MIT APP INVENTOR 6

di FRANCESCO FICILI

Continuiamo il nostro viaggio alla scoperta di MIT App Inventor, un tool di sviluppo per applicazioni Android creato dal MIT. In questa sesta e conclusiva puntata analizzeremo le famiglie di componenti maps e connectivity e vedremo come pubblicare le nostre app su google play.

Nella puntata precedente abbiamo proseguito lo studio e l'applicazione pratica dei componenti di MIT Appinventor, analizzando la famiglia di componenti **drawing and animations**, che ci permette di sviluppare applicazioni di gaming sui nostri dispositivi Android. Ricordiamo che i componenti sono gli elementi che permettono di aggiungere funzionalità all'app in fase di sviluppo con App Inventor.
Ebbene, in questa puntata concludiamo il discorso e presentiamo due ulteriori famiglie di componenti, ossia **maps** e **connectivity**, e vediamo anche come

fare per pubblicare, sullo store Google Play, le app che abbiamo creato.

La famiglia di componenti Maps

App Inventor dispone della famiglia di componenti Maps, appositamente creati per lo sviluppo di applicazioni di geolocalizzazione e geofencing (applicazioni che sfruttano l'interazione tra una mappa e le informazioni ricavate dal sensore di localizzazione). In **Fig. 1** è rappresentata un'immagine dei componenti della famiglia Maps.

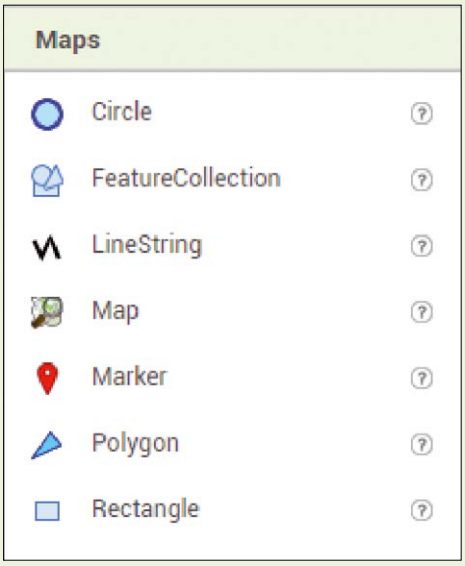

Fig. 1 - Famiglia di componenti Maps.

Non ci soffermeremo molto sulla descrizione di questi componenti in quanto non li utilizzeremo nell'esempio pratico presentato in questa puntata; ci limiteremo piuttosto a fornire una tabella riassuntiva che ne illustra le caratteristiche generali.

La famiglia di componenti connectivity

La famiglia connectivity di MIT App Inventor comprende questi quattro componenti:

- Activitiy Starter;
- Bluetooth Client;
- Blueetooth Server;
- Web.

Come potete dedurre dai loro nomi, si tratta principalmente di componenti per la gestione della connettività, in particolare Bluetooth e Web (sia tramite WiFi che tramite rete dati), oltre al componente **Activity Starter**, che permette di lanciare altre applicazioni. In **Fig. 2** è rappresentata la famiglia di componenti in oggetto.
Analizziamo più in dettaglio i vari componenti di questa famiglia, in modo da comprenderli più nel dettaglio.

Activity Starter

Il componente **Activity Starter** permette di lanciare un activity tramite il metodo start activity.
Una activity in Android può essere vista come una singola schermata (screen) che un utente può vedere sul suo dispositivo in un determinato momento. Sostanzialmente una app è composta da più activi-

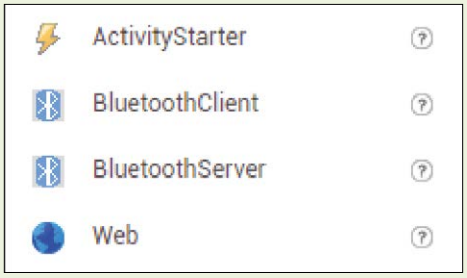

Fig. 2 - Famiglia di componenti connectivity.

COMPONENTE	DESCRIZIONE
Circle	Il circle component permette di visualizzare un cerchio di un dato raggio in metri and una certa latitudine e longitudine, all'interno di una mappa. Questo componente può essere utilizzato per l'implementazione di applicazioni di geofencing, utilizzando i metodi di interazione con il sensore di localizzazione.
FeatureCollection	Il FeatureCollection component raggruppa le feature di una o più mappe insieme.
LineString	Il LineString component permette di disegnare una sequenza continua di linee su una mappa.
Map	Il Map component è un container bidimensionale che permette di renderizzare mappe e piazzare vari tipi di marker al suo interno per identificare punti o zone. Le mappe sono fornite dal servizio OpenStreetMaps.
Marker	Il componente Marker permette di indicare punti in una mappa, come edifici o altri punti di interesse. I marker possono essere customizzati in diversi modi, come ad esempio cambiandone l'aspetto grafico utilizzando un asset grafico dell'app.
Polygon	Il componente Polygon permette di racchiudere un'area di dimensioni arbitrarie su di una mappa. L'uso principale di questo componente è quello di delimitare zone geografiche.
Rectangle	Il componente Rectangle permette di delimitare zone all'interno di un poligono rettangolare, con valori fissi per latitudine e longitudine dei vertici.

Tabella 1 - Componenti della famiglia Maps.

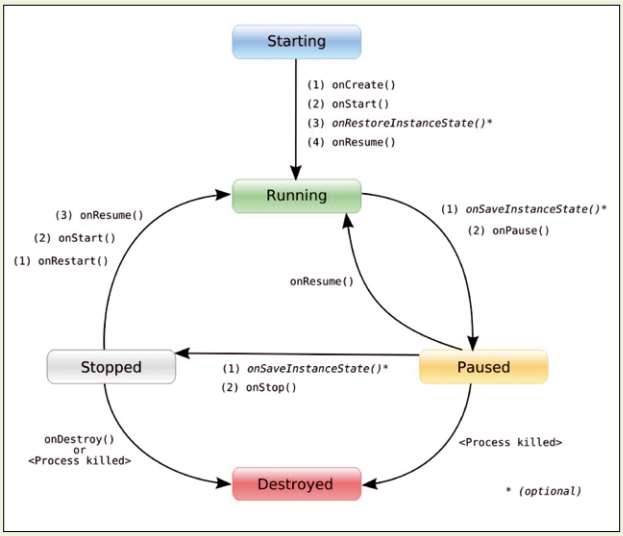

Fig. 3 - Ciclo di vita di una activity Android.

ty, ognuna delle quali può essere vista ed invocata da un'altra app, ed è a questa funzione che il componente **Activity Starter** assolve.
Una volta invocata, una activity ha anche un ciclo di vita, con tutta una serie di eventi correlati.
Il ciclo di vita delle activity in Android è riportato in **Fig. 3**.
Nella **Tabella 2** sono riportati, come di consueto, eventi, metodi e proprietà del componente.
Tra le activity che possono essere lanciate dal componente activity starter possiamo citare, giusto per fare qualche esempio.

- Avviare l'applicazione camera, cosa che può essere fatto impostando:
  - *Action: android.intent.action.MAIN*
  - *ActivityPackage: com.android.camera*
  - *ActivityClass: com.android.camera.Camera*

- Eseguire una ricerca su web. Assumendo di voler cercare "ElettronicaIn", bisognerebbe impostare:
  - *Action: android.intent.action.WEB_SEARCH*
  - *ExtraKey: query*
  - *ExtraValue: ElettronicaIn*
  - *ActivityPackage: com.google.android.providers.enhancedgooglesearch*
  - *ActivityClass: com.google.android.providers.enhancedgooglesearch.Launcher*

- Aprire il browser web ad una specifica pagina. Assumendo di voler aprire la pagina web di Elettronica In, bisognerebbe impostare:
  - *Action: android.intent.action.VIEW*
  - *DataUri: https://www.elettronicain.it/*

Bluetooth Client/Server

I moderni smartphone basati su Android sono normalmente dotati di un modulo bluetooth integrato. App Inventor modella il collegamento bluetooth tramite uno schema client/server; esistono quindi due componenti per la gestione del BT, a seconda che si voglia implementare il lato client o il lato server.
Il profilo BT maggiormente supportato è il profilo SPP (Serial Port Profile) che è il profilo base utilizzato dalla maggior parte dei metodi. C'è comunque la possibilità di connettersi tramite un profilo generico, ma il supporto specifico a livello di metodi è molto

PROPRIETÀ	
SINTASSI	DESCRIZIONE
Action	Action della activity.
Activity Class	Class della activity.
Activity Package	Package della activity.
Data Type	DataType della actrivity.
Data Uri	Data Uri della activity.
Extras	Questa proprietà accetta una lista di coppie key/value che possono poi essere utilizzate nel campo Extras della activity stessa. Sostituisce le proprietà obsolete ExtraKey ed ExtraValue.
EVENTI	
SINTASSI	DESCRIZIONE
AfterActivity(text result)	Evento triggerato quando il metodo StartActivity ritorna.
ActivityCanceled()	Evento triggerato quando il metodo StartActivity ritorna prematuramente, nel caso l'activity sia stata cancellata.
METODI	
SINTASSI	DESCRIZIONE
text ResolveActivity()	Ritorna il nome dell'activity corrispondente all'activityStarter, oppure una stringa vuota se l'activity corrispondente non viene trrovata.
StartActivity()	Lancia l'acitivity corrispondente all'activityStarter.

Tabella 2 - Eventi, metodi e proprietà del componente Activity Starter.

limitato in questo caso.
Nella **Tabella 3** sono riportati eventi, metodi e proprietà del componente BT client.
Invece nella **Tabella 4** sono riportati eventi, metodi e proprietà del componente BT Server.

Web

Il componente **Web** è un elemento di App Inventor non visibile che permette di gestire richieste http GET, POST, PUT e DELETE. Il componente prescinde dal tipo di connessione fisica utilizzata (WiFi o GSM/GPRS), fornendo un'interfaccia uniforme per l'accesso alla rete.
Nella **Tabella 5** sono riportati, come di consueto, eventi, metodi e proprietà del componente.

Esempio Pratico

Passiamo adesso, come di consueto, al nostro esempio pratico, per il quale utilizzeremo il componente BT, creando un semplice terminale BT con profilo SPP (Serial Port Profile).
Come di consueto partiamo dal layout; per realizzare il nostro terminale ci occorre:

- un componente BT client, che chiameremo Cmp_BtClient;
- un componente Clock, Cmp_Clock;
- un ListPicker, che chiameremo Lst_DevPicker;
- quattro Label, che chiameremo rispettivamente Lbl_StatusLbl, Lbl_BtSts, Lbl_Receive ed Lbl_ReceiveLbl;
- un pulsante Btn_Send;
- una TextBox Txt_Send.

Il layout dell'app può essere organizzato in tanti modi, in **Fig. 4** ne proponiamo uno.
In questo caso abbiamo posizionato il list picker in alto e questo elemento ci servirà per selezionare il dispositivo BT con il quale intendiamo iniziare una connessione. Sotto al picker abbiamo inserito una label che ci indicherà lo stato della connessione. Più in basso ancora la text box ed il pulsante per l'invio ed infine la label sulla quale stamperemo le stringhe in ricezione. Per quanto riguarda la ricezione configureremo il componente clock per generare un

PROPRIETÀ	
SINTASSI	DESCRIZIONE
AddressesAndNames	Proprietà che contiene indirizzi e nomi dei dispositivi BT accoppiati.
Available	Proprietà che indica se il modulo BT è presente sul dispositivo.
Enabled	Proprietà che indica se il modulo BT è abilitato.
IsConnected	Proprietà che indica se è stata stabilita una connessione.
DelimiterByte	Proprietà che permette di settare il delimiterByte (byte di delimitazione).
EVENTI	
SINTASSI	DESCRIZIONE
Nessuno	
METODI	
SINTASSI	DESCRIZIONE
boolean Connect(text address)	Metodo che permette di effettuare una connessione con il dispositivo il cui indirizzo è passato come parametro, mediante profilo SPP. Ritorna un boolean che indica se la connessione è andata a buon fine.
boolean ConnectWithUUID(text address, text uuid)	Metodo che permette di effettuare una connessione con il dispositivo il cui indirizzo è passato come parametro, mediante un profilo generico (UUID). Ritorna un boolean che indica se la connessione è andata a buon fine.
Disconnect()	Metodo che permette di disconnettersi dal dispositivo BT con il quale si è connessi.
boolean IsDevicePaired(text address)	Metodo che indica se un determinato dispositivo, il cui indirizzo è passato come parametro, è accoppiato.
text ReceiveText(number numberOfBytes)	Metodo che ritorna una stringa di testo ricevuta dal dispositivo connesso, di lunghezza pari al parametro NumberOfBytes. Se il parametro NumberOfBytes è minore di 0 vengono letti tutti i bytes ricevuti prima del carattere di delimitazione.
list ReceiveSignedBytes(number numberOfBytes)	Metodo che ritorna lista di bytes con segno ricevuta dal dispositivo connesso, di lunghezza pari al parametro NumberOfBytes. Se il parametro NumberOfBytes è minore di 0 vengono letti tutti i bytes ricevuti prima del carattere di delimitazione.
list ReceiveUnsignedBytes(number numberOfBytes)	Metodo che ritorna lista di bytes senza segno ricevuta dal dispositivo connesso, di lunghezza pari al parametro NumberOfBytes. Se il parametro NumberOfBytes è minore di 0 vengono letti tutti i bytes ricevuti prima del carattere di delimitazione.
SendBytes(list list)	Metodo che consente di inviare una lista di bytes al dispositivo BT connesso.
SendText(text text)	Metodo che consente di inviare una stringa di caratteri al dispositivo BT connesso.

Tabella 3 - Eventi, metodi e proprietà del componente BT Client.

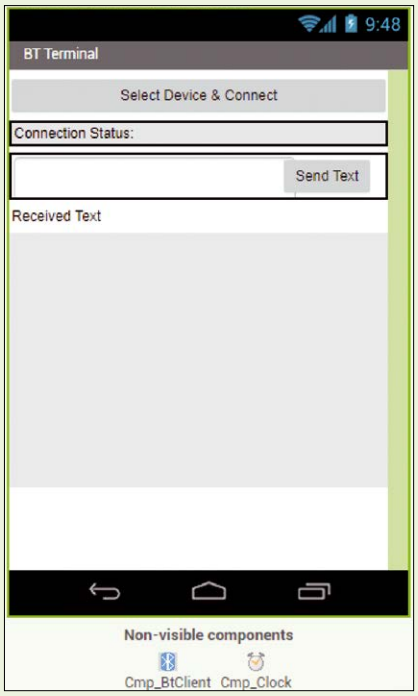

Fig. 4 - Layout dell'applicazione di esempio.

evento ogni 200ms e all'interno dell'evento accederemo le stringhe ricevute dal BT se ce ne sono.
Analizziamo adesso i behaviours, partendo dall'inizializzazione e dalla connessione con il server BT, operazioni rappresentate dai blocchi in **Fig. 5**.
In corrispondenza dell'evento "Initialize" dello screen viene inizializzata la label Lbl_BtSts. Successivamente si popola la lista Lst_DevPicker con indirizzi e nomi dei dispositivi BT associati, prendendoli dalla proprietà AddresseAndNames del componente, in corrispondenza dell'evento "BeforePicking". La selezione ottenuta in questa fase (Lst_DevPicker.Selection) viene utilizzata in corrispondenza dell'evento "AfterPicking" per connettersi col dispositivo sfruttando il metodo "Connect" del componente BT Client. Sempre in questo evento viene aggiornato il valore della label Lbl_BtStatus a "Connected". La gestione dell'invio e della ricezione dei dati è molto semplice. Per inviare una stringa, in corrispondenza dell'evento "Btn_Send.Click" viene invocato il metodo "SendText" del componente BT Client (viene preventivamente verificato che il componente sia connesso), come illustrato in **Fig. 6**.

PROPRIETÀ	
SINTASSI	DESCRIZIONE
Available	Proprietà che indica se il modulo BT è presente sul dispositivo.
DelimiterByte	Proprietà che permette di settare il delimiterByte (byte di delimitazione).
Enabled	Proprietà che indica se il modulo BT è abilitato.
IsAccepting	Proprietà che indica se il componente BT server sta accettando una connessione
IsConnected	Proprietà che indica se è stata stabilita una connessione.
EVENTI	
SINTASSI	DESCRIZIONE
ConnectionAccepted()	Evento che segnala che la connessione BT è stata accettata.
METODI	
SINTASSI	DESCRIZIONE
AcceptConnection(text serviceName)	Accetta una connessione via BT tramite profilo SPP.
AcceptConnectionWithUUID(text serviceName, text uuid)	Accetta una connessione via BT tramite profilo generico.
Disconnect()	Metodo che permette di disconnettersi dal dispositivo BT con il quale si è connessi.
number BytesAvailableToReceive()	Metodo che ritorna il numero di bytes che possono essere ricevuti.
list ReceiveSignedBytes (number numberOfBytes)	Metodo che ritorna lista di bytes con segno ricevuta dal dispositivo connesso, di lunghezza pari al parametro NumberOfBytes. Se il parametro NumberOfBytes è minore di 0 vengono letti tutti i bytes ricevuti prima del carattere di delimitazione.
text ReceiveText (number numberOfBytes)	Metodo che ritorna una stringa di testo ricevuta dal dispositivo connesso, di lunghezza pari al parametro NumberOfBytes. Se il parametro NumberOfBytes è minore di 0 vengono letti tutti i bytes ricevuti prima del carattere di delimitazione.
list ReceiveUnsignedBytes (number numberOfBytes)	Metodo che ritorna lista di bytes senza segno ricevuta dal dispositivo connesso, di lunghezza pari al parametro NumberOfBytes. Se il parametro NumberOfBytes è minore di 0 vengono letti tutti i bytes ricevuti prima del carattere di delimitazione.
SendBytes(list list)	Metodo che consente di inviare una lista di bytes al dispositivo BT connesso.
SendText(text text)	Metodo che consente di inviare una stringa di caratteri al dispositivo BT connesso.
StopAccepting()	Metodo per interrompere l'accettazione di una connessione.

***Tabella 4** - Eventi, metodi e proprietà del componente BT Server.*

Per la ricezione invece si sfrutta il componente clock. Ad ogni occorrenza dell'evento "Time" viene aggiornata la label Lbl_Receive, accodando i caratteri ricevuti sul canale BT mediante l'applicazione del metodo "ReceiveText", come illustrato in **Fig. 7**. Anche in questo caso si verifica se il componete BT è connesso).
In **Fig. 8** è visibile uno screenshot che rappresenta il funzionamento su un dispositivo reale: è stato utilizzato un HC-05 (disponibile sul sito *www.futurashop.it/HC05*) che effettua un loopback sulla linea seriale.

Pubblicare le app su google play

Passiamo adesso ad una breve guida alla pubblicazione di una applicazione sullo store Google Play. Gli step necessari alla pubblicazione di un'app sono diversi e molti sono opzionali a seconda di quanto dettagliate si vogliano fare le descrizioni, eventuali contenuti aggiuntivi, etc. In questa sede noi ci limiteremo a spiegare i passi minimi indispensabili per la pubblicazione, lasciando al lettore interessato l'esplorazione dei vari step opzionali.
Per pubblicare un'app sullo store di Google, il primo step da fare è registrarsi come sviluppatore,

PROPRIETÀ	
SINTASSI	DESCRIZIONE
AllowCookies	Proprietà che indica se i cookies provenienti da una risposta debbano essere salvati ed utilizzati in richieste successive. Il supporto ai cookies è disponibile solo da Android 2.3.
RequestHeaders	Header della richiesta, come lista di due elementi. Il primo elemento rappresenta il field name della richiesta, mentre il secondo rappresenta il valore.
SaveResponse	Proprietà che indica se la risposta debba essere salvata su di un file.
ResponseFileName	Il nome del file nel quale salvare la risposta. Se la proprietà SaveResponse è settata su True e ResponseFileName è vuoto, viene creato un nuovo file con un nome di default.
Url	Url della web request.
EVENTI	
SINTASSI	DESCRIZIONE
GotFile(text url, number responseCode, text responseType, text fileName)	Evento che indica che una richiesta è stata servita e la risposta è stata salvata su un file.
GotText(text url, number responseCode, text responseType, text responseContent)	Evento che indica che una richiesta è stata servita ed il contenuto della risposta viene passato come stringa.
METODI	
SINTASSI	DESCRIZIONE
text BuildRequestData(list list)	Metodo che converte una lista di coppie di elementi nome-valore in una stringa formattata correttamente per l'uso del metodo PostText.
ClearCookies()	Cancella tutti i cookies del web component.
Delete()	Esegue una richiesta http DELETE utilizzando la proprietà url e recupera la risposta. Se la proprietà SaveResponse è settata a true, la risposta verrà salvata nel file indicato e verrà triggerato l'evento GotFile, altrimenti verrà triggerato l'evento GotText.
Get()	Esegue una richiesta http GET utilizzando la proprietà url e recupera la risposta. Se la proprietà SaveResponse è settata a true, la risposta verrà salvata nel file indicato e verrà triggerato l'evento GotFile, altrimenti verrà triggerato l'evento GotText.
PostFile(text path)	Esegue una richiesta http POST utilizzando la proprietà url ed i dati provenienti dal file specificato come parametro. Se la proprietà SaveResponse è settata a true, la risposta verrà salvata nel file indicato e verrà triggerato l'evento GotFile, altrimenti verrà triggerato l'evento GotText.
PostText(text text)	Esegue una richiesta http POST utilizzando la proprietà url e la stringa dati passata come parametro. I caratteri della stringa dati saranno codificati mediante codifica UTF-8. Se la proprietà SaveResponse è settata a true, la risposta verrà salvata nel file indicato e verrà triggerato l'evento GotFile, altrimenti verrà triggerato l'evento GotText.
PutFile(text path)	Esegue una richiesta http PUT utilizzando la proprietà url ed i dati provenienti dal file specificato come parametro. Se la proprietà SaveResponse è settata a true, la risposta verrà salvata nel file indicato e verrà triggerato l'evento GotFile, altrimenti verrà triggerato l'evento GotText.
PutText(text text)	Esegue una richiesta http PUT utilizzando la proprietà url e la stringa dati passata come parametro. I caratteri della stringa dati saranno codificati mediante codifica UTF-8. Se la proprietà SaveResponse è settata a true, la risposta verrà salvata nel file indicato e verrà triggerato l'evento GotFile, altrimenti verrà triggerato l'evento GotText.
any JsonTextDecode(text jsonText)	Decodifica una stringa JSON. Molto utile per la decodifica delle risposte server.
any XMLTextDecode(text XmlText)	Decodifica una stringa XML. Molto utile per la decodifica delle risposte server.

***Tabella 5** - Eventi, metodi e proprietà del componente Web.*

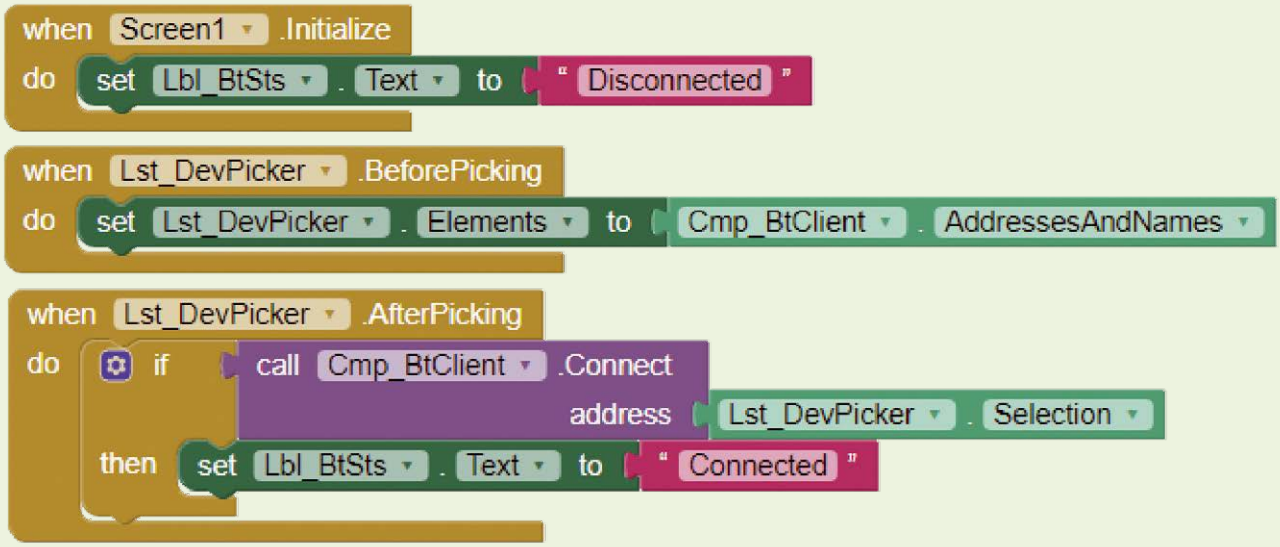

Fig. 5 - Inizializzazione e connessione con il dispositivo.

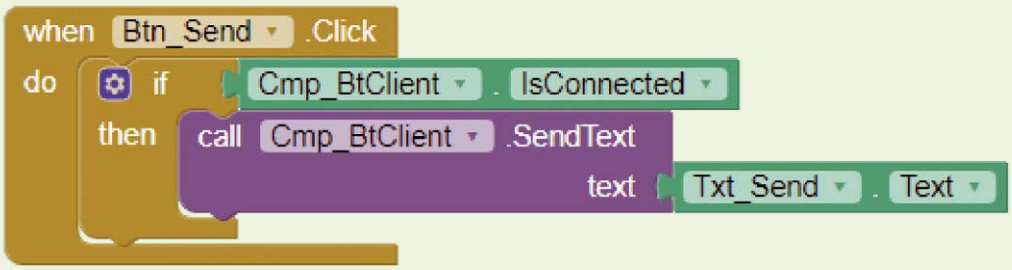

Fig. 6 - Invio dei dati.

cosa che può essere fatta dal seguente link: *https://play.google.com/apps/publish/signup/*.
Notate che per consentirvi di effettuare la registrazione Google richiede una fee (un diritto, vale a dire la **Quota di registrazione**) di 25 dollari, quindi è necessario disporre di una carta di credito per effettuare il pagamento mediante il form proposto.
La fee è a vita, quindi una volta che l'avrete corrisposta, non dovrete affrontare alcun costo aggiuntivo per ottenere un account da sviluppatore.
Una volta ottenuto l'account da sviluppatore potete accedere alla Google Play Console ed iniziare la procedura per la pubblicazione della vostra app.
Per iniziare la procedura cliccate sul pulsante "Pubblica un'applicazione Android su Google Play", come illustrato in **Fig. 9**.
A questo punto si aprirà la finestra di pop-up di creazione dell'applicazione (**Fig. 10**) dove dovrete inserire la lingua predefinita ed il titolo, per poi cliccare sul pulsante "crea".
Dopo aver creato l'istanza per la distribuzione, bisogna compilare una per una le sezioni evidenziate in **Fig. 11**, ossia:

- scheda dello store;
- versioni dell'app;
- classificazione dei contenuti;
- prezzi e distribuzione.

Una volta completata una sezione comparirà l'icona con la spunta di colore verde a fianco.
Tenete in considerazione che tutti i campi che è

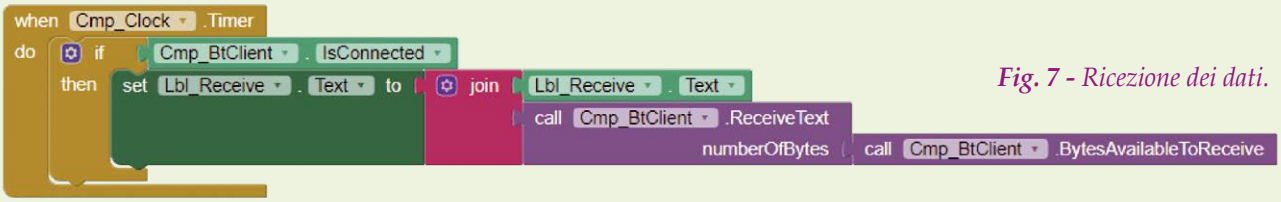

Fig. 7 - Ricezione dei dati.

Fig. 8 - Applicazione di esempio.

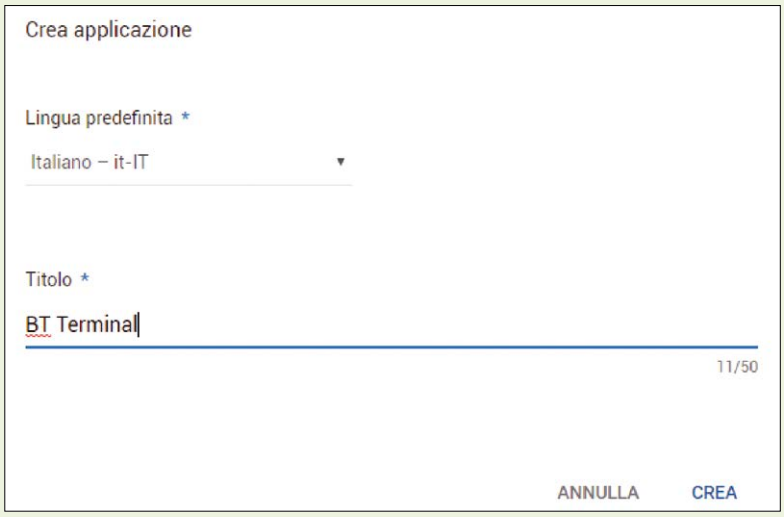

Fig. 10 - Finestra di pop-up di creazione applicazione.

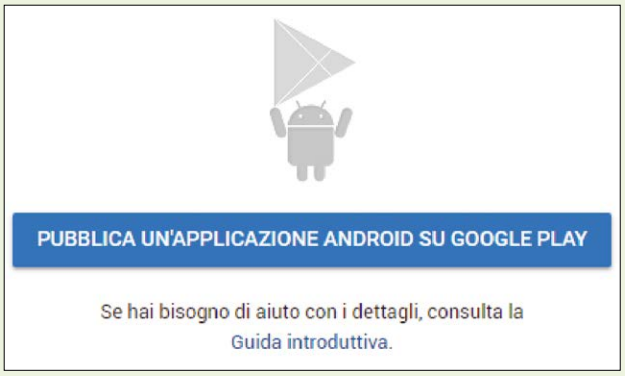

Fig. 9 - Pulsante per avviare la procedura di pubblicazione.

obbligatorio compilare sono contrassegnati con un asterisco.
In **Fig. 12** è rappresentato un esempio di compilazione dei campi iniziali della sezione "Scheda dello store".
Una volta riempite le varie sezioni bisogna tornare sulla sezione Versioni dell'App e cliccare sul pulsante blu "Esamina" in basso.
Nella schermata successiva troveremo un breve riepilogo e potremo pubblicare l'app cliccando sul pulsante "Inizia implementazione in versione di produzione", come illustrato in **Fig. 13**.
A questo punto l'app passerà nello stato "in attesa di pubblicazione" e Google Play vi notificherà non appena la pubblicazione sarà effettiva.

Conclusioni

In questa sesta e conclusiva puntata abbiamo visto le famiglie di componenti maps e connectivity, approfondendo tramite un esempio pratico un componente di quest'ultima famiglia. Inoltre abbiamo visto come pubblicare le app sullo store ufficiale di Google (Google Play).

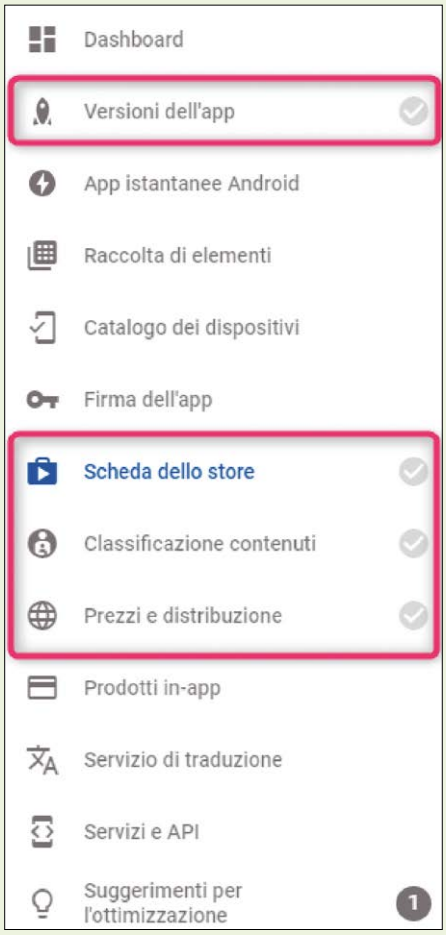

Fig. 11 - Sezioni da compilare per la pubblicazione sullo store.

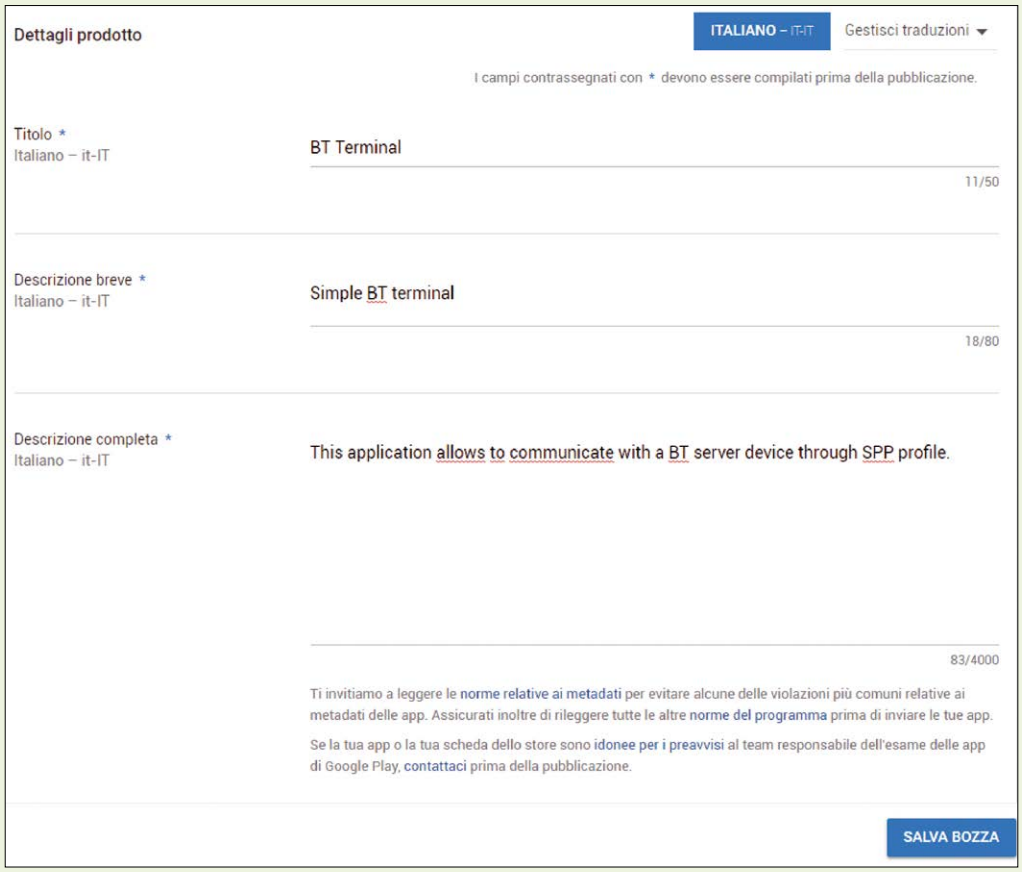

Fig. 12 - Campi iniziali della sezione scheda dello store.

Questa puntata completa la nostra panoramica su questo semplice ed efficace strumento di sviluppo per applicazioni Android e getta le basi per l'utilizzo di App Inventor quale complemento di vari dispositivi elettronici gestiti da app e progetti futuri che vi proporremo.

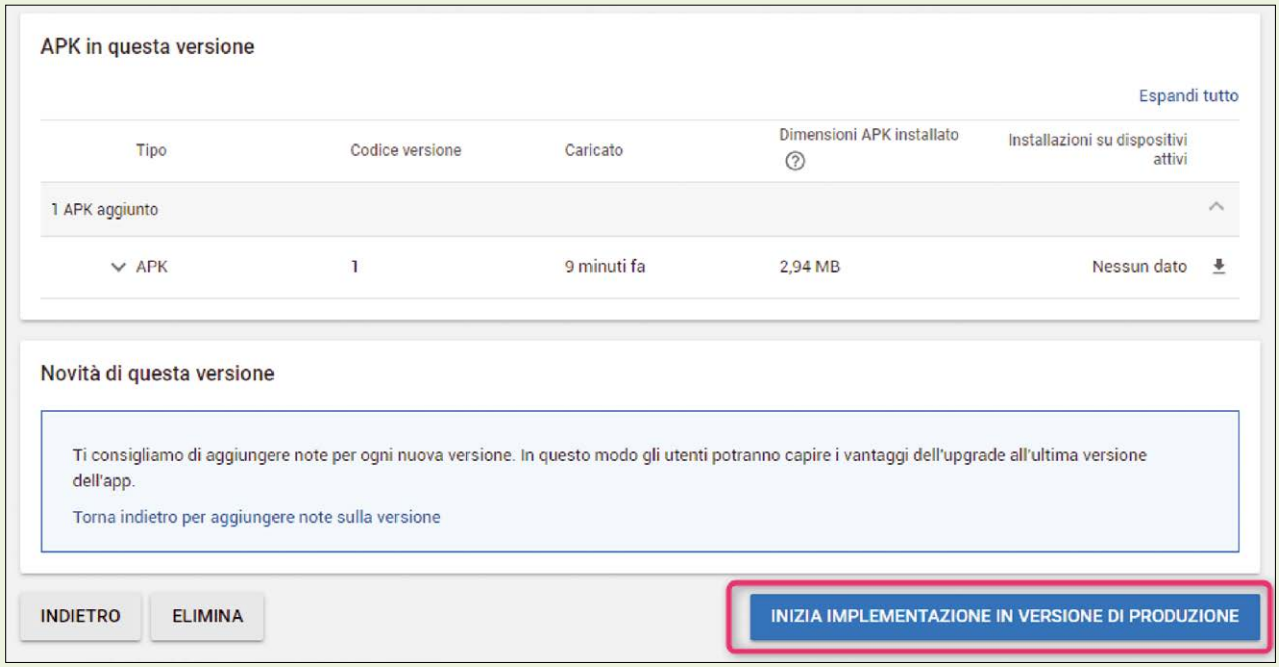

Fig. 13 - Pubblicazione dell'app sullo store.