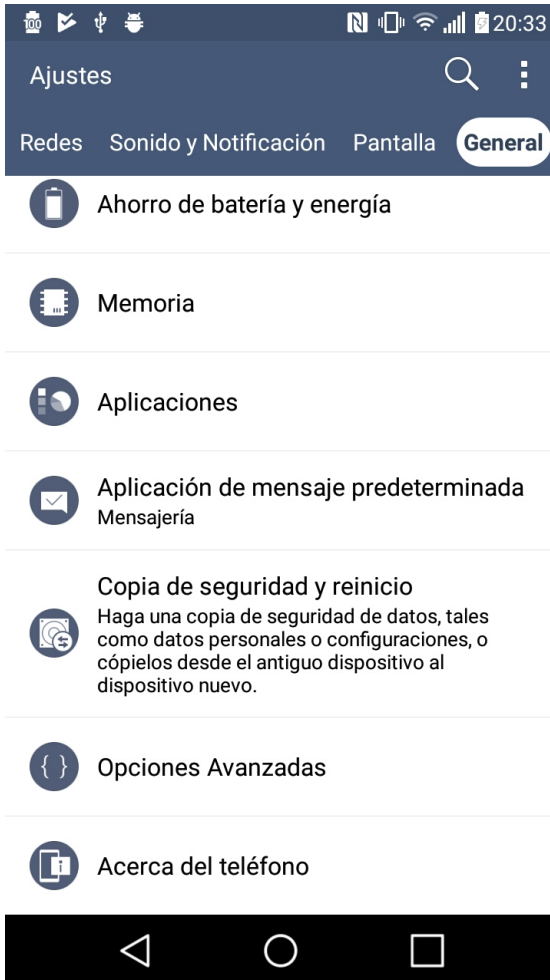
- To send or receive information online, as well as to represent the results captured by our App on the web, we will need a server. We have one dedicated for this course:

<https://ictw.agr.upv.es/>

### Services that we can use, with one account per team:

- Apache Web Server with PHP
- MySQL database
- Storage space managed by FTP
- Basic Google Maps/OSM API scripts to use maps

# USB Debug Mode



# File types in App Inventor



- Because the file extension
  - **.apk** is **THE APP** itself, ready to be installed, opening it in the device (for example, passing it as an attachment by email, hanging it on a web, or uploading it to Google Play for distribution)
  - **.aia** are the **backup** files of each project, to save a copy or to share the "code" with other users
  - **.keystore** is the file of digital private key used to **sign** our projects, unique for each user, essential when we publish on Google Play (keep a copy of it)



# A quick look at what we will find when working with App Inventor

Before starting to work with the browser, and without going into detail, let's quickly see how the different screens we will use looks like

# Main Projects screen

[Proyectos ▾](#)[Conectar ▾](#)[Generar ▾](#)[Ayuda ▾](#)[Mis proyectos](#)[Gallery](#)[Guía](#)[Informar de un problema](#)[Español ▾](#)[r1200services@gmail.com ▾](#)[Comenzar un proyecto nuevo...](#)[Borrar proyecto](#)[Publish to Gallery](#)

## Proyectos

Nombre	Fecha de creación	Fecha modificación ▾	Published
<input type="checkbox"/> ProyectoVISUAL	28/3/2018 17:51:54	28/3/2018 20:07:26	No
<input type="checkbox"/> BalizApp	16/7/2016 1:10:41	28/3/2018 19:28:57	No
<input type="checkbox"/> Scoreboard	11/2/2018 12:40:51	19/2/2018 20:41:23	No
<input type="checkbox"/> Scoreboard_checkpoint1	13/2/2018 20:23:08	13/2/2018 20:23:08	No
<input type="checkbox"/> ETSIAMN_IRO_GIS	25/1/2015 0:25:47	13/2/2018 16:09:59	No
<input type="checkbox"/> ETSIAMN_IRO_GIS_ICTW	4/7/2015 11:26:49	13/2/2018 16:09:39	No
<input type="checkbox"/> Scoreboard_checkpoint_final_diseno	11/2/2018 14:30:36	11/2/2018 14:30:36	No
<input type="checkbox"/> CAU_remoto	31/1/2018 19:06:40	8/2/2018 19:22:08	No
<input type="checkbox"/> ProjectValencia1	6/7/2017 14:45:14	7/2/2018 22:29:48	No
<input type="checkbox"/> ParrotProject	4/7/2016 12:27:59	4/2/2018 23:09:36	No
<input type="checkbox"/> Gestor_del_Arbolado_2_5_checkpoint2	24/1/2015 23:26:58	31/1/2018 20:10:57	No
<input type="checkbox"/> light	31/1/2018 18:52:06	31/1/2018 19:50:22	No
<input type="checkbox"/> timelapse	22/1/2018 23:27:51	31/1/2018 18:18:59	No
<input type="checkbox"/> camera	22/1/2018 13:00:37	31/1/2018 16:53:56	No
<input type="checkbox"/> AutomatiCam	22/1/2018 15:52:34	31/1/2018 16:12:04	No
<input type="checkbox"/> Proyecto_2018	19/1/2018 16:15:07	22/1/2018 12:48:09	No
<input type="checkbox"/> CultiApp	10/9/2015 19:41:20	19/1/2018 16:26:42	No
<input type="checkbox"/> CultiApp_checkpoint4	4/10/2015 23:39:19	19/1/2018 16:23:37	No
<input type="checkbox"/> NFC_pruebas	23/10/2017 13:03:00	19/1/2018 16:08:35	No
<input type="checkbox"/> Hablator	23/10/2017 13:03:54	23/10/2017 13:03:54	No
<input type="checkbox"/> LED_RGB	9/11/2014 19:18:08	18/7/2017 16:23:57	No
<input type="checkbox"/> List	6/7/2017 14:41:32	6/7/2017 15:34:23	No
<input type="checkbox"/> MapToCanvas	6/7/2017 10:37:03	6/7/2017 11:03:57	No
<input type="checkbox"/> CultiApp_checkpoint3	28/9/2015 15:41:43	6/7/2017 10:58:57	No
<input type="checkbox"/> Maps	4/7/2017 12:16:29	6/7/2017 10:56:06	No
<input type="checkbox"/> maps2button	4/11/2015 20:23:02	6/7/2017 10:35:53	No
<input type="checkbox"/> SendingVars	1/7/2016 21:37:53	4/7/2017 18:15:49	No
<input type="checkbox"/> LocationBasics	6/7/2016 12:21:40	4/7/2017 18:06:51	No
<input type="checkbox"/> AndroidWheresMyCar_MIT	2/7/2015 23:57:24	4/7/2017 17:58:25	No

# Components of the Designer view

[Proyectos ▾](#)[Conectar ▾](#)[Generar ▾](#)[Ayuda ▾](#)[Mis proyectos](#)[Gallery](#)[Guía](#)[Informar de un problema](#)[Español ▾](#)[r1200services@gmail.com ▾](#)**ProyectoVISUAL**

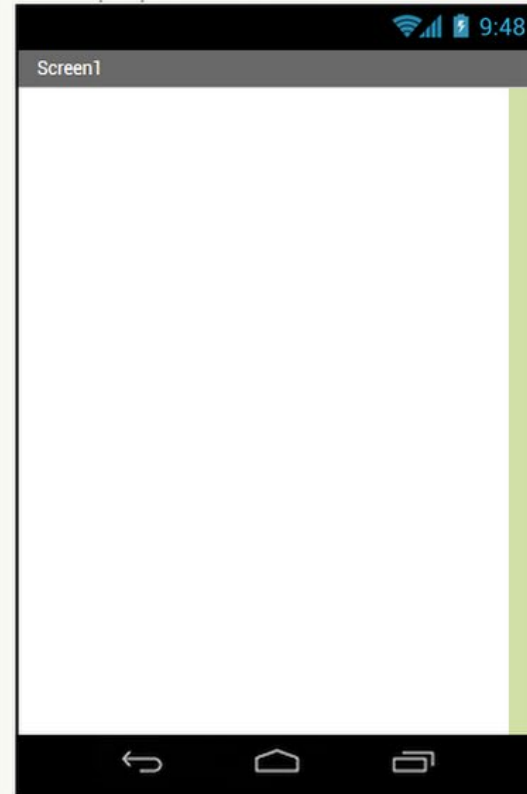
Screen1 ▾

Añadir ventana

Eliminar ventana

Diseñador

Bloques

**Paleta****Interfaz de usuario** Botón ? CasillaDeVerificación ? SelectorDeFecha ? Imagen ? Etiqueta ? SelectorDeLista ? VisorDeLista ? Notificador ? CampoDeContraseña ? Deslizador ? Desplegable ? CampoDeTexto ? SelectorDeHora ? VisorWeb ?**Disposición****Medios****Dibujo y animación****Maps****Sensores****Visor**☐ Mostrar en el Visor los componentes ocultos☐ Marcar para previsualizar al tamaño de la tablet

Componentes no visibles

**Componentes**

Screen1



Cambiar nombre

Borrar

**Medios**

Subir archivo...

**Propiedades**

Reloj1

TemporizadorSiempreSeDispara

☒

TemporizadorHabilitado

☒

IntervaloDelTemporizador



# Components of the Block Editor view

The screenshot displays the MIT App Inventor 2 Block Editor interface. The top navigation bar includes the MIT App Inventor 2 Beta logo, a menu (Projects, Connect, Build, Help), and user information (My Projects, Gallery, Guide, Report an Issue, English, r1200services@gmail.com).

The main workspace is divided into three panels:

- Blocks Panel (Left):** Contains a 'Built-in' section with categories like Control, Logic, Math, Text, Lists, Colors, Variables, and Procedures. Below this is a 'Screen1' section showing a 'HorizontalArrangement' block containing 'Image1', 'Especie', 'Spinner1', 'Image2', 'Afeccion', and 'Spinner2'. At the bottom of this panel are 'Rename' and 'Delete' buttons.
- Media Panel (Bottom Left):** Lists various image files (1\_33.png, 2\_33.png, 3\_33.png, 4\_33.png, 5\_33.png, gps1.png, gps2.png, gps3.png, logocuatrado.jpg) and an 'Upload File ...' button.
- Viewer Panel (Right):** Displays the code for the selected screen, 'ETSIAMN\_IRO\_GIS'. The code is organized into a 'when Screen1.Initialize' block, which contains a 'do' loop with the following blocks:
  - Initialize global 'BotonPulsado' to 0.
  - Initialize global 'strACCESSKEY' to 'ebtsept14'.
  - Initialize global 'Formateado' to ' '.
  - Initialize global 'Version' to '2.0'.
  - Initialize global 'strFilename' to ' '.
  - Initialize global 'foto' to ' '.
  - Initialize global 'SinCoordenadas' to 'I don't have valid position. If you switched ON the GPS (more precise), you often can experience some delays'.
  - Set 'Spinner1' ElementsFromString to 'No team, Red Team, Green Team, Blue Team, Yellow Team, Black Team, White Team, Orange Team'.
  - Set 'Spinner2' ElementsFromString to 'No clue, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20'.
  - Set 'Posicion' Text to 'Actual position... searching...(please, switch ON the GPS from the configuration screen)'.
  - Set 'Especie' TextColor to green.
  - Set 'Afeccion' TextColor to green.
  - Set 'lblResult' TextColor to green.
  - Set 'Label2' BackgroundColor to green.
  - Set 'Label6' BackgroundColor to green.
  - Set 'Label8' BackgroundColor to green.
  - Call 'TinyDB1' StoreValue with tag 'strACCESSKEY' and valueToStore 'get global strACCESSKEY'.
  - Call 'TinyDB1' StoreValue with tag 'Version' and valueToStore 'get global Version'.
  - If 'call TinyDB1.GetValue' with tag 'micorreio' and valueIfTagNotThere '0' is equal to '0', then 'open another screen screenName' to 'Configuracion'.



# Summary of resources and tools to use

- App Inventor (AI2, preferably with Google Chrome browser)
- For debugging:
  - Android emulator if necessary due to lack of physical device
- To connect the Smartphone with the web AI2 web
  - On the mobile device: MIT AI2 Companion app (you must have the DEBUG MODE active and the cable connected if you connect via USB)
  - On the computer: [aiStarter](#) program
- To show my device in the projector I will use [scrcpy](#)
- Server of the course <https://ictw.agr.upv.es> (contains all the necessary resources)
- FTP to transfer files to the server (Filezilla)
- For the management of the database:
  - PHPMYAdmin from <http://ictw.agr.upv.es/phpmyadmin>
- To access the server in console mode (advanced users only):
  - PuTTY for SSH for Secure shell and telnet (or PowerShell)
- Basic PHP scripts that we can edit on the PC with:
  - Notepad ++, Brackets ...
  - And for web design Netbeans, Brackets ...

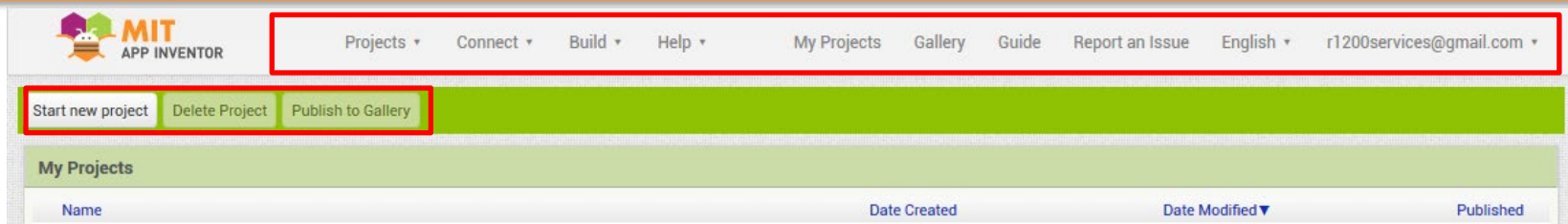
# Let's do it!

**Time to open your browser!**

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

**and use a Google account to login**

# Identifying the elements of App Inventor



- **Projects**

- My projects
- Start new project
- Import project (.aia) from my computer ...
- Import project (.aia) from a repository ...
- Delete Project
- Save project
- Save project as ...
- Checkpoint
- Export selected project (.aia) to my computer
- Export all projects
- Import keystore
- Export keystore
- Delete keystore

- **Connect**

- AI Companion
- Emulator
- USB
- Reset Connection
- Hard Reset

- **Build**

- App ( provide QR code for .apk )
- App ( save .apk to my computer )

- **Help**

- About
- Library
- Extensions
- Tutorials
- Troubleshooting
- Forums
- Report an Issue
- Companion Information
- Update the Companion
- Show Splash Screen

- **My Projects**

- **Gallery**

- **Guide**

- **Report an Issue**

- **Language**

- **Current Google Account login**

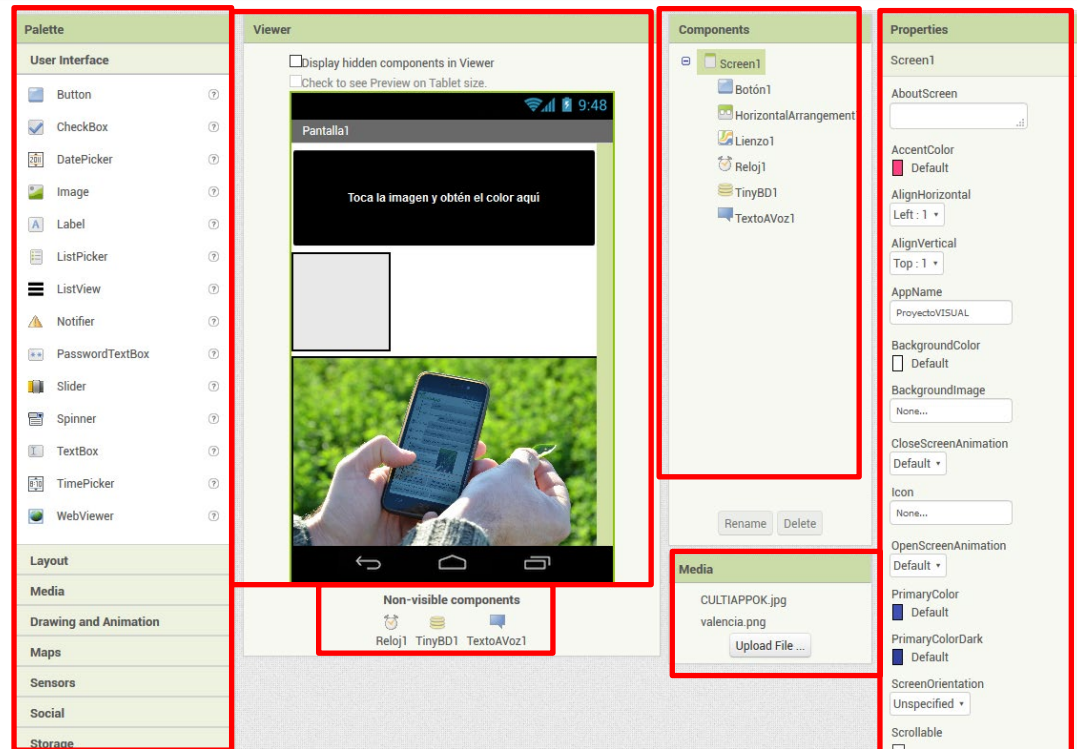
- **Start new project...**

- **Delete Project**

- **Publish to Gallery**

# Identifying the elements on screen inside a Project (**Designer**)

- Palette
- Viewer
- Components
- Non visible components
- Media
- Properties of the selected component



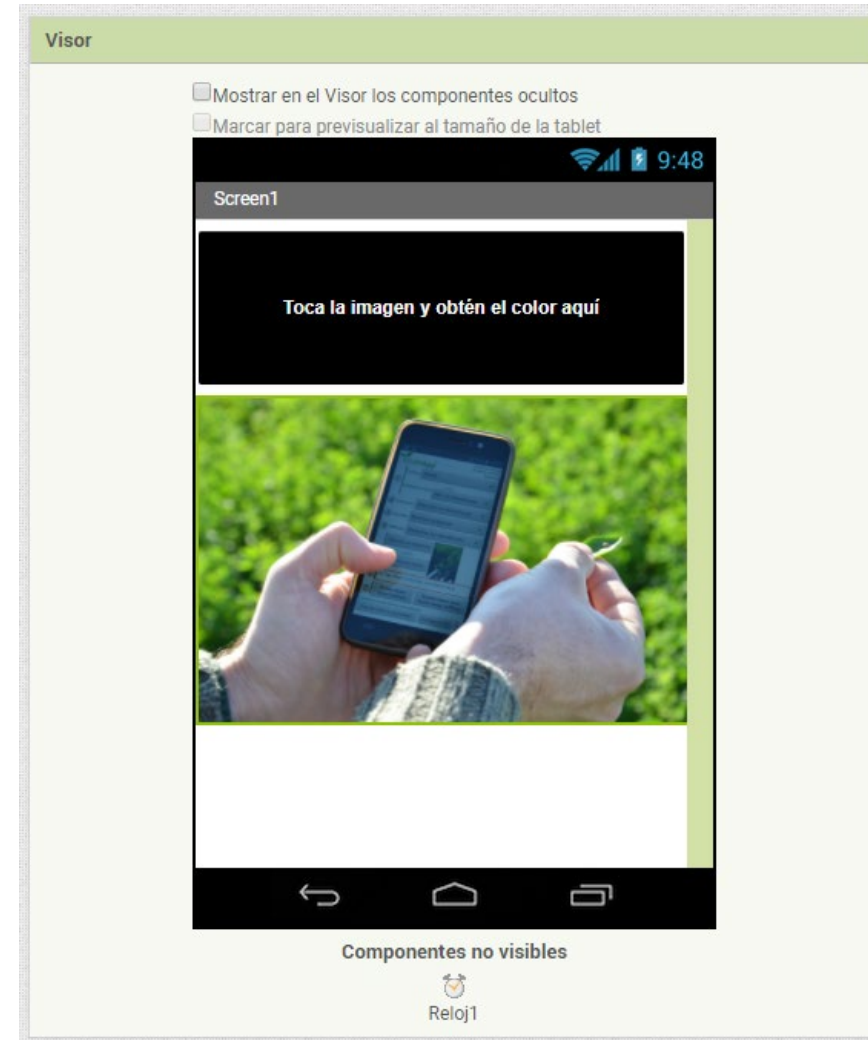
# Identifying the Palette elements

- They are this:
  - User interface
  - Layout
  - Media
  - Drawing and Animation
  - Maps
  - Sensors
  - Social
  - Storage
  - Connectivity
  - LEGO® MINDSTORMS®
  - Experimental
  - Extension

Palette
User Interface
Layout
Media
Drawing and Animation
Maps
Sensors
Social
Storage
Connectivity
LEGO® MINDSTORMS®
Experimental
Extension

# Identifying the **Viewer** elements

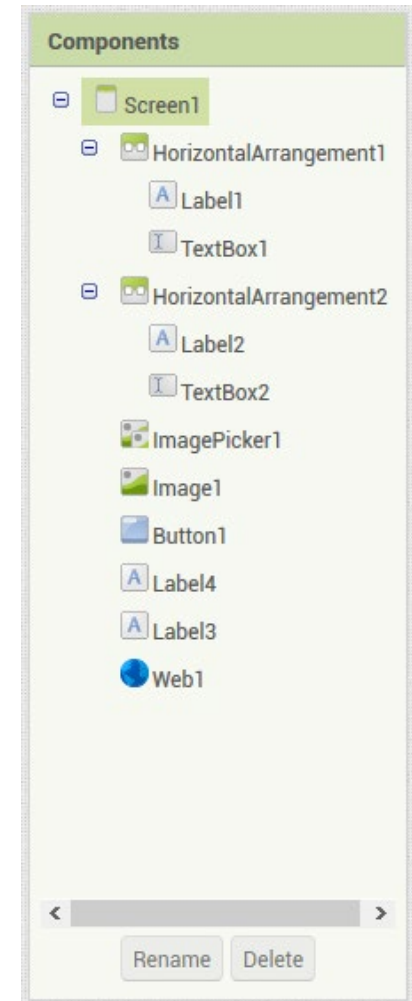
- Represents the app screen:
  - Even when very similar, his aspect if NOT DEFINITIVE, because has some help guides (green guide marks)





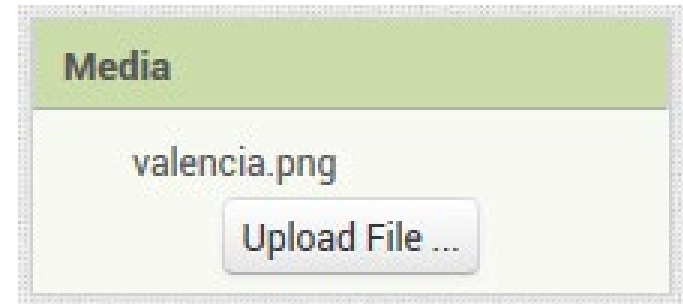
# Identifying the **Components** elements

- Arborescent hierarchical structure :
  - Shows all the elements integrated on the app screen
  - The PARENT element is **Screen1**, because all the elements are included on it, and if there are several screens, the first one is always "Screen1"



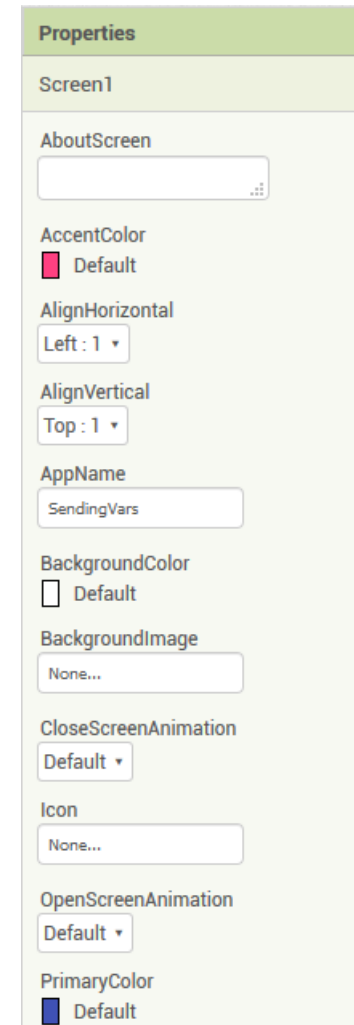
# Identifying the Media elements

- They are the **multimedia** files (Images, sounds...) uploaded to the server and available in App Inventor
  - We can upload more from here, or delete them if they were not necessary, in order to reduce the weight of the App
  - They can also be uploaded directly from the properties of each components, and will appear here



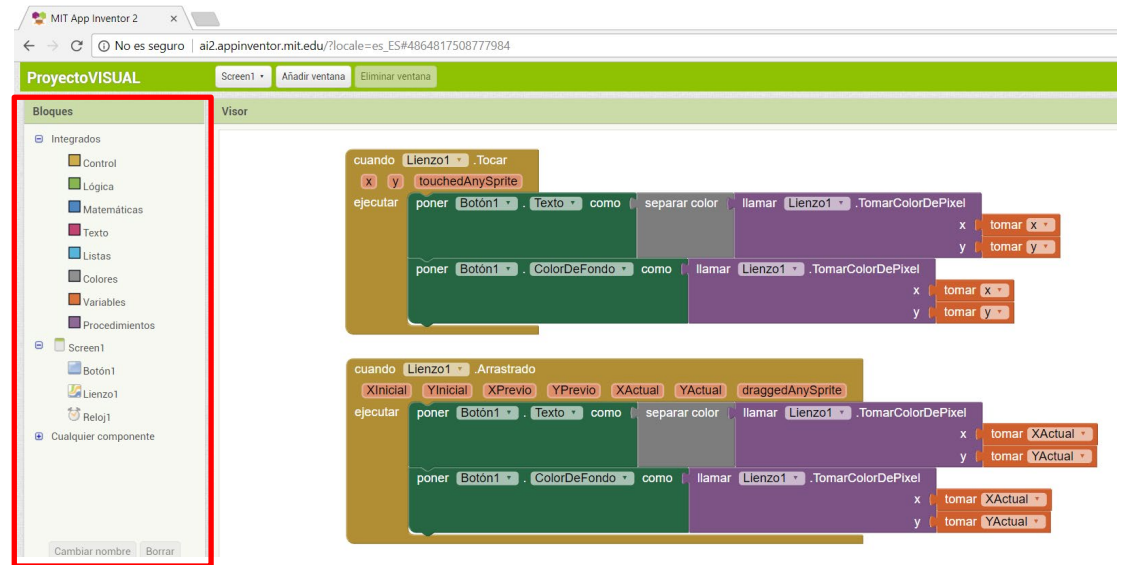
# Identifying the **Properties** elements

- They are the properties of the selected Component
  - We also have the possibility to modify these properties at runtime, that is, in block programming, not only now, during the design

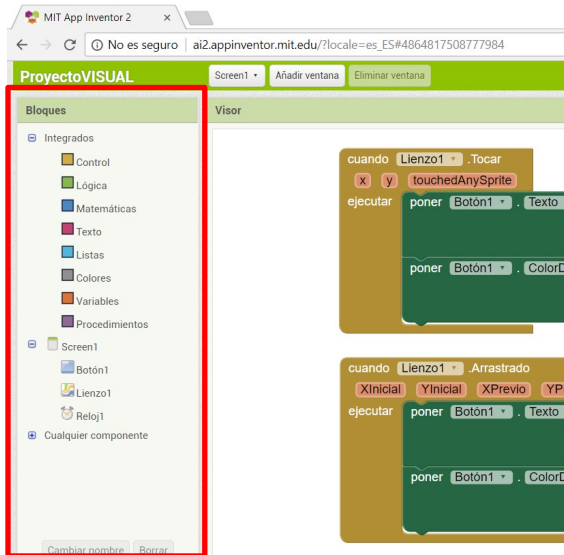


# Identifying the screen elements inside a Project (Blocks) (1/2)

This is the part where we will establish the programming itself, the "code", only using puzzle blocks that can only fit with the appropriate blocks



# Identifying the screen elements inside a Project (Blocks) (2/2)



Blocks (available elements to program):  
each color is because a function

- Built-in, the basic, grouped by:
  - Control (programming structures)
  - Logic
  - Math
  - Text
  - Lists
  - Colors
  - Variables
  - Procedures
- Own components (what you drop)
- Any component (affecting to all the elements in the group)

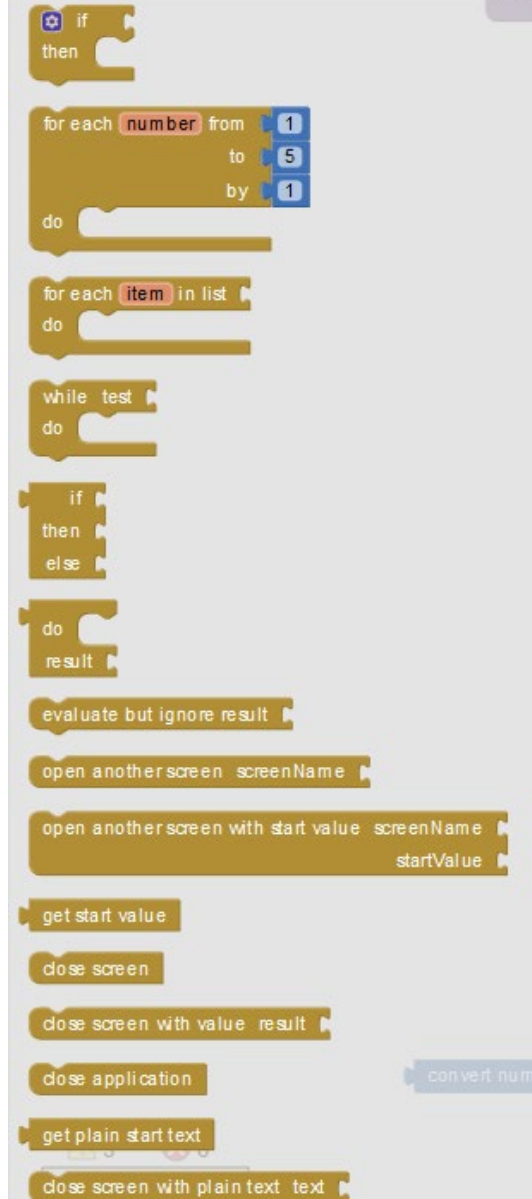


Backpack: to save objects to be reused later (permanent copy & paste)



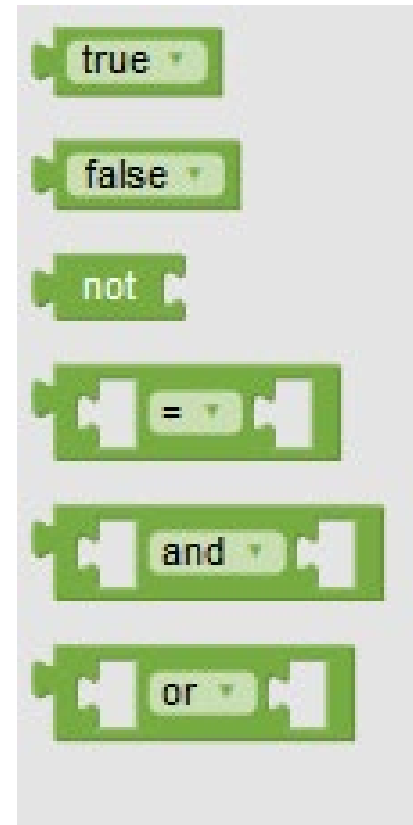
# Control Blocks

- If then...
- For each number from ... to ... in increments of ... run ...
- For each item in the list ...
- While condition ... execute ...
- Yes ... then ... if not ...
- Run ... result ...
- Evaluate but ignore result ...
- Open another screen with name ...
- Open another screen with name ... with an initial value ...
- Take the initial value (from the screen that called us)
- Close screen
- Close screen with a value ... (it will be given to the other screen that called us)
- Close the application
- Take the initial text (from the app that called us)
- Close screen with text ... (it will be given to the other application that called us)



# Logic Blocks

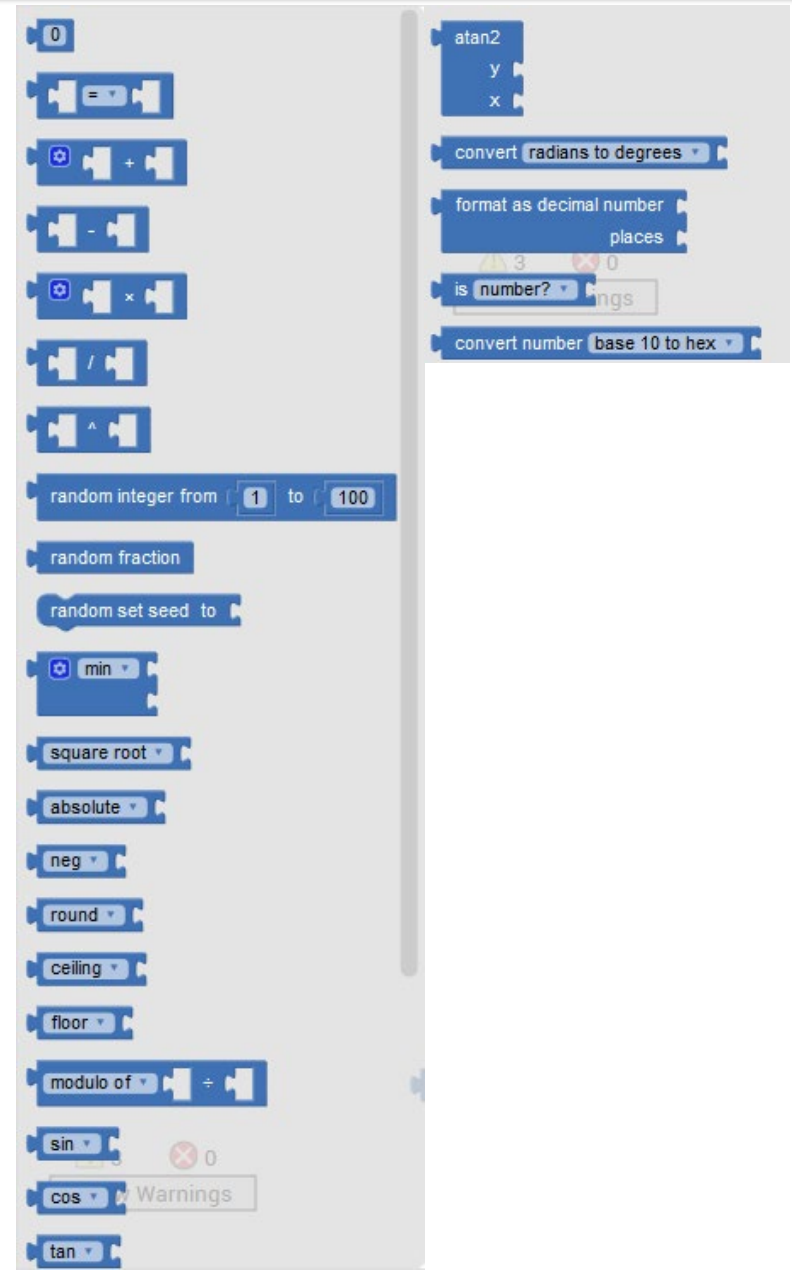
- True (True value)
- False (False value)
- Not (denial of ...)
- Equal = (comparison, returns True or False if equality is fulfilled or not)
- Not Equal  $\neq$
- AND (returns True if both elements are True "AND")
- OR (returns True if any element is True "OR")



# Math Blocks

Basic mathematical operations:

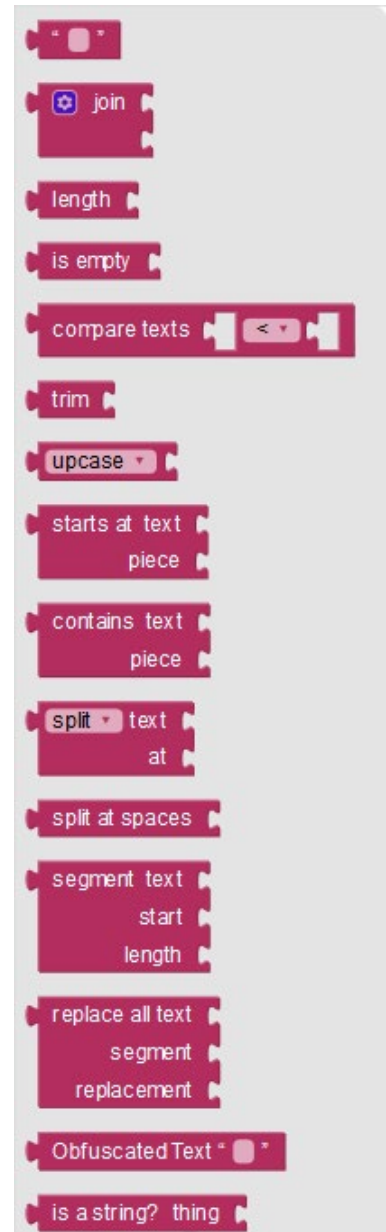
- Set a numeric value
- Logical value if both numbers are equal
- +, -, \*, /, ^
- Random numbers (and their seed)
  - Integer between ... and ...
  - Random fraction (between 0 and 1)
- Minimum or maximum value
- Square root of
- Absolute value of | ... |
- Negative of ...
- Round out...
- Upper and lower integer of a decimal number
- Modulo of ... / ... (rest division)
- Sine, cosine, tangent, arctangent ...
- Conversion radians / degrees
- Convert to decimal with ... decimals
- Is it a number? (or it's text and another class)
- Base conversion (binary, decimal, hexadecimal)



# Text Blocks

## Basic operations with text strings

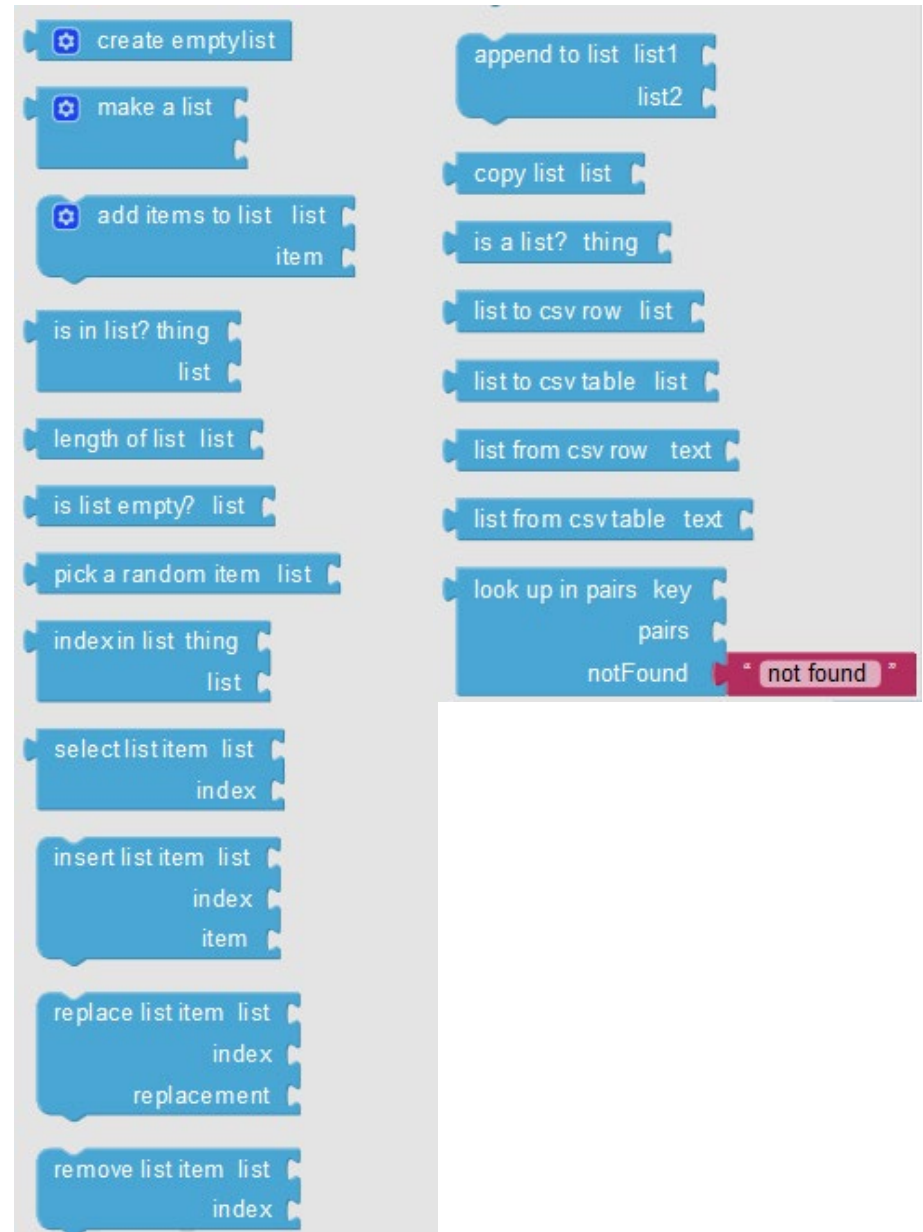
- Set text string
- Join chains
- Length (chain length)
- It is empty? (if it has any value)
- Text comparison
- Trim (removes spaces in front and behind chain)
- Conversion to uppercase or lowercase
- Beginning of the text (position of one string within another)
- Contains text (checking the inclusion of one string in another)
- Split (cut out on any given character) / Split at any (given chain)
- Split text by spaces
- Text segment from position ... in ... characters
- Replacement of a chain ... in text ... with substitute ...
- Obfuscation of texts in the resulting .apk file (within the app it will be the same as a text)
- Is a text string this ...?



# Lists Blocks

Operations with elements of Lists (Vector concept):

- Create empty list
- Create list of n elements
- Add items to an existing list
- Search for item in list
- Number of items in a list
- Is the list empty?
- Extract item at random from a list
- Find element position in list (0 if not found)
- Select item number ... from the list
- Insert item in list at the specified position
- Substitution of element in a given position
- Remove the item from the given position
- Add all the elements of the list2 to the list1 (the 1 will happen to contain both, while the 2 will remain unchanged)
- Copy list in another
- Is this a list ...?
- Convert list to record / CSV table (values separated by comma)
- Convert record / CSV table to list
- Search for key pairs (returns the value associated with the key in the list of partners)





# Dictionaries Blocks

Dictionaries, called in other languages terms such as maps, associative arrays or lists, are data structures that associate one value, often called the key, with another value. A common way of displaying dictionaries is using the JavaScript Object Notation (JSON), for example:

```
{
  "id": 1,
  "name": "Tim the Beaver",
  "school": {
    "name": "Massachusetts Institute of Technology"
  },
  "enrolled": true,
  "classes": ["6.001", "18.01", "8.01"]
}
```

The above example shows that in JSON the keys (quoted text before the :) can map to different types of values. The allowed types are number, text, other dictionaries, booleans, and lists. In the blocks language, you can build this dictionary as follows:

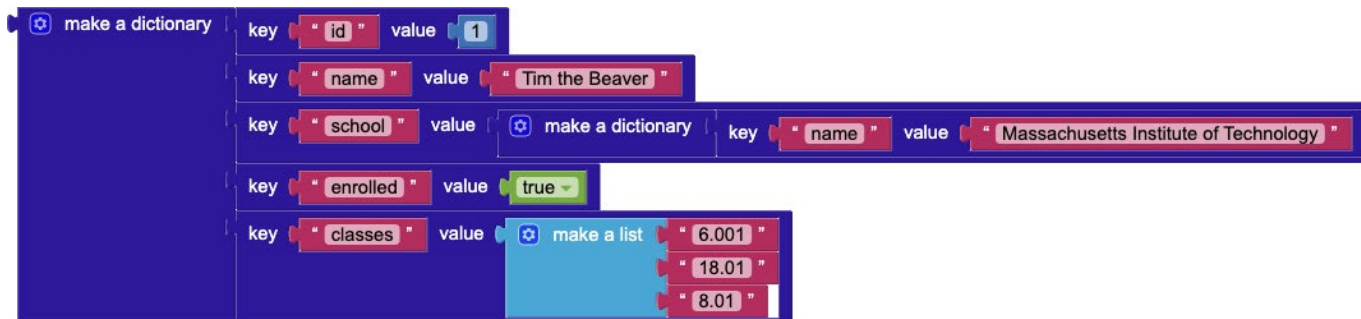
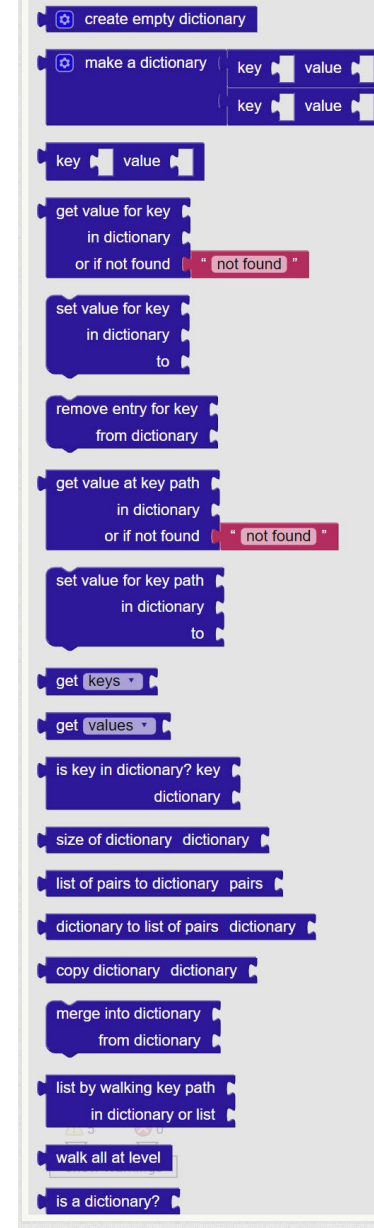


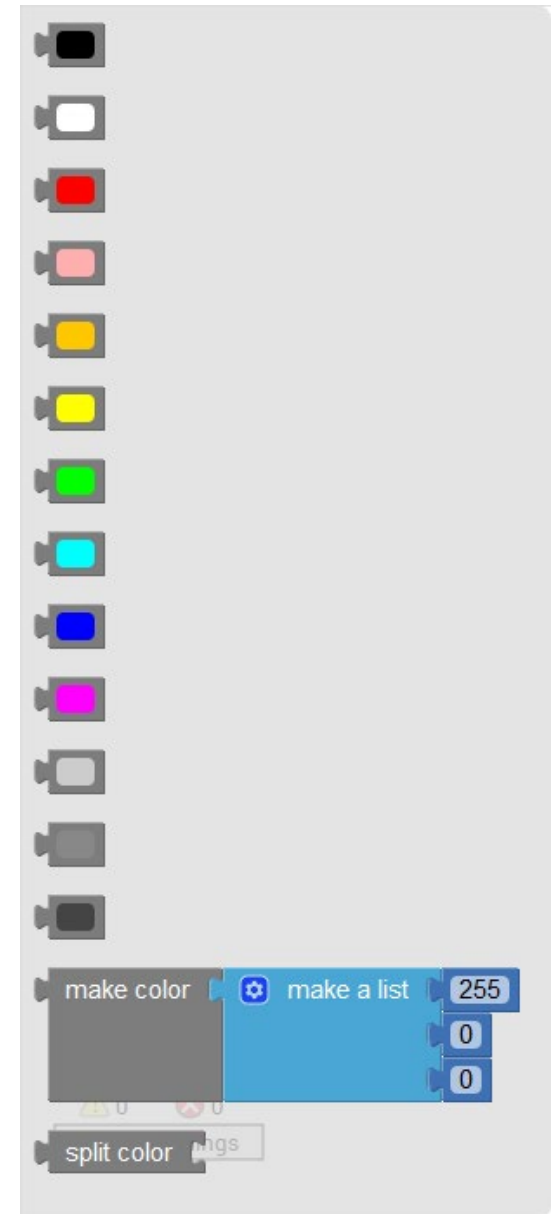
Figure 1: A blocks representation of the JSON code snippet shown above.



# Colors Blocks

Basic color operations:

- Set a color to some of the color properties of some component, for example, text color or background color (there are more colors if we click on the color in question)
- Create RGB color (red, green, blue and optionally alpha channel for transparency). Three or four elements list.
- Separate RGB color (in a red, green, blue and alpha channel list for transparency)



# Variables Blocks

Operations with variables:

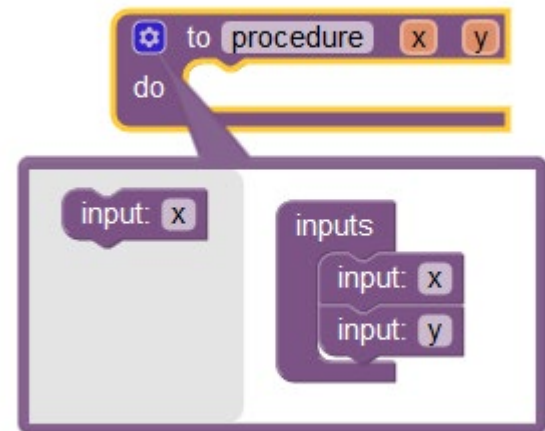
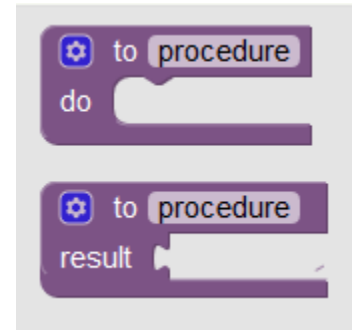
- Establishment of values (**setters**)
- Taking of values (**getters**)
- Globally (to the screen) or locally (to the block or procedure)
- For global variables to the entire project, we must have TinyBD, a kind of local database on the application, which lasts over the time, once the app is closed, and can be read from different screens



# Procedures Blocks

Creation of procedures, for when we have to repeat assiduously a succession of actions, so we only create it once and then we call it as many times as necessary:

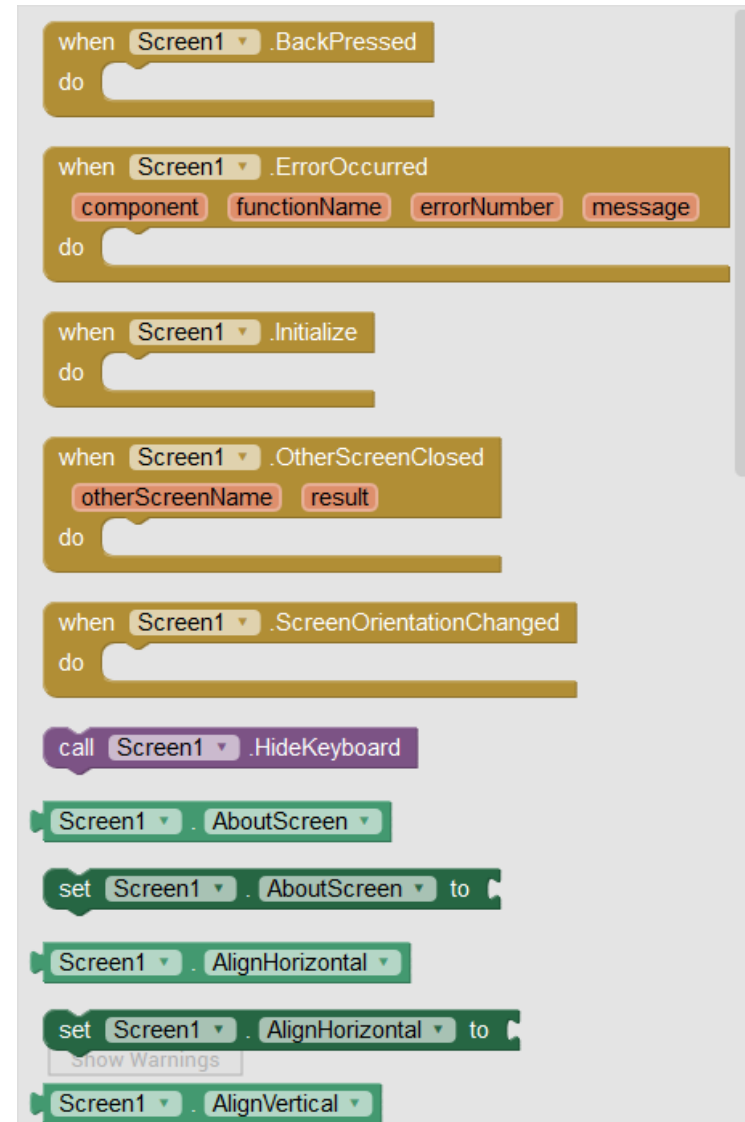
- No result (proper *procedure*)
- With result (typical *function* concept)



# Screen Blocks

Now we are going to observe that for a component (like Screen, that is hierarchically the superior element), a series of blocks of different colors appear, in function of group to which they belong:

- **Golden:** Control of actions, **events**
- **Green:**
  - **Light green: getters** (get property values)
  - **Dark green: setters** (set or read values in properties)
- **Purple:** Calls to procedures

























# Palette Objects (1/3)

## Palette











### User Interface

	Button	<a href="#">?</a>
	CheckBox	<a href="#">?</a>
	DatePicker	<a href="#">?</a>
	Image	<a href="#">?</a>
	Label	<a href="#">?</a>
	ListPicker	<a href="#">?</a>
	ListView	<a href="#">?</a>
	Notifier	<a href="#">?</a>
	PasswordTextBox	<a href="#">?</a>
	Slider	<a href="#">?</a>
	Spinner	<a href="#">?</a>
	Switch	<a href="#">?</a>
	TextBox	<a href="#">?</a>
	TimePicker	<a href="#">?</a>
	WebView	<a href="#">?</a>

## Layout


	HorizontalArrangement	<a href="#">?</a>
	HorizontalScrollArrangement	<a href="#">?</a>
	TableArrangement	<a href="#">?</a>
	VerticalArrangement	<a href="#">?</a>
	VerticalScrollArrangement	<a href="#">?</a>

## Media

















	Camcorder	<a href="#">?</a>
	Camera	<a href="#">?</a>
	ImagePicker	<a href="#">?</a>
	Player	<a href="#">?</a>
	Sound	<a href="#">?</a>
	SoundRecorder	<a href="#">?</a>
	SpeechRecognizer	<a href="#">?</a>
	TextToSpeech	<a href="#">?</a>
	Translator	<a href="#">?</a>
	VideoPlayer	<a href="#">?</a>

# Palette Objects (2/3)





## Drawing and Animation

-  Ball 
-  Canvas 
-  ImageSprite 





























## Maps

-  Circle 
-  FeatureCollection 
-  LineString 
-  Map 
-  Marker 
-  Navigation 
-  Polygon 
-  Rectangle 















## Charts

-  Chart 
-  ChartData2D 













## Sensors

-  AccelerometerSensor 
-  BarcodeScanner 
-  Barometer 
-  Clock 
-  GyroscopeSensor 
-  Hygrometer 
-  LightSensor 
-  LocationSensor 
-  MagneticFieldSensor 
-  NearField 
-  OrientationSensor 
-  Pedometer 
-  ProximitySensor 
-  Thermometer 

## Social











-  ContactPicker 
-  EmailPicker 
-  PhoneCall 
-  PhoneNumberPicker 
-  Sharing 
-  Texting 
-  Twitter 

## Storage































-  CloudDB 
-  DataFile 
-  File 
-  Spreadsheet 
-  TinyDB 
-  TinyWebDB 

# Palette Objects (3/3)







## Connectivity

	ActivityStarter	
	BluetoothClient	
	BluetoothServer	
	Serial	
	Web	

## LEGO® MINDSTORMS®

	NxtDrive	
	NxtColorSensor	
	NxtLightSensor	
	NxtSoundSensor	
	NxtTouchSensor	
	NxtUltrasonicSensor	
	NxtDirectCommands	
	Ev3Motors	
	Ev3ColorSensor	
	Ev3GyroSensor	
	Ev3TouchSensor	
	Ev3UltrasonicSensor	
	Ev3Sound	
	Ev3UI	
	Ev3Commands	

## Experimental

	ChatBot	
	FirebaseDB	
	ImageBot	

## Extension

[Import extension](#)

# First activity

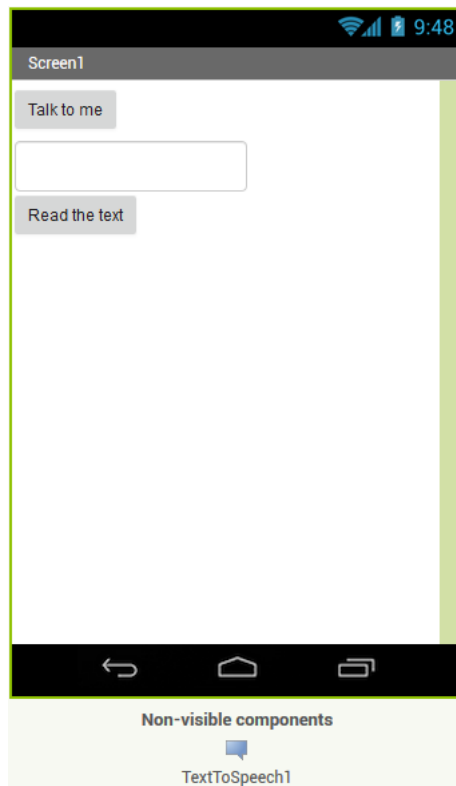
- Creation of our first "Hello World!" App, using a Button object and another Label, so that when you click on the button, the text "Hello World!" Appears on the label, just to check the cycle of:
  1. **Screen designer**
  2. **Design of Blocks**
  3. **Connect with the mobile device for debugging**

# Starting the activities

- We begin to explore the possibilities through some "result" apps to start with, which will surely hook you up and make you see how easy it is to achieve important results with little effort:
  - Talk to Me

# “Talk to Me”

## Designer:



## Blocks:





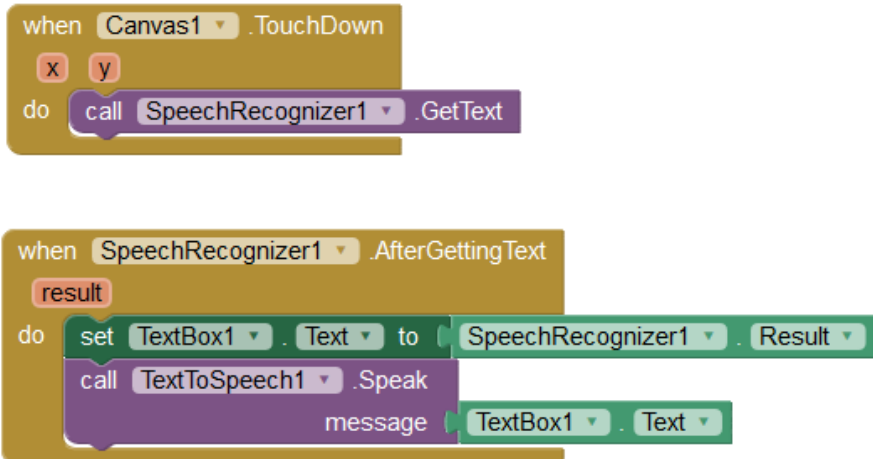
# “Parrot”

An application that listens to what you say and then... repeats it!

Designer:



Blocks:



# Important Concepts in App Inventor 2

- [Publishing Apps to Google Play Store](#)
- [Understanding Local and Global Variables](#)
- [Using Lists](#)
- [Commands and Expressions](#)
- [Control Flow](#)
- [Arranging Components on the Screen](#)
- [Manipulating Component State: Getters and Setters](#)
- [Using Conditional Blocks](#)
- [Events and Event Handlers](#)
- [Using Multiple Screens in One App](#)
- [PseudoRandom Number Generator and Random Set Seed](#)
- [Data and Databases](#)
- [Using the Activity Starter to launch external apps like the phone's web browser](#)
- [Blocks with Dropdowns](#)
- [Working with Images and Sounds](#)

# Conclusions

## Session 1 Introduction

---

We know the components, mechanisms and tools necessary to operate the complete cycle (design and debugging) both from the PC side and from the side of the Android mobile device

---

We will work with an absolutely valid tool for the real world, and supported by the web services that a server provides us

---

The developed Apps are guaranteed by MIT and a long-term commitment by the maintainer. Actually, they are starting to support iOS

---

Credits:

- **Wikipedia**
- **Web MIT App Inventor [Attribution-ShareAlike 3.0 Unported (CC BY-SA 3.0)]**