ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN
SCIENZE DELL'INFORMAZIONE

UNIRADIO CESENA: PROGETTAZIONE E IMPLEMENTAZIONE DI UNA MOBILE APP PER WEB RADIO

Relazione finale in
ALGORITMI E STRUTTURE DATI

Relatore
Prof. Luciano Margara

Tesi presentata da
Fabio Politi

Sessione II

Anno accademico 2013-2014

Indice

Introduzione

Oggi, con l'avvento delle nuove tecnologie e dell'utilizzo degli Smartphone, molte realtà vedono spesso accompagnare i propri progetti da l'utilizzo di App Mobile. Questo per permettere all'utente di poter accedere ai servizi, facilmente ed in mobilità, su multi-piattaforma. E' il caso di *Uniradio Cesena*, applicazione mobile della Web Radio degli studenti di Cesena, disponibile sia per Android che per iOS.
Il rapporto che si è venuto a creare tra gli sviluppatori e la realtà della radio ha permesso, e permetterà ulteriormente, di creare un prodotto di qualità sempre più completo e intuitivo per l'utente.

Obiettivo di questa tesi è quello di illustrare l'evoluzione di un'applicazione mobile per la fruizione di contenuti, streaming e non, di una Web Radio. In particolare vedremo lo sviluppo su piattaforma Android a scopo didattico, nonostante sia presente anche una versione analoga per iOS.
E' stata fondamentale la collaborazione tra sviluppatore ed utente finale poiché ha permesso, a partire da un Know-How di strumenti informatici, di identificare in maniera precisa una suddivisione interna all'App in sezioni.

Nel primo capitolo daremo una rapida descrizione dei sistema Android e del perché si è scelto quest'ultimo per implementare *Uniradio Cesena.* Nel secondo, dopo una breve descrizione progetto, parleremo del percorso di analisi dei requisiti e dei casi d'uso derivati, che hanno portato all'attuale struttura dell'App. Nel terzo parleremo del primo livello di contatto con l'utente, ovvero il Front-End; analizzeremo, sezione per sezione, il lavoro di *design di interazione* effettuato su XML e codice In-Line e la *gestione delle risoluzioni* sui molteplici dispositivi, al fine di garantire l'utilizzo al maggior numero di utenti. Nel quarto, vedremo come è stato strutturato il Back-End dell'applicazione, illustrando per le sezioni da me curate gli strumenti utilizzati, con

particolare attenzione a tecnologie quali *JSON* e *API di Facebook*. Infine, nel quinto capitolo trarremo le conclusioni e parleremo degli sviluppi futuri.

1 Le Applicazioni Mobile

1.1 Mondo mobile

Lo sviluppo di applicazioni per dispositivi mobili, ha come obiettivo quello di fornire all'utente finale una collezione di strumenti facili da utilizzare e da poter sempre portare con se. Ci basti pensare a dispositivi come *Smartphone* e *Tablet*, oggi molto diffusi e facenti parte della vita quotidiana. Questo fondamentale aspetto di versatilità e portabilità ci ha permesso di non dover utilizzare più così frequentemente PC fissi e Laptop, poiché ingombranti e scomodi in caso spostamenti.
Quindi tra i vantaggi offerti da questi piccoli dispositivi abbiamo:

- Riduzione delle dimensioni: ne consegue la possibilità di utilizzare il dispositivo ovunque e comodamente.
- Touch Screen: il ridimensionamento dei dispositivi ha portato a disfarsi di periferiche come mouse e tastiera, avantaggio dell'utilizzo di schermi tattili, con i quali è possibile interagire tramite tocco. Questo dà la sensazione dall'utente di controllare in prima persona quello che sta facendo.
- Connettività: poter portare con se un dispositivo ha generato la necessità di aver facilmente accesso ad Internet. Questo è stato possibile grazie alle tecnologie delle infrastrutture di telecomunicazione, come Wi-Fi e reti di operatori mobili (3G, LTE, GPRS, EDGE).
- Versatilità: oggi probabilmente la funzione di telefonia è quella meno utilizzata su gli attuali dispositivi. Questo perché quello che prima si faceva con più dispositivi, oggi lo facciamo solo con lo Smartphone. Ci basti pensare al servizio *fotocamera*, sempre più evoluto; partire da uno *Sharp J-SH04* (novembre 2000) con una sensore ottico da 0.1 MP, per arrivare ad un *Samsung Galaxy S5*

(febbraio 2014) che è dotato di un sensore ottico da 16.0 MP. A corredo di tale tecnologie ci sono software, o meglio definite App, che permettono di modificare anche gli scatti effettuati in maniera quasi del tutto professionale.

Ma anche altre tecnologie fanno dello Smartphone un vero e proprio dispositivo *ALL IN ONE*: *GPS* per il tracciamento della posizione e per la possibilità di utilizzare tali risorse in applicazioni di navigazione. *Accelerometro* (sensore) in grado di rilevare l'accelerazione e quindi lo spostamento. *Bluetooth* come protocollo di scambio dati Smartphone <> PC o Smartphone <> Smartphone, oppure come protocollo di comunicazione per associare allo Smartphone periferiche esterne, wireless appunto, come auricolari, tastiere etc.

Ma ci sono anche *microfoni ambientali*, *sensori di luminosità*, moduli *NFC* e molto altro. Insomma gli attuali dispositivi sono davvero sempre più completi.

Ma possiede anche dei contro:

- Diversificazione: sul mercato sono presenti diverse tipologie di terminali, principalmente classificati per caratteristiche. Ci sono terminali di fascia medio-bassa che hanno caratteristiche hardware contenute, e ci sono terminali definiti *Top di gamma*, che invece hanno sufficienti caratteristiche hardware per soddisfare appieno i requisiti minimi delle applicazioni, in modo da garantirne fluidità durante l'utilizzo.
- Autonomia: altro svantaggio è quello causato dalla batteria. Questo perché per mantenere dimensioni e peso contenuti bisogna scendere a compromessi con le dimensioni della batteria, e quindi con la capacità. Ottimizzando il software si riesce sicuramente a prolungarne l'autonomia, ma la durata resta sempre un grosso limite. Ad oggi per avere un dispositivo che garantisce un utilizzo di almeno un giorno, con un solo ciclo di carica, bisogna puntare a terminali di fascia alta e quindi costosi; questo ci ha portato a non essere più abituati a pensare alla durata della batteria in termini di giorni, ma di ore.
- Memoria: altra differenza tra terminali di fascia bassa ed alta è nel quantitativo di memoria RAM installata

Compito di un bravo sviluppatore di applicazioni è quello di limitare tali svantaggi, agendo in fase di progettazione sulla gestione delle risorse come la RAM e la gestione dei processi attivi in background ma non più utilizzati, al fine di poter supportare terminali appartenenti a fasce differenti.

1.2 Piattaforma ed ambiente di sviluppo Android

Android è un Sistema Operativo per dispositivi mobili, creato da Google Inc. basato su Kernel Linux. Esso non trova spazio solo tra i dispositivi mobili, ma vede il suo utilizzo anche in altri contesti come televisori (Android TV), automobili (Android Auto) ed orologi da polso (Android Wear) adattandone le interfacce d'utilizzo.

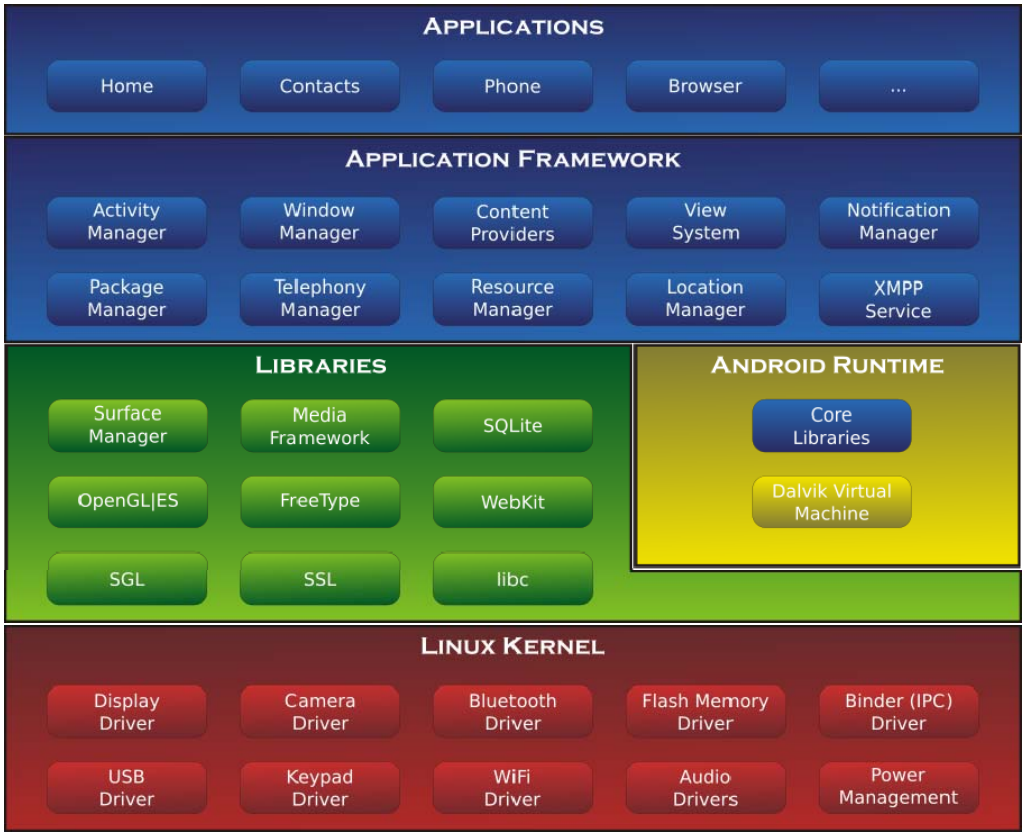
Salta immediatamente all'attenzione la peculiarità del sistema Android di essere di tipo *Open Source***,** ovvero che la licenza sotto la quale viene rilasciato, permette di modificarne e distribuirne liberamente il codice sorgente. Questo permette alle case produttrici di device ed a gruppi di sviluppatori, di adattare Android ai diversi disposativi e garantire così una vasta compatibilità. Il SO di casa Google vanta anche una vastissima community di sviluppatori, che mettono in gioco le proprie competenze di progettisti e programmatori, al fine di sfruttare appieno le potenzialità dei terminali sviluppando applicazioni di varia entità, principalmente scritte in linguaggio di programmazione *JAVA*.

Questa "libertà" ha permesso di rendere Android il sistema mobile più utilizzato e diffuso.

La prima *Release* di Android (*Android 1.0*) è stata rilasciata nel 2008, e comprendeva *il Market, il Browser, il Directory Manager, Client di Posta, Wi-Fi, Fotocamera* e le prime *Google App* per Android. Tra una release e la successiva, durante tutto l'anno, vengono effettuati continui aggiornamenti per migliorare il Sistema Operativo. Questo garantisce di avere sempre il proprio terminale aggiornato fino alla release ultima supportata.

1.2.1 La panoramica delle versioni di Android:

Abbiamo detto che le applicazioni sono sviluppate prevalentemente in linguaggio di programmazione JAVA, ed è per questo che il sistema Android è costituito da una precisa architettura che ne permette la corretta esecuzione. Analizziamo di seguito come è strutturata l'architettura di Android.



Ogni sistema Android si basa su Kernel Linux, versione 2.6. Recentemente, da Android 4.0 [2011] in poi, si utilizza la versione 3.x. Nel Kernel sono contenuti di driver necessari al controllo dell'hardware del dispositivo: driver per la tastiera, per lo schermo, per il touchpad, il Wi-Fi, il Bluetooth, il controllo dell'audio etc.

Al di sopra del Kernel abbiamo le librerie principali, anche queste derivate dal modo Open Source. Tra le principali abbiamo: *OpenGL* per la grafica, *SQLite* per la gestione dei dati e *WebKit* per visualizzare le pagine web.

L'architettura prevede l'*Android Runtime*, costituita da una Virtual Machine e da un insieme di librerie ad essa relative. Questa macchina virtuale si chiama *Dalvik Virtual Machine (DVM)*, ed altro non è che una Java Virtual Machine (JVM), ma con delle piccole differenze di fondo. È noto che la JVM esegue il *bytecode* standard generato all'atto della compilazione. Per migliorare le prestazioni sui dispositivi mobile la DVM esegue, invece, sempre del bytecode, ma generato in un altro linguaggio chiamato *DEX (Dalvik EXecutable)*, studiato appositamente. Grazie a l'Android SDK (Software Development Kit), ci sembrerà di lavorare con una normale Java Virtual Machine.

Al di sopra librerie e dell'Android Runtime troviamo, l'*Application Framework*, contenente i gestori e le applicazioni di base del sistema. Ci sono gestori per le

applicazioni, per le risorse, per le telefonate e molti altri.

Infine in cima, troviamo gli applicativi utilizzati dall'utente finale, molti dei quali già presenti nel sistema base (i.e. media player e browser web). Ed è proprio qui che gli sviluppatori troveranno spazio per le loro nuove applicazioni.

1.2.2 Strumenti per Android

Al fine di implementare applicazioni utilizzabili sui sistemi Android, è necessario ricorrere ad un apposito strumento di sviluppo da installare sul proprio PC: stiamo parlando di *Android SDK (Software Development Kit).* Questo Kit di sviluppo viene rilasciato gratuitamente per tutti i sistemi operativi (Windows, Linux e OS X). Al fine dell'installazione, è requisito fondamentale la presenza di un *Java SDK* (JDK) di versione 5 o più recente; questo perché Android si programma in Java e quindi in assenza di JDK non è possibile compilare il codice.

L'IDE (*Integrated Development Environment*) ufficiale supportato per lo sviluppo di applicativi per sistemi Android è *Eclipse*.

Proprio per Eclipse è presente anche un pacchetto di strumenti, rilasciato come plug-in, chiamato *Android Development Tools (ADT)*, in grado di fornire al programmatore un ambiente integrato che contiene tutti i tool necessari per sviluppare agevolmente le applicazioni Android.

1.2.3 Le versioni di Android

Sono state rilasciate diverse versioni di Android a partire dal 2008. Queste non avevano, in principio, nomi specifici. Ma a partire dalla versione 1.5 si è deciso di attribuire nomi di dolci.

La prima versione è stata: 1.0 – Bender – API 1 – 2008.

Da sinistra verso destra troviamo la versione (*1.0*), il nome (*Bender*), l'API Level (*API*) ovvero l'intero che la indentifica univocamente, e l'anno di rilascio (*2008*).

Di seguito riassumiamo in una tabella, tutte le versioni di Android presenti ad oggi.

Versione	Nome	API Level	Anno
1.0	Bender	1	2008
1.1	Bender	2	2009
1.5	Cupcake	3	2009
1.6	Donut	4	2009
2.0	Eclair	5	2009
2.0.1	Eclair	6	2009
2.1	Eclair	7	2010
2.1	Froyo	8	2011
2.3	Gingerbread	9	2011
2.3.3	Gingerbread	10	2011
3.0	Honeycomb	11	2011
3.1	Honeycomb	12	2011
3.2	Honeycomb	13	2011
4.0	Ice Cream Sandwich	14	2011
4.0.4	Ice Cream Sandwich	15	2012
4.1	Jelly Bean	16	2012
4.2	Jelly Bean	17	2012
4.3	Jelly Bean	18	2013
4.4	KitKat	19	2013
4.4.4	KitKat	19	2014

Al momento del rilascio di una nuova versione, Google provvede anche a rilasciare il relativo SDK, in modo che gli sviluppatori possano implementare applicazioni compatibili.

1.2.4 Componenti Android

Quando si decide di sviluppare un'applicazione Android, si viene a contatto con 4 componenti fondamentali, necessari per procedere nell'implementazione.
Questi sono ***Activity***, ***Service***, ***Content Provider*** *e* ***Broadcast Receiver***. Analizziamoli più da vicino:

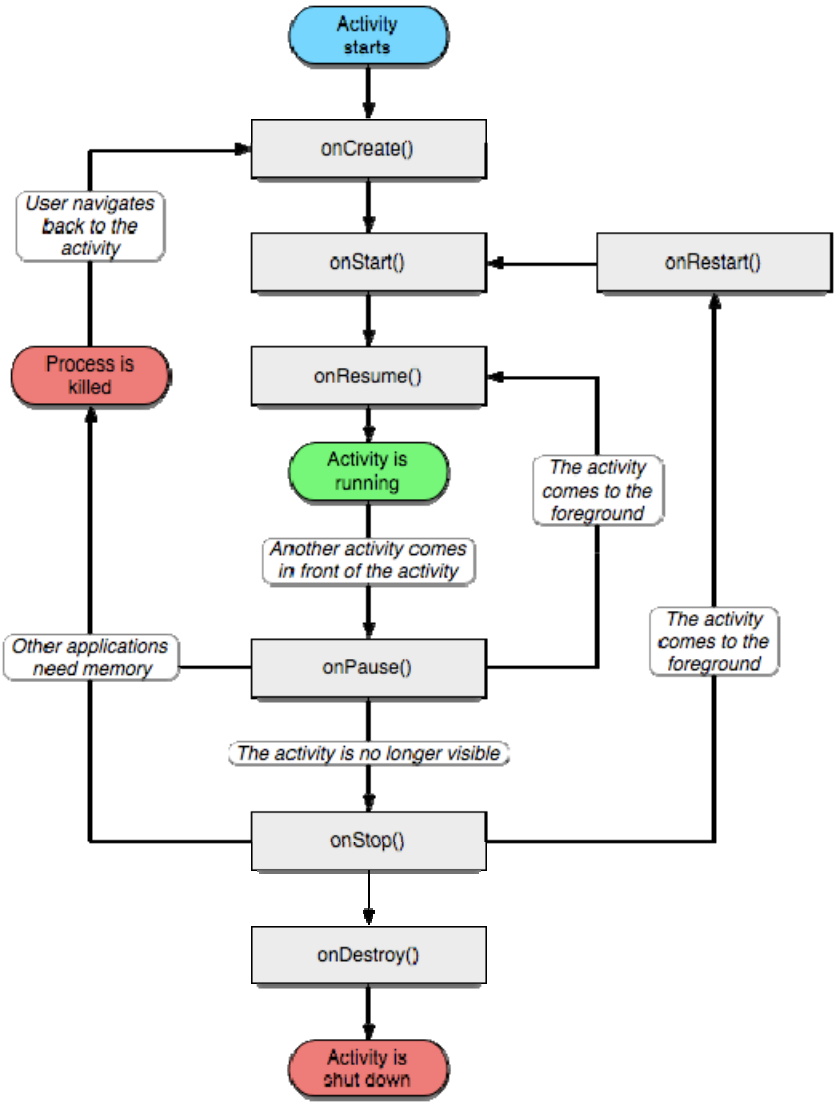
- *Activity*: un'attività, stando alla documentazione ufficiale, è *"una singola e precisa cosa che l'utente può fare"*. Questo significa che l'utente per fare qualcosa deve interagire con il terminale. Nel caso di uno smartphone, questo avviene per mezzo dei dispositivi di input, come tastiera e touchscreen. Questo però non basta per far sapere all'utente cosa può fare e come può farlo; e c'è bisogno anche di un canale aggiuntivo per presentare all'utente quanto ha fatto. Questo canale normalmente è il display del terminale. Quello che accade è che il software disegna tutti gli oggetti con cui l'utente può interagire, per poi produrre un risultato sempre mostrato sullo schermo. Quindi in definitiva possiamo dire che le attività sono componenti di un applicativo Android che fanno uso del display e che interagiscono con l'utente.

  Dal lato della programmazione un'Activity altro non è che una classe che estende *android.app.Activity* e quindi lo sviluppatore si serve tutti i metodi ereditati da *Activity*, per acquisire gli input dell'utente e quindi per interagire con il sistema

  Ricordando che tra i vantaggi dei dispositivi mobili avevamo parlato delle loro dimensioni contenute, quindi anche quelle del display, e tenendo presente che i terminali hanno modeste risorse di calcolo, per questi motivi le Activity di Android hanno carattere di esclusività. Questo vuol dire che possiamo si eseguire più attività in contemporanea, ma la visualizzazione sul display è di una alla volta. Quella che occupa il display (*Foregroud*) è in esecuzione ed interagisce direttamente con l'utente; le altre, invece, sono in stato di ibernazione e tenute in *Background*, in modo da ridurre al minimo il consumo delle risorse di calcolo e quindi per aumentare l'autonomia del dispositivo. È possibile ripristinare in qualsiasi momento un'attività presente in background, riportandola in primo piano, facendo andare in ibernazione quella

presente inizialmente in foreground.

Vediamo meglio come evolve il ciclo di vita di un'attività per mezzo di una flow chart.



La figura precedente mostra la sequenza di chiamate ai metodi di Activity che sono eseguite durante i passaggi di stato dell'attività. Dettagliamo tali metodi:

- *protected void onCreate()* : richiamiamo questo metodo all'atto della creazione dell'attività.

  È possibile passare come argomento *savedInstanceState* che serve per riportare un eventuale stato dell'attività salvato in precendenza da un'altra istanza terminata. L'argomento è *null* nel caso in cui l'attività non abbia uno stato salvato.

- *protected void onRestart()* : richiamiamo questo metodo per segnalare che l'attività sta per essere riavviata in conseguenza all'arresto.
- *protected void onStart()*: richiamiamo questo metodo per segnalare che l'attività sta per essere avviata e quindi mostrata in foreground.
- *protected void onResume():* richiamiamo questo metodo per segnalare che l'attività sta per iniziare l'interazione con l'utente.
- *protected void onPause():* richiamiamo questo metodo per segnalare che l'attività non sta più interagendo con l'utente.
- *protected void onStop():* richiamiamo questo metodo per segnalare che l'attività non è più in foreground
- *protected void onDestroy():* richiamiamo questo metodo per segnalare che l'applicazione sta per essere teminata.

È buona norma di programmazione, inserire come prima riga di codice di ciascuno di questi metodi, l'implementazione di base del metodo chiamato. Vediamo un esempio:

```
protected void onStart (){
          super.onStart();
//…}
```

- *Service*: un servizio a differenza di un'attività, svolge un ruolo lungo e continuativo, che normalmente avviene in background senza che l'utente ne entri in contatto. Il ruolo dei Service è quello di preparare in background i dati necessari da mostrare all'utente una volta fatta una qual si voglia richiesta; questo garantisce reattività responsiva a favore dell'utilizzatore dell'applicazione all'atto della visualizzazione.
- *Content Provider*: è una parte di un applicativo Android che ha il compito di rendere disponibili dei dati. Ogni applicazione può specificare uno o più tipi di dato e renderli disponibili esponendo uno o più Content Provider. Viceversa una qualsiasi App, può aver bisogno di accedere ad un preciso tipo di dato, e quindi il sistema la metterà in contatto con il relativo Content Provider precedentemente installato.

Ogni contenuto esposto tramite Content Provider è identificato da un URI, ovvero un indirizzo univoco. Ad esempio, per accedere alla lista dei contatti in rubrica, scriviamo:

```
Content://com.android.contacts/contacts
```

Una volta noto l'URI di una tipologia, l'azione di interazione con il provider che lo eroga è quasi del tutto simile ad una classica interrogazione ad un DB.

Si recupera l'istanza dell'oggetto *android.content.ContentResolver*; se ci troviamo all'interno di una Activity (o comunque ci è noto l'oggetto *android.app.Context*), possiamo usare il metodo *getContentResolver()*.

```
ContentResolver cr = getContentResolver();
```

Ottenuto l'oggetto di tipo ContentResolver, possiamo utilizzare i metodi che della classe ContentResolver. Questi sono:

- query()
- insert()
- update()
- delete()
- getType()
- onCreate()

- *Broadcast Receiver:* sono tra i componenti più importanti del sistema Android. Il loro ruolo è quello di restare in ascolto per messaggi di tipo *Intent Broadcast.* Questi ultimi sono dei particolari tipi di *intent* (vedremo successivamente cosa sono), spediti attraverso il metodo *sendBroadcast().*

  L'unico metodo implementato dal Broadcast Receiver è il metodo onRecieve(), chiamato dal sistema all'atto della ricezione di un Intent Broadcast.

Per mettere in comunicazione queste quattro fondamentali componenti, si fa ricorso ad un'altra componente fondamentale dei sistemi Android: *gli Intent.*

Come accennato in precedenza, con il termine intent identifichiamo una forma di comunicazione gestita dal sistema. Gl'intent permettono alle componenti di un'applicazione di comunicare tra loro.

Normalmente con un Intent si richiede di:

- *Avviare un'Activity;*
- *Avviare un Service;*
- *Inviare un messaggio in Broadcast che può essere ricevuto da ogni applicazione;*

Aspetto molto importante degl'Intent è che nel momento in cui questi recapitano un messaggio, hanno a disposizione un "contenitore" dove sono presenti i dati che posso essere letti dal destinatario. Questi valori condivisi mediante Intent, sono chiamati *Extras.*

Esistono due tipi di Intent: quello esplicito e quello implicito. Nel primo caso l'applicazione può definire il componente di destinazione direttamente nell'intent; nel secondo l'applicazione chiede al sistema Android di valutare i componenti registrati sulla base dei dati dell'Intent.

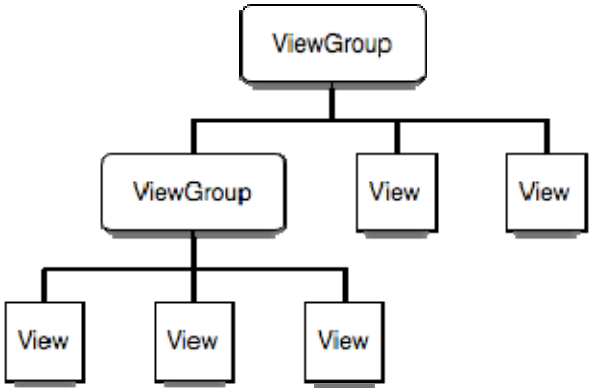
View

Altro componente molto importante nei sistemi Android sono le *View.*

Android utilizza *View* e *ViewGroup* per organizzare tutto quello che appare sullo schermo del nostro dispositivo.

Quindi bottoni, campi di testo, icone e tutto ciò che appartiene ad un'interfaccia grafica sono *View.* .

Le ViewGroup possiamo definirle, invece, come contenitori che uniscono più oggetti di tipo *View*. Anche i *ViewGroup* sono oggetti di tipo *View* e possono essere contenuti da altri *ViewGroup*.



I componenti *View* estendono la classe base *android.view.View.* Il sistema Android mette già a disposizione molti componenti di questo tipo. La maggior parte la troviamo nel pacchetto *android.widget.* Ma l'utente ha sempre la possibilità di poter estendere la

classe *View* e realizzare i propri componenti.

I *Widget* sono quei componenti di base utili al fine di interagire con l'utente, come bottoni, check box, liste, campi di testo, etc.

La classe *android.view.ViewGroup* è un'estensione di *View*. Esistono particolari tipi di ViewGroup denominati Adapter e si occupano di collegare ad un insieme di View, dei dati. Questi dati possono essere array, allora otteniamo Array Adapter, oppure cursori provenienti da DB, e quindi otteniamo Cursor Adapter.
Gli Adapter hanno come ruolo, infine, quello di creare un numero di View pari a quello dei dati recuperati.

Layout

Con il termine *Layout* identifichiamo tutti quei *ViewGroup* che sono utilizzati per posizionare i *Widget* sul display. Alcuni di questi layout sono già messi a disposizione da Android.

2 Uniradio Cesena

2.1 Panoramica

Uniradio Cesena è un'applicazione per Smartphone e Tablet rivolta ad un pubblico eterogeneo che ha come obiettivo quello di portare sempre con se i contenuti streaming e non della Web Radio degli studenti universitari di Cesena. Per mezzo di questa applicazione l'utente può consultare il palinsesto della radio, gli eventi ai quali la radio partecipa, le schede di tutti i programmi che vanno in onda con le relative informazioni, accedere ai servizi SPIN per richiedere un brano a piacimento oppure lasciare un messaggio vocale alla regia della radio. È possibile, inoltre essere rimandati al sito di SoundCloud dove sono presenti i Podcast delle puntate dei programmi andate in onda.

Ma la funzione principale dell'applicazione è quella di Streaming audio, contornata dalle informazioni relative a quanto si sta ascoltando e dalla possibilità di condividere tali informazioni tramite Social Network.

Per funzionare l'applicazione si appoggia ad una serie di Web Service che per mettono di recuperare tutte le informazioni necessarie per le differenti sezioni. Questi servizi Web sono stati implementati da terze persone sempre partecipati al progetto della radio, poiché l'infrastruttura è molto estesa e comprende anche una regia automatica (MB Studio), un server per lo streaming audio (Shoutcast) ed il server su cui è hostato il sito web della radio.

Uniradio Cesena è stata sviluppata per piattaforma Android e pubblicata sul Play Store. È presente anche l'analoga versione per iOS presente sull'App Store di Apple. Entrambe le versioni sono scaricabili ed utilizzabili gratuitamente.

2.2 Perché un'App per una Web Radio

La scelta di creare un'applicazione mobile per la fruizione dei contenuti radiofonici è nata dalla necessita di raggiungere sempre più utenti. Sapevamo che in prevalenza questi ultimi erano ragazzi e ragazze di età compresa tra i 18 e 30 anni, che sempre di più utilizzano i dispositivi Mobile per accedere a contenuti sul web.

L'incontro tra la necessità del progetto di aumentare sempre di più i propri ascoltatori e la versatilità dei dispositivi odierni, ha condotto a prendere in considerazione, nella campagna Marketing, lo sviluppo di un'applicazione dedicata allo scopo.

La versatilità di un'applicazione si traduce in 3 caratteristiche fondamentali:

- *Comodità*: possiamo portare sempre con noi i nostri terminali e quindi le nostre applicazioni ed accedere ai contenuti dove e quando vogliamo.
- *Compattezza*: in una sola applicazione, divisa per sezioni, possiamo collezionare tutte le informazioni che ci interessano, come ad esempio lo *Streaming*, gli *eventi*, i *programmi* e così via.
- *Condivisione*: fondamentale è l'aspetto Social, che ci permette di condividere con gli amici quanto stiamo facendo con la nostra applicazione.

Di seguito faremo un'analisi dei requisiti fondamentali che ci ha portato a progettare e strutturare l'applicazione Uniradio Cesena rispetto alle esigenze prefisse.

2.3 Analisi dei requisiti

Dal confronto tra noi sviluppatori e la direzione della radio, è stata stilata una lista di funzioni che hanno portato ad una divisione dell'applicazione in sezioni:

- Streaming
- Palinsesto
- Chi Siamo
- Eventi
- Programmi
- SPIN

2.3.1 Streaming

La principale funzionalità dell'applicazione è proprio quella di *Streaming*. Per questo motivo deve avere un'interfaccia molto semplice che permetta all'utente di avere subito le informazioni relative ai contenuti Streaming (*artista*, *titolo*, *immagine album*), e di sapere quali sono le azioni che può compiere in questa sezione (*play*, *pause*, *condividi*).

2.3.2 Palinsesto

Come già il nome della sezione suggerisce, questa funzione permette di visualizzare in maniera ordinata il *Palinsesto* della radio. Le informazioni da gestire sono *nome del programma* o *della rotazione*, l'*orario* di messa in onda e quali sono i *conduttori*. Per semplificare la visualizzazione e garantire ordine all'interfaccia, il palinsesto viene diviso in giorni della settimana.

2.3.3 Chi Siamo

Nella sezione *Chi siamo* c'era la necessità di dare una breve descrizione della radio così da permettere all'ascoltatore di sapere con cosa si stesse "interfacciando"; a supporto abbiamo dei link come *sito web*, *collabora*, *mail* ed *indirizzo* utili all'utente finale per poter entrare a contatto con la radio.

2.3.4 Eventi

La necessità della radio di promuovere gli eventi da essa direttamente organizzati o altri eventi importati ai quali essa prendesse parte come collaboratrice, ha portato alla progettazione di una sezione eventi. L'idea è quella di avere principalmente un listato di eventi composto da *data*, *titolo* e *luogo*. Secondariamente cliccando su una delle righe è possibile espandere i dettagli in una nuova finestra aggiungendo dettagli come *l'immagine* ed una *descrizione* minimale.

2.3.5 Programmi

Avendo un palinsesto che indica lo slot di messa in onda dei diversi programmi, è stato automatico aggiungere una sezione dedicata a questi. Una pagina iniziale visualizza la griglia di tutti i programmi con *immagine*, *nome*, *giorno* e range *orario* di messa in onda. Una seconda pagina, per ogni programma, visualizza altri dettagli quali *descrizione*, *link* alla *pagina Facebook* e alla *pagina SoundCloud* coi Podcast delle puntate andate in onda.

2.3.6 SPIN

L'idea alla base di SPIN era quella di ricreare una piattaforma multimediale che permettesse all'utente di poter interagire con la radio. SPIN è diviso in due sezioni: nella prima, è possibile richiedere una canzone alla regia automatica (MB Studio), che la manderà nella fascia oraria dedicata a SPIN; nella seconda sezione è possibile registrare un breve messaggio vocale da inviare alla nostra Redazione che, una volta esaminato il messaggio, farà in modo di mandarla On Air. Per comodità il servizio di richiesta del brano è attivo solo in determinate fasce orarie in modo da permettere il regolare flusso della radio composto da rotazioni e programmi.

2.4 Casi d'uso

Dei tipici scenari di utilizzo dell'applicazione sono:

- Un utente vuole ascoltare cosa sta andando in onda su Uniradio Cesena. Lanciando l'applicazione dall'apposita icona, si ritrova immediatamente nella sezione Streaming. Qui potrà visualizzare il brano (*titolo, autore* e *copertina*) oppure lo spot in onda, e tramite i pulsati *Play* e *Spot* potrà gestire lo streaming.
- L'ascoltatore vuole condividere il contenuto attualmente in onda con conoscenti. È presente il pulsante di condivisione, tramite il quale è possibile pubblicare sulla propria bacheca di Facebook il brano in onda.
- L'utente vuole spere quali sono gli eventi ai quali la radio prende parte. Nella sezione eventi verrà visualizzata la lista di eventi qualora presenti.
- L'utente vuole avere una descrizione più dettagliata di un programma che sta attualmente andando in onda. Nella sezione programmi è possibile consultare la pagina del programma desiderato.

- L’utente vuole richiedere un brano alla regia automatica della radio, oppure inviare un breve messaggio vocale alla redazione.

3 Front-End (Interaction Design)

Un'adeguata progettazione dell'interfaccia, permette all'utente di utilizzare con più facilità ed intuizione l'applicazione. All'avvio, deve essere ben chiaro quali sono le cose che si possono fare. Inizialmente quella che verrà visualizzata è la funzione principale, ovvero quella dello streaming corrente.

Basterà semplicemente scorrere da sinistra verso destra sullo schermo touch con un movimento detto di *Swipe*, per accedere al menu dell'applicazione dove sono listate tutte le sezioni di cui l'App si compone. Prima di analizzarle una alla volta, faremo una panoramica su gli strumenti utilizzati che hanno reso possibile la loro progettazione grafica.

3.1 Strumenti grafici

3.1.1 XML

Come abbiamo accennato nel Capitolo 1, definiamo *Layout* la struttura grafica di un'Activity. È molto importante progettare un layout ben strutturato poiché è ciò che semplifica la navigazione all'utente; per questo si è deciso di concentrarsi molto su questo aspetto sia provando in prima persona quanto creato, sia prendendo dei gruppi di persone, di eterogenea provenienza, in qualità di tester.

L'IDE Eclipse metta a disposizione un approccio *visuale* alla creazione di GUI (*Graphical User Interface*), dando la possibilità di trascinare componenti sull'editor, per poi adattarli ad occhio. Sarà poi l'editor ad interpretare e a generare il codice necessario. Questo semplifica si allo sviluppatore il lavoro di creazione dell'interfaccia, ma in fin dei conti il codice generato non è sempre ben strutturato e spesso è difficile da gestire.

Infatti ambienti di sviluppo che sono dotati di un buon Editor Visuale sono davvero pochi.

Ecco perché viene introdotto il linguaggio a marcatori XML (*eXtensible Markup Language*), che ben si presta all'operazione di descrizione di componenti, proprio come il linguaggio HTML fa sul Web.

XML è più semplice da leggere e da scrivere, sia per lo sviluppatore sia per gl'editor visuali. Android permette di realizzare GUI con un classico approccio basato sul codice di programmazione Java, ma spesso è più semplice e comodo ricorrere ad un approccio più recente basato su XML che il sistema mette a disposizione nativamente.

3.1.2 Risoluzioni

Contrariamente allo sviluppo di interfacce per iOS, dove sono note le risoluzioni dei display, per Android questo aspetto è più complesso. Il sistema open source di Google ha la possibilità di essere installato su una vasta gamma di dispositivi compatibili. Questo non significa che dobbiamo sviluppare un'interfaccia per ogni risoluzione. Rispetto a delle statistiche siamo riusciti a risalire alla risoluzioni più diffusa tra i terminali Android: HVGA (*Half-size Video Grafic Array*) ovvero 320 x 480 pixel. Però questa nozione non ci permette di poter progettare in maniera assoluta le interfacce, poiché questa logia creerebbe grafiche sformate su terminali a risoluzioni diverse.

Per semplificare il lavoro, gli sviluppatori hanno raggruppato le dimensioni dei display secondo una combinazione <*risoluzione*, *densità pixel*>. Riportiamo di seguito la tabella di riferimento.

	Low density (120), *ldpi*	Medium density (160), *mdpi*	High density (240), *hdpi*	Extra high density (320), *xhdpi*
Small screen	**QVGA (240x320)**		480x640	
Normal screen	**WQVGA400 (240x 400)** **WQVGA (240x432)**	**HVGA (320x480)**	**WVGA (480x800)** **WVGA854 (480x854)** 600x1024	640x960
Large screen	**WVGA800 (480x800)** **WVGA854 (400x854)**	**WVGA800 (480x800)** **WVGA854 (480x854)** 600x1024		
Extra large screen	1024x600	**WXGA (1280x800)** 1024x768 1280x768	1536x1152 1920x1152 1920x1200	2048x1536 2560x1536 2560x1600

Abbiamo quindi una suddivisione delle risoluzioni in 4 categorie: *Small*, *Normal*, *Large* ed *Extra Large*. Ed abbiamo una suddivisione delle densità in *ldpi (low density)*, *mdpi (medium density)*, *hdpi (high density)* e *xhdpi (extra high density)*.

Mantenendo le proporzioni, l'attenzione si sposta sulla densità dello schermo. È qui che viene introdotto il concetto di misura indipendente dai pixel chiamata *density-independent pixel*.

> *The density-independent pixel is equivalent to one physical pixel on a 160 dpi screen, which is the baseline density assumed by the system for a "medium" density screen. At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use. The conversion of dp units to screen pixels is simple: px = dp * (dpi / 160). For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels. You should always use dp units when defining your application's UI, to ensure proper display of your UI on screens with different densities.*
>
> Cit. developer.android.com

Android ci viene in contro su questo aspetto, in particolare per la gestione di immagini. Infatti esso gestisce le 4 risoluzioni trattate in precedenza. Vediamo come su Android bisogna scalare le immagini.

La citazione precedente apre con la frase: *"Un Density-Independent Pixel (dp) è equivalente ad 1px su uno schermo da 160dpi, ovvero la densità base riferita ad uno schermo a densità media"*. Quindi se dobbiamo gestire un immagine da 100x100 pixel dobbiamo creare quattro immagini adattate alle quattro densità secondo il *rapporto 3:4:6:8*. Presa come risoluzione base la 100x100, le altre risoluzioni si derivano in questo modo:

- *ldpi*: 100 * (3/4) = 75px
- *mdpi*: 100 * (4/4) = 100px
- *hdpi*: 100 * (6/4) = 150px
- *xhdpi*: 100 * (8/4) = 200px

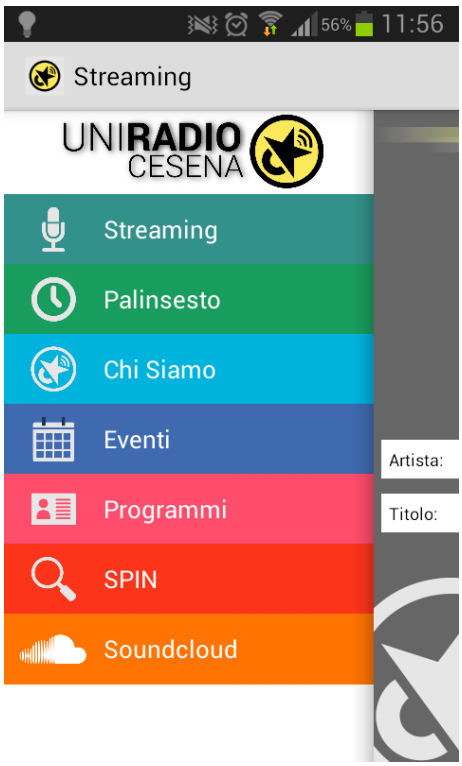
Conseguentemente Android ci dà la possibilità di creare nella cartella *res*, del nostro progetto, altre sottocartelle rinominate in maniera standard per gestire le differenti risoluzioni:

- *drawable*
- *drawable-ldpi*
- *drawable-mdpi*
- *drawable-hdpi*
- *drawable-xhdpi*

La prima conterrà tutte le risorse che non dipendono dalla risoluzione dello schermo.

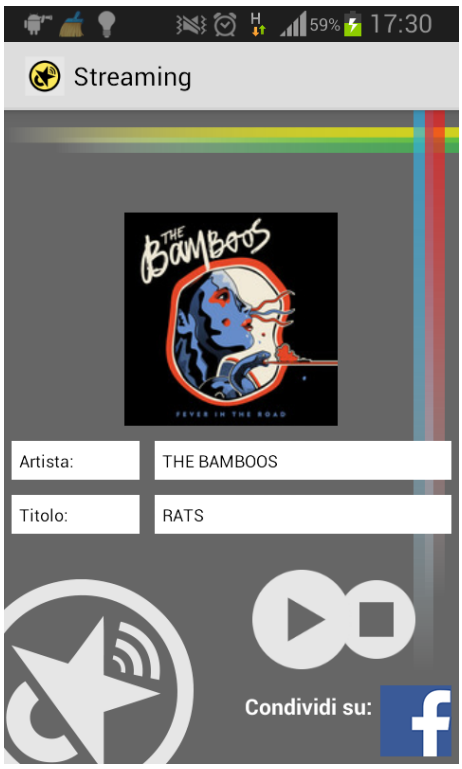
3.2 Sezioni

3.2.1 Menu



Tramite una *Gesture* di *Swipe*, oppure per mezzo della pressione del tasto opzioni, è possibile visualizzare il Menu dell'App e quindi la lista delle funzioni presenti: *Streaming*, *Palinsesto*, *Chi Siamo*, *Eventi*, *Programmi*, *SPIN*, *Soundcloud*. Di seguito diamo una panoramica sulle tecniche utilizzate per produrre un'interfaccia semplice e intuitiva, che quindi permetta, anche ad un utente non troppo esperto, di poter raggiungere la risorsa desiderata in massimo 3 tocchi. Questa è una prerogativa che ci si è posti al fine di rendere la navigazione snella; possiamo chiamare questa specifica *navigazione a tre livelli*.

3.2.2 Streaming



La schermata di Streaming è quella che per prima viene mostrata all'utente. Questo è ovvio, poiché lo streaming della risorsa audio è la funzione principale dell'applicazione presa in esame. Si compone di: un'immagine che identifica la copertina dell'album da cui la canzone è tratta; due Label, *Artista* e *Titolo* che hanno accanto le rispettive *TextView* dove verranno mostrati i dati richiesti; i pulsanti del lettore multimediale *Play* e *Stop*; infine, il tasto di sharing tramite il Social Network *Facebook*, che permette all'utente di condividere sulla propria bacheca quanto sta ascoltando.

Immagine di copertina

```
<ImageView
                android:id="@+id/imageCopertina"
                android:layout_width="150dp"
                android:layout_height="150dp"
                android:layout_margin="5dp"
                android:layout_marginTop="73dp"
                android:maxHeight="250dp"
                android:maxWidth="250dp"
                android:scaleType="fitXY"
                android:src="@drawable/icona" />
```

Informazioni

Prendiamo in esame solo la riga dedicata all'Artista, poiché per il titolo il funzionamento è analogo.

```
<LinearLayout
    android:id="@+id/layout_artista"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
```

```
    <TextView
     android:id="@+id/txtLblArtista"
     android:layout_width="@dimen/streaming_label_dim"
     android:layout_height="wrap_content"
     android:layout_margin="5dp"
     android:background="@color/white"
     android:padding="5dp"
     android:text="Artista"
     android:textColor="@color/black" />

    <TextView
     android:id="@+id/txtArtista"
     android:layout_width="0dp"
     android:layout_height="wrap_content"
     android:layout_margin="@dimen/margin"
     android:layout_weight="3"
     android:background="@color/white"
     android:padding="@dimen/padding"
     android:text="Nome artista"
      android:textColor="@color/black" />
</LinearLayout>
```

Tasti Multimediali

```
<RelativeLayout
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_centerInParent="true" >

   <ImageButton
      android:id="@+id/button_stop"
      android:layout_width="55dp"
      android:layout_height="55dp"
      android:layout_centerVertical="true"
      android:layout_marginLeft="60dp"
      android:background="@null"
      android:contentDescription="@drawable/stopbutton"
      android:scaleType="fitXY"
      android:src="@drawable/stopbutton" />
   <ImageButton
      android:id="@+id/button_play"
      android:layout_width="70dp"
      android:layout_height="70dp"
      android:layout_centerVertical="true"
      android:background="@null"
      android:contentDescription="@drawable/playbutton"
      android:scaleType="fitXY"
```

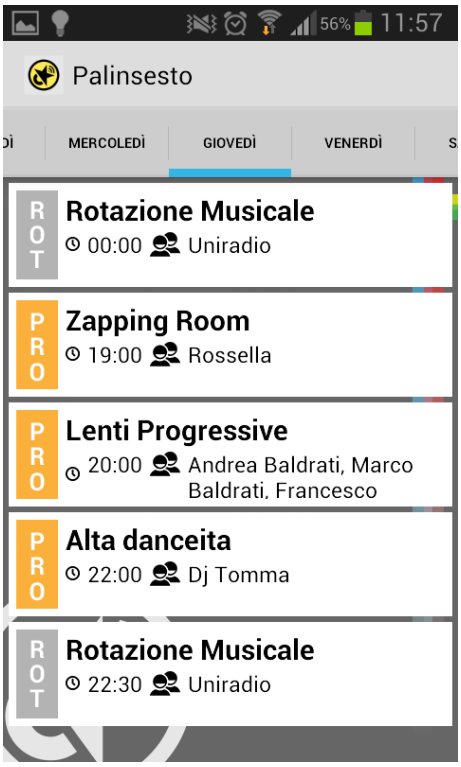
```
      android:src="@drawable/playbutton" />
</RelativeLayout>
```

Condividi su Facebook

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true" >
    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginBottom="10dp"
        android:text="Condividi su:"
        android:textAppearance="?android:attr/
                                textAppearanceMedium"
        android:textColor="@color/white"
        android:textStyle="bold" />
    <ImageButton
        android:id="@+id/imageViewFacebook"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_gravity="center_vertical"
        android:layout_margin="5dp"
        android:background="@null"
        android:src="@drawable/facebook" />
</LinearLayout>
```

Il tasto di condivisione su Facebook apre la classica finestra di condivisione messa a disposizione dal social network

3.2.3 Palinsesto



Il palinsesto è la funzione che permetta all'utente di poter sempre sapere in che giorno ed a che ora un programma o una rotazione sono trasmessi.

Esso si compone di sette Tab scorrevoli o selezionabili, una per ogni giorno della settimana. Per ogni Tab abbiamo una lista di *Item*. Queste sono composte da: un'etichetta verticale che indentifica se si tratta di una rotazione, di un programma o di una replica; il titolo del programma o della rotazione; orario di messa in onda; elenco dei conduttori. Nel caso delle rotazioni, non c'è un conduttore, ma è Uniradio ad essere identificato come conduttore.

Item palinsesto

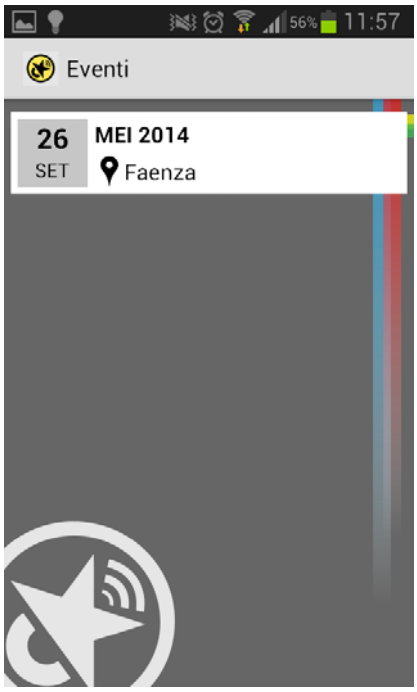
```
<LinearLayout
     android:id="@+id/linearLayout1"
     android:layout_width="match_parent"
     android:layout_height="wrap_content" >

     <ImageView
     android:id="@+id/palinsesto_item_image_left"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:src="@drawable/palinsesto_rot" />

    <LinearLayout
     android:layout_width="match_parent"
     android:layout_height="match_parent"
     android:layout_gravity="fill"
     android:layout_marginLeft="5dp"
     android:orientation="vertical" >

          <TextView
               android:id="@+id/
```

```
                palinsesto_item_titolo_trasmissione"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Titolo trasmissione"
            android:textColor="@color/black"
            android:textSize="20dp"
            android:textStyle="bold" />

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="bottom"
            android:orientation="horizontal" >

            <ImageView
                android:id="@+id/imageView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center
                                _vertical"
                android:src="@drawable/ora_programma"
            />

            <TextView
                android:id="@+id/palinsesto
                                _item_orario"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="5dp"
                android:text="Orario"
                android:textColor="@color/black"
                android:textSize="15dp" />
            <ImageView
                android:id="@+id/imageView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="5dp"
                android:src="@drawable/conduttori" />
            <TextView
                android:id="@+id/palinsesto
                                _item_conduttori"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginLeft="5dp"
                android:text="Conduttori"
                android:textColor="@color/black"
                android:textSize="15dp" />
```

```
            </LinearLayout>
        </LinearLayout>
</LinearLayout>
```

3.2.4 Eventi

La sezione eventi è molto semplice e minimale. Anche qui troviamo un listato di una o più *Item*. Per ognuna di queste le informazioni immediate che ci vengono mostrate sono il *Titolo*, la *Data* ed il *Luogo*.
Selezionando uno degli eventi presenti nella lista è possibile accedere ad un'ulteriore Activity dove ci sono i dettagli dell'evento.



Item evento

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:background="@color/grigio_chiaro"
        android:paddingLeft="10dp"
        android:paddingRight="10dp"
        android:gravity="center"
        android:orientation="vertical" >
        <TextView
           android:id="@+id/eventi_giorno"
           android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:textSize="33px"
            android:textStyle="bold"
            android:textColor="@color/black"
            android:text="TextView" />
        <TextView
            android:id="@+id/eventi_mese"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:textSize="23px"
            android:textColor="@color/black"
            android:text="TextView" />
        <TextView
            android:id="@+id/eventi_anno"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center_horizontal"
            android:textSize="23px"
            android:textColor="@color/black"
            android:text="TextView" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_marginLeft="5dp"  >
        <TextView
            android:id="@+id/eventi_titolo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:maxLines="2"
            android:textSize="26px"
            android:text="Large Text"
            android:textStyle="bold"
            android:textColor="@color/black"
            android:textAppearance="?android:attr/
                                    textAppearanceLarge"
            android:layout_gravity="center_
                                    vertical|left"/>

            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
```

```
                android:gravity="center_vertical"
                android:layout_marginTop="5dp">
               <ImageView
                android:id="@+id/eventi_logo_luogo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                 android:src="@drawable/dove_siamo"
                android:layout_gravity="center
                          _vertical|center_horizontal"/>
                <TextView
                 android:id="@+id/eventi_luogo"
                 android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                 android:textColor="@color/black"
                 android:textSize="26px"
                 android:layout_gravity="center
                          _vertical|center_horizontal"
                 android:text="TextView" />
            </LinearLayout>
    </LinearLayout>
</LinearLayout>
```

Layout dettagli

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentTop="true"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_marginTop="10dp"
    android:background="@color/white"
    android:padding="5dp" >
    <RelativeLayout
          android:layout_width="100dp"
          android:layout_height="match_parent"
          android:orientation="vertical" >
          <ImageView
                android:id="@+id/evento_immagine"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="40dp"
                android:src="@drawable/icona" />
                <LinearLayout
                     android:layout_width="match_parent"
```

```
                android:layout_height="wrap_content"
                android:layout_alignParentBottom=
                                        "true"
                android:layout_alignParentLeft=
                                        "true"
                android:layout_alignParentRight=
                                        "true"
                android:layout_below="@id/evento
                                        _immagine"
                android:orientation="vertical" >
                <TextView
                android:id="@+id/evento_data"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center
                                    _horizontal"
                android:text="10 / AGO"
                android:textColor="@color/black"
                android:textSize="24dp" />
                <TextView
                android:id="@+id/evento_giorno"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center
                                    _horizontal"
                android:background="#cccccc"
                android:padding="3dp"
                android:text="Giovedi"
                android:textColor="@color/black"
                android:textSize="20dp" />
                <TextView
                android:id="@+id/evento_annoEvento"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center
                                    _horizontal"
                android:text="2010"
                android:textColor="@color/black"
                android:textSize="24dp" />
            </LinearLayout>
    </RelativeLayout>
    <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="match_parent"
          android:layout_marginLeft="5dp"
          android:layout_weight="1"
          android:orientation="vertical" >
```

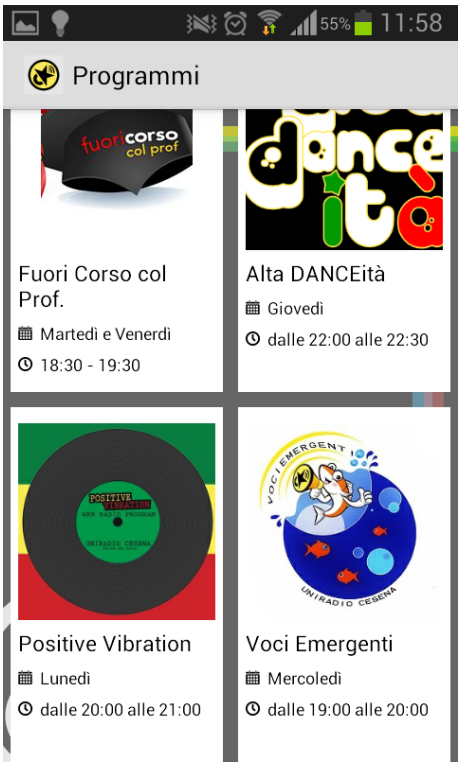
```
        <TextView
            android:id="@+id/evento_titolo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Large Text"
            android:textAppearance="?android:attr/
                                    textAppearanceLarge"
            android:textColor="@color/black" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical"
            android:paddingBottom="15dp"
            android:paddingTop="10dp" >
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="3dp"
                android:layout_weight="5" >
                <TextView
                android:id="@+id/evento_lblLuogo"
                android:layout_width="50dp"
                android:layout_height="wrap_content"
                android:layout_marginRight="3dp"
                android:background="#cccccc"
                android:gravity="center
                                _vertical|left"
                android:padding="2dp"
                android:text="Luogo"
                android:textColor="@color/black" />
                <TextView
                android:id="@+id/evento_Luogo"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_weight="4"
                android:background="#cccccc"
                android:gravity="center
                                _vertical|left"
                android:padding="1dp"
                android:text="TextView"
                android:textColor="@color/black" />
            </LinearLayout>
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_marginBottom="3dp"
                android:layout_weight="5" >
```

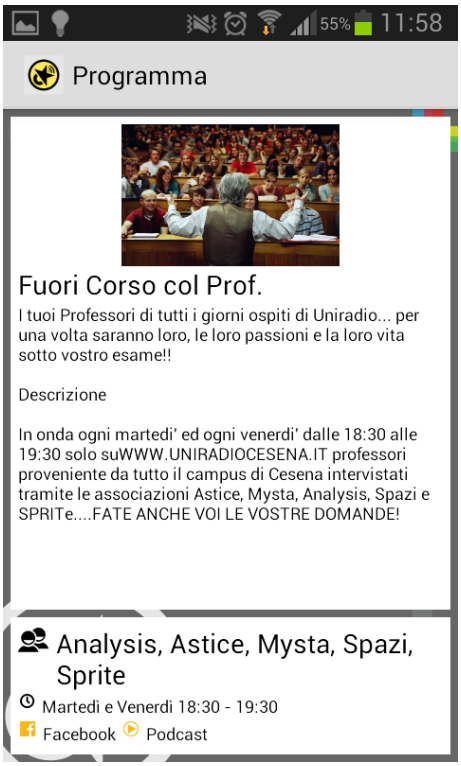
```
                    <TextView
                          android:id="@+id/evento_lblVia"
                          android:layout_width="50dp"
                          android:layout_height="wrap
                                                 _content"
                          android:layout_marginRight="3dp"
                          android:background="#cccccc"
                          android:gravity="center
                                          _vertical|left"
                          android:padding="2dp"
                          android:text="Via"
                          android:textColor="@color/black"
                                                  />

                    <TextView
                    android:id="@+id/evento_Via"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_weight="4"
                    android:background="#cccccc"
                    android:gravity="center
                                    _vertical|left"
                    android:padding="1dp"
                    android:text="TextView"
                    android:textColor="@color/black" />
               </LinearLayout>
               <LinearLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_marginBottom="3dp"
                    android:layout_weight="5" >
                    <TextView
                    android:id="@+id/evento_lblOra"
                    android:layout_width="50dp"
                    android:layout_height="wrap_content"
                    android:layout_marginRight="3dp"
                    android:background="#cccccc"
                    android:gravity="center
                                    _vertical|left"
                    android:padding="2dp"
                    android:text="Ora"
                    android:textColor="@color/black" />
                    <TextView
                    android:id="@+id/evento_Ora"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_weight="4"
```

```
                    android:background="#cccccc"
                    android:gravity="center
                                        _vertical|left"
                    android:padding="1dp"
                    android:text="TextView"
                    android:textColor="@color/black" />
            </LinearLayout>
        </LinearLayout>
        <TextView
            android:id="@+id/evento_descrizione"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="4"
            android:scrollbars="vertical"
            android:text="TextView"
            android:textColor="@color/black" />
    </LinearLayout>
</LinearLayout>
```

3.2.5 Programmi

La sezione programmi si sviluppa, come quella eventi, su due livelli. Nel primo abbiamo una griglia dove in anteprima possiamo vedere tutti i programmi presenti nel palinsesto della radio. Per ciascuno di essi abbiamo il Logo, il nome del programma, il giorno e l'ora di messa in onda. Nel secondo livello abbiamo invece il dettaglio del programma; in aggiunta alle informazioni precedenti troviamo una breve descrizione, i conduttori, il link alla pagina Facebook del programma ed il link ai Podcast di Soundcloud.





Item programma

```
<LinearLayout
     android:layout_width="200dp"
     android:layout_height="250dp"
     android:background="@color/white"
     android:orientation="vertical"
     android:padding="5dp"
     android:layout_margin="5dp"
     android:layout_weight="1" >

     <ImageView
        android:id="@+id/listprogrammi_logo"
        android:layout_width="wrap_content"
        android:layout_height="150dp"
        android:src="@drawable/icona"
        android:layout_gravity="center_horizontal|top" />

     <TextView
        android:id="@+id/listprogrammi_titolo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Titolo Programma"
        android:textAppearance="?android:attr/
                                textAppearanceMedium"
        android:textColor="@color/black" />
     <LinearLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingTop="5dp" >
        <ImageView
            android:id="@+id/listprogrammi_logo_giorno"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/giorno_programma" />
        <TextView
            android:id="@+id/list_programmi_giorno"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="Giorno programma"
            android:textColor="@color/black" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:paddingTop="5dp" >
        <ImageView
            android:id="@+id/listprogrammi_logo_ora"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ora_programma" />
        <TextView
            android:id="@+id/listprogrammi_ora"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="Orario"
            android:textColor="@color/black" />
    </LinearLayout>
</LinearLayout>
```

Dettaglio Programma

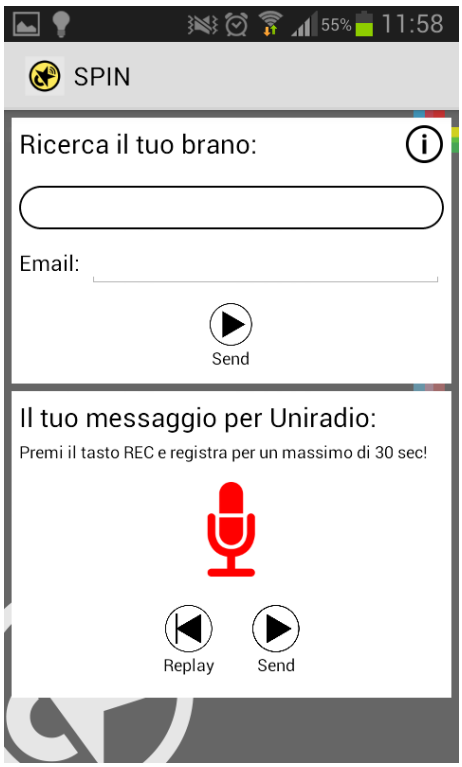
```
<LinearLayout
    android:id="@+id/linearLayout1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_above="@+id/linearLayout2"
    android:layout_alignParentTop="true"
    android:layout_margin="5dp"
    android:layout_marginBottom="5dp"
    android:background="@color/white"
```

```
    android:orientation="vertical"
    android:padding="5dp" >
    <ImageView
        android:id="@+id/programma_copertina"
        android:layout_width="match_parent"
        android:layout_height="100dp" />
    <TextView
        android:id="@+id/programma_titolo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Titolo Programma"
        android:textAppearance="?android:attr/
                                textAppearanceLarge"
        android:textColor="@color/black" />
    <TextView
        android:id="@+id/programma_descrizione"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:scrollbars="vertical"
        android:text="TextView"
        android:textColor="@color/black" />
</LinearLayout>
<LinearLayout
    android:id="@+id/linearLayout2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_alignParentRight="true"
    android:layout_marginBottom="5dp"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:background="@color/white"
    android:orientation="vertical"
    android:padding="5dp" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >
        <ImageView
        android:id="@+id/programma_logo_conduttori"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/conduttori" />
        <TextView
        android:id="@+id/programma_nomeconduttori"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="5dp"
```

```
        android:text="Conduttori"
        android:textAppearance="?android:attr/
                                textAppearanceLarge"
        android:textColor="@color/black" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="1dp" >
        <ImageView
            android:id="@+id/programma_logo_ora"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ora_programma" />
        <TextView
            android:id="@+id/programma_ora"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="Orario Programma"
            android:textColor="@color/black" />
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="1dp" >
        <ImageButton
            android:id="@+id/programma_logo_fb"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@null"
            android:src="@drawable/fb_programma" />
        <TextView
            android:id="@+id/programma_facebook"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="Facebook"
            android:textColor="@color/black" />
        <ImageButton
            android:id="@+id/programma_logo_podcast"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:background="@null"
            android:paddingLeft="5dp"
            android:src="@drawable/podcast_programma"
                                                    />
```

```
            <TextView
                  android:id="@+id/programma_podcast"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:paddingLeft="5dp"
                  android:text="Podcast"
                  android:textColor="@color/black" />
      </LinearLayout>
</LinearLayout>
```

3.2.6 SPIN



La sezione SPIN è uno degli esperimenti del reparto tecnico di Uniradio Cesena. L'obiettivo era quello di creare una piattaforma multimediale per dare la possibilità all'utente di poter interagire con la radio. Per questo motivo si è pensato di dotare questa sezione di due particolari funzioni. La prima, in alto, dà la possibilità di ricercare e richiedere un brano alla regia automatica della radio (MB Studio); data la presenza del brano nel database e fornita la propria e-mail, è possibile inviare al richiesta premendo *Send*. La seconda funzione permette invece di inviare un breve messaggio vocale di 30 secondi alla redazione di Uniradio. La registrazione del messaggio non viene inviata direttamente, ma si ha la possibilità di poterla riascoltare tramite il tasto *Replay*. Fino a quando l'utente non preme il tasto *Send* è possibile registrare ed ascoltare il proprio messaggio quante volte si vuole.

Ricerca brano

```
<LinearLayout
    android:id="@+id/spin_layout_ricerca"
```

```
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="5dp"
    android:layout_marginBottom="5dp"
    android:background="@color/white"
    android:orientation="vertical" >
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="5dp"
        android:layout_marginTop="5dp"
        android:weightSum="3" >
        <TextView
            android:id="@+id/spin_ricerca"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingLeft="5dp"
            android:text="Ricerca il tuo brano: "
            android:textAppearance="?android:attr/
                                    textAppearanceLarge"
            android:textColor="@color/black" />
        <ImageButton
            android:id="@+id/spin_info_logo"
            android:layout_width="wrap_content"
            android:layout_height="27dp"
            android:layout_alignParentRight="true"
            android:layout_marginRight="5dp"
            android:background="@null"
            android:scaleType="matrix"
            android:src="@drawable/spin_info" />
    </RelativeLayout>
    <AutoCompleteTextView
        android:id="@+id/city"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/cornice"
        android:popupBackground="#FFFFFF"
        android:textColor="@color/black"
        android:textColorHighlight="@color/black"
        android:lines="1"
        android:textColorHint="@color/black" />
    <LinearLayout
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:gravity="center"
        android:padding="5dp" >
        <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:paddingRight="5dp"
            android:text="Email:"
            android:textAppearance="?android:attr/
                                    textAppearanceMedium"
            android:textColor="@color/black" />
        <EditText
            android:id="@+id/spin_email"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textEmailAddress"
            android:textColor="@color/black" >
        <requestFocus />
        </EditText>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:paddingBottom="10dp"
        android:paddingTop="5dp" >
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:gravity="center_horizontal" >
            <ImageButton
                android:id="@+id/spin_richiedi_brano"
                android:layout_width="30dp"
                android:layout_height="30dp"
                android:background="@null"
                android:scaleType="fitXY"
                android:src="@drawable/spin_richiedi"
            <TextView
                android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="Send"
                android:textColor="@color/black"
                android:gravity="center_horizontal"/>
```

```
            </LinearLayout>
        </LinearLayout>
</LinearLayout>
```

Registra messaggio

Il Layout per la parte di registrazione del messaggio da inviare alla regia non è stato totalmente progettato in XML, ma è stato gestito attraverso codice Java. Il motivo è derivato dall'esigenza di voler applicare un effetto grafico al bottone di registrazione a forma di microfono.

XML

```
<LinearLayout
    android:id="@+id/spin_layout_registra"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_alignLeft="@+id/spin_layout_ricerca"
    android:layout_alignRight="@+id/spin_layout_ricerca"
    android:layout_below="@+id/spin_layout_ricerca"
    android:background="@color/white"
    android:orientation="vertical"
    android:padding="5dp" >
    <TextView
        android:id="@+id/spin_titolo_registra"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_vertical|left"
        android:paddingBottom="3dp"
        android:text="Il tuo messaggio per Uniradio: "
        android:textAppearance="?android:attr/
                                textAppearanceLarge"
        android:textColor="@color/black" />
    <TextView
        android:id="@+id/spin_testo_registra"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingBottom="3dp"
        android:text="Premi il tasto REC e registra per
                          un massimo di 30 sec!"
        android:textColor="@color/black"
        android:textSize="12dp" />
    <LinearLayout
        android:id="@+id/linearLayoutBottoniSpin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
            android:layout_margin="10dp"
            android:gravity="center_horizontal"
            android:orientation="horizontal"
            android:layout_weight="1" >
        </LinearLayout>
        <LinearLayout
            android:id="@+id/linearLayoutBottoniSpin2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_margin="10dp"
            android:layout_weight="1"
            android:gravity="center_horizontal"
            android:orientation="horizontal" >
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_gravity="center_horizontal" >
        </LinearLayout>
</LinearLayout>
```

Java

```
LinearLayout ll = (LinearLayout)
    findViewById(R.id.linearLayoutBottoniSpin);
        LinearLayout ll2 = (LinearLayout)

    findViewById(R.id.linearLayoutBottoniSpin2);
        mPlayButton = new PlayButton(this);
        mRecordButton = new RecordButton(this);
        ImageButton inviaRegistrazione = new
                                    ImageButton(this);
        LinearLayout view1 = new LinearLayout(this);
        LinearLayout view3 = new LinearLayout(this);

        LinearLayout l1 = new LinearLayout(this);
        LinearLayout l2 = new LinearLayout(this);
        l1.setOrientation(LinearLayout.VERTICAL);
        l1.setGravity(Gravity.CENTER);
        l2.setOrientation(LinearLayout.VERTICAL);
        l2.setGravity(Gravity.CENTER);

        TextView testo = new TextView(this);
        testo.setText("Replay");
        testo.setGravity(Gravity.CENTER_HORIZONTAL);
        testo.setTextColor(Color.BLACK);
```

```
          TextView testo2 = new TextView(this);
          testo2.setText("Send");
          testo2.setGravity(Gravity.CENTER);
          testo2.setTextColor(Color.BLACK);

inviaRegistrazione.setBackgroundResource(R.drawable.spin_
richiedi);

<. . . Altro codice. . .>

LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(30, 30);
          params.gravity = Gravity.CENTER_VERTICAL;

          LinearLayout.LayoutParams params2 = new
LinearLayout.LayoutParams(
                    ViewGroup.LayoutParams.WRAP_CONTENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT);

          LinearLayout.LayoutParams margin = new
LinearLayout.LayoutParams(

    LinearLayout.LayoutParams.FILL_PARENT,

    LinearLayout.LayoutParams.WRAP_CONTENT);

    margin.setMargins(10, 0, 10, 0);

    params2.gravity = Gravity.CENTER;

    mRecordButton.setLayoutParams(params);
    mPlayButton.setLayoutParams(params);
    inviaRegistrazione.setLayoutParams(params);

    view1.addView(mRecordButton, new
               LinearLayout.LayoutParams(100, 100, 1));
    l1.addView(mPlayButton, new
               LinearLayout.LayoutParams(50, 50, 1));

    l1.addView(testo, new LinearLayout.LayoutParams(
          ViewGroup.LayoutParams.WRAP_CONTENT,
          ViewGroup.LayoutParams.WRAP_CONTENT, 0));
    l2.addView(inviaRegistrazione, new
          LinearLayout.LayoutParams(50, 50, 1));

    l2.addView(testo2, new LinearLayout.LayoutParams(
          ViewGroup.LayoutParams.WRAP_CONTENT,
```

```
            ViewGroup.LayoutParams.WRAP_CONTENT, 0));
        ll.addView(view1, new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.WRAP_CONTENT, 1));
        ll2.addView(l1, new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.MATCH_PARENT,
            ViewGroup.LayoutParams.WRAP_CONTENT, 1));
        ll2.addView(view3, new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT, 0));
        ll2.addView(l2, new LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT, 0));
        l1.setLayoutParams(params2);
        testo2.setLayoutParams(params2);
        view1.setLayoutParams(params2);
        l1.setPadding(0, 0, 20, 0);
        l2.setPadding(20, 0, 0, 0);
```

3.2.7 Chi Siamo

La sezione Chi Siamo, ha un'interfaccia molto semplice e facilmente comprensibile. Nella parte superiore della schermata è presente una descrizione della radio attorno al logo della radio. Al di sotto abbiamo affiancato quelli che sono i link alle informazioni principali utili ad un nuovo utente per entrare in contatto con la radio.

XML

```
<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dp"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_marginTop="5dp"
    android:orientation="vertical" >
    <LinearLayout
        android:id="@+id/linearLayout1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@color/white"
        android:orientation="vertical"
        android:padding="5dp" >
        <TextView
            android:id="@+id/chi_siamo_descrizione1"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp"
            android:text="@string/chi_siamo
                                    _testo_sopra"
            android:textColor="@color/black" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:weightSum="2" >
            <ImageView
                android:id="@+id/chi_siamo
                                    _logo_uniradio"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginBottom="10dp"
                android:layout_marginRight="10dp"
                android:layout_weight="1"
                android:contentDescription="@string/
                            chi_siamo_testo_centrale"
                android:src="@drawable/chi_siamo" />
            <TextView
                android:id="@+id/chi_siamo
                                    _descrizione2"
                android:layout_width="160dp"
                android:layout_height="match_parent"
                android:layout_weight="1"
                android:padding="5dp"
```

```
                android:text="@string/chi_siamo
                                        _testo_centrale"
                android:textColor="@color/black" />
        </LinearLayout>
        <TextView
            android:id="@+id/chi_siamo_descrizione3"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent"
            android:padding="5dp"
            android:text="@string/chi_siamo
                                        _testo_sotto"
            android:textColor="@color/black" />
    </LinearLayout>
    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:background="@color/white"
        android:orientation="vertical"
        android:padding="10dp" >
        <TableRow android:layout_weight="4" >
            <ImageButton
                android:id="@+id/chi_siamo_logo_sito"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center
                                        _vertical"
                android:layout_weight="1"
                android:contentDescription="@drawable
                                        /sito_web"
                android:onClick="openSitoWeb"
                android:src="@drawable/sito_web"
                android:background="@null"/>
            <ImageButton
                android:id="@+id/chi_siamo
                                        _logo_collabora"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_gravity="center
                                        _vertical"
                android:layout_weight="1"
                android:contentDescription="@drawable
                                        /collabora"
                android:onClick="openCollabora"
                android:src="@drawable/collabora"
                android:background="@null"/>
            <ImageButton
```

```
                android:id="@+id/chi_siamo
                                    _logo_email"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:contentDescription="@drawable
                                        /email"
                android:onClick="openEmail"
                android:src="@drawable/email"
                android:background="@null"/>
          <ImageButton
                android:id="@+id/chi_siamo
                                    _logo_indirizzo"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:contentDescription="@drawable
                                    /dove_siamo"
                android:onClick="openContatti"
                android:src="@drawable/dove_siamo"
                android:background="@null"/>
     </TableRow>
     <TableRow
          android:layout_width="fill_parent"
          android:layout_weight="4" >
          <TextView
                android:id="@+id/chi_siamo_sito"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:gravity="center"
                android:text="@string/chi
                     _siamo_footer_menu_sito_web"
                android:textColor="@color/black" />
          <TextView
                android:id="@+id/chi_siamo_collabora"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_weight="1"
                android:gravity="center"
                android:text="@string/chi
                     _siamo_footer_menu_collabora"
                android:textColor="@color/black" />
          <TextView
                android:id="@+id/chi_siamo_email"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
```

```
                    android:layout_weight="1"
                    android:gravity="center"
                    android:text="@string/chi_siamo
                                    _footer_menu_mail"
                    android:textColor="@color/black" />
                <TextView
                    android:id="@+id/chi_siamo_indirizzo"
                    android:layout_width="0dp"
                    android:layout_height="wrap_content"
                    android:layout_weight="1"
                    android:gravity="center"
                    android:text="@string/chi
                         _siamo_footer_menu_indirizzo"
                    android:textColor="@color/black" />
            </TableRow>
        </TableLayout>
</LinearLayout>
```

4 Back End

In questo capito vedremo tutti gli aspetti nascosti all'utente, ovvero tutto quello che si cela dietro l'interfaccia grafica. Osserveremo per le sezioni principali, come otteniamo le risorse che poi mostriamo con *stile* nel Front-End. Prima di fare questo faremo una piccola introduzione teoria a qualche strumento utilizzato per i nostri fini: *JSON*.

4.1 Strumenti

ServiceAudio

Dovendo dare all'utente la possibilità di poter ascoltare lo streaming, è stato fondamentale la creazione di una classe *ServiceAudio* che estendesse la classe *Service.*
È stato necessario eseguire l'override dei metodi dei principali metodi di callback:

- *StarService()*: il sistema vene avviato ed eseguito in background a tempo indeterminato;
- *StopService()*: il sistema viene interrotto;
- *onBind():* il sistema chiama questo metodo quando un altro componente vuole legare con il servizio. Nell'implementazione di questo metodo, è necessario fornire un'interfaccia che i client utilizzano per comunicare con il servizio, restituendo un *IBinder*. Si deve sempre implementare questo metodo, ma se non si lo si considera vincolante, allora si dovrebbe restituire *null*:
- *onCreate():* il sistema chiama questo metodo quando il servizio viene creato;

- *onDestroy:* il sistema chiama questo metodo quando il servizio non è più utilizzato e deve essere distrutto. Il servizio deve implementare questo metodo per poter pulire risorse quali *thread*, *listeners* registrati, *receivers*, etc.;

JSON

Il sistema Android include le librerie *json* che permettono di lavorare facilmente con questo tipo di file. Quello che a noi interessa è com'è fatto un file JSON. Esso non è altro che un *array* di *array associativi*, chiamati oggetti, ognuno di questi composto da una coppia chiave valore.

[{"value":"New","action":"CreateNewDoc"},{"value":"Open","action":"OpenDoc"}]

Definiamo i concetti di array e di array associativi:

- *Array*: sequenze ordinate di valori, separati da virgole e racchiusi in parentesi quadre [ ];
- *Array associativi*: sequenze coppie chiave-valore separate da virgole racchiuse in parentesi graffe { };

Parser JSON

Fornito il file JSON, prima di utilizzarlo bisogna leggerlo in modo da ottenere le informazioni contenute al suo interno. Questa operazione è detta di *parsing*, ovvero di analisi. Vedremo successivamente come abbiamo affrontato questa esigenza nella sezione palinsesto.

Risorse esterne

Una cosa molto importante da dire è che l'applicazione è nata dopo la creazione del sito web della radio. Quest'ultimo ha già un suo player che necessita di risorse quali, ad esempio, l'immagine di copertina dell'album, l'artista ed il titolo del brano in onda, ed ovviamente la sorgente audio che permette lo streaming della canzone.
Nei paragrafi a seguire si farà riferimento a gli URL di tali risorse.

4.2 Streaming

Copertina

La copertina dell'Album la recuperiamo dal seguente URL:

http://service.uniradiocesena.it/OnAir.jpg

Tramite il seguente codice utilizziamo il metodo BitmapDownloaderTask per ottenere la bitmap da associare poi nell' ImageView *imageCopertina*

```
new BitmapDownloaderTask(
(ImageView)findViewById(R.id.imageCopertina)).execute("http://s
ervice.uniradiocesena.it/OnAir.jpg");
```

Info brano

Le aree di testo sono un po' più complesse da gestire dell'immagine di copertina. Infatti ci possiamo trovare in più situazioni. Prima però vediamo come ottenere il titolo del brano:

```
titolo = getPage();

private String getPage()throws MalformedURLException,
IOException {
    HttpURLConnection con = (HttpURLConnection) new URL(urlOvh)
                .openConnection();
    con.connect();
if (con.getResponseCode() == HttpURLConnection.HTTP_OK) {
          return inputStreamToString(con.getInputStream());
    }else{
          return null;
    }
}
```

Ottenuto il titolo intero (i.e. *QUEEN - ANOTHER ONE BITES THE DUST)*, adesso bisogna dividerlo in Artista e Titolo. Questo lavoro lo facciamo per mezzo del costruttore *StringTokenizer* della classe *StringTokenizer*:

```
StringTokenizer tokens = new StringTokenizer(titolo, "-");
```

Il costruttore *StringTokanizer* suddivide la stringa *titolo* in più token utilizzando il trattino come elemento divisorio.

Come dicevamo, possiamo trovarci in più situazioni:

```
if (tokens.countTokens() == 0) {
                txtArtista.setVisibility(View.GONE);
                txtTitolo.setVisibility(View.GONE);
                txtLblArtista.setVisibility(View.GONE);
                txtLblTitolo.setVisibility(View.GONE);
          } else if (tokens.countTokens() == 1) {
                txtArtista.setVisibility(View.VISIBLE);
                txtTitolo.setVisibility(View.GONE);
                txtLblArtista.setVisibility(View.GONE);
                txtLblTitolo.setVisibility(View.GONE);

     txtArtista.setGravity(Gravity.CENTER_HORIZONTAL);
LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(
                LayoutParams.MATCH_PARENT,
                LayoutParams.MATCH_PARENT);
                params.setMargins(90, 0, 90, 0);
                txtArtista.setLayoutParams(params);

     txtArtista.setText((tokens.nextToken()).replaceAll("\n",
""));
          } else if (tokens.countTokens() == 2) {
                txtArtista.setVisibility(View.VISIBLE);
                txtTitolo.setVisibility(View.VISIBLE);
                txtLblArtista.setVisibility(View.VISIBLE);
                txtLblTitolo.setVisibility(View.VISIBLE);
                String first =
     (tokens.nextToken()).replaceAll("\n", "");
                String second = tokens.nextToken();
                try {
                     if (second.charAt(0) == ' ')
                          second = second.replaceFirst(" ",
"");
                     second = second.replaceAll("\n", "");
                } catch (Exception e) {}
                txtArtista.setText(first);
                txtTitolo.setText(second);
          } else if (tokens.countTokens() == 3) {
                txtArtista.setVisibility(View.VISIBLE);
                txtTitolo.setVisibility(View.VISIBLE);
                txtLblArtista.setVisibility(View.VISIBLE);
                txtLblTitolo.setVisibility(View.VISIBLE);
                String first = tokens.nextToken();
                String second = tokens.nextToken();
                try {
                     if (second.charAt(0) == ' ')
```

```
                        second = second.replaceFirst(" ",
"");
                } catch (Exception e) {    }
                String third = tokens.nextToken();
                txtArtista.setText(first);
                txtTitolo.setText(second + " " + third);
          }
```

- *tokens.countTokens() == 0;* nel caso in cui il numero di token è uguale a 0, non visualizza nessun tipo di informazione, quindi *.setVisibility(View.GONE);*
- *tokens.countTokens() == 1;* nel caso in cui il numero di token è uguale a 1, vuol dire che ha trovato ho il nome della rotazione oppure c'è un programma in onda e quindi visualizza solo la TextView dell'Artista.
- *tokens.countTokens() == 2;* nel caso in cui il numero di token è uguale a 2, viene messo in Artista il primo token e in Titolo il secondo e quindi visualizzato titolo ed artista.
- *tokens.countTokens() == 3;* nel caso in cui il numero di token è uguale a 3, il risultato è come il caso precedente, ma viene messo in Artista il primo e in Titolo l'append del secondo col terzo.

Player multimediale

All'apertura dell'applicazione, nel metodo *onCreate()* della *StreamingActivity*, verranno inizializzati i pulsanti play e stop e quindi settati i rispettivi OnClickListener:

```
private void Inizializza() {
    playButton =
    (ImageButton)findViewById(R.id.button_play);
    playButton.setOnClickListener(new View.OnClickListener()
    {
          public void onClick(View view) {
                service.avviaStreaming();
                creaNotifica();
                playButton.setEnabled(false);
                service.isPlaying = true;
                playButton.setImageResource(
                      R.drawable.playbutton_on);
          }
    });
    stopButton = (ImageButton)
    findViewById(R.id.button_stop);
```

```
    stopButton.setOnClickListener(new View.OnClickListener()
    {
          public void onClick(View view) {
                service.stopStream();
                cancellaNotifica();
                playButton.setEnabled(true);
                service.isPlaying = false;
                playButton.setImageResource(
                      R.drawable.playbutton);
          }
    });
}
```

Alla pressione del tasto play, viene avviato lo streaming e creata nella barra in alto l'icona di notifica che ci permetterà di poter riaprire l'applicazione in qualsiasi momento mentre stiamo utilizzando il terminale in altri contesti. Verrà anche caricata l'immagine di un nuovo tasto Play, con però questa volta uno sfondo colorato per far anche vedere che lo streaming è attivo.

Al contrario invece, alla pressione del tasto Stop, lo streaming verrà interrotto e l'icona di notifica verrà eleminata. L'immagine del tasto Play tornerà ad essere quella iniziale.

La risorsa per lo streaming è recuperabile al seguente indirizzo:

```
public String url_audiostream =
"http://stream.uniradiocesena.it:9002/";
```

e tramite il costruttore *serviceAudio()*, inizializzo l'oggetto *service* di tipo *serviceAudio*

```
service = new serviceAudio();
```

I metodi utilizzabili su questo service sono:

- **public void** avviaStreaming()
- **void** loadStream(StreamingActivity activity)
- **synchronized public void** startStream()
- **synchronized public void** stopStream()
- **public void** onDestroy()

Tasto Facebook

Quando viene premuto il tasto Facebook per la condivisione della risorsa audio che si sta ascoltando, viene richiamato il metodo *postToWall()*:

```
pubblica.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                     postToWall();
                }
});
```

Il metodo *postToWall* controlla se la sessione di Facebook è valida. Se non lo è, esegue quanto scritto nel blocco if, altrimenti quanto scritto nel blocco else:

```
public void postToWall() {
    if (!facebook.isSessionValid()) {
    }else{
          Bundle params = new Bundle();
          params.putString(
          "name", "Sto ascoltando " + stringaTitolo
                          + " su UniRadio App per Android");
          params.putString("picture",
                "http://service.uniradiocesena.it/OnAir.jpg");
          params.putString("link",
                "http://www.uniradiocesena.it");
          facebook.dialog(this, "feed", params, new
          DialogListener() {
                @Override
                public void onFacebookError(FacebookError e) {
                }
                @Override
                public void onError(DialogError e) {
                }
                @Override
                public void onComplete(Bundle values) {
                }
                @Override
                public void onCancel() {
                }
          });
    }
}
```

4.3 Palinsesto

La gestione del palinsesto la possiamo suddividere il più step.
Il primo concerne il recupero delle informazioni da visualizzare successivamente nella struttura a Tab scorrevoli. Nel metodo *onCreate()* della classe *PalinsestoActivity* viene richiamato il costruttore per avviare l'AsyncTask

```
new MyAsyncTask().execute();
```

La classe *MyAsyncTask* estende la classe *AsyncTask<String, Integer, Double>*. Quest'ultima consente di effettuare delle operazioni affidate ad un thread in background, per poi mostrare i risultati sull'User Interface del thread principale. Nella stesura della nostra nuova classe *MyAsyncTask* abbiamo fatto l'*Override* dei metodi della super classe, per poter eseguire quanto a noi interessava.
I metodi a disposizione sono:

- **onPreExecute**: operazioni di preparazione effettuate prima del task.
- **doInBackground**: le operazioni che il task deve fare. In questa fase non è possibile modificare elementi della UI.
- **onProgressUpdat**: usato per mostrare lo stato di avanzamento del task.
- **onPostExecute**: operazioni da eseguire dopo il task.
- **onCancelled**: invocato se il task viene annullato.

La nostra nuova classe ha richiesto solo l'utilizzo di alcuni di questi metodi. Vediamo quali

doInBackground

```
protected Double doInBackground(String... params) {
    try {
```

Passiamo l'url del web service php che genera il Json del palinsesto a partire da una query sul Database del sito della radio. Il web service è stato creato dall' amministratore del sito appositamente per l'utilizzo nell'applicazione mobile.

```
          String url =
    "http://service.uniradiocesena.it/.../db_palinsesto.php";
```

Di conseguenza creiamo un client http, sul quale per mezzo di chiamate come *HttpGet(uri)*, client.execute(request), entity.getContent() otteniamo il contenuto

```
          HttpClient client = new DefaultHttpClient();
          HttpGet request = new HttpGet(uri);
          HttpResponse response;
          response = client.execute(request);
          HttpEntity entity = response.getEntity();
          InputStream content = entity.getContent();
```

Attraverso metodi di lettura e di costruzione stringhe, viene infine inizializzata una variabile *result* di tipo stringa, che conterrà l'intera stringa JSON.

```
          BufferedReader reader = new BufferedReader(
               new InputStreamReader(content));
          StringBuilder builder = new StringBuilder();
          String line;
          while ((line = reader.readLine()) != null) {
               builder.append(line); }
          content.close();
          String result;
          result = builder.toString();
```

Adesso viene richiamato il metodo per l'utilizzo della stringa Json contenuta in result *parseJson(result).*

```
parseJson(result);
    } catch (ClientProtocolException e) {
          e.printStackTrace();
    } catch (IOException e) {
          e.printStackTrace();
   }
    return null;
}
```

Vediamo nel dettaglio cosa avviene richiamando il metodo *parseJson(result).*

```
public void parseJson(String result) {
```

Dichiaro sette ArrayList, uno per ogni giorno della settimana.

```
    LunediList = new ArrayList<HashMap<String, String>>();
    MartediList = new ArrayList<HashMap<String, String>>();
    MercolediList = new ArrayList<HashMap<String,String>>();
```

```
    GiovediList = new ArrayList<HashMap<String, String>>();
    VenerdiList = new ArrayList<HashMap<String, String>>();
    SabatoList = new ArrayList<HashMap<String, String>>();
    DomenicaList = new ArrayList<HashMap<String, String>>();
```

Mantengo delle variabili temporanee che mi serviranno durante la lettura del JSON

```
Integer giorno;
         String titolo;
         String ora_inizio, ora_fine;
         String conduttori;
         String descrizione;
         try {
```

Controllo se è già presente un'istanza globale di una variabiel di tipo JSONArray, così non devo re-inizializzarne una nuova per mezzo di una locale.

```
JSONArray jArray;
               if (jArrayPalinsesto == null) {
                    jArray = new JSONArray(result);
                    jArrayPalinsesto = jArray;
               } else
                    jArray = jArrayPalinsesto;
```

Sapendo che un file JSON è composto da un array di JSON object e questi sono composti s loro volta da coppie di dati <chiave, valore>, utilizzo un ciclo di acquisizione dati dove, per ogni JSON object, assegno il valore letto alla variabile temporanea corrispondente e precedentemente creata.

```
for (int i = 0; i < jArray.length(); i++) {
    JSONObject json_data =
    jArray.getJSONObject(i);
    giorno = Integer.parseInt(json_data.getString("weekday"));
    titolo = json_data.getString("class_name");
    ora_inizio = json_data.getString("start_hour");
    ora_fine = json_data.getString("end_hour");
    conduttori = json_data.getString("instructor_name");
    descrizione = son_data.getString("instructor_description");
```

Creo un oggetto di tipo HashMap che conterrà un sequenza di coppie chiave valore che poi andranno a popolare gli ArrayList di ogni singolo giorno della settimana. Quest'ultima operazione verrà eseguita per mezzo di uno switch.

```
               HashMap<String, String> map = new
                    HashMap<String, String>();
```

```
                titolo = titolo.replace("\\\'", "'");
                String[] separated = titolo.split("#");
                titolo = separated[0];
                map.put("titolo", titolo);
                map.put("giorno", giorno.toString());
                map.put("conduttori", conduttori);
                map.put("descrizione", descrizione);
                map.put("oraInizio", ora_inizio);
                map.put("oraFine", ora_fine);
                switch (giorno) {
                     case 1:
                          LunediList.add(map);
                          break;
                     case 2:
                          MartediList.add(map);
                          break;
                     case 3:
                          MercolediList.add(map);
                          break;
                     case 4:
                          GiovediList.add(map);
                          break;
                     case 5:
                          VenerdiList.add(map);
                          break;
                     case 6:
                          SabatoList.add(map);
                          break;
                     case 0:
                          DomenicaList.add(map);
                          break;
                     }
                }
           } catch (JSONException e) {
                Log.e("log_tag", "Error parsing data " +
                     e.toString());
           }
}
```

doInBackground

```
protected void onPostExecute(Double result) {
                riempiTabs();
           }
```

Come anticipato nell'onPostExecute troviamo le operazioni da eseguire dopo il task. In questo caso quello che faremo sarà riempire le Tab giorno per giorno. Vediamo i passaggi principali che eseguono questa operazione.

Inizializziamo un viewpageradapter che conterrà tutti gli ArrayList ottenuti

```
          viewpageradapter.items0 = LunediList;
          viewpageradapter.items1 = MartediList;
          viewpageradapter.items2 = MercolediList;
          viewpageradapter.items3 = GiovediList;
          viewpageradapter.items4 = VenerdiList;
          viewpageradapter.items5 = SabatoList;
          viewpageradapter.items6 = DomenicaList;
```

Setto il viewpageradapter all'interno del ViewPager chiamato mPager

```
mPager.setAdapter(viewpageradapter);
```

Una volta creati i tabListener dobbiamo associare ad ogni Tab l'etichetta che identifica il giorno. Questa operazione ci permette di selezionare la ListTiew corretta che ci interessa.

```
ActionBar.TabListener tabListener = new
ActionBar.TabListener(){};

   tab =
mActionBar.newTab().setText("Lunedì").setTabListener(tabListene
r);
    mActionBar.addTab(tab);
    tab = mActionBar.newTab().setText("Martedì")
                    .setTabListener(tabListener);
    mActionBar.addTab(tab);
    tab = mActionBar.newTab().setText("Mercoledì")
                    .setTabListener(tabListener);
    mActionBar.addTab(tab);
    tab = mActionBar.newTab().setText("Giovedì")
                    .setTabListener(tabListener);
    mActionBar.addTab(tab);
    tab = mActionBar.newTab().setText("Venerdì")
                    .setTabListener(tabListener);
    mActionBar.addTab(tab);
    tab = mActionBar.newTab().setText()"Sabato"
                    .setTabListener(tabListener);
    mActionBar.addTab(tab);
    tab = mActionBar.newTab().setText("Domenica")
                    .setTabListener(tabListener);
    mActionBar.addTab(tab);
```

Infine abbiamo il metodo *selezionaGiorno()* che ci permette, a partire dall'acquisizione del giorno corrente, di mostrare automaticamente la scheda corrispondete all'apertura del palinsesto.

```
private void selezionaGiorno() {
          Calendar calendar = Calendar.getInstance();
          int day = 0;
          switch (calendar.get(Calendar.DAY_OF_WEEK)) {
          case 1:
               day = 6;   //Corrisponde alla Domenica
               break;
          case 2:
               day = 0;   //Corrisponde al Lunedì
               break;
          case 3:
               day = 1;   //Corrisponde al Martedì
               break;
          case 4:
               day = 2;   //Corrisponde al Mercoledì
               break;
          case 5:
               day = 3;   //Corrisponde al Giovedì
               break;
          case 6:
               day = 4;   //Corrisponde al Venerdì
               break;
          case 7:
               day = 5;   //Corrisponde al Sabato
               break;
          }
          mPager.setCurrentItem(day);
          progressBar.setVisibility(View.GONE);
}
```

5 Conclusioni e sviluppi futuri

L'idea di produrre una tesi sullo sviluppo di una applicazione mobile per una Web Radio, è derivata da fattori di diversa entità: primo fra tutti, l'applicare le conoscenze acquisite durante il percorso di studi e vedere quindi realizzato un proprio prodotto utilizzato da un bacino più o meno grande di utenti; secondariamente, il partecipare attivamente al progetto Uniradio Cesena ha portato a ponderare la necessità di dare all'utente la possibilità di fruire lo streaming radiofonico non solo da una postazione web, ma anche dal proprio Smartphone o Tablet e quindi in mobilità.

La creazione di questa applicazione ha richiesto l'approfondimento di linguaggi quali Java per Android e XML.

È stata sviluppata parallelamente anche la versione per iOS, che però non è stata presa in considerazione ai fini di questa tesi.

Molto importante è stato anche il lavoro che è stato fatto dal reparto tecnico della radio, che saputo progettare al meglio il sito ed il database dal quale, attraverso dei web service, si è potuto utilizzare le risorse nell'applicazione.

Gli sviluppi futuri saranno molteplici poiché si vuole puntare molto su questa applicazione per dare visibilità e potenzialità al progetto delle radio. Si è cercato di creare un elenco di funzioni da aggiungere ed altre da migliorare:

- *Sveglia*: vogliamo dare la possibilità all'utente affezionato di potersi svegliare ascoltando la nostra radio.
- *Souncloud*: quello che adesso è semplicemente un collegamento alla pagina web di Soundcloud, dove sono presenti tutte le registrazioni delle puntante andate in onda dei programmi, lo integreremo nella schermata di dettaglio di ogni programma. Verrà quindi visualizzata la lista delle puntate da poter

riascoltare. Questo aggiornamento avverrà quando sarà presente la licenza podcast

- *Chat con la regia*: vogliamo dare la possibilità all'utente di poter scrivere, durante la diretta di un programma, alla regia.
- *Icona di notifica*: vogliamo migliorare l'icona di notifica, che si viene a creare nella barra in alto, aggiungendo la possibilità di manipolare il flusso multimediale con le classiche funzioni stop, pause e play. Così facendo, l'ascoltatore, se è impegnato nella consultazione di una pagina web, non deve necessariamente tornare all'applicazione per interrompere il flusso.
- *Grafica*: verrà riprogettata la grafica al fine di migliora l'esperienza d'uso in visione del prossimo aggiornamento del sistema Android alla nuova versione.
- *Social*: nella prossima versione integreremo le API di Twitter e Google+ per aumentare le possibilità di condivisioni.

Bibliografia

1. Android Developers Guide
   http://developer.android.com/

2. Sito Uniradio Cesena
   http://www.uniradiocesena.it