



# Obbiettivi

---

1. Permettere di riprodurre un testo scritto con l'accento linguistico opportuno.
2. Permettere di ascoltare una frase e visualizzare il testo scritto.
3. Permettere di salvare frasi ascoltate e riprodurle.
4. Localizzazione in Inglese e Italiano.



# Main Activity

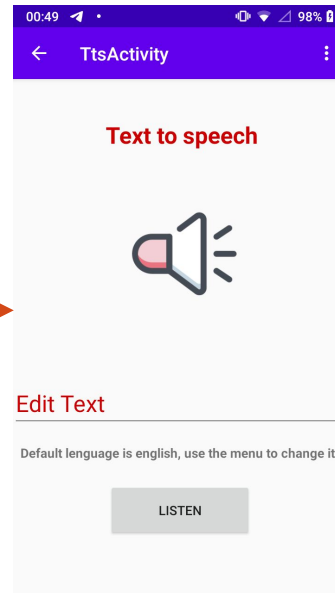
L'app si presenta come una semplice interfaccia, le scelte possibili sono:

- 1) Riprodurre un testo scritto (Text to Speech)
- 2) ascoltare una frase e visualizzare il testo scritto (Speech Recognizer)
- 3) Aggiungere o Riprodurre un messaggio salvato.

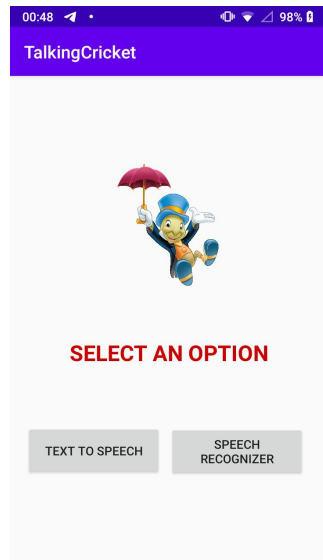
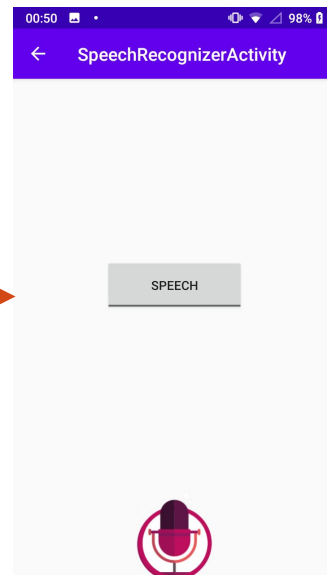




## Activity for Text to Speech



## Activity for Speech Recognizer

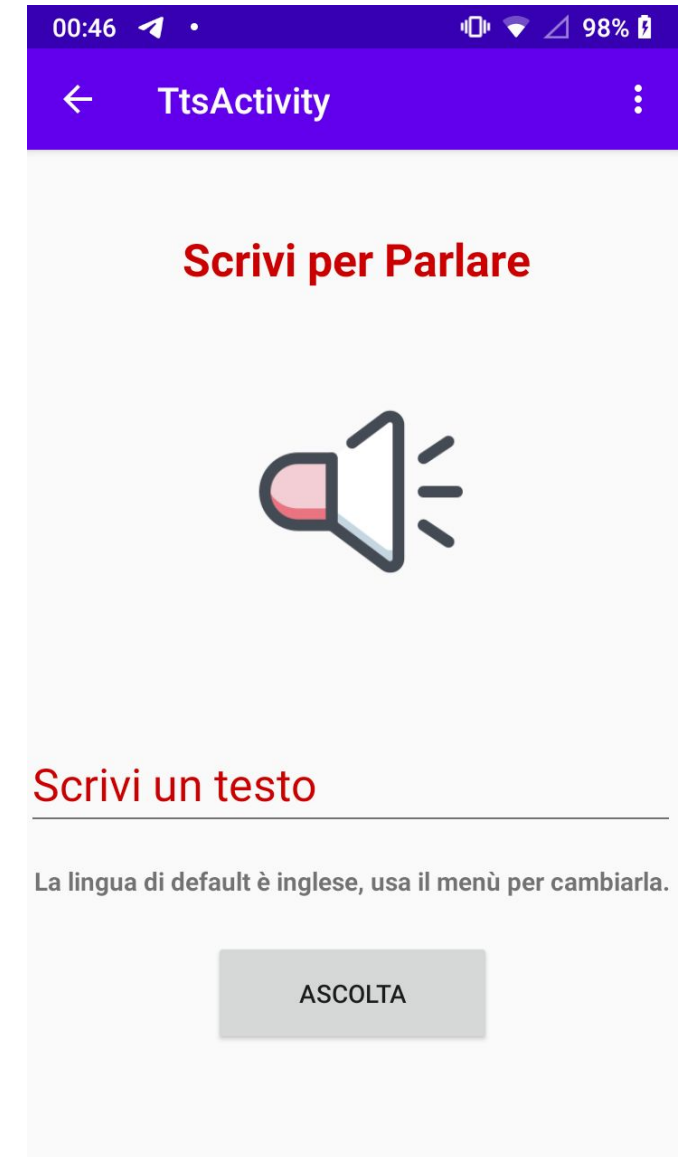
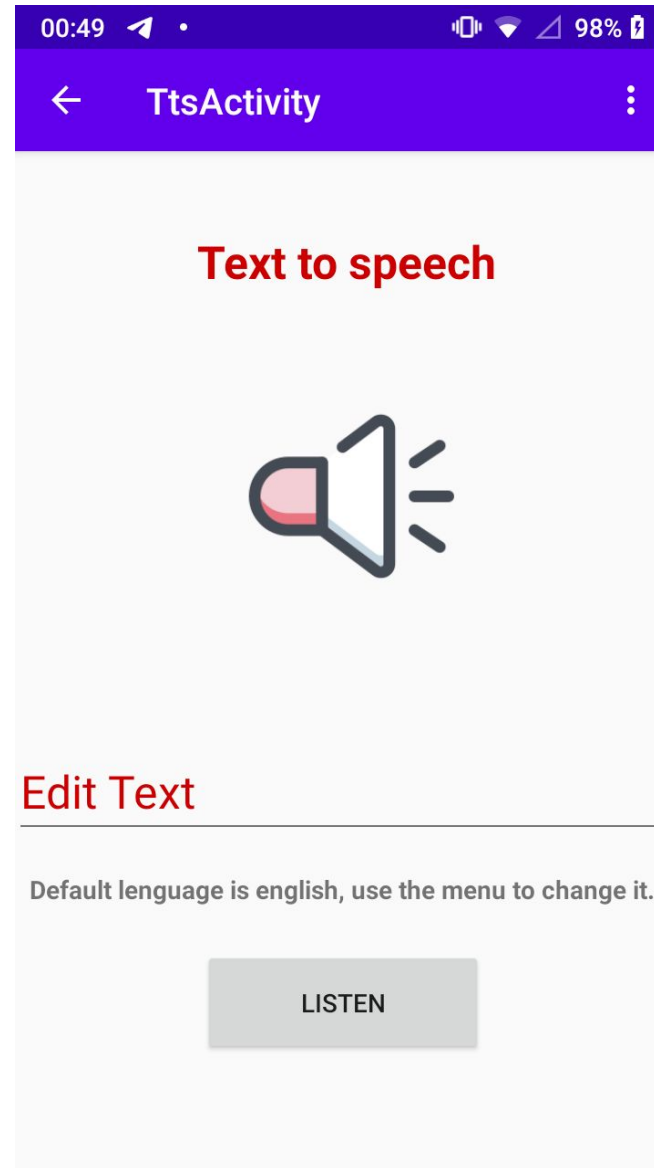


## Main Activity





# TTS: layout





# TTS: metodo speak()

---

Salvato il testo in una variabile string [toSpeak] da un edit text

La gestione di un oggetto *TextToSpeech* è molto semplice in quanto viene invocato il metodo

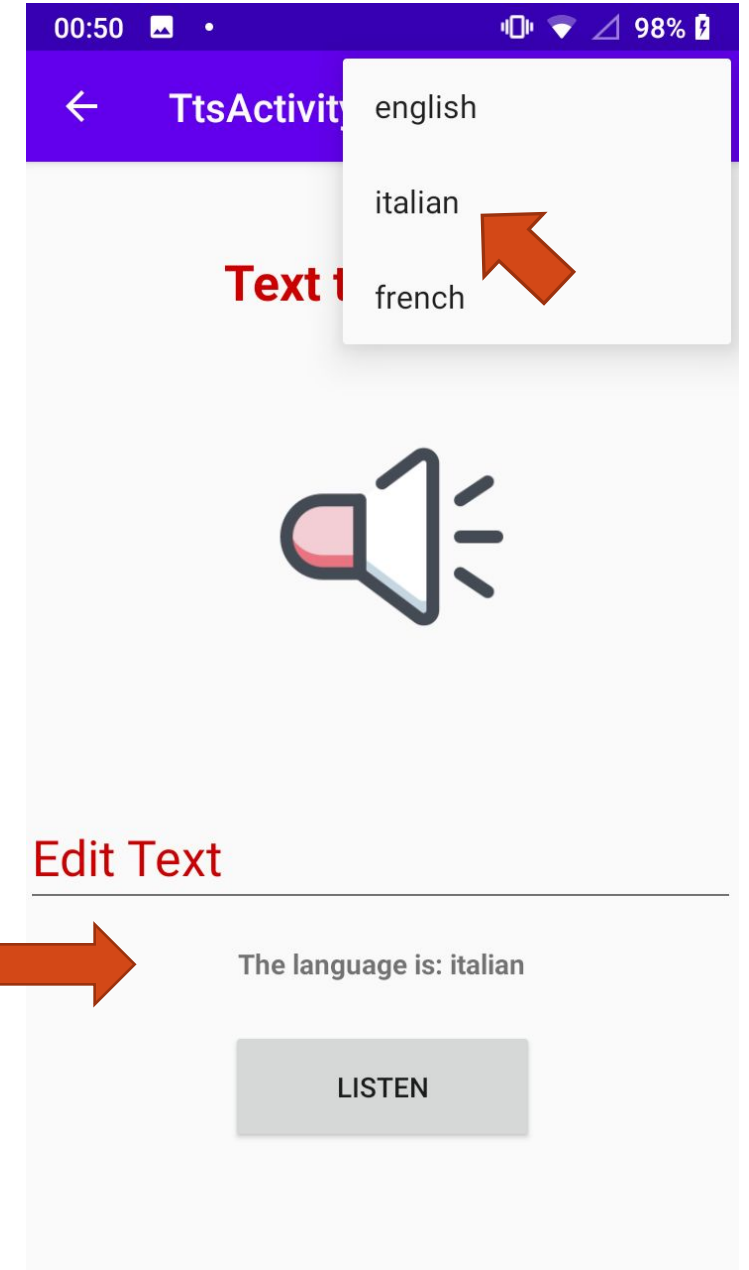
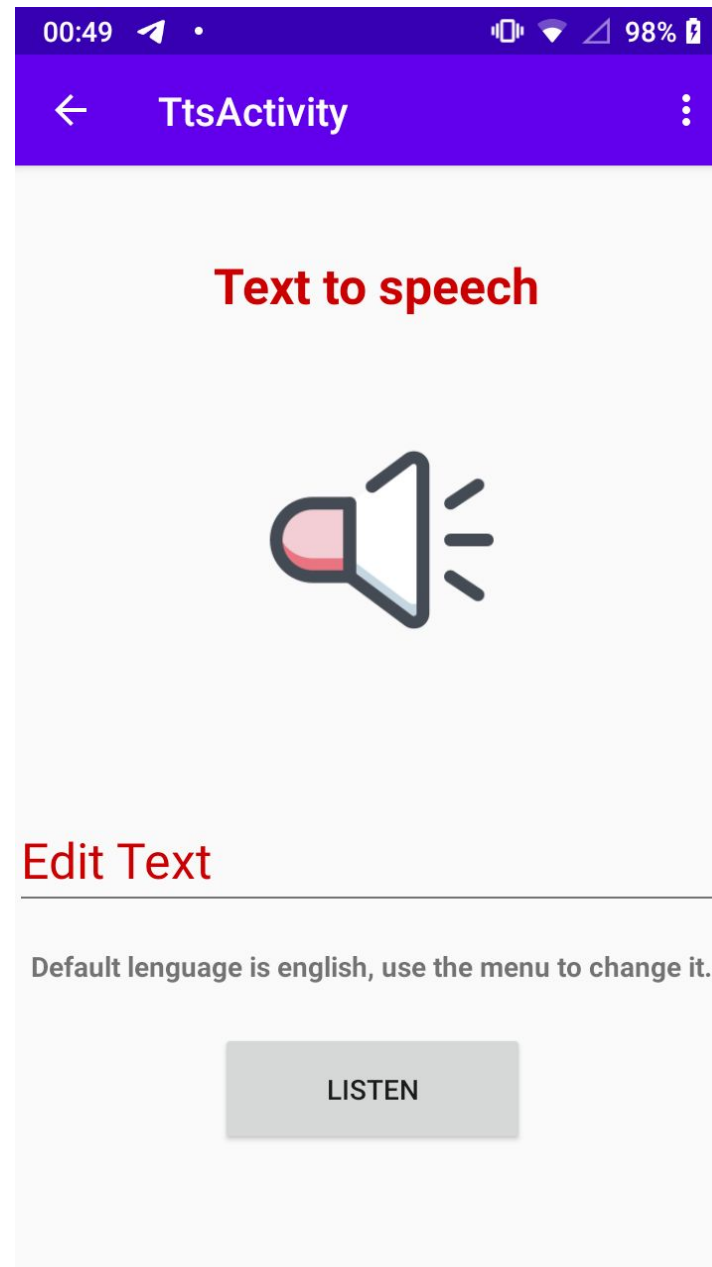
*speak( String string, int queueMode, params)*

- a) string è la stringa acquistata
- b) queueMode è parametro che mi permette di scegliere di aggiungere voci alla riproduzione (QUEUE\_ADD) o di rilasciare tutto in unico flusso (QUEUE\_FLUSH) e ho scelto questa ultima.
- c) parms è un parametro che ho impostato a null ma che può essere impostato in modo differente per specificare ad esempio il volume.





# TTS: Funzionalità del menù





# TTS: Gestione del menù

---

Ho scelto una gestione del menù molto semplice:

- a. Realizzazione di un file *res/menu/main.xml* in cui ho impostato il layout del menu
- b. In *TtsActivity* ho creato un metodo *onCreateOptionsMenu()* per creare il menù nell'activity del Text To Speech
- c. Ho creato un metodo *onOptionsItemSelected()* il quale si occupa, usando una struttura switch case di gestire il codice di impostazione dell'accentazione linguistica sfruttando il metodo *setLanguage()*. Nei vari casi ho anche opportunamente impostato il testo che si visualizza nel cambio della lingua.





# SR: Speech Recognizer

---

1. Questa classe permette di tradurre un input vocale in un output testuale.
2. Da un apposito button si avvia la registrazione del messaggio vocale.
3. Una progressBar permette di avere un feedback visuale dell'avvio della conversazione.
4. Il testo di output poi appare su textView.

## Criticità

- L'API usa server remoti per il riconoscimento vocale.
1. Non consente un riconoscimento continuo del vocale per non consumare la batteria.
  2. Necessita di una connessione dati per comunicare con il server.

L'API per il motivo 1 non consente quindi di riconoscere una frase molto lunga.

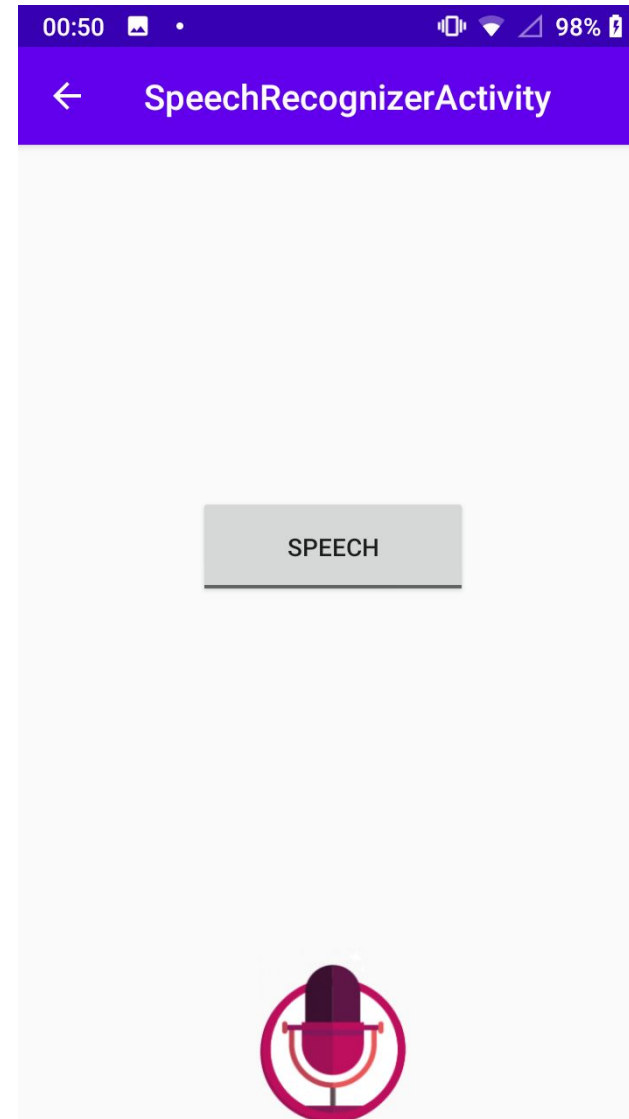






# SR: Layout

La disposizione grafica degli elementi è concepita per aiutare l'utente ad avvicinare la bocca al microfono del cellulare in modo intuitivo.





# SR: Metodi per l'API

---

1. Creato un nuovo oggetto *SpeechRecognizer*.
2. La cosa più importante è che il metodo *setRecognitionListener()* deve essere sempre invocato prima di fare qualsiasi operazione.
3. Si passa al metodo *onRequestPermissionsResult()*.
4. Ottenuti i permessi si passa ad un apposito metodo in cui si può invocare un *startListening(intent)* che prende in pasto un intent appositamente realizzato.
5. La gestione degli errori è fondamentale per comprendere quali impedimenti potrebbero non far funzionare la mia activity. Per questo viene realizzato un metodo opportuno *getErrorText()* che attraverso una struttura di switch gestisce i messaggi di errore.





# SR: ProgressBar

---

Una parte consistente del codice gestisce graficamente della progressBar, ne gestisce l'animazione in modo da renderla graficamente apprezzabile.

*setVisibility()* è il metodo che la gestisce la visualizzazione e insieme al metodo *setIndeterminate(true)* che fa in modo rendere indeterminato l'avanzamento della barra di progresso.



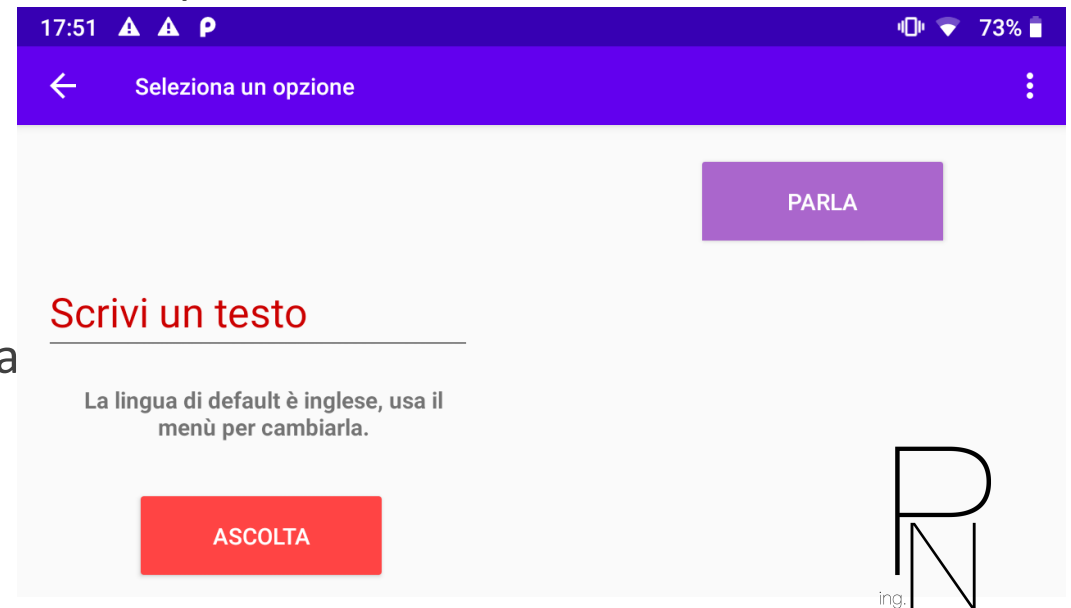


# Fragment e Landscape

Le due principali activity di TTS e SR sono state implementate tramite i loro rispettivi fragment.

Per poter sfruttare appieno le loro potenzialità senza però modificare la semplicità dell'applicazione. Ho pensato di fare in modo che i due fragment comparissero in contemporanea quando l'applicazione si trova in Landscape.

Questo aspetto è stato possibile semplicemente modificando opportunamente il file xml della versione del layout che si trova nella cartella dal nome Land.





# Manifest

---

MainActivity è l'activity padre invece per le altre activity vanno specificati i *parentActivityName=".MainActivity"*

Per l'API SpeechRecognizer devo avere l'autorizzazione *permission.RECORD\_AUDIO* e *permission.INTERNET*

Per installarla sul dispositivo reale:

1. *Xmls:tools="http://schemas.android.com/tools"*
2. In application togliere *hardwareAccelerated="false"*
3. Aggiungere *debuggable="true"* e *tools:ignore="HardcodeDebugMode"*





# DataBase Comandi

---

CON UN SEMPLICE DATABE SQLITEOPENHELPER LOCALE





# Obiettivo

---

**Salvare un testo acquisito con SR e riprodurlo  
in un secondo momento con TTS**





# DB: Parti fondamentali

---

1. *DbHandler.class* dove vengono istanziati tutti i metodi per realizzazione del Database. Ho un (id, nome, comando, posizione), fondamentale il parametro posizione per salvare la posizione iniziale di quando genero il dato. All'inizio id = posizione.
2. *SetActivity* invoca il metodo *insertUserDetails(name, commad)* e genera così gli elementi del database.
3. *DetailActivity* si occupa di visualizzare gli elementi del database su un ListView e consente di accedere all'activity per l'eliminazione, la modifica e la riproduzione.
4. *CricketSpeechActivity* tramite il suo fragment svolge le operazioni di riproduzione e di eliminazione degli elementi del database.
5. Punti chiave 1) passaggio di dati tra activity e fragment e viceversa 2) gestione degli id e bug risolto dal paramento posizione.





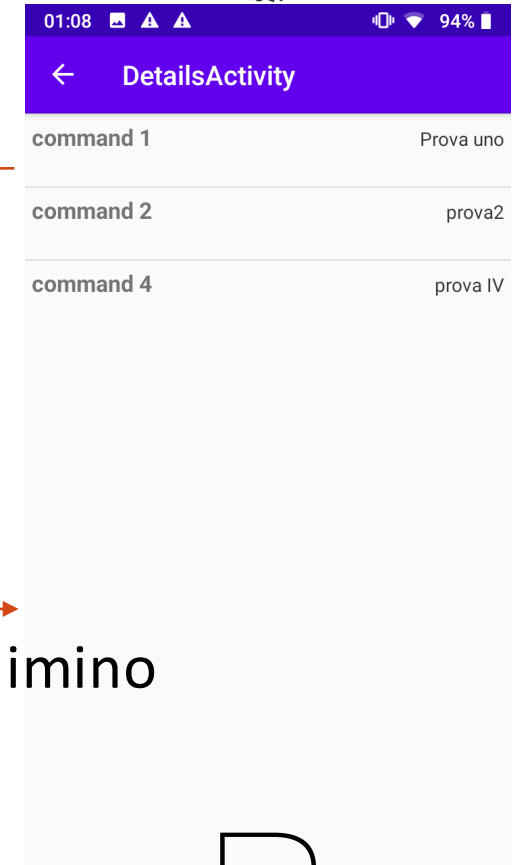
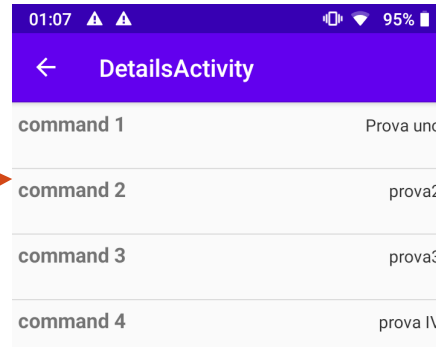


# comando

## Nuovo Comando

## Imposta Comando o l'immagine del grillo

## Se elimino





# DB: passaggio di dati

---

1. *DetailsActivity* passa tramite INTENT con metodo *intent.putExtra("id", id)* e il comando a *CricketSpeechActivity*.
2. *CricketSpeechActivity* recepisce i dati tramite *intent.getIntExtra("id", defaultValue)*
3. *CricketSpeechActivity* tramite BUNDLE passa i dati al fragment con il metodo *bundle.putInt("id", id)* a *CricketSpeechFragment*.
4. *CricketSpeechFragment* recepisce i dati tramite metodo *bundle.getInt("id")*
5. Da qui il dato viene semplicemente trattato come una normale stringa per il comando TTS e usato l'id per consentire l'eliminazione nel caso lo si desidera.

**NOTA: SI PASSA IL PARAMETRO ID CHE CONSENTE DI ACCEDERE AGLI ELEMENTI DEL DATABASE**





# DB: risoluzione del BUG

---

**Problema:** ogni volta che vengono eliminati dati, non c'è corrispondenza della riga con gli elementi del database.

**Perché?** Il ListView genera un id del dato sulla sua posizione nella riga, NON coincide con id database.

**Soluzione:** salvare la posizione dell'id del dato al momento della creazione, recuperarlo dal dato ListView (estraendolo da una stringa). Confrontarlo in un ciclo while con la riga del ListView, e incrementando il numero di riga di ListView se non coincidono o sono null i riferimenti al database.



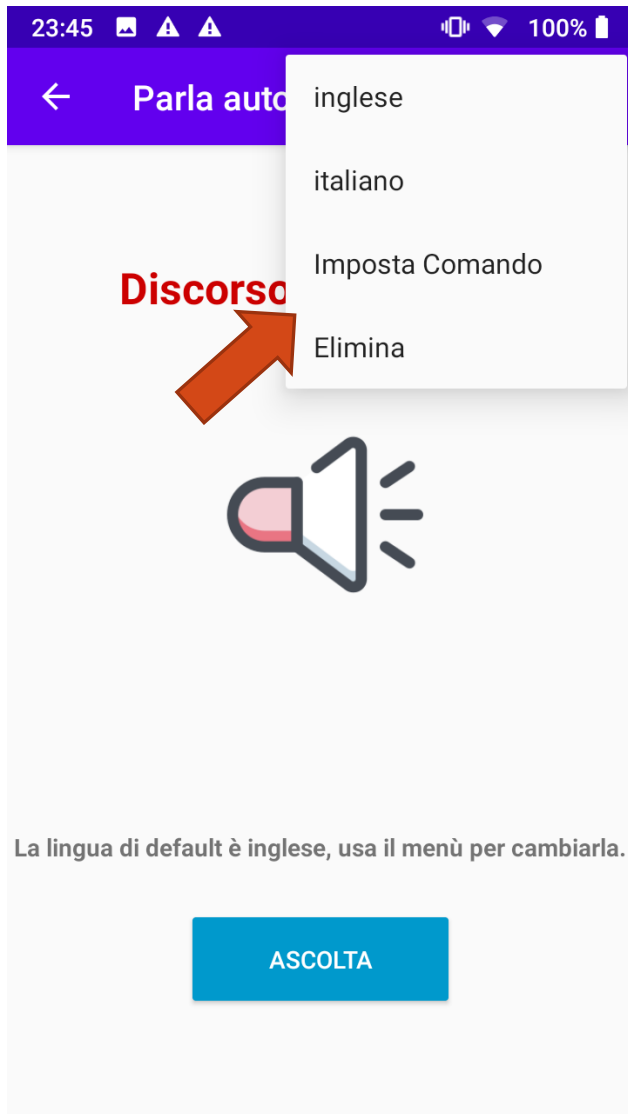
id-list	id-db	nome
1	1	com1
2	2	com2
3	3	com3
4	4	com4
5	5	com5
6	6	com6

id-list	id-db	nome
1	1	com1
2	4	com4
3	6	com6
4		
5		
6		

## HO BISOGNO DI CONOSCERE ID-DB PER SAPERE ID-DB

id-list	id-db	nome	pos
1	1	com1	1
2	4	com4	4
3	6	com6	6
4			
5			
6			

- Accedo solo a id-list
- Se chiedo la risorsa all'id-list 2 mi da null perché id-db 2 non esiste.
- id-list++ fino a != null e trovo id-db 4 e va bene
- Ma se voglio il contenuto di id-list 3 con id-list++ mi fermo al id-db 4 e non trovo com6.
- Allora aggiungo un controllo in più chiedendo anche id-list == position
- Nel ciclo while mi fermo solo quando accade che non sia ne null ne diverso dal parametro posizione.





# Bibliografia

---

- a) Lezioni e materiale del professor Livio Tenze [http://www2.units.it/carrato/didatt/E\\_web/index.html](http://www2.units.it/carrato/didatt/E_web/index.html)
- b) <https://developer.android.com/reference/android/speech/tts/TextToSpeech>
- c) <https://developer.android.com/reference/android/speech/SpeechRecognizer>
- d) <https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase>
- e) [https://developer.android.com/guide/fragments?gclid=Cj0KCQiAh4j-BRCsARIsAGeV12C24T3WEH9Sm6drrxWMav6VdReejneI0fGoa62huqDMToGKkI3vMcMaArACEALw\\_wcB&gclid=aw.ds](https://developer.android.com/guide/fragments?gclid=Cj0KCQiAh4j-BRCsARIsAGeV12C24T3WEH9Sm6drrxWMav6VdReejneI0fGoa62huqDMToGKkI3vMcMaArACEALw_wcB&gclid=aw.ds)





# Fine

---

GRAZIE PER L'ATTENZIONE

