Università degli Studi di Milano

Facoltà di Scienze Matematiche, Fisiche e Naturali

Dipartimento di Informatica e Comunicazione

Corso di Laurea in Informatica

Utilizzo di Android nel campo delle Tecnologie assistive rivolte agli anziani: studi di fattibilità e realizzazione di un prototipo

Candidato:
Alessandro Vincenti

Relatore:
Dott. Andrea Trentini

Correlatore:
Prof. Mattia Monga

Anno Accademico 2010/2011

Alla mia famiglia

Indice

Capitolo 1

Anziani, vita quotidiana e tecnologie assistive

1. Terminologia

Gli ausili sono "strumenti tecnologici che consentono di superare certe barriere all'accessibilità, o di compensare certe limitazioni funzionali ai fini di facilitare o rendere possibili determinate attività della vita quotidiana" (Commissione Europea, 1995)[1].

Lo Standard Internazionale ISO 9999:2011 - alla luce del modello ICF del funzionamento umano, della disabilità e della salute (International Classification of Functioning, Disability and Health)[2] - definisce più specificamente ausilio "qualsiasi prodotto (inclusi dispositivi, apparecchiature, strumenti, sistemi tecnologici, software), di produzione specializzata o di comune commercio, atto a prevenire, compensare, tenere sotto controllo, alleviare o eliminare menomazioni, limitazioni nelle attività, o ostacoli alla partecipazione".

Sulla scia della dizione americana "Assistive Technology Products", si sta diffondendo anche in Italiano come in altre lingue il termine Tecnologie Assistive, preferito soprattutto quando ci si riferisce non a singoli oggetti ma ad insiemi di prodotti, componenti e software che, opportunamente assemblati e configurati, rispondono ad un'esigenza di autonomia della persona con disabilità.

Una tecnologia può essere considerata un ausilio quando questa è specificamente utilizzata per compensare limitazioni funzionali, facilitare la vita indipendente, e far sì che le persone anziane e le persone disabili possano realizzare le loro piene potenzialità.

[1] Commissione Europea (1995), *Workplan Programma TIDE*. Citato in (EUSTAT, 1999)

[2] http://www.who.int/classifications/icf/en/

Tale termine non si applica quindi solo a tecnologie progettate specificamente per le persone disabili: si estende anche a quelle tecnologie di uso comune che, quando necessario, possono diventare di ausilio a chi ha una disabilità.[3]

2. Introduzione

Il nostro pianeta è sempre più popoloso, ma i suoi abitanti tendono ad essere sempre più anziani. La crescita demografica, che recentemente ha portato la popolazione mondiale a superare i 7 miliardi, continuerà ancora per 30-40 anni, fino a raggiungere gli 8-9 miliardi[4].

Si stima che entro il 2050 le persone con più di 60 anni saranno più dei bambini, arrivando a ben 2 miliardi, contro gli attuali 650 milioni[5].

L'invecchiamento della popolazione europea è strettamente legato al miglioramento della salute delle persone, che tendono ad essere più attive e a vivere più a lungo.

La "Comunicazione sul Futuro Demografico dell'Europa" della Commissione Europea ha infatti individuato una serie di trend demografici (le conseguenze del baby-boom post bellico sulla popolazione; il forte aumento dell'aspettativa di vita dal 1960 in avanti; il numero di nascite inferiore al tasso di rimpiazzo generazionale) il cui effetto complessivo è rappresentato dall'aumento della popolazione nella fascia di età anziana. Attualmente, nei ventisette paesi dell'Unione Europea vi sono 18,2 milioni di abitanti di età superiore ad 80 anni, pari al 4% della popolazione, e l'Eurostat prevede che entro il 2014 ve ne saranno 24,1 (ovvero il 5,2% della popolazione).[6]

Il numero delle persone di età compresa fra 65 e 79 anni è aumentato significativamente dal 2000 in avanti, ed il trend rimarrà tale fin verso il 2050.

---

[3] Andrich, R. (2008), *Concetti generali sugli ausili*. In Caracciolo, A. & Redaelli, T. & Valsecchi, L.: *Terapia occupazionale: ausili e metodologie per l'autonomia*. pp. 105-138. Milano: Raffaello Cortina.

[4] U.S. Census Bureau (2011), *International Data Base, World Population: 1950-2050*, June 2011 update

[5] ONU (2002), *Invecchiamento della popolazione mondiale: 1950-2050*, Secondo rapporto, Madrid

[6] Leonardi, M. & Chatterji, S. & Bickenbach, J. (2008), *Funzionamento e disabilità nell'invecchiamento della popolazione europea: quale politica per quale intervento?*, Quaderni Europei sul nuovo Welfare, Quaderno n.10.

Gli effetti dell'invecchiamento della popolazione sulle cure saranno amplificati da un aumento sproporzionato di demenza, depressione ed altre malattie neurologiche e psichiatriche (Draper, 2004). Più in generale, l'invecchiamento, come sottolineato dalla Commissione Europea nel Disability Action Plan 2006-07, è fortemente correlato alla prevalenza della disabilità. Circa il 30% delle persone in età compresa fra 55 e 64 anni denunciano una disabilità, e il 63% delle persone con una disabilità hanno più di 45 anni.
Una risposta alla disabilità può venire oggi dall'utilizzo delle moderne tecnologie.
Spesso il primo approccio agli ausili tecnologici avviene grazie alle informazioni fornite alla famiglia da medici, terapisti, insegnanti, associazioni che già conoscono gli ausili per la natura stessa della loro professione, per la specificità del loro campo di azione o per esperienza diretta.
Quando le difficoltà insorgono in età senile o vengono comunque attribuite al normale processo di invecchiamento, la reazione dell'anziano e spesso anche dell'ambiente che lo circonda è quella di accettare con fatalismo le difficoltà dovute ad un peggioramento della vista, dell'udito, della memoria, della mobilità.
Questo comporta di frequente un netto peggioramento della qualità della vita, con un progressivo abbandono di attività importanti per l'autonomia o semplicemente per occupare il tempo in modo piacevole.
L'anziano è spesso circondato da un ambiente sociale abbastanza ristretto e man mano che le difficoltà peggiorano le occasioni di scambio sociale tendono a ridursi ancora di più.
A questa oggettiva difficoltà nell'ottenere informazioni riguardo la disponibilità di ausili che possano offrire aiuto, si aggiunge di solito una notevole resistenza (sia da parte dell'anziano, sia da parte dell'ambiente che lo circonda) nel ricorrere ad ausili tecnologici.
Questo spiega l'utilizzo ancora assai limitato di questo tipo di ausili in età geriatrica.

3. Idee e soluzioni

Un recente studio inglese[7] ha delineato come le tecnologie assistive favoriscano l'autonomia nella vita quotidiana di persone anziane e disabili, individuando le seguenti principali possibilità:

- offrire una migliore e più economica assistenza sociale e sanitaria a domicilio attraverso la *teleassistenza* e i servizi di *telemedicina*;
- fornire servizi che intrattengano, educhino e stimolino l'interazione sociale, in modo da arricchire la vita relazionale degli anziani e dei disabili che vivono in casa (*servizi di partecipazione digitale*);
- fornire servizi che incoraggino gli utenti a restare in forma e ad adottare sani stili di vita (*servizi per il benessere*);
- consentire agli anziani e ai disabili di lavorare da casa in modo da partecipare di più nell'economia e nella società (*telelavoro*).

L'evoluzione tecnologica e di mercato dovrebbe portare a una riduzione sostanziale dei costi dei sistemi di telemedicina e teleassistenza, a un ampliamento della gamma di servizi e tecnologie assistive a disposizione di anziani e disabili. Una conseguenza evidente dell'ampliamento del mercato è il cambiamento della natura stessa del settore che fornisce le attrezzature: da uno in cui le aziende specializzate forniscono sistemi *stand-alone* "chiusi" (con proprio software e hardware integrato), ad uno in cui le aziende specializzate sviluppano specifiche applicazioni software per le persone anziane e disabili, utilizzando però le normali piattaforme presenti sul mercato di massa, come i televisori o i dispositivi mobili (cellulari, palmari, tablet ecc.).

Un aiuto importante può venire da ausili estremamente semplici da utilizzare, come ad esempio il video-ingranditore: un ripiano su cui posare il testo da leggere, sormontato da un monitor dove viene proiettata l'immagine ingrandita del testo sottostante. Ingrandimento, contrasto, luminosità sono quasi sempre ampiamente regolabili.

Se la lettura richiede comunque uno sforzo eccessivo, soprattutto quando il testo è lungo, si può ricorrere ad una macchina di lettura che, tramite pochi e semplici comandi, legge il testo scritto con una sintesi vocale.

[7] Lewing D., Adshead S., Glennon B., Williamson B., Moore T., Damodaran L., Hansell P. (2010), *Assisted living technologies for older and disabled people in 2030*, Londra (Regno Unito), Plum Consulting

Per rimanere nell'ambito della lettura, chi non può utilizzare gli arti superiori o, immobilizzato a letto, non può mantenere a lungo le braccia sollevate, può ricorrere al volta-pagine: una sorta di braccio meccanico che, a comando, sfoglia le pagine.

Un'attività importantissima dal punto di vista sociale è la comunicazione tramite telefono.

Molti anziani hanno difficoltà uditive e faticano a sostenere la conversazione telefonica dove non è consentito il supporto della lettura labiale. Inoltre risulta a molti difficile comporre i numeri, così come memorizzarli o consultare l'agenda telefonica.

Per ovviare a questo problema sono stati creati telefoni con tasti numerici ingranditi e quindi più visibili e con tasti aggiuntivi, sempre di grandi dimensioni, in cui memorizzare i numeri più importanti.

Il personal computer stesso, per coloro che sono in grado di utilizzarlo, può fornire assistenza in molteplici attività della vita quotidiana. La connessione ad internet ad esempio permette di fare acquisti, di leggere i quotidiani, compiere operazioni bancarie e accedere a qualsiasi informazione senza doversi spostare da casa.

Utilizzare la tecnologia, oltre a facilitare attività cui è difficile rinunciare, può anche avere una ricaduta importante a livello di autostima: aiuta a mantenere la mente "più flessibile" ed a sentirsi maggiormente adeguati al proprio tempo.

Capitolo 2

Il progetto e la scelta del prototipo

1. Introduzione

Dall'analisi delle considerazioni fatte nel capitolo precedente nasce l'idea che è alla base di questo progetto: implementare un'applicazione software che possa essere d'aiuto alle persone anziane nello svolgimento delle loro attività quotidiane.
Questa applicazione dovrà essere eseguita preferibilmente su dispositivi maneggevoli, con dimensioni ridotte e costi accessibili, come ad esempio telefoni cellulari, smartphone, tablet o hardware dedicati (Androidbox o dispositivi come Arduino[8]).

2. Possibili implementazioni

- Applicazione che permetta all'utente di gestire la spesa al supermercato: creare la lista dei prodotti necessari, acquistarli (interfacciandosi al sito web del negozio) ed ottenere indicazioni sui tempi di consegna, il tutto tramite un'interfaccia molto semplificata
- Applicazione che, sfruttando un display e una webcam, si comporti come una lente di ingrandimento, o (più complesso) che riconosca il testo ripreso dalla fotocamera (tramite funzionalità di riconoscimento ottico dei caratteri, OCR) e lo legga ad alta voce (tramite un sintetizzatore vocale)
- Applicazione in grado di interfacciarsi via bluetooth ad un cardiofrequenzimetro, che registri informazioni relative alla variabilità del battito cardiaco, avvisi l'utente al superamento di soglie d'allarme, e magari che sia in grado di

[8] Piattaforma hardware "open source", fornita insieme ad un ambiente di sviluppo pensato per chi non ha esperienza nel campo della programmazione. Per ulteriori informazioni consultare il sito web: http://www.arduino.cc/

telefonare/inviare un sms ad un parente nel caso in cui si verifichino complicazioni cardiache

- Applicazione che gestisca la segreteria telefonica (di un cellulare) tramite un'interfaccia semplificata
- Applicazione che semplifichi l'invio e la ricezione degli SMS (su un telefono cellulare), magari con funzionalità di sintesi e di riconoscimento vocale.
- Applicazione che possa gestire la programmazione TV, programmare la registrazione di determinate trasmissioni e offrire funzioni supplementari durante la riproduzione (informazioni aggiuntive, zoom, sottotitoli, mettere in pausa la riproduzione/continuare la riproduzione in differita…)

3. La scelta del prototipo

La scelta del prototipo da implementare è stata influenzata da diversi fattori, primo tra tutti una considerazione sull'hardware da utilizzare: lo strumento ideale per accompagnare quotidianamente un anziano deve essere piccolo, leggero, facile da trasportare e possibilmente economico.

Nonostante gli indubbi vantaggi di un dispositivo come Arduino (opensource, economico, facile da programmare e da adattare a qualsiasi tipo di applicazione), esiste un altro dispositivo che soddisfa tutti questi requisiti, e che probabilmente, data la sua larghissima diffusione, è già in possesso di buona parte degli interessati: il telefono cellulare.

La scelta di questo strumento pone però due problemi fondamentali: l'eterogeneità dei software presenti sui telefoni cellulari (da cui la difficoltà nel creare un software multipiattaforma, portabile su diversi tipi di dispositivi) e la complessità d'uso per individui non avvezzi alla tecnologia, magari con problemi alla vista.

Si è pensato di risolvere il primo problema adottando un sistema operativo opensource che in questi anni sta subendo un'enorme diffusione, installato su dispositivi le cui fasce di prezzo sono in continua discesa[9]: Android.

[9] Si veda l'articolo: "The rise of the low-priced Android smartphone market", di Luke Lin, DIGITIMES Research, Taipei, aprile 2011
http://www.digitimes.com/Reports/Report.asp?datepublish=2011/4/18&pages=RS&seq=400)

Riguardo alla complessità d'uso, si è pensato di creare il prototipo di un'applicazione che avesse proprio l'obiettivo di facilitare l'utilizzo del telefono cellulare.

Dall'intervista[10] condotta durante lo studio e la progettazione di questo prototipo, è emerso che una delle funzionalità del telefono meno usate dagli utenti anziani è l'invio/la lettura degli sms, proprio a causa della sua complessità: la maggior parte degli utenti ha grandi problemi nella digitazione sulla tastiera, i cui tasti sono troppo piccoli o scomodi, e nella lettura dei messaggi ricevuti, a causa delle ridotte dimensioni dello schermo e della complessità dei passaggi da seguire nel menu per accedere alla lettura dei messaggi.

Da qui l'idea di creare una applicazione in grado di intercettare l'arrivo di nuovi messaggi, leggerli ad alta voce (sfruttando le API Android TextToSpeech fornite da Google), mostrarli automaticamente sullo schermo con caratteri chiari e grandi, e offrire la possibilità di rispondere premendo un solo tasto e dettando a voce la risposta da inserire nel messaggio.

[10] Si veda il Capitolo 5

Capitolo 3

Android

Android è un sistema operativo per dispositivi mobili inizialmente sviluppato da una certa "Android Inc.", la quale nel 2005 fu acquisita da "Google Inc." come parte della sua strategia per entrare nel mercato dei dispositivi mobili.

I cofondatori di Android Inc.[11] iniziarono a lavorare per Google e svilupparono una piattaforma basata sulla versione 2.6 del kernel Linux.

Il 5 Novembre 2007 fu creato l'Open Handset Alliance (OHA), un consorzio di produttori[12] il cui obiettivo è sviluppare "standard aperti" per dispositivi mobili. Nello stesso giorno dell'annuncio della sua formazione, l'OHA presentò pubblicamente Android e nel giro di pochi giorni rilasciò i relativi strumenti di sviluppo (SDK).

1. Perché Android

Google ha voluto che Android fosse un sistema aperto e gratuito. La maggior parte del suo codice è rilasciata tramite la licenza open-source Apache, il che significa che chiunque sia interessato ad utilizzare questo sistema può farlo semplicemente scaricando il codice sorgente completo.

Un altro grande vantaggio nell'utilizzo di Android è che offre un approccio unificato allo sviluppo di applicazioni. Lo sviluppatore che crea un'applicazione per Android può contare sul fatto che essa sarà eseguibile su una vasta moltitudine di dispositivi, anche di marche e caratteristiche hardware diverse.

---

[11] Andy Rubin (cofondatore di Danger), Rich Miner (cofondatore di Wildfire Communications Inc.), Nick Sears (vicepresidente di T-Mobile) e Chris White (principale autore dell'interfaccia grafica di WebTV)

[12] Allo stato attuale, l'OHA è composta da 35 membri, tra cui: operatori di telefonia mobile, produttori di semiconduttori, produttori di dispositivi mobili, produttori di software e compagnie per la commercializzazione. Per un elenco completo si faccia riferimento a: http://www.openhandsetalliance.com/oha_members.html

Non ci sono limitazioni né riguardo alla configurazione hardware né riguardo alla configurazione software: i venditori (tipicamente i produttori di hardware) possono aggiungervi le loro estensioni proprietarie, e personalizzare il sistema e le interfacce per differenziare i propri prodotti.

Android risolve insomma alcuni dei problemi che fino ad oggi hanno bloccato lo sviluppo di applicazioni per smartphone, come ad esempio:

- Frammentazione: nel 2007 furono venduti 70 milioni di smartphone, e tuttavia ogni marchio presentava un proprio ambiente applicativo. Java fu introdotto per migliorare questa situazione[13], fornendo un ambiente comune a tutti i dispositivi. Sfortunatamente, quasi tutti i dispositivi che supportano J2ME supportano anche estensioni proprietarie, che limitano la portabilità delle applicazioni.
- Sistemi operativi proprietari: la maggior parte dei dispositivi mobili utilizza sistemi operativi proprietari e non open-source, che possono essere modificati solo dai produttori.
- Reti chiuse: spesso gli operatori telefonici bloccano i dispositivi con il pretesto di preservare l'integrità delle loro reti, limitando così l'accesso degli utenti alle applicazioni fornite dagli operatori stessi.

Tutte queste caratteristiche hanno contribuito all'enorme diffusione di Android, tanto da portarlo non solo a superare il sistema operativo Apple IOS, ma addirittura a diventare, nel luglio del 2011, il secondo sistema operativo per dispositivi mobili più diffuso in Europa[14].

2. Architettura

Il sistema operativo Android è suddiviso in cinque sezioni, raggruppate in quattro livelli principali:

[13] tramite J2ME e le Wireless Java Recommendations

[14] Si veda l'articolo "Android Captures #2 Ranking Among Smartphone Platforms in EU5", Londra, Regno Unito, Settembre 2011 - comScore, Inc. (NASDAQ: SCOR)
http://www.comscore.com/Press_Events/Press_Releases/2011/9/Android_Captures_number_2_Ranking_Among_Smartphone_Platforms_in_EU5

- **Linux Kernel**: il Kernel su cui è basato Android. Questo livello contiene tutti i driver di basso livello dei vari componenti hardware
- **Librerie**: contengono tutto il codice che fornisce le funzioni principali del sistema operativo. Per esempio, la libreria SQLite fornisce il supporto ai database in modo che una applicazione li possa usare per archiviare i dati. La libreria WebKit fornisce funzionalità per la navigazione web.
- **Android Runtime**: fornisce un nucleo di librerie che permette agli sviluppatori di scrivere applicazioni Android usando il linguaggio di programmazione Java. Include anche la macchina virtuale Dalvik[15], che consente ad ogni applicazione Android di essere eseguita in un processo e con una istanza a lei riservati.

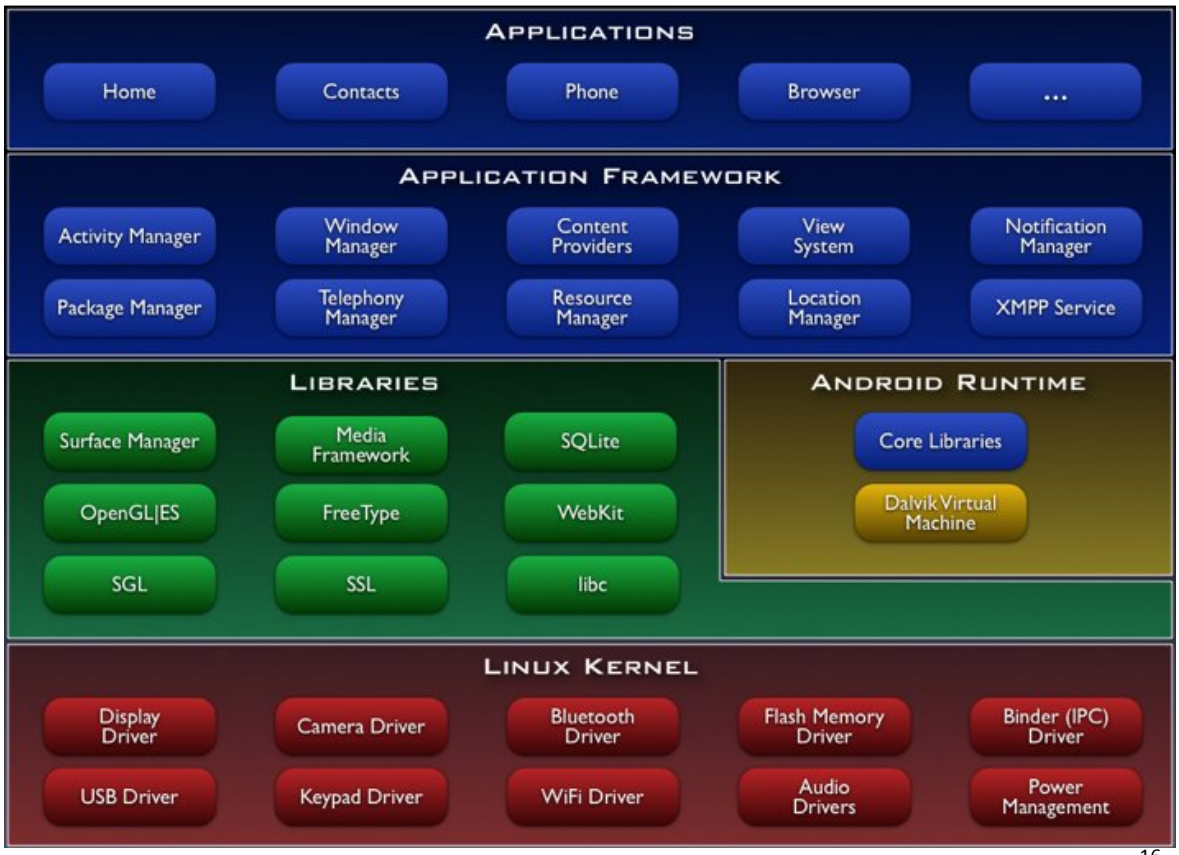

Figura 3.1 [16]

[15] Dalvik è una macchina virtuale specializzata progettata specificamente per Android ed ottimizzata per dispositivi mobili alimentati a batteria e con memoria e CPU limitate

[16] Figura 3.1: http://upload.wikimedia.org/wikipedia/commons/6/63/System-architecture.jpg

- **Application Framework**: mette a disposizione degli sviluppatori le varie risorse del sistema operativo, in modo che possano sfruttarle nelle proprie applicazioni.
- **Applicazioni**: in questo livello si trovano sia le applicazioni di base fornite con i dispositivi Android (come ad esempio il telefono, i contatti, il browser ecc.) sia le applicazioni prodotte da terzi scaricabili ed installabili tramite l'Android Market.

3. Ambiente di esecuzione

Le applicazioni in Android si differenziano da quelle relative ad ambienti desktop e server per alcuni aspetti legati sia al mondo mobile sia alle specifiche intenzioni di Google:

- Risorse limitate: il limite principale di un dispositivo mobile è la capacità della batteria. Ogni clock del processore, ogni aggiornamento della memoria, ogni pixel illuminato sullo schermo prende energia dalla batteria. Di conseguenza le risorse computazionali sono limitate – la frequenza di clock è nell'ordine di centinaia di MHz, la memoria è al massimo qualche gigabyte ecc.
- Mash-up nelle applicazioni: Android estende il concetto di mash-up[17] dal mondo di internet al mondo mobile. Rende possibile la creazione di nuove applicazioni che ne includano di già esistenti.
- Intercambiabilità delle applicazioni: Android incorpora un meccanismo che è indipendente dall'implementazione di applicazioni specifiche. E' ad esempio possibile inviare una mail tramite qualsiasi applicazione sia in grado di farlo: è il sistema operativo che si occupa di individuare quali applicazioni possono inviare e-mail, avviare l'applicazione se necessario, e collegare la richiesta in modo che l'email possa essere inviata. L'utente può sostituire il browser, il lettore mp3 o il client di posta a suo piacimento, e Android si adatta automaticamente.

4. Componenti di una applicazione Android

Le applicazioni sono costruite a partire da quattro tipi di componenti di base definiti dall'architettura di Android:

- **Activity**: sono parti di codice eseguibile che vengono inizializzate dall'utente o dal sistema operativo ed eseguite solo finché sono necessarie. Possono interagire con l'utente e richiedere dati o servizi da altre attività o servizi tramite query o intent.

[17] Mash-up: letteralmente: "poltiglia", in termini informatici indica un'applicazione che usa contenuto da più sorgenti per creare un servizio completamente nuovo. Il contenuto dei mash-up è normalmente preso da terzi via API, tramite feed o Javascript

La maggior parte del codice eseguibile di una applicazione viene eseguito nel contesto di una attività. Le attività solitamente corrispondono alle schermate: ogni attività mostra all'utente una schermata.

Quando non è in esecuzione, una attività può essere terminata dal sistema operativo per liberare memoria.

- **Service**: sono parti di codice eseguibile che normalmente restano in background dal momento della istanziazione fino a quando il dispositivo viene spento. Normalmente non dispongono di una interfaccia utente.
- **Broadcast e Intent Receivers**: rispondono a richieste di servizi da un'altra applicazione. Un Broadcast Receiver risponde ad una comunicazione di un evento che riguarda tutto il sistema. Queste comunicazioni possono provenire dal sistema stesso o da qualsiasi applicazione in esecuzione nel sistema.

  Un'attività o un servizio fornisce alle altre applicazioni accesso alle proprie funzionalità attraverso l'esecuzione di un Intent Receiver, una piccola parte di codice eseguibile che risponde alle richieste di dati o servizi provenienti da altre attività. L'attività richiedente (client) invia un Intent, e lascia al sistema operativo il compito di stabilire quale applicazione debba riceverlo e reagire ad esso.
- **Content Providers**: sono creati per condividere dati con altre attività o servizi. Un Content Provider utilizza una interfaccia standard nella forma di un indirizzo URI[18] per adempiere alle richieste di altre applicazioni che potrebbero anche non conoscere quale Content Provider stanno utilizzando.

  Quando un'applicazione invia una richiesta con un determinato URI, il sistema operativo controlla quali applicazioni si sono registrate come Content Providers per l'URI specificato ed inoltra la richiesta all'applicazione corretta (avviandola se non è già in esecuzione). Se esiste

[18] Uniform Resource Identifier: (URI, acronimo più generico rispetto ad "URL") è una stringa che identifica univocamente una risorsa generica che può essere un indirizzo Web, un documento, un'immagine, un file, un servizio, un indirizzo di posta elettronica, ecc.

più di un Content Provider per l'URI specificato, il sistema chiede all'utente quale desideri utilizzare.

5. Ciclo di vita di una activity Android

Android è stato progettato sulla base degli specifici requisiti delle applicazioni mobile. In particolare è caratterizzato da meccanismi che mirano a preservare le risorse che sono limitate sui dispositivi mobili, come ad esempio memoria e batteria.

Questi meccanismi sono evidenti nel ciclo di vita delle attività, che definisce gli stati o gli eventi attraverso i quali una activity si trova a passare dal momento della sua creazione fino alla termine della sua esecuzione.

Android tiene traccia della sequenza di visualizzazione delle activity in uno stack. Ogni volta che una applicazione chiede di visualizzare una activity questa viene posta in cima allo stack e quella che era precedentemente visualizzata diventa la penultima.

Gli stati in cui una qualsiasi activity si può trovare in un dato instante (indipendentemente dal fatto che sia in cima o in fondo allo stack) sono quattro:

- **INACTIVE**: Una activity in questo stato è pronta per essere eseguita. Questo è anche lo stato in cui si trova quando viene distrutta dal sistema.
- **RUNNING**: Quando una activity è in questo stato significa che è in cima allo stack (in ogni istante una sola activity può essere in questo stato). Android non terminerà mai il processo legato a questa activity, anche in condizioni di scarse risorse disponibili.
- **PAUSED**: Una activity si trova in questo stato quando è penultima nello stack ed è ancora parzialmente visibile (cioè quando l'activity principale è una dialog o ha un effetto trasparenza).
- **STOPPED**: Una activity in questo stato non è più visibile ed è quindi candidata per la rimozione.

E' importante conoscere lo stato in cui si trova una activity perché il sistema, per liberare risorse, può decidere di chiuderne più di una. Le activity che non cercherà mai di chiudere sono quelle in stato di RUNNING, mentre quelle che cercherà di chiudere per ultime sono quelle in stato PAUSED. Le activity in stato STOPPED e INACTIVE possono essere eliminate tranquillamente.

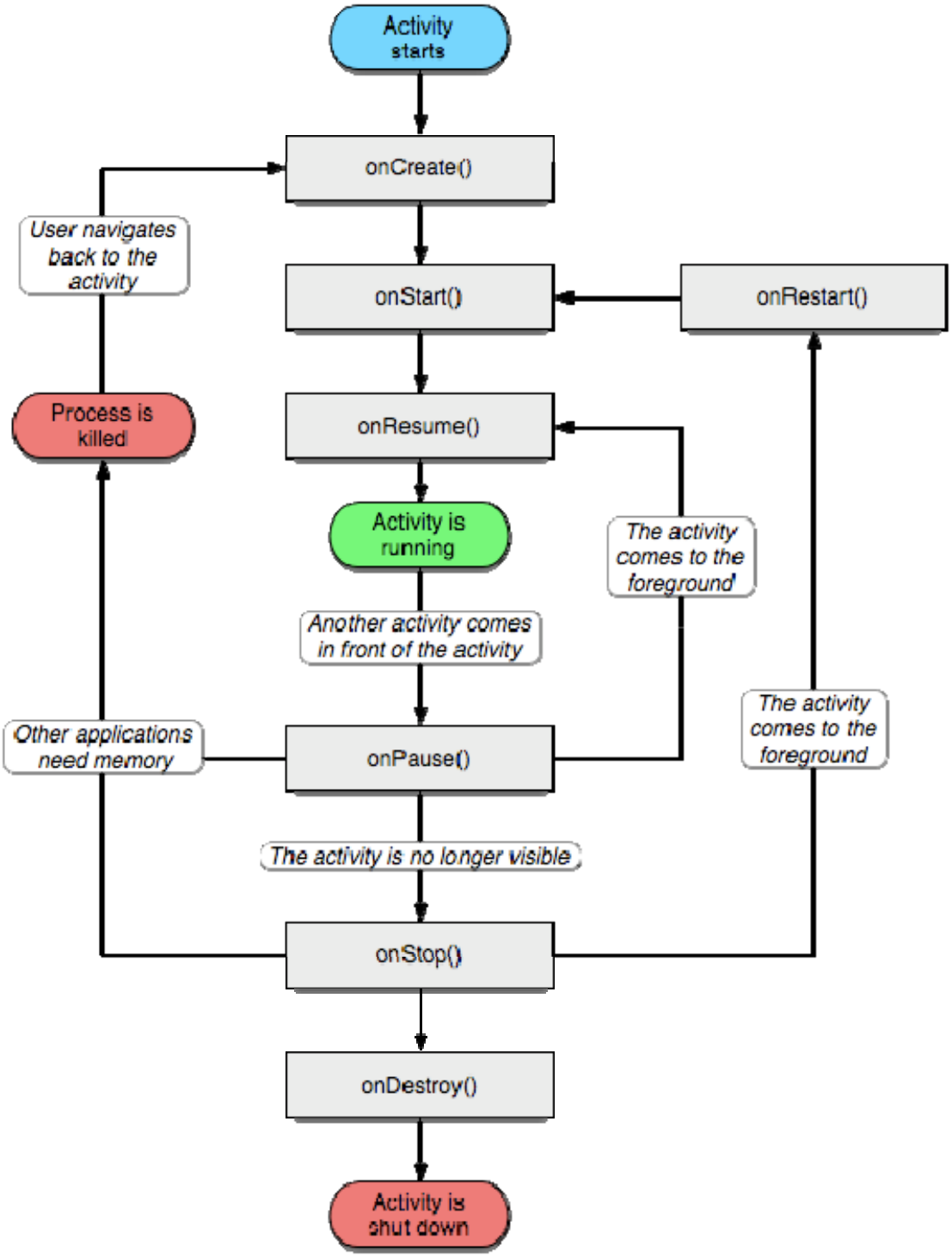

Figura 3.2 [19]

Ogni activity segue e reagisce agli eventi istanziando dei metodi che riscrivono, per ogni evento, i metodi della classe Activity:

- onCreate: invocato quando una activity viene creata per la prima volta.

[19] Figura 3.2: http://developer.android.com/images/activity_lifecycle.png

- onStart: invocato appena prima che la activity diventi visibile sullo schermo. Quando completa il suo corso, il controllo passa ad onResume (se la activity passa in primo piano sullo schermo) oppure ad onStop (se per qualche motivo l'attività non può passare in primo piano).
- onResume: a questo punto la activity si trova nello stato RUNNING e sta interagendo con l'utente.
- onPause: invocato quando il sistema sta per passare il controllo ad una activity differente
- onStop: invocato quando un'altra activity è passata in primo piano sullo schermo
- onDestroy: invocato quando una activity sta per essere distrutta

6. Ciclo di vita di un service Android

Il ciclo di vita di un service è simile a quello di una activity, ma differisce in pochi importanti dettagli:

- onCreate e onStart: i service possono essere avviati quando un client invoca il metodo *Context.startService(Intent)*. Se il service non è già in esecuzione, Android lo avvia ed invoca i metodi *onCreate* e *onStart*. Se il service è già in esecuzione, il suo metodo *onStart* è invocato nuovamente con il nuovo intent. E' quindi normale che il metodo *onStart* di un service sia invocato ripetutamente in una singola esecuzione del service.
- onResume, onPause e onStop non sono necessari: non avendo un'interfaccia utente ed essendo in esecuzione sempre in background, un service non ha bisogno di questi metodi.
- onBind: se un client ha bisogno di una connessione persistente ad un servizio, può invocare il metodo *Context.bindService*. Questo crea il service se non è già in esecuzione, e invoca il metodo *onCreate* ma non *onStart*. Invece, il metodo *onBind* è invocato con l'intent del client, e restituisce un oggetto *IBind* che può essere usato dal client per successive chiamate al service. E' normale che un service abbia contemporaneamente client ad esso legati (bind) e client che lo avviano.

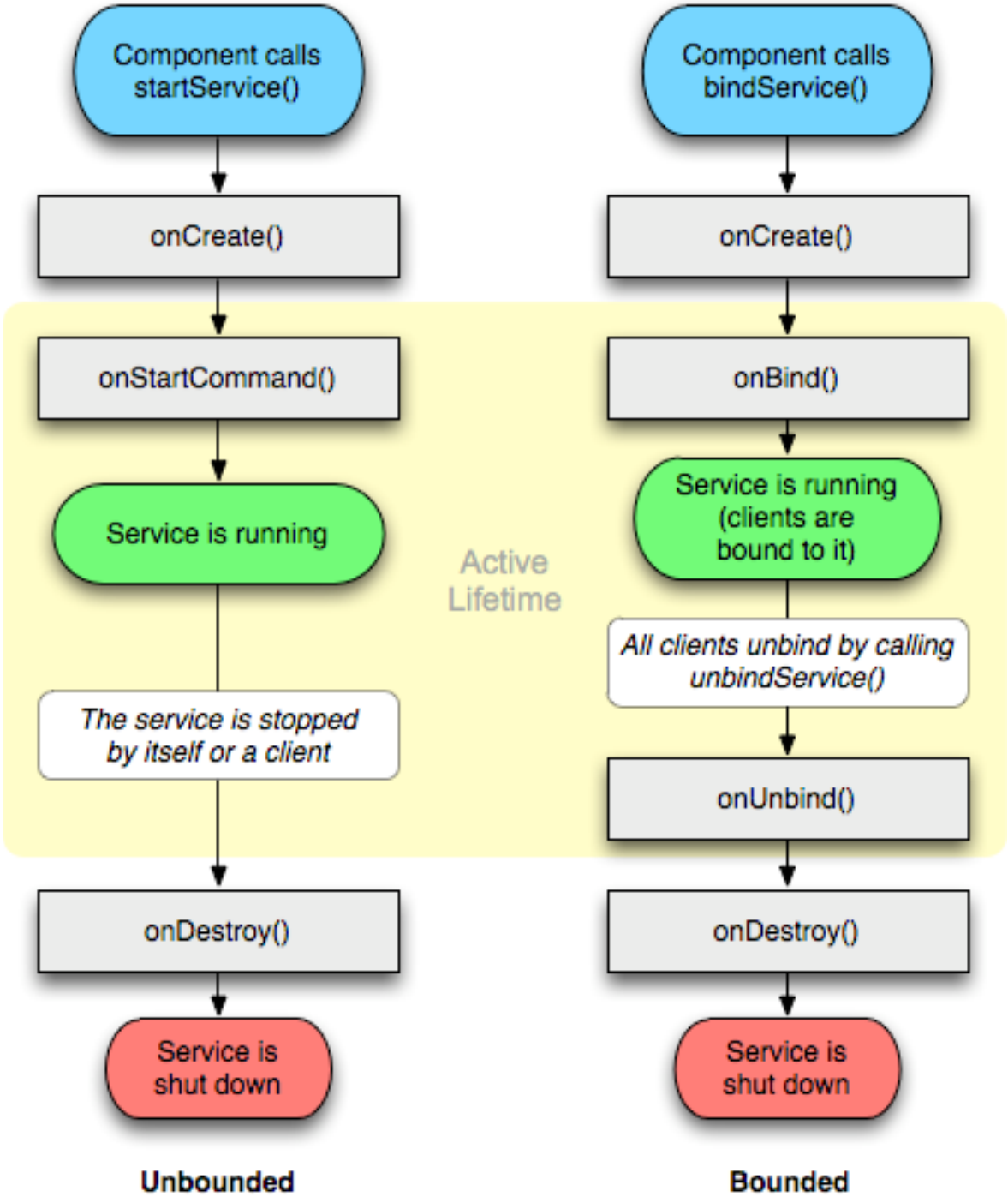

Figura 3.3 [20]

- onDestroy: è invocato quando il service sta per essere terminato, come per le activity. Android termina un servizio quando non ci sono più client legati ad esso, o nel caso in cui la memoria disponibile sia poca (in questo caso il service viene riavviato quando la memoria torna ad essere disponibile).

[20] Figura 3.3: http://developer.android.com/images/service_lifecycle.png

Capitolo 4

Il prototipo

1. Descrizione e specifiche

Il prototipo consiste in un'applicazione Android, SMSAloud, che facilita le operazioni di lettura e di scrittura di un SMS sfruttando la sintesi e il riconoscimento vocale.
È stato sviluppato tramite l'ambiente di sviluppo integrato (IDE) "Eclipse"[21] previa installazione del pacchetto degli strumenti di sviluppo per Android (SDK) forniti da Google (per l'installazione e la configurazione dell'IDE sono state seguite le istruzioni presenti sul sito ufficiale di Android, http://developer.android.com/sdk/installing.html).
L'IDE tra le altre cose fornisce la possibilità di creare delle macchine virtuali Android, che sono state molto utili durante la fase di debugging.

Prima dell'implementazione si è deciso che il prototipo dovesse soddisfare le seguenti specifiche:

- Deve essere in grado di intercettare l'arrivo di un nuovo SMS, leggere ed inviare SMS
- All'avvio deve controllare che il motore di sintesi vocale (TextToSpeech) sia installato. In caso negativo, deve esserne avviata l'installazione
- All'arrivo di un SMS questo deve essere automaticamente mostrato a video (con caratteri grandi e sfondo bianco per facilitare la lettura) e letto ad alta voce (tramite il sintetizzatore vocale)
- Nel leggere e mostrare il messaggio a video, devono essere date indicazioni anche sul mittente del messaggio. In particolare, se il numero del mittente è

[21] Per informazioni si veda: http://www.eclipse.org/org/

presente in rubrica, il mittente deve essere riconosciuto e il suo nome deve essere indicato esplicitamente

- L'utente deve avere la possibilità di riascoltare l'ultimo messaggio ricevuto anche in un secondo momento. Questo comando deve essere inviato nel modo più semplice possibile (tramite il "tocco" dello schermo o tramite la selezione del tasto "Rileggi" nel menu)
- L'utente deve avere la possibilità di rispondere all'ultimo messaggio ricevuto in modo semplice, premendo un solo tasto e (se il riconoscimento vocale è installato) dettando a voce il testo del messaggio di risposta
- Se il riconoscimento vocale non è installato sul dispositivo o se l'utente lo preferisce, deve essere possibile rispondere ad un messaggio tramite il gestore degli SMS di Android
- Deve essere possibile modificare (e salvare per esecuzioni future) le impostazioni riguardanti il timbro di voce e la velocità di pronuncia del sintetizzatore vocale. Deve inoltre essere possibile specificare se si desidera inviare direttamente l'SMS dopo il riconoscimento del parlato o se si desidera visualizzare il testo riconosciuto all'interno del gestore degli SMS di Android, per confermare l'invio manualmente.
- L'applicazione deve disporre di una icona di sistema che ne segnali l'esecuzione e che, se cliccata, permetta di portare l'applicazione in primo piano.
- All'avvio l'applicazione deve essere minimizzata nell'icona di sistema, e l'avvio deve essere notificato con un messaggio audio/video
- L'applicazione non deve interferire con la gestione standard degli SMS, quindi non deve impedirne la visualizzazione tramite altri gestori di messaggistica
- Deve essere possibile minimizzare l'applicazione per effettuare altre attività, senza che questo impedisca all'applicazione di tornare in primo piano alla ricezione di un SMS e continuare la propria esecuzione
- L'applicazione deve disporre di un tasto "Chiudi" che ne termini l'esecuzione, senza influenzare le altre funzionalità di messaggistica del dispositivo.
- Deve essere presente una sezione in cui sia possibile ottenere informazioni sulla versione e sull'autore della applicazione.

- L'interfaccia deve essere chiara, semplice e pulita, a contrasto elevato e caratteri facilmente leggibili

2. Architettura

L'applicazione è basata su quattro activity principali (il cui codice sorgente è riportato in appendice):

- SMSReceiver
- MainActivity
- Impostazioni
- Info

Ogni activity corrisponde ad una schermata, ad eccezione di SMSReceiver che, essendo un ricevitore, ha la sola funzione di ricevere gli SMS e passarli all'activity principale, MainActivity, e non è quindi dotata di un'interfaccia grafica.
L'activity Impostazioni permette di modificare le impostazioni dell'applicazione, cambiando i valori di alcune variabili memorizzate nel file "impostazioni.dat", mentre l'activity Info ha il solo scopo di mostrare informazioni sulla versione e sull'autore dell'applicazione.

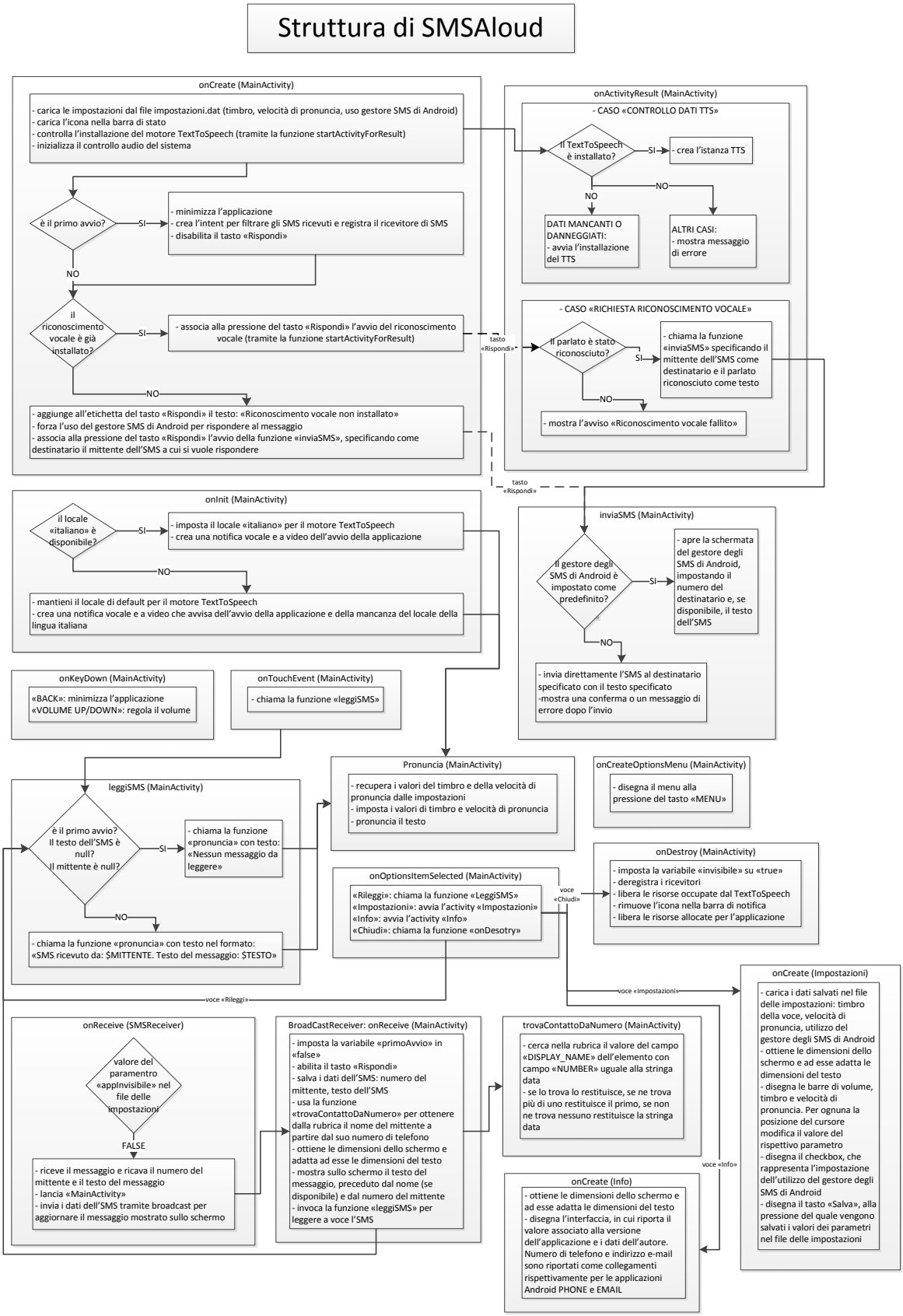

Figura 4.1: Struttura completa dell'applicazione SMSAloud

2.1. activity SMSReceiver

Questa activity estende la classe BroadcastRecevier ed è indicata come gestore degli SMS (*android.provider.Telephony.SMS_RECEIVED*) nel file AndroidManifest.xml. Questo la abilita ad intercettare gli SMS, che sono inviati tramite broadcast nel sistema.

Alla ricezione di un SMS viene invocato il metodo **onReceive**.

Questo per prima cosa controlla il valore del parametro "appInvisibile" all'interno del file delle impostazioni "impostazioni.dat". appInvisibile viene usato dalla MainActivity per comunicare il proprio stato: quando vale "TRUE" significa che la MainAcitivty non è in esecuzione, per cui non è necessario che il metodo OnReceive esegua alcuna operazione.

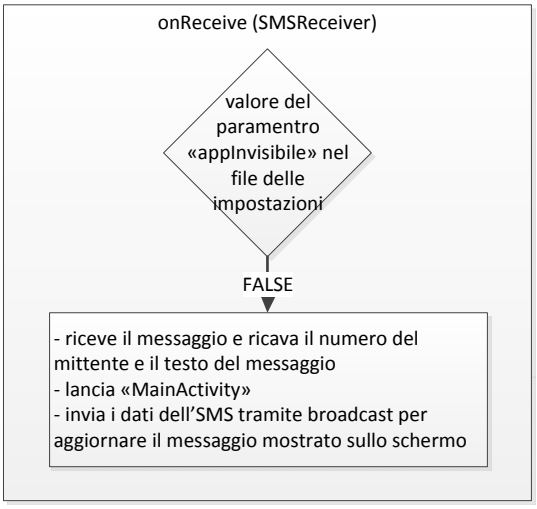

Figura 4.2: Struttura dell'activity SMSReceiver

Quando invece il parametro appInvisible vale "FALSE", onReceive prosegue la sua esecuzione: riceve i dati dell'SMS tramite intent direttamente dal sistema. Dall'intent estrae un elemento di tipo Bundle, dal quale byte per byte ricava il numero del mittente e il testo del messaggio.

A questo punto il metodo porta in primo piano l'activity MainActivity ed invia tramite broadcast un intent con flag "SMS_RECEIVED_ACTION" e contenente il numero del mittente e il testo del messaggio.

2.2 activity MainActivity

È l'activity principale dell'applicazione ed implementa la classe OnInitListener per poter usufruire dei servizi di sintesi vocale (TextToSpeech).

onCreate è il metodo che viene invocato quando l'activity viene creata.

Dal file "impostazioni.dat" carica le impostazioni riguardanti il timbro e la velocità di pronuncia del sintetizzatore vocale, e le preferenze circa l'utilizzo del gestore degli SMS

di Android. Poi imposta la variabile “appInvisible” al valore “FALSE” per comunicare all’activity SMSReceiver che MainActivity è in esecuzione.

Dopo aver caricato l’icona dell’applicazione nella barra di stato, invia un intent alla funzione “StartActivityForResult” con flag “TextToSpeech.Engine.ACTION_CHECK_TTS_ _DATA” e con requestCode “CONTROLLO_DATI_TTS” affinché controlli che il sintetizzatore vocale sia correttamente installato. Poi inizializza il controllo audio del sistema e controlla il valore della variabile “primoAvvio”: se vale “TRUE” (è il suo valore di default) significa che l’applicazione è appena stata avviata e non sono ancora stati ricevuti messaggi. Quindi disabilita il tasto “Rispondi”, registra il ricevitore degli SMS tramite la funzione “registerReceiver” usando il filtro “SMS_RECEIVED_ACTION” e minimizza l’applicazione.

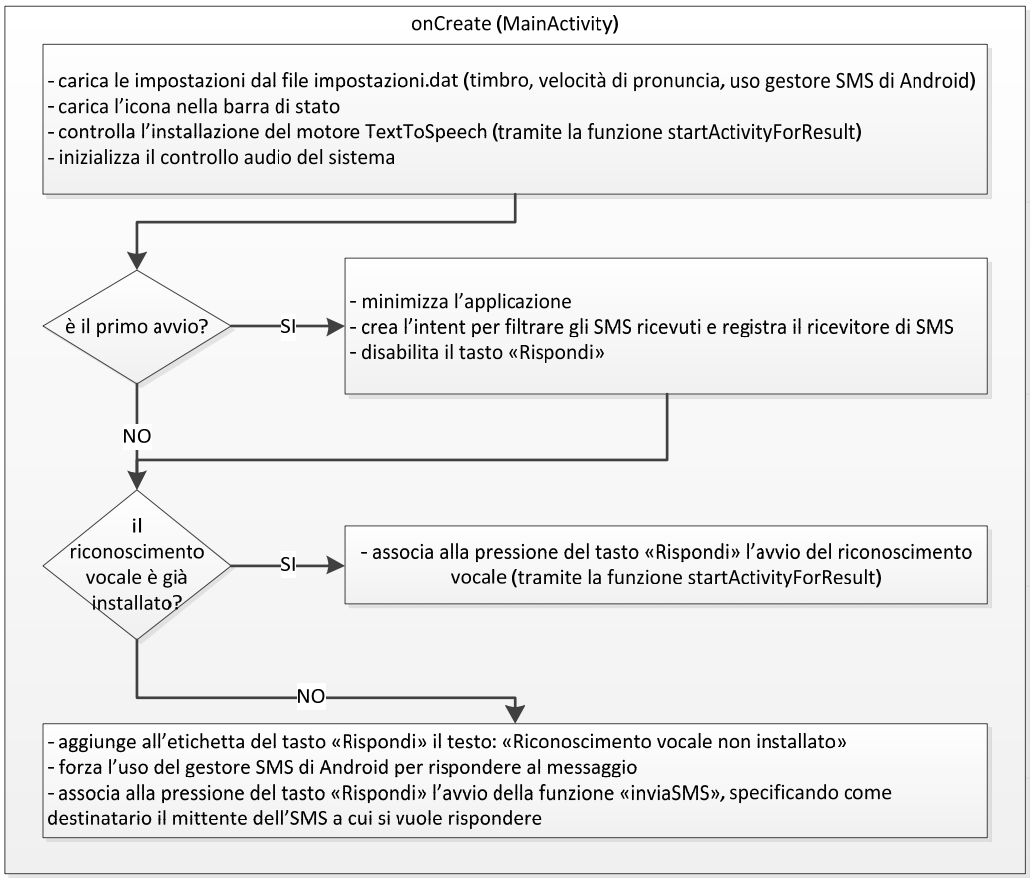

Figura 4.3: Stuttura del metodo onCreate di MainActivity

A questo punto verifica se il riconoscimento vocale è installato.

Se lo è, associa alla pressione del tasto “Rispondi” l’avvio della procedura di riconoscimento vocale tramite l’invio di un intent (di tipo “android.speech.action.

.RECOGNIZE_SPEECH" e con requestCode "RICHIESTA_RICONOSCIMENTO_VOCALE) alla funzione "StartActivityForResult".

Se il riconoscimento vocale non è installato, aggiunge all'etichetta del tasto "Rispondi" il testo "Riconoscimento vocale non installato", forza l'uso del gestore degli SMS di Android (impostando la variabile "usaGestoreSMS" a "TRUE") ed associa alla pressione del tasto "Rispondi" l'avvio della funzione "inviaSMS", passandogli come unico parametro il numero del mittente dell'SMS a cui si desidera rispondere.

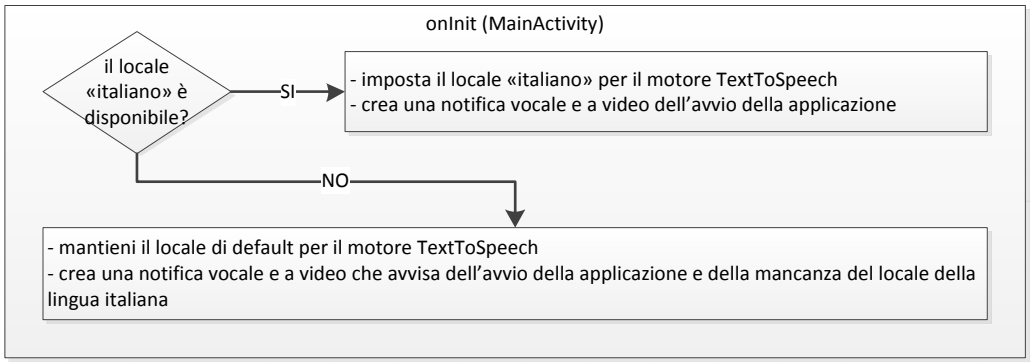

Figura 4.4: Struttura del metodo onInit di MainActivity

onInit è un metodo richiesto dalla funzionalità TextoToSpeech.

Viene anch'esso invocato alla creazione dell'activity, e in esso normalmente si imposta il locale della lingua.

In questo caso specifico verifica la disponibilità del locale di lingua italiana, e se presente lo imposta per il sintetizzatore vocale, altrimenti mantiene il locale di default.

Invia poi una notifica vocale e a video dell'avvio della applicazione e dell'eventuale mancanza della lingua italiana.

onReceive è il metodo definito nell'ambito del BroadCastReceiver: viene invocato ogni volta che viene ricevuto un nuovo SMS (dall'activity SMSReceiver).

Alla ricezione di un SMS imposta il valore della variabile "primoAvvio" in "FALSE" per comunicare al metodo onCreate che l'applicazione non è più al primo avvio.

Abilita il tasto “Rispondi”, ottiene il numero del mittente e il testo del messaggio dall’intent passato da SMSReceiver, passa il numero del mittente alla funzione “trovaContattoDaNumero” affinché cerchi nella rubrica il nome del mittente.
Dopodiché mostra a video il testo messaggio ricevuto, preceduto dal nome del mittente (se è stato trovato) e dal numero di telefono. Le dimensioni del testo sono adattate alle dimensioni dello schermo.
Infine chiama la funzione “leggiSMS” per far leggere il messaggio al sintetizzatore vocale.

onDestroy è il metodo che viene chiamato quando l’applicazione sta per essere chiusa.
I suoi compiti sono comunicare all’activity SMSReceiver che MainActivity non è più in esecuzione impostando la variabile “appInvisibile” al valore “TRUE”, de-registrare i ricevitori del BroadcastReceiver, liberare le risorse occupate dal motore di sintesi vocale e dall’applicazione stessa.

onActivityResult è un metodo che viene invocato dal sistema dopo l’esecuzione di un’operazione compiuta dal sistema stesso (avviata tramite startActivityForResult), ed ha il compito di proseguire l’esecuzione in funzione del risultato dell’operazione di sistema.
Operazioni di questo tipo sono utilizzate due volte in MainActivity, sempre all’interno del metodo onCreate: la prima volta per il controllo dell’installazione del sintetizzatore vocale, e la seconda per eseguire il riconoscimento vocale.

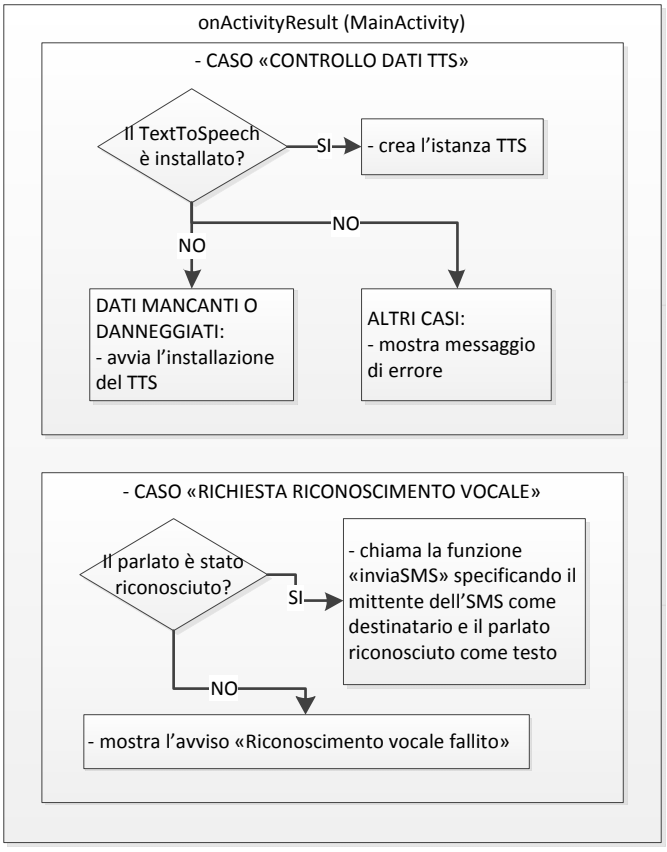

Figura 4.5: Struttura del metodo onAcitivityResult di MainActivity

Nel caso del requestCode “CONTROLLO_DATI_TTS”, se il risultato è positivo inizializza il motore TextToSpeech, se

risultano dei dati mancanti ne avvia l'installazione, e se il risultato è un errore mostra un messaggio di errore.
Nel caso del requestCode "RICHIESTA_RICONOSCIMENTO_VOCALE", se il risultato è positivo (cioè se il riconoscimento vocale è riuscito) invoca la funzione "inviaSMS" passandogli come parametri il numero del mittente e il testo riconosciuto. Se il risultato è negativo mostra un messaggio di errore.

Gestione del menu**:** due metodi si occupano di gestire il menu dell'applicazione, **onCreateOptionsMenu** (che disegna il menu seguendo le indicazioni contenute nel file /res/menu/sms_menu.xml) e **onOptionsItemSelected** (che associa ad ogni voce un'azione specifica: alla voce "Rileggi" associa il lancio della funzione "leggiSMS", alla voce "Impostazioni" associa l'avvio dell'activity "Impostazioni", alla voce "Info" associa l'avvio dell'activity "Info", e alla voce "Chiudi" associa il lancio del metodo "onDestroy").

Intercettazione dei tasti hardware: i metodi che si occupano di intercettare la pressione dei tasti sono due.
onKeyDown in MainActivity intercetta la pressione del tasto "BACK", a cui associa la minimizzazione dell'applicazione all'icona di sistema (in modo che l'applicazione non venga chiusa se non tramite la voce "Chiudi" del menu), e la pressione dei tasti "VOULUME_UP" e "VOLUME_DOWN", a cui associa rispettivamente l'innalzamento e l'abbassamento del volume dei flussi multimediali del sistema (questi pulsanti normalmente modificano il volume della suoneria).
onTouchEvent in MainActivity intercetta il tocco dello schermo, a cui associa la chiamata della funzione "leggiSMS".

Funzioni di utilità generale: all'interno di MainActivity sono definite alcune funzioni che svolgono compiti specifici, e che vengono utilizzate più volte dai metodi.

inviaSMS riceve come argomenti il numero di telefono del destinatario e il testo dell'SMS da inviare, e varia il proprio comportamento in funzione del valore della variabile "usaGestoreSMS" presente nel file delle impostazioni "impostazioni.dat".

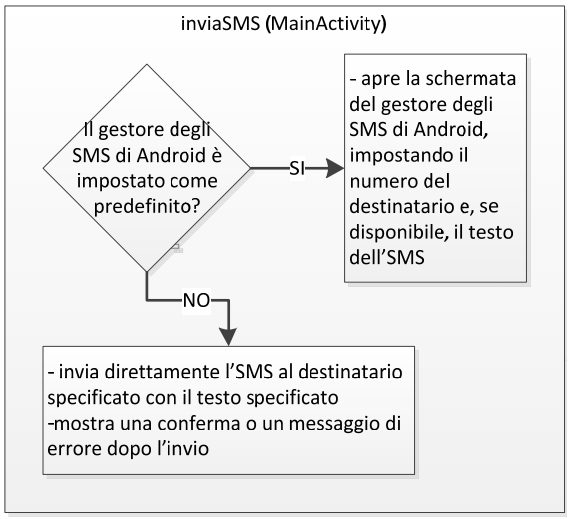

Figura 4.6: Struttura della funzione inviaSMS di MainActivity

Se il gestore degli SMS di Android è impostato come predefinito (e quindi se "usaGestoreSMS" ha valore "TRUE") la funzione chiama (tramite un intent) il gestore degli SMS di Android e gli passa il numero del destinatario e l'eventuale testo del messaggio.

Se il gestore degli SMS di Android non è impostato come predefinito (e quindi se "usaGestoreSMS" ha valore "FALSE") invia automaticamente un SMS con il testo specificato al destinatario specificato. Poi registra un BroadcastReceiver per ricevere informazioni sull'esito dell'invio, e quando le riceve mostra a video una notifica riguardante l'esito dell'invio.

trovaContattoDaNumero riceve come argomento una stringa contenente un numero di telefono e restituisce una stringa contenente il nome del contatto della rubrica corrispondente a quel numero.

In pratica esegue una sorta di query, cercando il "DISPLAY_NAME" dell'elemento della rubrica avente "NUMBER" uguale al numero di telefono dato.

Se la ricerca trova più risultati, la funzione restituisce solo il primo nominativo. Se non viene trovato nessun contatto, la funzione restituisce il numero di telefono fornito nell'argomento.

leggiSMS dopo aver controllato che il nome del mittente e il testo del messaggio non siano vuoti e che l'applicazione non sia appena stata avviata (e che quindi non ci siano messaggi da leggere), legge l'ultimo SMS ricevuto nel formato "SMS ricevuto da: $nomeDelMittente. Testo del messaggio: $testoSMS".

Se invece si tratta del primo avvio pronuncia la frase: “Nessun messaggio da leggere”.
Per pronunciare le frasi utilizza la funzione “pronuncia”.

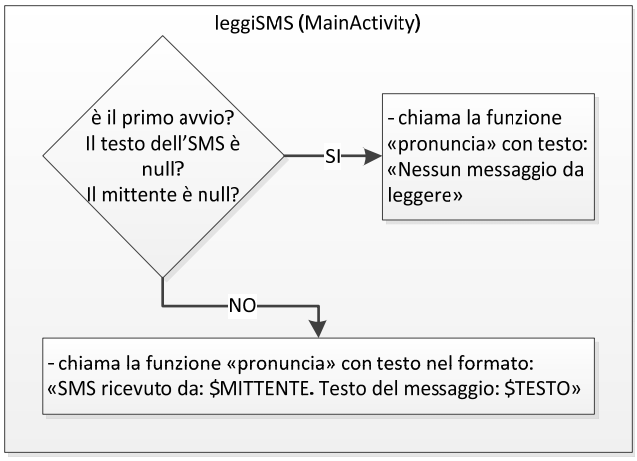

Figura 4.7: Struttura della funzione leggiSMS di MainActivity

pronuncia usa il sintetizzatore vocale TextToSpeech per leggere la stringa che gli viene passata come argomento.
Prima di avviare il sintetizzatore carica le impostazioni di timbro e velocità di pronuncia dal file “impostazioni.dat” (se le impostazioni non sono presenti usa i valori di default).

2.3 activity Impostazioni

Alla creazione viene invocato l’unico metodo di questa activity, **onCreate**.
Esso ha il compito di disegnare l’interfaccia grafica (seguendo le indicazioni contenute nel file /res/layout/Impostazioni.xml e adattando le dimensioni dei caratteri in funzione delle dimensioni dello schermo), caricare i valori delle impostazioni memorizzati nel file “impostazioni.dat” e gestire eventuali modifiche su questi valori.
In particolare disegna tre barre di scorrimento i cui cursori riportano i valori attuali di volume del sistema, timbro di voce e velocità di pronuncia del sintetizzatore vocale, e un checkbox, che indica se nelle impostazioni si sia scelto di usare il gestore degli SMS di Android prima dell’invio dei messaggi. Lo spostamento del cursore di ogni barra modifica il valore della relativa voce, e la presenza o meno della crocetta nel checkbox modifica il valore relativo (che può essere TRUE o FALSE).

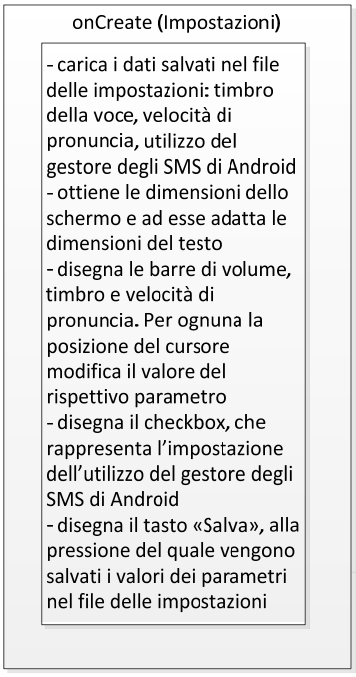

Figura 4.8: Struttura dell’activity Impostazioni

Premendo il tasto “Salva”, disegnato in fondo alla schermata, è possibile sovrascrivere i valori presenti nel file “impostazioni.dat” con i valori modificati e sostituire il valore del volume di sistema con il valore indicato. Un toast avvisa dell’avvenuto salvataggio delle impostazioni, dopodiché l’esecuzione del metodo (e dell’activity) termina.
L’esecuzione di questa activity può terminare anche senza salvare le modifiche, premendo il tasto “BACK” sul dispositivo.

2.4 activity Info

Alla creazione viene invocato l’unico metodo di questa activity, **onCreate**.
Il metodo ha l’unico compito di disegnare l’interfaccia grafica (seguendo le indicazioni contenute nel file /res/layout/Info.xml e adattando le dimensioni dei caratteri in funzione delle dimensioni dello schermo), che in questo caso mostra semplicemente delle informazioni sull’autore e sulla versione della applicazione.

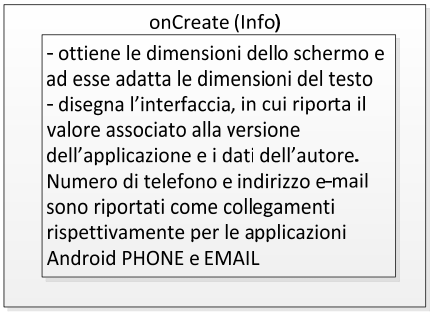

Figura 4.9: Struttura dell’activity Info

Il numero di telefono e l’indirizzo di posta elettronica sono indicati come di tipo “autoLink”, quindi se selezionati impostano automaticamente una telefonata al numero specificato o un nuovo messaggio di posta elettronica verso l’indirizzo di posta specificato.
L’esecuzione di questa activity termina alla pressione del tasto “BACK” sul dispositivo.

3. Interfacce Utente

Le interfacce utente sono state progettate per essere essenziali, semplici e di facile lettura.
Ogni schermata è stata volutamente disegnata con testo nero su sfondo bianco, per permettere la lettura anche in condizioni di luce non ottimali. Inoltre il carattere del testo viene adattato automaticamente alle dimensioni dello schermo per favorire la leggibilità.

Si è cercato di ridurre al minimo la necessità di interazione da parte dell'utente, ad esempio permettendo di rileggere il testo dell'ultimo messaggio ricevuto semplicemente toccando lo schermo in un punto qualsiasi, o portando automaticamente l'applicazione in primo piano alla ricezione di un nuovo SMS.

Figura 4.10: da sinistra verso destra: schermata dell'activity "MainActivity" prima della ricezione di un SMS (nel caso specifico in cui il riconoscimento vocale non è installato nel sistema); schermata dell'activity "MainActivity" dopo la ricezione di un SMS (sempre nel caso specifico in cui il riconoscimento vocale non è installato nel sistema).

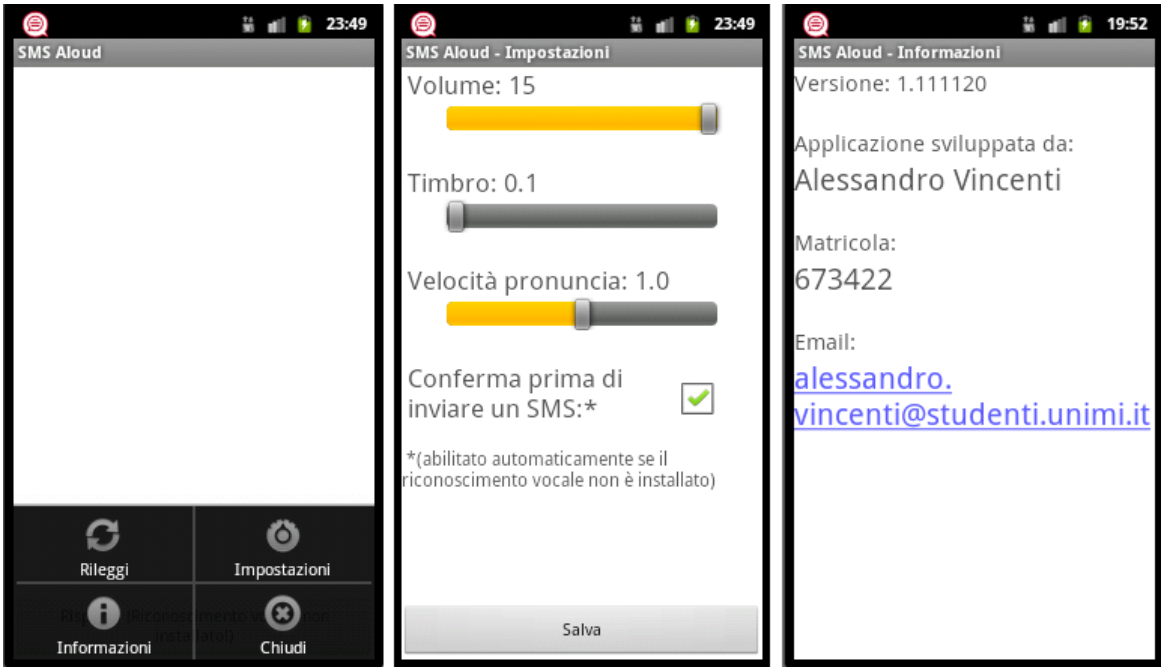

Figura 4.11: da sinistra verso destra: schermata della mainActivity alla pressione del tasto “Menu”; schermata dell’activity “Impostazioni”; schermata dell’activity “Info”

Capitolo 5

Valutazioni e sviluppi futuri

1. Indagine iniziale

Durante lo studio e la progettazione di questo prototipo è stata condotta un'indagine su un campione di 6 persone di età compresa tra 65 e 80 anni, alcune delle quali affette da limitazioni visive.

Inizialmente è stata condotta un'intervista con lo scopo di stabilire quali fossero le maggiori difficoltà riscontrare nell'uso del telefono cellulare e di ricavarne indicazioni per implementare un'applicazione che potesse risolverle o almeno alleviarle.

In questa fase è emerso che la maggior parte degli intervistati usa il cellulare esclusivamente per ricevere ed inviare telefonate. Pur riconoscendo l'utilità della comunicazione via SMS, questa non è abitualmente sfruttata per i seguenti motivi:

- difficoltà di navigazione all'interno dei menu del telefono
- scarsa leggibilità del testo (dovuta a caratteri troppo piccoli)
- difficoltà nella digitazione dei messaggi da inviare

Tutti gli intervistati hanno dimostrato interesse alla proposta di un'interfaccia vocale per rendere più agevole la gestione degli SMS.

La funzionalità più importante è stata individuata nel semplificare la lettura del messaggio ed eventualmente la risposta.

Durante lo sviluppo dell'applicazione sono stati utilizzati due tester, che con le loro osservazioni hanno contribuito all'affinamento della modalità operativa ed al miglioramento delle interfacce utente.

2. Indagine conclusiva

Al termine del lavoro l'applicazione è stata proposta a tutti i partecipanti all'intervista iniziale per verificare il raggiungimento dello scopo prefisso.
In generale l'applicazione ha ricevuto l'apprezzamento di tutti i partecipanti, seppur con le seguenti osservazioni:

- occorre il supporto di un utente "esperto" per l'installazione
- la lettura immediata alla ricezione di un SMS può non essere sempre opportuna
- il riconoscimento vocale usato per l'invio del messaggio di risposta richiede la connessione ad internet
- per ottenere un riconoscimento vocale accurato è necessario scandire le parole

3. Valutazione dei risultati del progetto

L'applicazione oggetto di questo progetto ha riscosso un buon successo fra tutti coloro che hanno avuto modo di provarla. Essa si è dimostrata utile anche in ambiti diversi da quelli che l'hanno originata, come ad esempio l'uso durante la guida di veicoli.
Alcune delle osservazioni emerse durante l'indagine conclusiva sono legate ai limiti attuali della tecnologia adottata e potranno ragionevolmente trovare soluzione con le prossime release di Android. Le modalità di installazione delle applicazioni, ad esempio, potranno essere semplificate ulteriormente, così come potrà essere rilasciata una libreria per il riconoscimento vocale che non necessiti di un collegamento internet.

4. Sviluppi futuri

Gli sviluppi futuri, prendendo anche in considerazione le altre osservazioni ricevute, potranno essere:

- implementare un'opzione per disabilitare la riproduzione vocale automatica alla ricezione di un SMS
- introdurre la possibilità di riprodurre vocalmente anche gli SMS presenti in archivio

- implementare un’opzione per abilitare l’avvio automatico dell’applicazione all’accensione del dispositivo
- aggiungere il supporto multilingue alle interfacce e ai processi di sintesi e riconoscimento vocale

BIBLIOGRAFIA

Block, B. (2011), *Android Captures #2 Ranking Among Smartphone Platforms in EU5*, Londra (Regno Unito), comScore Inc.

Gill, J. M. (2001), *Requirements for the Interconnection of Assistive Technology Devices and Information and Communication Technology Systems*, http://www.tiresias.org/research/reports/

Gill, J.M. (2004), *Access-Ability: Making Technology More Useable By People With Disabilities*, http://www.tiresias.org/research/reports/

Gill, J. M. (2007), Accessibility for Visitors, http://www.tiresias.org/research/reports/

Jeffords, J. & Harkin, T. (1988), The US technology-related assistance for individuals with disabilities act, U.S. Public Law 100-407, Section 3.1

Lee, W. M. (2011), Beginning Android Application Development, 1st edition, Indianapolis (Indiana, US), Wiley Publishing Inc.

Lewing D., Adshead S., Glennon B., Williamson B., Moore T., Damodaran L., Hansell P. (2010), *Assisted living technologies for older and disabled people in 2030*, Londra (Regno Unito), Plum Consulting

Lin, L. (2011), *The rise of the low-priced Android smartphone market*, Taipei (Taiwan), DIGITIMES Research

Petrie, H. (1997), User-centred design and evaluation of adaptive and assistive technology for disabled and elderly users, Informationstechnik und Technische Informatik, 39(2), 7-12

Roe, P. R. W. (2007), *Towards an Inclusive Future: impact and Wider Potential of Information Communication Technologies*, Bruxelles (Belgio), COST

Rogers, R. & Lombardo, J. (2009), *Android Application Development: Programming with the Google SDK*, Sebastopol (California, US), O'Reilly Media Inc.

http://developer.android.com/

http://www.openhandsetalliance.com/android_overview.html

Appendice

1. Codice sorgente dell'activity SMSReceiver

```
/*    Copyright (C) <2011>  <Alessandro Vincenti>

      This file is part of SMSAloud

      SMSAloud is free software: you can redistribute it and/or modify
      it under the terms of the GNU General Public License as published
      by the Free Software Foundation, either version 3 of the License,
      or (at your option) any later version.

      SMSAloud is distributed in the hope that it will be useful,
      but WITHOUT ANY WARRANTY; without even the implied warranty of
      MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
      GNU General Public License for more details.

      You should have received a copy of the GNU General Public License
      along with this program.  If not, see
      <http://www.gnu.org/licenses/>.
*/


package it.alvin.SMSAloud;


import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.telephony.SmsMessage;


public class SMSReceiver extends BroadcastReceiver {

      private SharedPreferences impostazioni;
      private boolean appInvisibile;

      @Override
      public void onReceive(Context context, Intent intent) {
            impostazioni =
context.getSharedPreferences("impostazioni.dat", 0);
            appInvisibile = impostazioni.getBoolean("appInvisibile",
false);
            //---se l'applicazione non è in modalità invisibile, riceve
            //---il messaggio e lo invia alla MainAcitvity---
            if (appInvisibile == false) {
                  //---riceve il messaggio---
                  Bundle bundle = intent.getExtras();
```

```
                    SmsMessage[] msgs = null;
                    String mittenteSms = "";
                    String testoSms = "";
                    if (bundle != null) {
                          //---legge il messaggio---
                          Object[] pdus = (Object[]) bundle.get("pdus");
                          msgs = new SmsMessage[pdus.length];
                          for (int i=0; i<msgs.length; i++){
                                msgs[i] =
SmsMessage.createFromPdu((byte[])pdus[i]);
                                mittenteSms +=
msgs[i].getOriginatingAddress().toString();
                                testoSms +=
msgs[i].getMessageBody().toString();
                          } // fine ciclo for
                          //---lancia l'attività principale---
                          Intent mainActivityIntent = new Intent(context,
MainActivity.class);

      mainActivityIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
                          context.startActivity(mainActivityIntent);
                          //---invia un broadcast per aggiornare il
                          //---messaggio mostrato nell'attività principale
                          Intent broadcastIntent = new Intent();

      broadcastIntent.setAction("SMS_RECEIVED_ACTION");
                          String[] datiSms = new String[] { mittenteSms,
testoSms };
                          broadcastIntent.putExtra("datiSms", datiSms);
                          context.sendBroadcast(broadcastIntent);
                    } // fine if
              } // fine if (se l'applicazione è in modalità invisibile non
                // fa nulla)
      } // fine onReceive

}
```

2. Codice sorgente dell'activity MainActivity

```
package it.alvin.SMSAloud;


import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.os.Bundle;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
```

```
import android.content.IntentFilter;
import android.content.SharedPreferences;
import android.content.pm.PackageManager;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.speech.tts.TextToSpeech;
import java.util.ArrayList;
import java.util.Locale;
import android.telephony.SmsManager;
import android.util.DisplayMetrics;
import android.util.Log;
import android.util.TypedValue;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import android.provider.ContactsContract.PhoneLookup;
import android.media.AudioManager;
import android.net.Uri;
import android.database.Cursor;


public class MainActivity extends Activity implements OnInitListener {


      Button btnRispondi;
      IntentFilter intentFilter;
      String[] datiSms;
      String mittenteSms = "";
      String nomeMittenteSms = "";
    String testoSms = "";
    Boolean primoAvvio = true;
      private TextToSpeech tts;
      private static final int CONTROLLO_DATI_TTS = 43214321;
      private static final int RICHIESTA_RICONOSCIMENTO_VOCALE =
12341234;
      private NotificationManager gestoreNotifiche;
      private static final int ID_NOTIFICA = 1;

      //---dati per le impostazioni
      private SharedPreferences impostazioni;
      private AudioManager audio;
      private float timbro, velPronuncia;
      private boolean usaGestoreSms;


      //---invocato quando l'attività viene creata---
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
      super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //---carica le impostazioni---
            impostazioni = getSharedPreferences("impostazioni.dat",
MODE_PRIVATE);
            timbro = impostazioni.getFloat("timbro", 0.1F);
            velPronuncia = impostazioni.getFloat("velPronuncia", 1.0F);
            usaGestoreSms = impostazioni.getBoolean("usaGestoreSms",
true);
            //---disinserisce la modalità invisibile---
            SharedPreferences.Editor modificaImpostazioni =
impostazioni.edit();
            modificaImpostazioni.putBoolean("appInvisibile", false);
            modificaImpostazioni.commit();
            //---carica l'icona nella barra di stato
        gestoreNotifiche = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
        Notification notifica = new Notification(R.drawable.smsaloud,
"", System.currentTimeMillis());
        Intent intentNotifica = new Intent(this, MainActivity.class);
        PendingIntent contentIntent = PendingIntent.getActivity(this, 0,
intentNotifica, 0);
        notifica.setLatestEventInfo(this, "SMS Aloud", "Clicca qui per
aprire SMS Aloud", contentIntent);
        notifica.flags |= Notification.FLAG_NO_CLEAR;
        gestoreNotifiche.notify(ID_NOTIFICA, notifica);
        // controlla l'installazione del TextToSpeech
        Intent intentControlloTTS = new Intent();

      intentControlloTTS.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_
DATA);
      startActivityForResult(intentControlloTTS, CONTROLLO_DATI_TTS);
      tts = new TextToSpeech(this, this);
        //---inizializza il controllo audio del sistema
        audio = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
            //---tasto "Rispondi"---
            btnRispondi = (Button) findViewById(R.id.btn_rispondi);
            //---se il programma è appena stato lanciato:
            if (primoAvvio == true) {
                  //---minimizza l'applicazione
                  this.moveTaskToBack(true);
                  //---registra il ricevitore di SMS---
                  intentFilter = new IntentFilter();
                  intentFilter.addAction("SMS_RECEIVED_ACTION");
                  registerReceiver(intentReceiver, intentFilter);
                  //---disabilita il tasto "Rispondi"---
            btnRispondi.setEnabled(false);
           }
           PackageManager pm = getPackageManager();
           Intent riconosciParlato = new
Intent("android.speech.action.RECOGNIZE_SPEECH");
          //---se il riconoscimento vocale è installato---
```

```
          if (pm.queryIntentActivities(riconosciParlato, 0).size() != 0)
{
            btnRispondi.setOnClickListener(new View.OnClickListener() {
                        @Override
                  public void onClick(View v) {
                              //---azione associata alla pressione del
                              //---tasto "Rispondi" con riconoscimento
                              //---vocale installato
                              pronuncia("Pronunciare il testo del
messaggio");
                              try {
                                    Thread.sleep(1800);
                              } catch (InterruptedException e) {
                                    Log.i("Eccezione nella funzione
sleep():", e.getMessage());
                              }
                              Intent intentRispondi = new
Intent("android.speech.action.RECOGNIZE_SPEECH");

      intentRispondi.putExtra("android.speech.extra.LANGUAGE_MODEL",
"free_form");

      intentRispondi.putExtra("android.speech.extra.PROMPT",
"Pronunciare il testo del messaggio di risposta");
                           startActivityForResult(intentRispondi,
RICHIESTA_RICONOSCIMENTO_VOCALE);
                              }
                  });
          //---se il riconoscimento vocale non è installato
          } else {
            btnRispondi.setText("Rispondi (Riconoscimento vocale non
installato!)");
            //---imposta l'uso del gestore di default degli SMS
                  modificaImpostazioni.putBoolean("usaGestoreSms",
true);
            btnRispondi.setOnClickListener(new View.OnClickListener() {
                        @Override
                  public void onClick(View v) {
                              //---richiama il gestore di default degli
                              //---SMS inviandogli il numero del
                              //---mittente
                              SharedPreferences.Editor
modificaImpostazioni = impostazioni.edit();

      modificaImpostazioni.putBoolean("usaGestoreSms", true);
                              modificaImpostazioni.commit();
                              inviaSMS(mittenteSms, "");
                        } // fine onClick
                  });
          } // fine else
    } // fine onCreate
```

```
    @Override
    public void onInit(int status) {
        //--- controlla che l'italiano sia installato---
      Locale italiano = new Locale("it", "IT");
            if (tts.isLanguageAvailable(italiano) >=
TextToSpeech.LANG_AVAILABLE) {
                  tts.setLanguage(italiano);
                  pronuncia("S.M.S. elààud avviato");
                  Toast.makeText(getBaseContext(), "SMS Aloud avviato",
Toast.LENGTH_LONG).show();
            } else {
                  pronuncia("S.M.S. aloud started. Italian Language is
not available");
                  Toast.makeText(getBaseContext(), "SMS Aloud started.
Italian Language is not available", Toast.LENGTH_LONG).show();
            }
    } // fine onInit


    private BroadcastReceiver intentReceiver = new BroadcastReceiver() {
          @Override
          public void onReceive(Context context, Intent intent) {
            primoAvvio = false;
            btnRispondi.setEnabled(true);
            datiSms = intent.getStringArrayExtra("datiSms");
            mittenteSms = datiSms[0];
            testoSms = datiSms[1];
            nomeMittenteSms = trovaContattoDaNumero(mittenteSms);
            //---ottiene le dimensioni dello schermo e ad esse adatta la
            //---dimensione del testo---
            DisplayMetrics metrics = new DisplayMetrics();
            getWindowManager().getDefaultDisplay().getMetrics(metrics);
            int larghezzaSchermo = metrics.widthPixels;
            int dimensioniTesto = (int)
(Math.floor(larghezzaSchermo/9));
            //---mostra l'SMS ricevuto in un TextView---
            TextView schermataSms = (TextView)
findViewById(R.id.textView1);
            schermataSms.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTesto);
            //---se il nome del mittente è stato trovato, mostra il
            //---messaggio nel formato "SMS ricevuto da nome (numero):
            //---testo"
            if (mittenteSms != nomeMittenteSms) {
                  schermataSms.setText("SMS ricevuto da " +
nomeMittenteSms  + " (" + mittenteSms +"):\n" + testoSms);
            //---se il nome del mittente NON è stato trovato, mostra il
            //---messaggio nel formato "SMS ricevuto da numero: testo"
            } else {
                  schermataSms.setText("SMS ricevuto da " +
nomeMittenteSms  + ":\n" + testoSms);
            }
            //---legge l'SMS---
```

```
            leggiSms(1);
            } // fine onReceive
      };


      @Override
    protected void onDestroy() {
      //---imposta la modalità invisibile---
      SharedPreferences.Editor modificaImpostazioni =
impostazioni.edit();
      modificaImpostazioni.putBoolean("appInvisibile", true);
      modificaImpostazioni.commit();
      //---de-registra i ricevitori---
        unregisterReceiver(intentReceiver);
        Intent smsReceiver = new Intent(this, SMSReceiver.class);
        stopService(smsReceiver);
        //---libera le risorse occupate dal motore TTS---
        tts.shutdown();
        //---rimuove l'icona nella barra di notifica---
        gestoreNotifiche.cancel(ID_NOTIFICA);
        //---libera le risorse---
        super.onDestroy();
        System.runFinalizersOnExit(true);
        System.exit(0);
    }


    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
      // caso richiesta riconoscimento vocale
      if (requestCode == RICHIESTA_RICONOSCIMENTO_VOCALE) {
            if (resultCode == RESULT_OK) {
                  ArrayList<String> matches =
data.getStringArrayListExtra("android.speech.extra.RESULTS");
                  String testo = matches.get(0).toString();
                  inviaSMS(mittenteSms, testo);
            } else {
                  Toast.makeText(getBaseContext(), "Riconoscimento
vocale fallito", Toast.LENGTH_LONG).show();
            }
      } // fine if (caso richiesta riconoscimento vocale)
      // caso controllo dati TTS
      if (requestCode == CONTROLLO_DATI_TTS) {
            switch (resultCode) {
            case TextToSpeech.Engine.CHECK_VOICE_DATA_PASS:
                  // dati già installati, crea l'istanza TTS
                  tts = new TextToSpeech(this, this);
                  break;
            case TextToSpeech.Engine.CHECK_VOICE_DATA_BAD_DATA:
            case TextToSpeech.Engine.CHECK_VOICE_DATA_MISSING_DATA:
            case TextToSpeech.Engine.CHECK_VOICE_DATA_MISSING_VOLUME:
                  // dati mancanti, li installa
                  Intent intentIstallaTTS = new Intent();
```

```
      intentIstallaTTS.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_
DATA);
                  startActivity(intentIstallaTTS);
                 break;
            case TextToSpeech.Engine.CHECK_VOICE_DATA_FAIL:
            default:
                Log.e("TTS test", "Errore. TTS non disponibile");
            }
      } // fine if (caso controllo dati TTS)
      super.onActivityResult(requestCode, resultCode, data);
    }


//==========================GESTIONE DEL MENU==========================


      // disegna il menu
      @Override
      public boolean onCreateOptionsMenu(Menu menu) {
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.sms_menu, menu);
            return true;
      }


      // gestisce i pulsanti del menu
      @Override
      public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case R.id.rileggi:
                  leggiSms(1);
                  return true;
            case R.id.impostazioni:
                  startActivity(new Intent(this, Impostazioni.class));
                  return true;
            case R.id.info:
                  startActivity(new Intent(this, Info.class));
                  return true;
            case R.id.chiudi:
                  onDestroy();
            default:
                  return super.onOptionsItemSelected(item);
            } // fine switch
      }


//==========================INTERCETTA TASTI==========================


      public boolean onKeyDown(int keyCode, KeyEvent event) {
            //---intercetta la pressione del tasto "BACK": fa in modo
            //---che il programma non venga chiuso premendo il tasto
```

```
            //---"Indietro", ma solo tramite l'apposito tasto "Chiudi"
            //---nel menù---
            if (keyCode == KeyEvent.KEYCODE_BACK &&
event.getRepeatCount() == 0) {
                  this.moveTaskToBack(true);
                  return true;
            } // fine if
            //---intercetta la pressione dei comandi del volume nella
            //---applicazione principale: fa in modo che modifichino le
            //---impostazioni del volume delle applicazioni multimediali
            //---(questa applicazione compresa) invece del volume della
            //---suoneria
            if (keyCode == KeyEvent.KEYCODE_VOLUME_UP) {
              audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
AudioManager.ADJUST_RAISE, AudioManager.FLAG_SHOW_UI);
              return true;
            } // fine if
            if (keyCode == KeyEvent.KEYCODE_VOLUME_DOWN) {
              audio.adjustStreamVolume(AudioManager.STREAM_MUSIC,
AudioManager.ADJUST_LOWER, AudioManager.FLAG_SHOW_UI);
              return true;
            } // fine if
            return super.onKeyDown(keyCode, event);
      } // fine onKeyDown


      //---intercetta il tocco dello schermo: se vi è presente un SMS,
      //---lo rilegge
      public boolean onTouchEvent(MotionEvent event) {
            leggiSms(1);
            return true;
      } // fine onTouchEvent


//==============================FUNZIONI=================================

      //---invia un SMS a un altro dispositivo---
    private void inviaSMS(String numeroTel, String testoSms) {
      usaGestoreSms = impostazioni.getBoolean("usaGestoreSms", true);
      //---se il gestore degli SMS è impostato come predefinito, apre la
      //---schermata del gestore degli SMS prima di inviare il
      //---messaggio, passandogli il numero del destinatario e, se c'è,
      //---il testo riconosciuto
      if (usaGestoreSms == true) {
            Intent i = new Intent (android.content.Intent.ACTION_VIEW);
            i.putExtra("address", numeroTel);
            i.putExtra("sms_body", testoSms);
            i.setType("vnd.android-dir/mms-sms");
            startActivity(i);
            //---se il gestore degli sms NON è impostato come
            //---predefinito, invia il messaggio automaticamente---
            } else {
                  String SENT = "SMS_SENT";
```

```
                String DELIVERED = "SMS_DELIVERED";
                PendingIntent sentPI =
PendingIntent.getBroadcast(this, 0, new Intent(SENT), 0);
                PendingIntent deliveredPI =
PendingIntent.getBroadcast(this, 0, new Intent(DELIVERED), 0);
                SmsManager sms = SmsManager.getDefault();
                sms.sendTextMessage(numeroTel, null, testoSms, sentPI,
deliveredPI);
                //---quando l'SMS è stato inviato---
                registerReceiver(new BroadcastReceiver() {
                      @Override
                      public void onReceive(Context arg0, Intent arg1)
{
                            switch (getResultCode()) {
                            case Activity.RESULT_OK:
                                  Toast.makeText(getBaseContext(),
"SMS inviato!", Toast.LENGTH_SHORT).show();
                                  pronuncia("S.M.S. inviato");
                                  break;
                     case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                          Toast.makeText(getBaseContext(), "Errore!",
Toast.LENGTH_SHORT).show();
                          pronuncia("Errore! S.M.S. non inviato");
                          break;
                     case SmsManager.RESULT_ERROR_NO_SERVICE:
                          Toast.makeText(getBaseContext(), "Nessun
servizio!", Toast.LENGTH_SHORT).show();
                          pronuncia("Nessun servizio! S.M.S. non
inviato");
                          break;
                     case SmsManager.RESULT_ERROR_NULL_PDU:
                          Toast.makeText(getBaseContext(), "PDU vuoto!",
Toast.LENGTH_SHORT).show();
                          tts.speak("Errore! S.M.S. non inviato",
TextToSpeech.QUEUE_FLUSH, null);
                          break;
                     case SmsManager.RESULT_ERROR_RADIO_OFF:
                          Toast.makeText(getBaseContext(), "Radio
spenta!", Toast.LENGTH_SHORT).show();
                          pronuncia("La radio è spenta! S.M.S. non
inviato");
                          break;
                                } // fine dello switch
                            } // fine onReceive
                }, new IntentFilter(SENT));
            } // fine else (caso in cui il gestore SMS non è impostato
              // come predefinito)
    } // fine della funzione inviaSMS


    // trova il nome di un contatto a partire dal suo numero di telefono
    private String trovaContattoDaNumero(String numero) {
            // definisco le colonne che volglio che mi vengano
```

```
            // restituite dalla query
            String[] campiContatto = new String[] {
                        PhoneLookup.DISPLAY_NAME,
                        PhoneLookup.NUMBER};
            // codifica il numero di telefono e crea il filtro URI
            Uri uriContatto =
Uri.withAppendedPath(PhoneLookup.CONTENT_FILTER_URI,
Uri.encode(numero));
            // esegue la query
            Cursor c = getContentResolver().query(uriContatto,
campiContatto, null, null, null);
            // se la query restituisce più risultati, restituisce
            // comunque solo il primo
            if (c.moveToFirst()) {
                  String nome =
c.getString(c.getColumnIndex(PhoneLookup.DISPLAY_NAME));
                  return nome;
            }
            // se non trova nessun contatto, restituisce il numero di
            // telefono
            return numero;
      }


    // legge l'ultimo SMS ricevuto
    private void leggiSms (int stato) {
      if (nomeMittenteSms != null && testoSms != null && primoAvvio ==
false ) {
            pronuncia("S.M.S. ricevuto da: " + nomeMittenteSms + ".
Testo del messaggio. " + testoSms);
      } else {
            Toast.makeText(getBaseContext(), "Nessun messaggio da
leggere", Toast.LENGTH_SHORT).show();
            pronuncia("Nessun messaggio da leggere.");
      } // fine if-else
    } // fine leggiSms


    // pronuncia una stringa
    private void pronuncia(String stringa) {
      timbro = impostazioni.getFloat("timbro", 0.1F);
            velPronuncia = impostazioni.getFloat("velPronuncia", 1.0F);
            tts.setPitch(timbro);
            tts.setSpeechRate(velPronuncia);
            tts.speak(stringa, TextToSpeech.QUEUE_FLUSH, null);
    }

}
```

3. Codice sorgente dell'activity Impostazioni

```
package it.alvin.SMSAloud;

import android.app.Activity;
import android.content.Context;
import android.content.SharedPreferences;
import android.media.AudioManager;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.TypedValue;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.SeekBar;
import android.widget.Toast;
import android.widget.TextView;

public class Impostazioni extends Activity {

      private CheckBox checkBox;
      private TextView testoVolume, testoTimbro, testoVelocita,
testoConfermaSms, notaConfermaSms;
      private SeekBar barraVolume, barraTimbro, barraVelocita;

      /** dati per le impostazioni */
      private SharedPreferences impostazioni;
      private AudioManager audio;
      private int volume;
      private float timbro, velocita;
      private boolean usaGestoreSms;

      //---alla creazione dell'attività---
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
        setContentView(R.layout.impostazioni);

            //---ottiene le dimensioni dello schermo e ad esse adatta la
            //---dimensione del testo---
      DisplayMetrics metrics = new DisplayMetrics();
      getWindowManager().getDefaultDisplay().getMetrics(metrics);
      int larghezzaSchermo = metrics.widthPixels;
      int dimensioniTesto = (int) (Math.floor(larghezzaSchermo/14));
      int dimensioniNota = (int) (Math.floor(larghezzaSchermo/20));

      //---carica i dati salvati---
        impostazioni = getSharedPreferences("impostazioni.dat",
MODE_PRIVATE);
        timbro = impostazioni.getFloat("timbro", 0.1F);
            velocita = impostazioni.getFloat("velPronuncia", 1.0F);
            usaGestoreSms = impostazioni.getBoolean("usaGestoreSms",
true);

          /** GESTIONE DEL VOLUME */
```

```
            //---ottiene il valore del volume del flusso media dal
            //---sistema---
            audio = (AudioManager)
getSystemService(Context.AUDIO_SERVICE);
            volume = audio.getStreamVolume(AudioManager.STREAM_MUSIC);
        //---disegna l'interfaccia di controllo del volume---
            testoVolume = (TextView) findViewById(R.id.testo_volume);
            testoVolume.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTesto);
        barraVolume = (SeekBar) findViewById(R.id.barra_volume);
        testoVolume.setText(" Volume: " + volume);
        barraVolume.setProgress(volume);
        barraVolume.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            //---i movimenti del cursore sulla barra modificano il
            //---valore della variabile volume---
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                  volume = progress;
                  testoVolume.setText(" Volume: " + volume);
            }
                  @Override
                  public void onStartTrackingTouch(SeekBar arg0) {}
                  @Override
                  public void onStopTrackingTouch(SeekBar seekBar) {}
        });

        /** GESTIONE DEL TIMBRO */
        //---disegna l'interfaccia di controllo del timbro---
        testoTimbro = (TextView) findViewById(R.id.testo_timbro);
        testoTimbro.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTesto);
        barraTimbro = (SeekBar) findViewById(R.id.barra_timbro);
        testoTimbro.setText("\n Timbro: " + timbro);
        int temp = (int) (timbro*10)-1;
        barraTimbro.setProgress(temp);
        barraTimbro.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            //---i movimenti del cursore sulla barra modificano il
            //---valore della variabile timbro---
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                  timbro = (float) (progress+1)/10 ;
                  testoTimbro.setText("\n Timbro: " + timbro);
            }
                  @Override
                  public void onStartTrackingTouch(SeekBar arg0) {}
                  @Override
                  public void onStopTrackingTouch(SeekBar seekBar) {}
        });
```

```
        /** GESTIONE DELLA VELOCITA' DI PRONUNCIA */
        //---disegna l'interfaccia di controllo della velocità di
        //---pronuncia---
        testoVelocita = (TextView) findViewById(R.id.testo_velocita);
        testoVelocita.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTesto);
        barraVelocita = (SeekBar) findViewById(R.id.barra_velocita);
        testoVelocita.setText("\n Velocità pronuncia: " + velocita);
        int temp2 = (int) (velocita*10)-5;
        barraVelocita.setProgress(temp2);
        barraVelocita.setOnSeekBarChangeListener(new
SeekBar.OnSeekBarChangeListener() {
            @Override
            //---i movimenti del cursore sulla barra modificano il
            //---valore della variabile velocita---
            public void onProgressChanged(SeekBar seekBar, int progress,
boolean fromUser) {
                  velocita = (float) (progress+5)/10 ;
                  testoVelocita.setText("\n Velocità pronuncia: " +
velocita);
            }
                  @Override
                  public void onStartTrackingTouch(SeekBar arg0) {}
                  @Override
                  public void onStopTrackingTouch(SeekBar seekBar) {}
        });

        /** CHECKBOX "conferma prima di inviare SMS" */
        testoConfermaSms = (TextView)
findViewById(R.id.testo_conferma_sms);
        testoConfermaSms.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTesto);
        checkBox = (CheckBox) findViewById(R.id.conferma_sms);
        //---prepara il checkbox con le impostazioni salvate---
        if (usaGestoreSms) {
            checkBox.setChecked(true);
        } else {
            checkBox.setChecked(false);
        }
        //---se viene modificato, modifica il valore di "usaGestoreSms"
        checkBox.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                  //---se crocettato, imposta il valore di
"usaGestoreSms" su TRUE---
                  if (checkBox.isChecked()) {
                        usaGestoreSms = true;
                  //---altrimenti lo imposta su FLASE
                  } else {
                        usaGestoreSms = false;
                  }
            }
        });
```

```
        notaConfermaSms = (TextView)
findViewById(R.id.nota_conferma_sms);
        notaConfermaSms.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniNota);


        /** TASTO "Salva" */
            Button btnSalva = (Button) findViewById(R.id.btnSalva);
            //---azione associata alla pressione del tasto "Salva"---
            btnSalva.setOnClickListener(new View.OnClickListener() {
                  public void onClick(View v) {
                              //---salva le impostazioni---
                              SharedPreferences.Editor
modificaImpostazioni = impostazioni.edit();
                              modificaImpostazioni.putFloat("timbro",
timbro);

      modificaImpostazioni.putFloat("velPronuncia", velocita);

      modificaImpostazioni.putBoolean("usaGestoreSms", usaGestoreSms);
                              modificaImpostazioni.commit();
                              //---imposta il volume---

      audio.setStreamVolume(AudioManager.STREAM_MUSIC,
Integer.valueOf(volume), 0);
                              Toast.makeText(getBaseContext(),
"Impostazioni salvate!", Toast.LENGTH_SHORT).show();
                              finish();
                  } // fine onClick
            }); // fine setOnClickListener

    } // fine OnCreate


}
```

4. Codice sorgente dell'activity Info

```
package it.alvin.SMSAloud;

import android.app.Activity;
import android.content.pm.PackageManager.NameNotFoundException;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.util.TypedValue;
import android.widget.TextView;

public class Info extends Activity {

      //---invocato quando l'attività viene creata---
          @Override
```

```
        public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.info);
          //---ottiene le dimensioni dello schermo e ad esse adatta la
          //---dimensione del testo---
          DisplayMetrics metrics = new DisplayMetrics();
          getWindowManager().getDefaultDisplay().getMetrics(metrics);
          int larghezzaSchermo = metrics.widthPixels;
          int dimensioniTestoGrande = (int)
(Math.floor(larghezzaSchermo/12));
          int dimensioniTestoPiccolo = (int)
(Math.floor(larghezzaSchermo/16));
          //---disegna l'interfaccia
          String versione = "Non disponibile";
          try {
                versione =
getPackageManager().getPackageInfo(getPackageName(), 0).versionName;
          } catch (NameNotFoundException e) {}
          TextView ver = (TextView) findViewById(R.id.ver);
          ver.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTestoPiccolo);
          ver.setText("Versione: " + versione +"\n");
          TextView app = (TextView) findViewById(R.id.app);
          app.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTestoPiccolo);
          TextView nome = (TextView) findViewById(R.id.nome);
          nome.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTestoGrande);
          TextView matr = (TextView) findViewById(R.id.matr);
          matr.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTestoPiccolo);
          TextView nrMatr = (TextView) findViewById(R.id.nrMatr);
          nrMatr.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTestoGrande);
          TextView email = (TextView) findViewById(R.id.email);
          email.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTestoPiccolo);
          TextView indEmail = (TextView) findViewById(R.id.indEmail);
          indEmail.setTextSize(TypedValue.COMPLEX_UNIT_PX ,
dimensioniTestoGrande);

        } // fine OnCreate
    }
```