APP di gestione dei lettori RFID

I lettori RFID BLE di TERTIUM Technology

TERTIUM Technology produce e commercializza con il nome **BlueBerry** dispositivi lettori RFID HF (ISO) e UHF (EPC) portatili che presentano un'interfaccia BLE (*Bluetooth Low Energy*) utilizzabile da uno smartphone o da un tablet.

I clienti di TERTIUM Technology che acquistano lettori RFID **BlueBerry** sviluppano APP di gestione per i più diffusi sistemi operativi per smartphone e tablet che supportano lo standard di comunicazione BLE, in particolare Google Android e Apple iOS.

Le librerie native di gestione a basso livello dei dispositivi BLE di TERTIUM Technology

TERTIUM Technology ha sviluppato 3 librerie native denominate **`TxRxLib`** per la gestione a basso livello dei propri dispositivi dotati di interfaccia BLE:

- libreria per sistema operativo Android in linguaggio Java;
- libreria per sistema operativo iOS in linguaggio Objective-C;
- libreria per sistema operativo iOS in linguaggio Swift.

Le librerie **`TxRxLib`**, oltre alle funzionalità di ricerca e riconoscimento dei dispositivi BLE, espongono una API per l'invio di comandi e la ricezione di risposte nella forma di stringhe composte da caratteri ASCII; in conseguenza del fatto che tutti i dispositivi BLE realizzati da TERTIUM Technology espongono un protocollo di comandi e risposte utilizzando due caratteristiche denominate rispettivamente *Tx* e *Rx*, le librerie di basso livello **`TxRxLib`** consentono di gestire qualsiasi dispositivo dotato di interfaccia BLE realizzato da TERTIUM Technology, non solo i lettori RFID HF/UHF **BlueBerry**[1].

Le librerie di gestione a basso livello dei dispositivi BLE di TERTIUM Technology sono documentate secondo gli standard previsti dagli IDE utilizzati per lo sviluppo.

Le APP di test delle librerie native di gestione a basso livello dei dispositivi BLE di TERTIUM Technology

Allo scopo di testare le librerie native di gestione a basso livello dei propri dispositivi dotati di interfaccia BLE, TERTIUM Technology ha realizzato tre APP:

- APP nativa per sistema operativo Android realizzata in linguaggio Java;
- APP nativa per sistema operativo iOS realizzata in linguaggio Objective-C;
- APP nativa per sistema operativo iOS realizzata in linguaggio Swift;

Le APP di test espongono tutte una GUI costituita da 2 viste:

- vista con elenco dei dispositivi BLE rilevati e possibilità di selezione e connessione ad un dispositivo specifico;
    - vista con elenco dei comandi inviati ad un dispositivo BLE di TERTIUM Technology, delle relative risposte e delle condizioni di errore generate, controllo di input per digitare ed inviare una stringa di comando.

[1] in particolare le librerie native di basso livello **TxRxLib** consento la gestione dei cosiddetti dispositivi "attivi": sensori BLE e *gateway* di accesso alle reti di sensori *wireless*

Le APP di test native inoltre espongono sulla rete Wi-Fi un *socket* TCP al quale è possibile connettersi per inviare ad un dispositivo BLE connesso stringhe di comando e ricevere le relative risposte e le condizioni di errore generate.

La figura che segue illustra l'architettura software complessiva delle librerie native e delle APP di test per la gestione a basso livello dei dispositivi BLE di TERTIUM Technology:

<table>
<tr><td>APP di test nativa
(Java)</td><td>APP di test nativa
(Objective-C)</td><td>APP di test nativa
(Swift)</td></tr>
<tr><td>libreria nativa Tx/Rx
(Java)</td><td>libreria nativa Tx/Rx
(Objective-C)</td><td>libreria nativa Tx/Rx
(Swift)</td></tr>
<tr><td>sistema operativo Android</td><td colspan="2">sistema operativo iOS</td></tr>
</table>

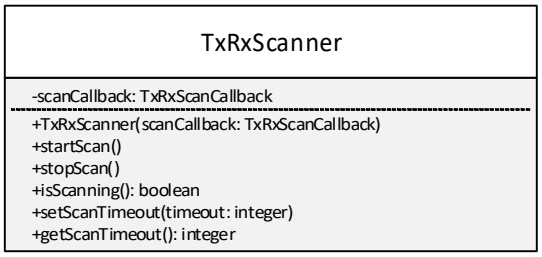
Le funzionalità delle librerie **`TxRxLib`** e delle relative APP native di test sono descritte nei seguenti documenti allegati:

- "TERTIUM_TxRxApp per sistema operativo Android"
- "TERTIUM_TxRxApp per sistema operativo iOS"

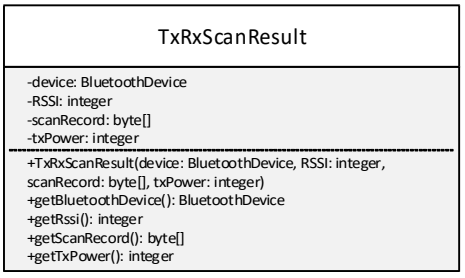
La API di gestione ad alto livello dei lettori RFID BLE di TERTIUM Technology

Per i lettori RFID HF/UHF BLE **BlueBerry** di TERTIUM Technology è stata progettata ed implementata, sia per sistema operativo Android che per sistema operativo iOS, una API di gestione asincrona ad alto livello che incapsula le funzionalità della libreria di basso livello **`TxRxLib`** basata sulle seguenti classi:

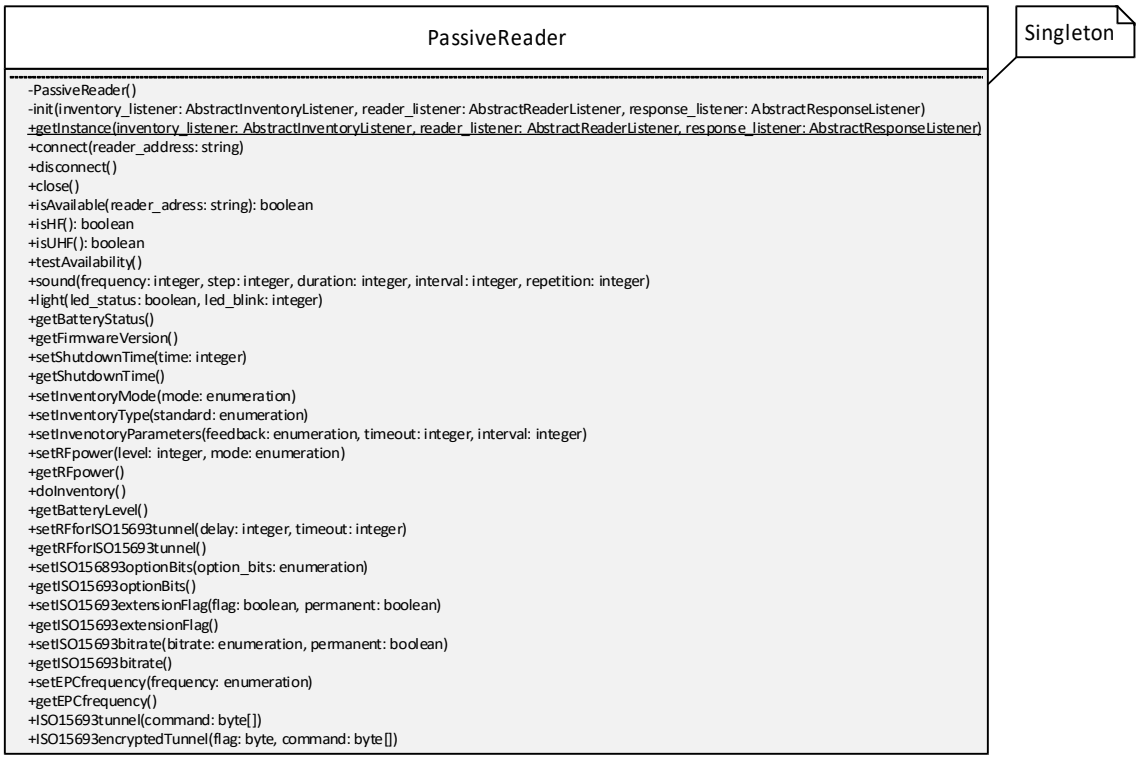
- ***TxRxScanner*** – espone i metodi per la ricerca di dispositivi BLE



- ***TxRxScanResult*** – incapsula un oggetto istanza della classe *BluetoothDevice* che rappresenta un dispositivo rilevato più altre informazioni relative alla rilevazione (RSSI del segnale radio, potenza di trasmissione del beacon, contenuto di advertisement del beacon, …)

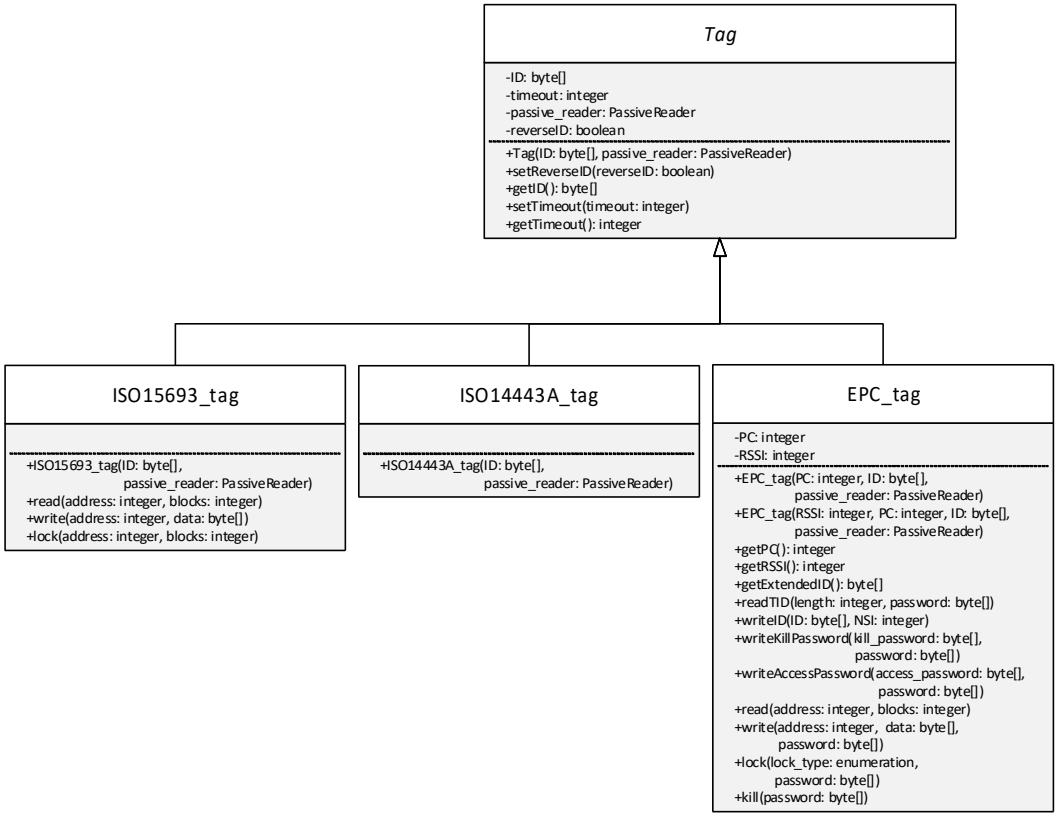


- ***PassiveReader*** – espone i metodi per il controllo del lettore RFID HF/UHF fornendo un livello di astrazione indipendente dalle stringhe di comando del protocollo



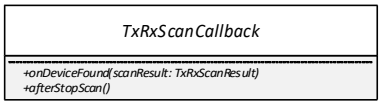
-

- ***Tag*** – rappresenta un *tag* RFID, le classi derivate ***ISO15693_tag***, ***ISO14443_tag*** e ***EPC_tag*** espongono i metodi di lettura, scrittura e gestione di un tag di uno specifico tipo anche in questo caso fornendo un livello di astrazione indipendente dalle stringhe di comando del protocollo

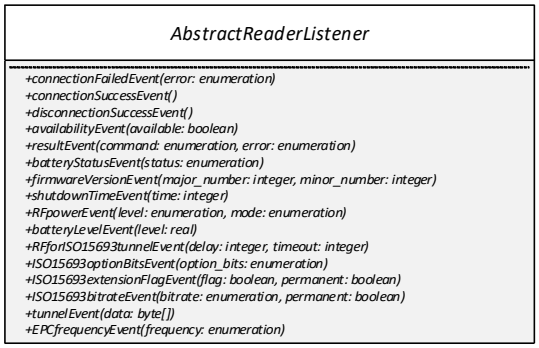


e sulle seguenti interfacce:

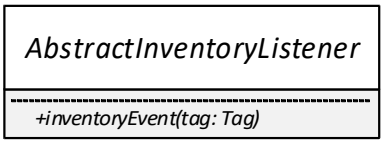
- ***TxRxScanCallback*** – definisce i metodi di callback per la rilevazione dei dispositivi BLE



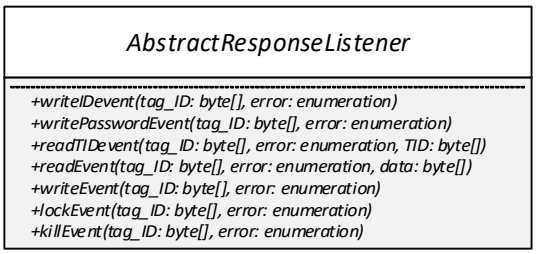
- ***AbstractReaderListener*** – definisce i metodi di callback relativi ai metodi della classe *PassiveReader*



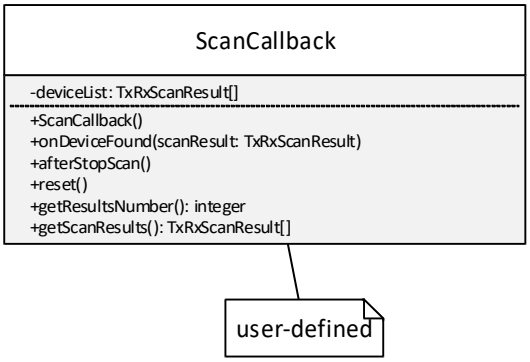
- ***AbstractInventoryListener*** – definisce il metodo di callback per la rilevazione di tag RFID



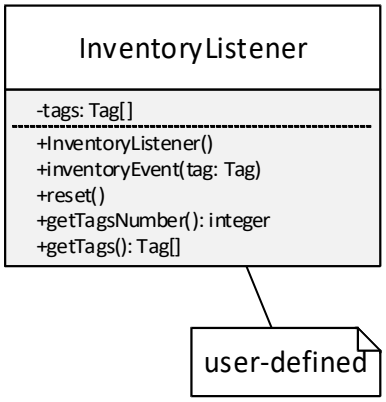
- ***AbstractResponseListener*** - definisce i metodi di callback relativi ai metodi delle classi derivate dalla classe *Tag*



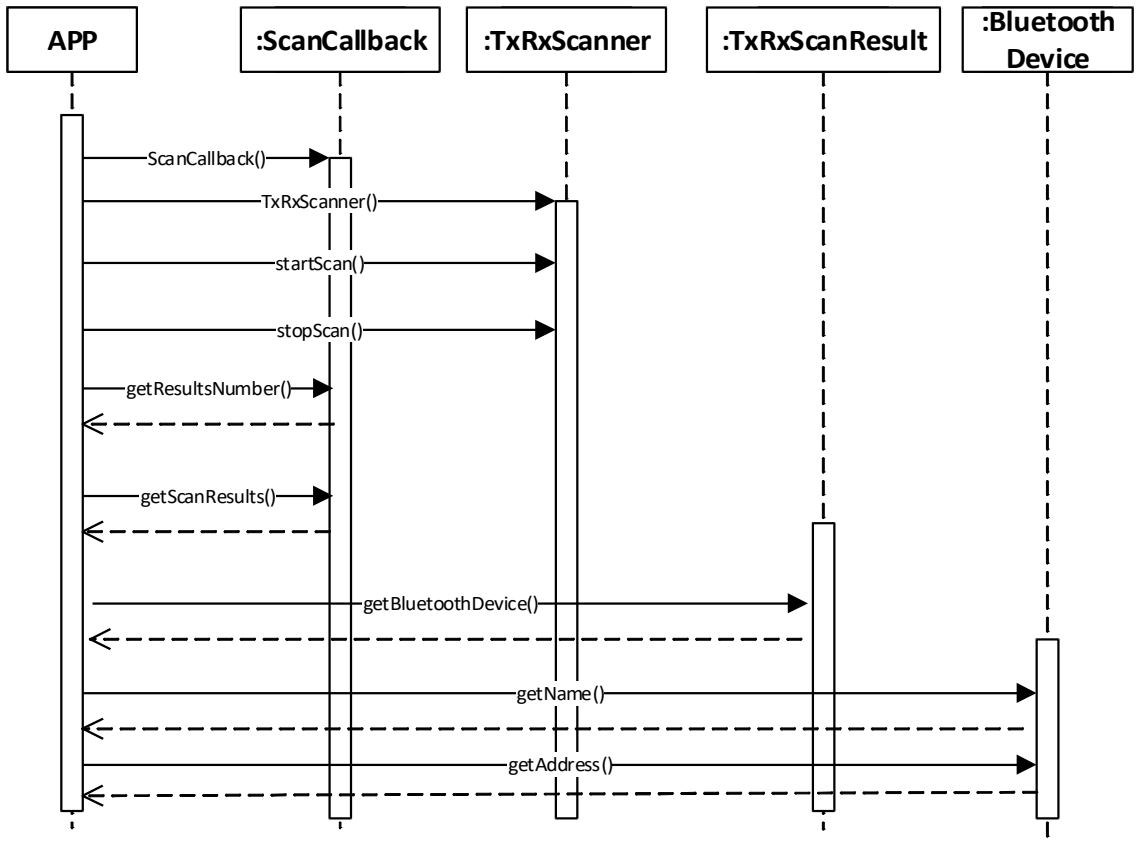
Nell'ipotesi di implementare l'interfaccia *TxRxScanCallback* con la classe *ScanCallback* che espone metodi di gestione della collezione dei tag RFID rilevati



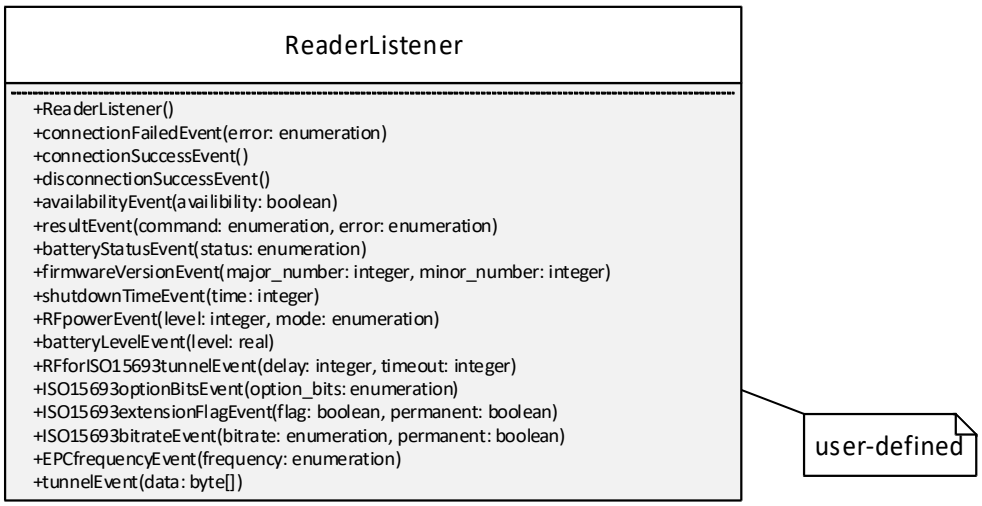
e di implementare l'interfaccia *AbstractInventoryListener* con la classe *InventoryListener* che espone metodi di gestione della collezione di *tag* RFID rilevati



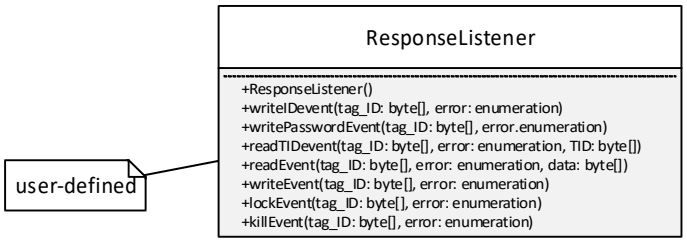
l'interazione tipica di una APP con le API ha inizio con la ricerca di un lettore RFID BLE documentata dal seguente diagramma UML di sequenza:



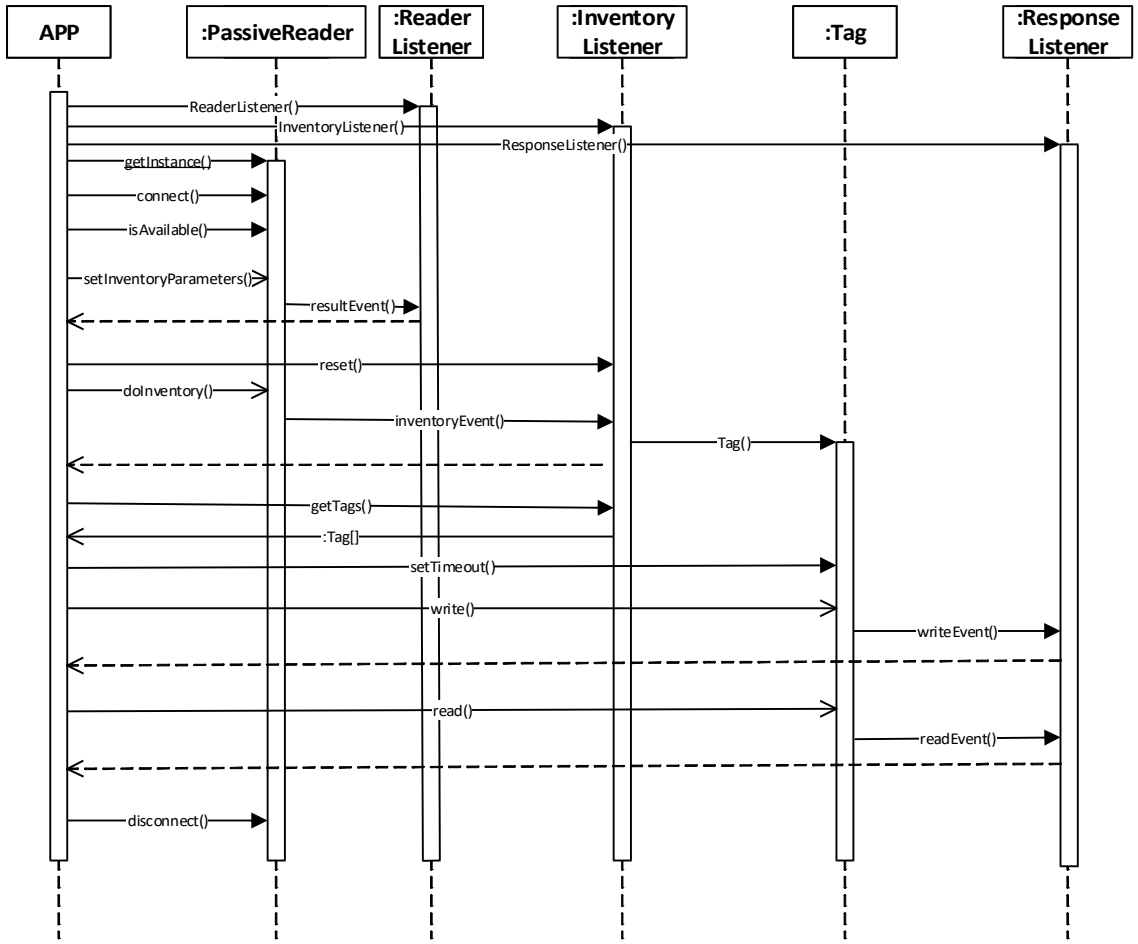
Nell'ipotesi di implementare l'interfaccia *AbstractReaderListener* con la classe *ReaderListener*



e la classe *AbstractResponseListener* con la classe *ResponseListener*



l'interazione tipica di una APP con le API prosegue con la rilevazione di tag RFID (inventory) e l'esecuzione di operazioni su uno di essi, fasi entrambe documentate nel seguente diagramma UML di sequenza:



Le librerie native di implementazione delle API di gestione ad alto livello dei lettori RFID BLE di TERTIUM Technology

TERTIUM Technology ha sviluppato 3 librerie native denominate **`RfidPassiveApiLib`** che implementano le API di gestione ad alto livello dei propri lettori RFID dotati di interfaccia BLE:

- libreria per sistema operativo Android in linguaggio Java;
- libreria per sistema operativo iOS in linguaggio Objective-C;
- libreria per sistema operativo iOS in linguaggio Swift.

Le librerie di implementazione delle API di gestione ad alto livello dei lettori RFID BLE di TERTIUM Technology sono documentate secondo gli standard previste dagli IDE utilizzati per lo sviluppo; in particolare per la libreria in linguaggio Java è stata generata la documentazione in formato HTML secondo lo standard JavaDoc.

Le APP di test delle librerie native di implementazione delle API di gestione ad alto livello dei lettori RFID BLE di TERTIUM Technology

Allo scopo di testare le librerie native di implementazione delle API di gestione ad alto livello dei propri lettori RFID dotati di interfaccia BLE, TERTIUM Technology ha realizzato 3 APP:

- APP nativa per sistema operativo Android realizzata in linguaggio Java;
- APP nativa per sistema operativo iOS realizzata in linguaggio Objective-C;
- APP nativa per sistema operativo iOS realizzata in linguaggio Swift;

Le APP di test espongono tutte una GUI costituita da 2 viste:

- vista con elenco dei dispositivi BLE rilevati con possibilità di selezione e connessione ad un dispositivo specifico;
- vista suddivisa in 3 sezioni:

    - visualizzazione dell'esito dell'invocazione dei metodi della API per l'inizializzazione del lettore RFID effettuata al momento della connessione,
    - visualizzazione dell'esito dell'invocazione dei metodi della API per l'interrogazione dello stato del lettore RFID effettuata periodicamente,

- controllo per la selezione[2] e l'invocazione con parametri predefiniti di un metodo della API di gestione del lettore RFID o di interazione con un *tag* rilevato e visualizzazione del relativo esito (comprende la visualizzazione degli ID dei *tag* rilevati in seguito alla selezione del metodo *doInventory* della API).

La figura che segue illustra l'architettura software complessiva delle librerie native e delle APP di test delle API per la gestione ad alto livello dei dispositivi BLE di TERTIUM Technology:

APP di test nativa (Java)	APP di test nativa (Objective-C)	APP di test nativa (Swift)
libreria nativa API (Java)	libreria nativa API (Objective-C)	libreria nativa API (Swift)
libreria nativa Tx/Rx (Java)	libreria nativa Tx/Rx (Objective-C)	libreria nativa Tx/Rx (Swift)
sistema operativo Android	**sistema operativo iOS**	

[2]è disponibile per la selezione un sottoinsieme significativo dei metodi della API

I *repository* del codice delle librerie/APP di gestione dei lettori RFID di TERTIUM Technology

Il codice delle librerie e delle APP di gestione dei lettori RFID prodotti da TERTIUM Technology è rilasciato con licenza *open-source* MIT (opensource.org/licenses/MIT) che ne consente l'impiego sia in progetti software *open-source* che proprietari, senza obbligo di ridistribuzione del codice sorgente.

Il codice sorgente delle librerie e delle APP di gestione è distribuito sulla piattaforma github.com nei seguenti *repository* di TERTIUM Technology (github.com/tertiumtechnology/) :

Libreria/APP	***Repository***
libreria nativa **TxRxLib** Java/Android	`tt-txrx-lib-android`
libreria nativa **TxRxLib** Objective-C/iOS	`tt-txrx-lib-ios-objc`
libreria nativa **TxRxLib** Swift/iOS	`tt-txrx-lib-ios-swift`
APP nativa test **TxRxLIB** Java/Android	`tt-txrx-demoapp-android`
APP nativa test **TxRxLib** Objective-C/iOS	`tt-txrx-demoapp-ios-objc`
APP nativa test **TxRxLib** Swift/iOS	`tt-txrx-demoapp-ios-swift`
libreria nativa **RfidPassiveApiLib** Java/Android	`tt-rfid-passive-api-lib-android`
libreria nativa **RfidPassiveApiLib** Objective-C/iOS	`tt-rfid-passive-api-lib-ios-objc`
libreria nativa **RfidPassiveApiLib** Swift/iOS	`tt-rfid-passive-api-lib-ios-swift`
APP nativa test **RfidPassiveApiLib** Java/Android	`tt-rfid-passive-api-testapp-android`
APP nativa test **RfidPassiveApiLib** Objective-C/iOS	`tt-rfid-passive-api-testapp-ios-objc`
APP nativa test **RfidPassiveApiLib** Swift/iOS	`tt-rfid-passive-api-testapp-ios-swift`