MIT APP INVENTOR 1

di FRANCESCO FICILI

Iniziamo a scoprire MIT App Inventor, un tool di sviluppo per applicazioni Android creato dal MIT e basato sul linguaggio di programmazione grafico Scratch. In questa prima puntata cercheremo di prendere confidenza con l'ambiente di sviluppo e creeremo la nostra prima, semplice app. Prima puntata.

Le applicazioni per smartphone sono diventate una realtà dei nostri giorni, al punto che ormai ne scarichiamo ed utilizziamo una quantità considerevole per soddisfare le esigenze più disparate. In particolare, con l'avvento e la diffusione dei sistemi basati su Android, il numero di app è ulteriormente aumentato, trattandosi di applicazioni ancora più semplici da realizzare rispetto alla controparte iOS, vuoi per la maggiore semplicità della pubblicazione di una app sul Play Store rispetto all'Apple Store, vuoi per le caratteristiche dei linguaggi utilizzati e delle piattaforme di sviluppo.
Ciononostante, la realizzazione di una app Android richiede ancora delle competenze specifi-

Fig. 1 - Logo di Android.

Fig. 2 - Logo di MIT App Inventor.

che di programmazione, come la conoscenza del linguaggio Java ed una certa confidenza con il framework Android.
Per ovviare a questa problematica è stata sviluppata una serie di tools che permettono di sviluppare semplicemente app Android senza la necessità di essere dei programmatori Java provetti e conoscere dettagliatamente ambienti di sviluppo complessi. Uno dei tool di questo tipo più utilizzati (conta ad oggi circa 2 milioni di utenti sparsi per il mondo) è MIT App Inventor.

Introduzione a MIT App Inventor

Sviluppato originariamente da Google, nel 2011 il progetto è passato sotto la gestione del MIT (Massachussets Institute of Technology), prendendo la denominazione di MIT App Inventor EDU, o, più comunemente, MIT App Inventor; il relativo logo è riportato in **Fig. 2**.
L'uso di MIT App Inventor rende la realizzazione di una app Android un processo molto più intuitivo rispetto ai metodi tradizionali basati su linguaggi testuali; infatti per realizzare una app con App Inventor si fa uso di una serie di blocchi simili a pezzi di un puzzle per la creazione del codice. A partire da dicembre 2013 è stata rilasciata la versione 2 dell'ambiente, che va a sostituire la precedente versione beta (rinominata App Inventor Classic), introducendo una serie di migliorie e di fixing dei bug rispetto alla prima release.

Setup dell'ambiente e creazione di un Progetto

MIT App Inventor è un ambiente di sviluppo completamente on-line; per utilizzare l'ambiente bisogna dotarsi di un account Google e collegarsi all'indirizzo web *http://appinventor.mit.edu/explore/*. Nella pagina che si apre, bisogna poi cliccare sul pulsante "Create apps!", come illustrato in **Fig. 3**: verrà richiesto di selezionare un account Google e la relativa password ed una volta autenticati, anche di accettare le condizioni di utilizzo. Una volta accettate le condizioni, verrà aperta la pagina iniziale dell'ambiente di sviluppo on-line.
A questo punto è possibile selezionare il pulsante "Start New Project", presente in alto a sinistra, per creare un nuovo progetto. I file relativi ai progetti vengono mantenuti anch'essi su un database cloud e sono legati all'account Google utilizzato per l'autenticazione. Una volta selezionata l'opzione "Start New Project" si aprirà un pop-up all'interno del quale bisognerà inserire il nome del progetto; eseguita questa operazione il progetto comparirà tra l'elenco dei progetti ("My Projects", **Fig. 4**) e si aprirà la schermata di **Fig. 5**.

Debug ed Emulazione

Per poter testare e debuggare le applicazioni che andremo a creare ci occorre chiaramente un sistema

Fig. 3 - pagina iniziale di MIT App Inventor.

Fig. 4 - Nuovo progetto creato ed aggiunto alla lista "My Projects".

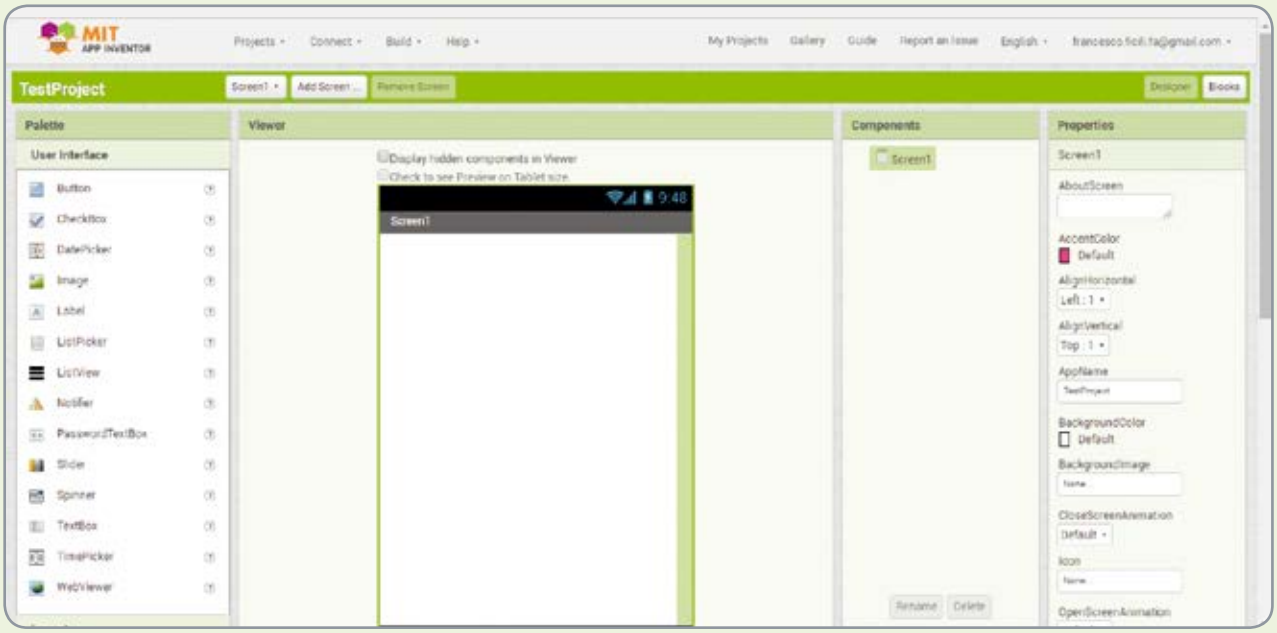

Fig. 5 - Schermata iniziale "TestProject".

per connetterci con un device di test, oppure un emulatore. App Inventor ci mette a disposizione ben tre opzioni per testare le nostre creazioni:

- test in real-time su device Android con connessione WiFi;
- test in real-time con Emulatore su PC,
- test in real-time su device Android con connessione USB.

In questa sezione descriveremo come effettuare il testing tramite le prime due opzioni (**Fig. 6**), in quanto consideriamo la terza semplicemente un sottocaso della prima, necessario solo in situazioni particolari nelle quali non sia disponibile la connessione WiFi sul nostro PC o sul device (caso peraltro estremamente raro). L'uso dell'emulatore è invece utile nel caso non si abbia a disposizione un dispositivo Android da utilizzare come device di test.

Test in real-time su device Android con WiFi

Occupiamoci della procedura per eseguire il test in real-time tramite connessione WiFi:

1. installare sul dispositivo target l'app "MIT AI2 Companion", che può essere comodamente scaricata dal Play Store; una volta scaricata l'app è sufficiente seguire i vari step per l'installazione;

2. connettere il PC di sviluppo e il dispositivo target sulla stessa rete WiFi;

3. connettere il progetto App Inventor al dispositivo, selezionando l'opzione "Connect→AI Companion" presente sulla project bar in alto, come illustrato in **Fig. 7**.

Una volta selezionata questa opzione apparirà una finestra di dialogo contenente un QR code: a questo punto potete lanciare l'app sul dispositivo e connetterla al PC tramite una delle due opzioni disponibili (scan del QR code oppure inserimento diretto del codice a 6 cifre). Dopo qualche secondo avverrà la connessione e dovreste vedere sullo schermo del vostro dispositivo l'app che state sviluppando. L'app stessa si aggiornerà ogni volta che farete una

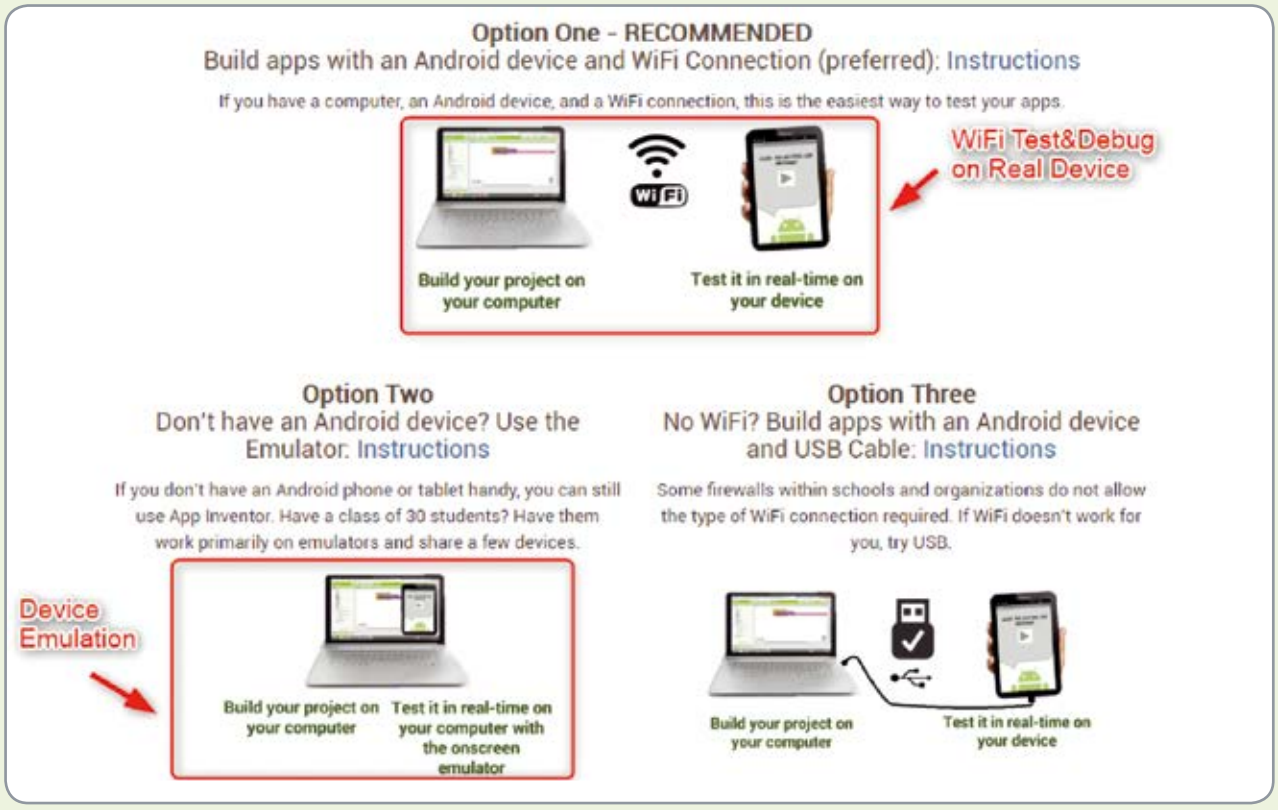

Fig. 6 - Possibili opzioni per il test e il debug delle app sviluppate con App Inventor.

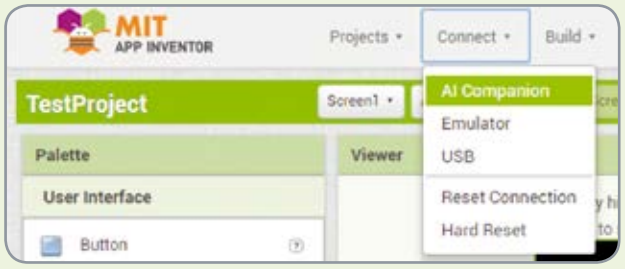

Fig. 7 - Connessione all'AI Companion.

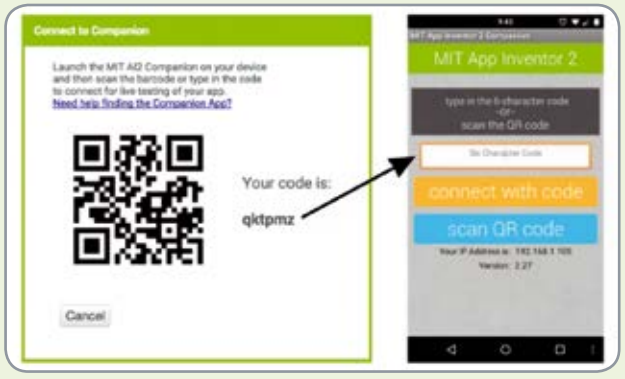

Fig. 8 - Scanning del QR code per connettere l'app.

modifica dall'ambiente di sviluppo, grazie ad una caratteristica chiamata "live testing".

Test in real-time con Emulatore

L'utilizzo dell'emulatore permette di testare le nostre app senza l'utilizzo di un device esterno Android, ma avvalendosi di un dispositivo virtuale che gira direttamente sul PC di sviluppo. È un'opzione molto utile in diverse circostanze e ce ne avvarremo per comodità didattica durante il corso, tutte le volte che sarà possibile.
Per poter utilizzare l'emulatore bisogna prima di tutto installarlo. Per farlo è necessario:
1. scaricare l'applicazione (questo link è valido per la versione Windows...) da *http://appinventor.mit.edu/explore/ai2/windows.html*; per le istruzioni complete per MAC OS e Linux vi rimandiamo all'apposita sezione sul sito di App Inventor (*http://appinventor.mit.edu/explore/ai2/setup-emulator.html#step2*).
2. una volta installato l'emulatore bisogna lanciarlo con un doppio clic sull'icona **ai Starter** che troverete sul desktop; verrà aperta una command window che visualizzerà sulla console i messaggi provenienti dall'aiStarter;

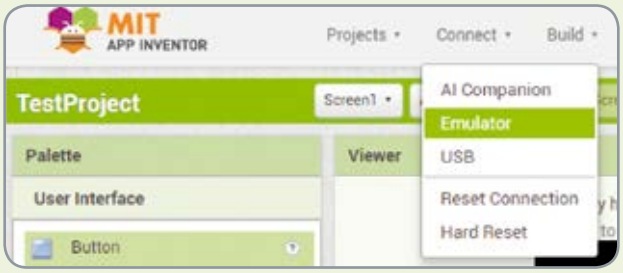

Fig. 9 - Connessione dell'emulatore dall'ambiente di sviluppo.

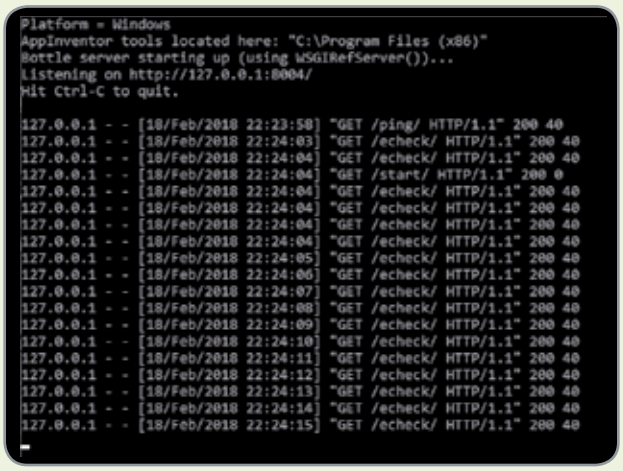

Fig. 10 - Connessione dell'emulatore, messaggi sulla console.

3. adesso è sufficiente selezionare l'opzione "Connect➔Emulator" presente sulla project bar in alto, come illustrato in **Fig. 9**.

A questo punto dovreste notare alcune attività in svolgimento sul lato aiStarter, come illustrato in **Fig. 10**, e dopo alcuni secondi verrà avviato anche l'emulatore del dispositivo; infine verrà sincronizzata l'app (**Fig. 11**) realizzata.
Da qui in poi il funzionamento è del tutto simile al caso del dispositivo esterno, a meno di limitazioni nell'uso delle periferiche.

Designer view e Block view

Passiamo ora all'analisi dell'ambiente di sviluppo ed in particolare soffermiamoci sulle due schermate principali nelle quali ci troveremo a lavorare; l'interfaccia di App Inventor è infatti costituita da due schermate principali di lavoro, chiamate **Designed View** e **Block View**: nella prima si sviluppa il layout della nostra app, ossia si costruisce la sua interfaccia grafica, mentre nella seconda si inserisce il codice vero e proprio, che, come abbiamo visto in precedenza, in App Inventor è totalmente grafico. Vediamo quindi come sono composte queste due View, partendo dalla Designer View ed aiutandoci con la **Fig. 12**: nella Designer View troviamo i seguenti componenti principali:

- ***Main Bar*** che è la barra principale di gestione del progetto, dalla quale è possibile gestire gli screen (aggiungere nuove viste o eliminare viste esistenti), effettuare tutte le operazioni di project management, eseguire il build di una app ed altro ancora;
- ***Components Palette***, che è la palette contenente i componenti e che, come di consueto, è divisa per categorie; i componenti sono una caratteristica molto importante di App Inventor, in quanto oltre a permettere la costruzione dell'interfaccia grafica e del layout, permettono di gestire praticamente tutte le periferiche presenti sul nostro Smartphone; consentono anche di eseguire operazioni avanzate, come ad esempio l'interfacciamento con i social, la gestione dello storage e della connettività, ecc.
- ***Viewer***: questa sezione rappresenta la nostra area di lavoro in fase di layout design, infatti è sagomata come lo schermo del nostro Smartphone; in

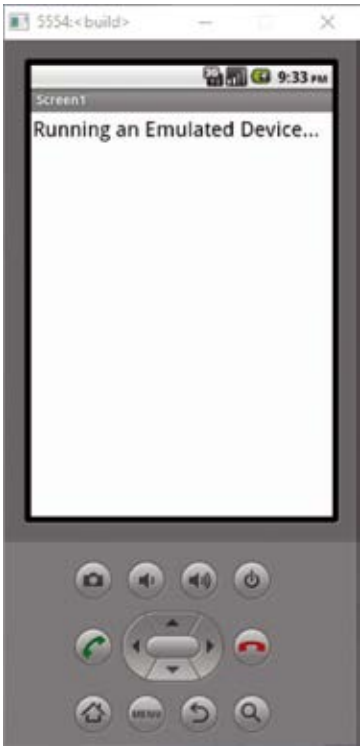

Fig. 11 - Sincronizzazione dell'app sul dispositivo emulato.

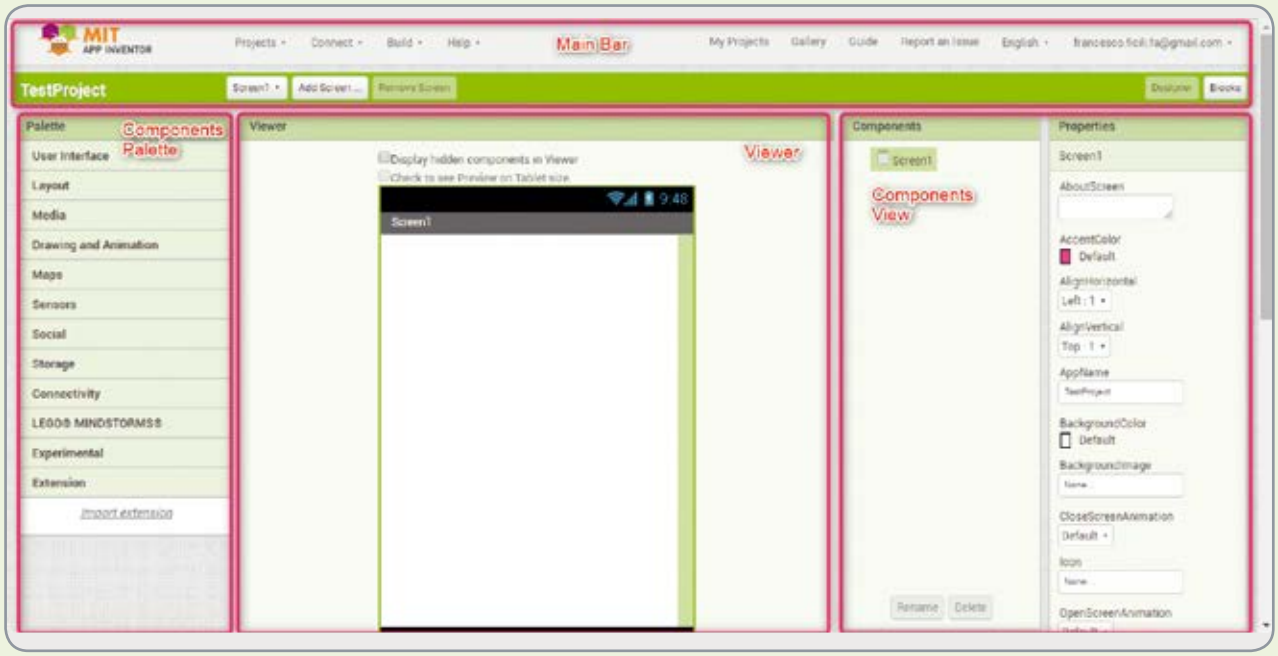

Fig. 12 - La Designer View.

Fig. 13 - Selezione della Block View.

essa è possibile trascinare i componenti prelevati dalla Components Palette.

- ***Components View***, dalla quale è possibile gestire le proprietà dei componenti, oltre che rinominare o eliminare gli stessi dal layout; questa view è divisa in due sezioni chiamate **Components** (da cui si può selezionare uno dei componenti presenti all'interno di un determinato screen) e **Properties**, dalla quale è possibile accedere le proprietà del componente selezionato.

Dalla Designer View è possibile passare alla Block View tramite il selettore presente alla estrema destra della Main Bar, selezionando la voce Blocks, come illustrato in **Fig. 13**. La Block View, visibile in **Fig. 14**, è la schermata all'interno della quale avviene lo sviluppo della logica della nostra app; essa è composta dai seguenti elementi:

- ***Main Bar***, in tutto e per tutto identica alla Main Bar esaminata nel caso della Designer View;
- ***Blocks Palette***, la quale contiene i blocchi per l'implementazione della nostra applicazione; tale palette è divisa in due parti: la prima (denominata Built-in) contiene tutti i blocchi base, come i blocchi logico/matematici, i blocchi per il controllo di esecuzione, le varie costanti e le varie operazioni di base che sono eseguibili sui blocchi, mentre la seconda contiene i blocchi specifici dei vari componenti del progetto e tale sezione si popola man mano che i componenti vengono aggiunti alla View lato design;
- ***Blocks Viewer***: questa è la sezione nella quale viene creato il codice grafico, trascinando all'interno di essa i vari blocchi presenti della Blocks Palette; per impedire errori nelle connessioni i blocchi in App Inventor sono sagomati a forma di puzzle e si possono incastrare tra di loro solo se risultano compatibili.

Esempio Grafico: Hello World

Ora che abbiamo preso un po' di confidenza con l'ambiente di sviluppo, passiamo alla realizzazione di un semplice esempio pratico che ci permetta di cogliere subito la potenza di questo strumento di programmazione: costruiamo il classico programma "Hello World", ma anziché far lampeggiare un LED o stampare la classica stringa "Hello World" a video, utilizziamo il TextToSPeech component per sintetizzare vocalmente la nostra stringa "Hello World".

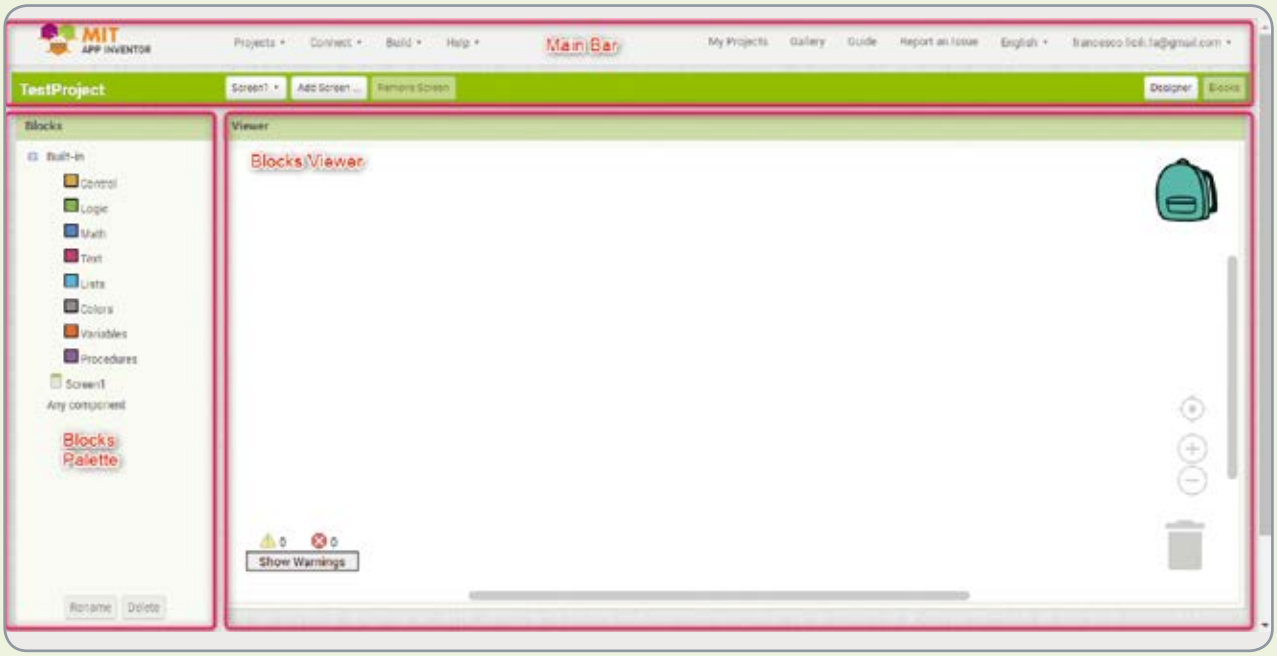

Fig. 14 - Block View.

Partiamo dalla Designer View ed inseriemo sul nostro layout i seguenti componenti:

- User Interface ➔ Button,
- Media ➔ TextToSpeech.

Come già accennato, per inserire i componenti sul layout occorre selezionarli dalla palette e poi trascinarli sulla Design Viewer. Una volta completata questa operazione, il nostro layout dovrebbe essere del tutto simile a quello riportato in **Fig. 15**.
A differenza del button il componente TextToSpeech è un componente non visibile, quindi la sua presenza all'interno del progetto viene indicata tramite una icona in basso sotto il layout.

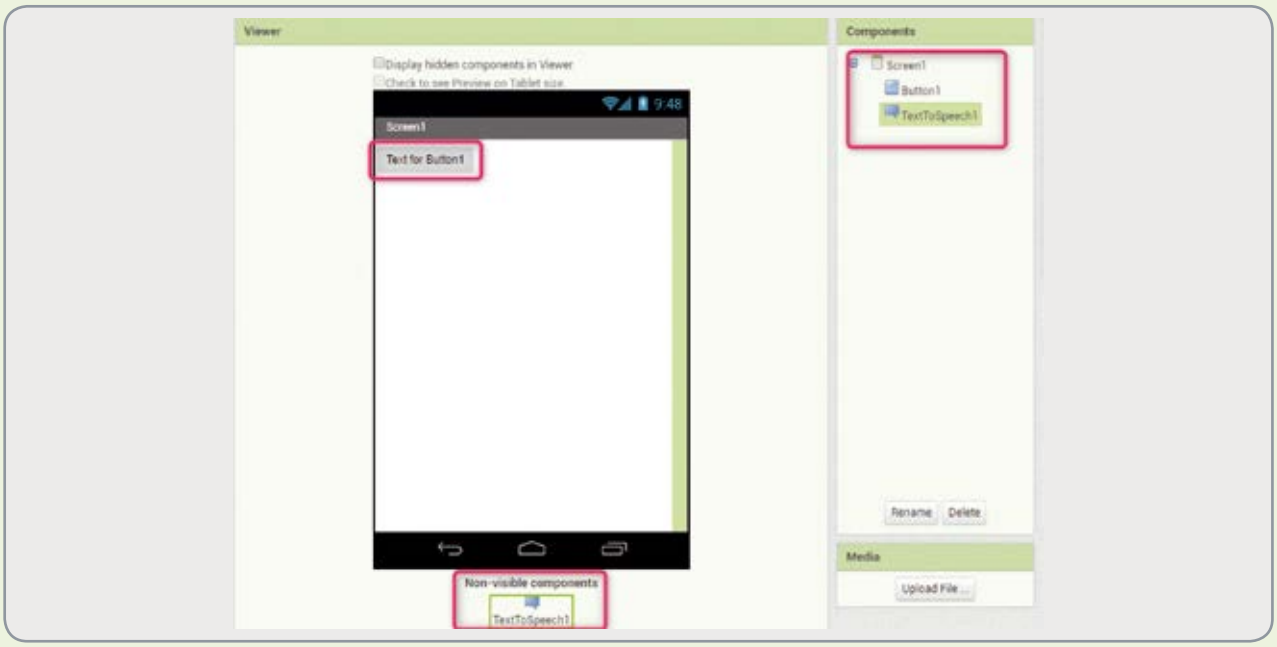

Fig. 15 - Inserimento dei componenti sulla Design View.

Breve storia di Android

Nell'ottobre del 2003 quattro programmatori statunitensi, Andy Rubin, Chris White, Nick Sears e Rich Miner fondarono una società, la Android Inc. per lo sviluppo di quello che Rubin definì "...dispositivi cellulari più consapevoli della posizione e delle preferenze del loro proprietario". Durante lo stesso anno il budget inizialmente stanziato per il progetto si esaurì ed un amico di Rubin, Steve Perlamn, rifinanziò il progetto con 10.000 dollari. Perlman consegnò a Rubin il denaro in una busta ma rifiutò ogni proposta di partecipazione alla società.
Il 17 agosto 2005 Google acquistò l'azienda, come acquisizione strategica per l'ingresso nel mercato della telefonia mobile. È in quel periodo che il team comincia a sviluppare un sistema operativo per dispositivi mobili basato su Linux. La presentazione ufficiale dell'OS avvenne il 5 novembre 2007 dalla neonata OHA (Open Handset Alliance), un consorzio di aziende del settore Hi Tech che include Google, produttori di smartphone, operatori di telefonia mobile e produttori di microprocessori come Qualcomm e TI. Il primo dispositivo equipaggiato con Android che venne lanciato sul mercato fu l'HTC Dream: era il 22 ottobre del 2008.
Dalla versione 1.5 di Android, ogni aggiornamento o rilascio dell'OS, segue una precisa convenzione alfabetica per i nomi, associando ogni release al nome di un dolciume:

- 1.5 ——— Cupcake,
- 1.6 ——— Donut,
- 2.0, 2.1 ——— Eclair,
- 2.2.x ——— Froyo,
- 2.3.x ——— Gingerbread,
- 3.0, 3.1, 3.2.x ——— Honeycomb,
- 4.0.x ——— Ice Cream Sandwich,
- 4.1.x, 4.2.x, 4.3.x — Jelly Bean,
- 4.4.x ——— KitKat
- 5.0.x, 5.1.x ——— Lollipop,
- 6.0.x ——— Marshmallow,
- 7.0, 7.1.x ——— Nougat,
- 8.0, 8.1 ——— Oreo

Fig. A - Logo di Android Oreo.

Per mantenere un minimo di ordine e coding style rinominiamo i componenti come segue:

- Button1 ➔ Btn_HelloWorld;
- TextToSpeech1 ➔ Cmp_TextToSpeech.

I componenti possono essere rinominati selezionandoli nella component view e cliccando sul tasto Rename, presente in basso sulla stessa view. Comparirà una finestra di dialogo dalla quale sarà possibile rinominare il componente.
Inoltre modifichiamo la proprietà Text del Button, cambiandola in "Hello World!!!", come indicato in **Fig. 16**. Poi settiamo la lingua predefinita per il componente TextToSpeech su "Italiano", attraverso il menu a tendina Language accessibile dalle proprietà del componente.
A questo punto possiamo passare nella Block View

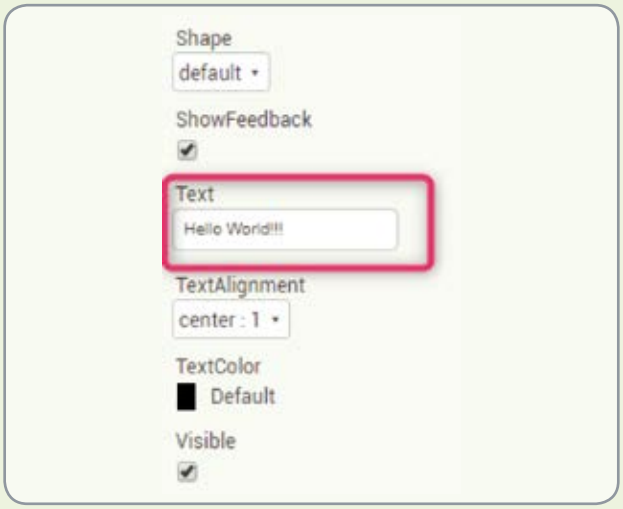

Fig. 16 - Aggiornamento proprietà Text del Button.

Fig. 17 - Blocco When Btn_HelloWorld.Click…do.

Fig. 18 - Blocco call Cmp_TextToSpeech.Speak.

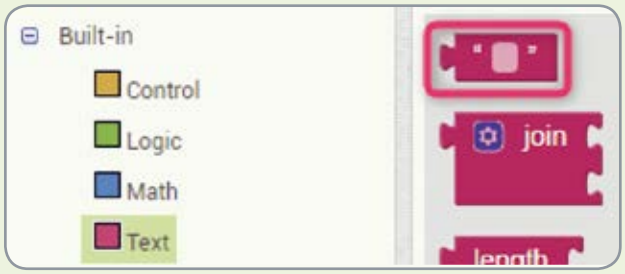

Fig. 19 - Blocco Text String.

per aggiungere il codice grafico alla nostra app. Procediamo come segue: per prima cosa posizioniamo sulla Viewer il blocco "When Btn_HelloWorld.

Fig. 20 - Codice grafico relativo all'esempio pratico.

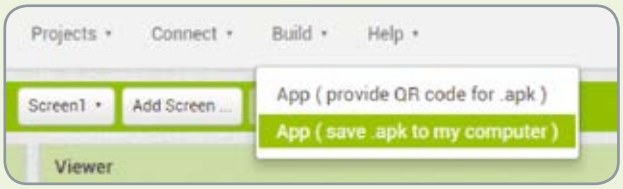

Fig. 21 - Building dell'App definitiva.

Click…do", presente tra i blocchi specifici del componente Btn_HelloWorld (**Fig. 17**). Poi incastriamo all'interno di quest'ultimo il blocco "call Cmp_TextToSpeech.Speak" del componente Cmp_TextToSpeech (Fig. 18). Infine colleghiamo alla connessione "message" di quest'ultimo un blocco text string (Fig. 19).
Infine scriviamo all'interno del blocco text string la stringa "Ciao Mondo!!!". Se avete fatto tutto correttamente il risultato dovrebbe essere visibile in **Fig. 20**.
Per testare la nostra app così creata possiamo utilizzare uno dei due metodi descritti precedentemente (test su dispositivo tramite connessione WiFi o uso dell'emulatore). Una volta sicuri del risultato possiamo anche eseguire il build della nostra app in formato .apk ed installarla sul nostro dispositivo mobile. Per farlo è sufficiente selezionare l'opzione Build➔App (save .apk on my computer) presente nella Main Bar, come illustrato in **Fig. 21**.
Completata la fase di Build, l'app verrà scaricata nella cartella predefinita per i download del browser che state utilizzando e potrete installarla sul vostro dispositivo mobile.

Conclusioni

Siamo arrivati alla conclusione di questa prima puntata di questo corso introduttivo a MIT App Inventor; in questa puntata abbiamo iniziato a prendere confidenza con l'ambiente, installato l'emulatore, analizzato l'interfaccia e creato un primo progetto di esempio. Dalla prossima puntata inizieremo con in primi esempi pratici di applicazioni Android, occupandoci dei concetti di base relativi alla programmazione grafica e dando un primo sguardo ai componenti, che sono uno degli elementi fondamentali di App Inventor.