Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Triennale in
STATISTICA E TECNOLOGIE INFORMATICHE


RELAZIONE FINALE

# VIAGGI: UN'APPLICAZIONE GEOLOGALIZZATA PER IL TURISMO

## *VIAGGI: A LOCATION-AWARE APPLICATION FOR TURISM*

Relatore Prof. Loris Nanni
Dipartimento di Ingegneria dell'Informazione

Laureando: Favaro Nicola
Matricola n°1000489

Anno Accademico 2012/2013

*Ringrazio i miei genitori che*
*mi hanno permesso di proseguire*
*gli studi e mi hanno sempre sostenuto, i*
*miei compagni di università per la aver reso*
*indimenticabili questi tre anni.*

# INDICE

# INTRODUZIONE

Uno smartphone (o in italiano telefonino intelligente, cellulare intelligente, telefonino multimediale) è un dispositivo mobile che abbina alle funzionalità di telefono cellulare quelle di gestione di dati personali che siano essi musica, immagini o documenti si qualsiasi genere.

Sebbene la storia degli smartphone inizi a partire dagli anni '90, il vero cambio di rotta si ha nel 2007, anno in cui Apple rilascia il primo iPhone . Si tratta di un prodotto innovativo che si posiziona nella fascia alta del mercato degli smartphone e offre le più avanzate funzionalità multimediali. E' dotato di uno schermo multi-touch e l'interazione con l'utente è coadiuvata da un accelerometro e un giroscopio digitale che insieme fungono da sensore di movimento; dispone anche di un sensore di prossimità e un sensore di luce. Apple ha depositato più di 300 brevetti legati a tale dispositivo.

Pochi mesi dopo il lancio dell'IPhone e quindi dal corrispondente sistema operativo iOS (come spiegato nel paragrafo 2.1.3) arriva il suo più grande concorrente: Android. La società Android Inc. viene fondata nel 2003 e nel 2005 viene acquistata da Google. Dopo due anni di sviluppo, nel novembre del 2007, viene presentato il nuovo sistema operativo in collaborazione con produttori di smartphone come HTC e Samsung. Da questa data gli aggiornamenti sono stati numerosi per eliminare bug e migliorare le prestazioni; ognuno di questi possiede la caratteristica di prendere il nome dai dolciumi: la versione 1.5 si chiama *cupcake* fino ad arrivare alla 4.3, la più recente, *jelly bean.*

Android e iOS insieme occupano più del 90% del mercato degli smartphone ma esistono altri sistemi operativi appartenenti a questo mondo come ad esempio Windows Phone legato principalmente alla Nokia, BlackBerry famoso inizialmente per la gestione della mail da dispositivo, Bada ecc.

Una delle caratteristiche principali di questi sistemi operativi è la possibilità di installare nuove applicazioni oltre a quelle predefinite già installate nel dispositivo: si tratta delle cosiddette app. Con il termini app, che è un neologismo abbreviazione di applicazione, si va a indicare un software leggero e semplice che ha lo scopo di fornire un determinato servizio all'utente.

Esempi di app sono ad esempio Spotify e Shazam per la musica, Whatsapp per la messaggistica istantanea (quindi necessita di una connessione Internet) e per finire anche tutti i giochi per smartphone.

Al fine di favorire la distribuzione delle applicazioni si utilizzano appositi distributori digitali meglio conosciuti come store o market. Ognuno di questi "negozi" è usualmente dedicato ad un singolo sistema operativo al quale "appartiene" e contiene solo applicazioni compatibili con quest'ultimo. Nonostante ciò, la maggior parte delle applicazioni di successo è disponibile per quasi tutti i sistemi operativi e questo comporta sviluppi diversi a seconda della piattaforma.

La diffusione degli smartphone e, di conseguenza, delle app, ha fatto sì che molte aziende di informatica si dedicassero al settore *mobile* e allo sviluppo di applicazioni per questi dispositivi.

# CAPITOLO 1 - LA NOSTRA IDEA

Durante il periodo di stage, che ho svolto alla Business Reasearch s.r.l (un'azienda di Padova che si occupa prevalentemente di siti internet e web application), mi sono occupato della realizzazione un'app per smartphone. L'idea è quella di realizzare un'applicazione location-based, quindi basata sulla geolocalizzazione[1], che ha come principale obiettivo quello di mettere in contatto agenzie viaggi e possibili turisti.
Come nome dell'applicazione è stato scelto "Viaggi".
Sia agenzie che turisti possono trarre beneficio dall'utilizzo di questo software: le prime avrebbero nuovi clienti, mentre i secondi potrebbero consultare le proposte delle agenzie vicine in qualsiasi momento.
Le agenzie con le quali ci si vuole interfacciare sono quelle specializzate nei viaggi su misura. In questo modo gli utenti possono avere a che fare con veri 'esperti del settore' e scegliere il viaggio più adatto alle loro esigenze. Si parla solo di determinate tipologie di viaggio: i cosiddetti viaggi di fascia alta come quelli di nozze.
Uno dei punti di forza dell'applicazione è la geolocalizzazione che consente di visualizzare solo i viaggi delle agenzie che stanno "intorno" all'utente. Questo permette, in una seconda fase, un contatto diretto tra cliente e agenzia per stabilire ogni singolo dettaglio del viaggio. Questo passaggio è, a mio parere, indispensabile trattandosi comunque di viaggi costosi.
Ad ogni modo un primo contatto con l'agenzia è possibile averlo direttamente dall'app che prevede le funzioni di invio mail e chiamata.
I clienti possono scegliere le categorie di viaggio da visualizzare. Le categorie vanno un po' a riassumere il concetto di "cosa voglio fare durante il viaggio": la maggior parte dei turisti non ha un'idea precisa di dove andare ma ha ben chiaro il tipo di vacanza che intende fare.
L'app prevede anche la possibilità di esprimere un giudizio su viaggi e agenzie per far sì che gli utenti possano comunicare tra di loro esprimendo opinioni: conoscere le idee altrui aiuta sicuramente nella scelta.

[1] Si tratta di individuare la posizione del device e quindi anche dell'utente.

## 1.1 Principio di funzionamento

Entrando un po' più nel dettaglio questa applicazione prevede due parti:

- L'app per smartphone;
- Parte web (sito internet).

### *1.1.1 L'app per smartphone*

L'app per smartphone è dedicata ai turisti e prevede le seguenti funzioni:

- Login/registrazione usando la mail o usando le credenziali di alcuni social networks;
- La visualizzazione dei viaggi disponibili;
- Invio mail e chiamata dell'agenzia;
- Votazione di agenzia e viaggio;
- Visualizzazione dei viaggi per categoria;
- Galleria delle foto dei viaggi.

### *1.1.2 Parte web (sito internet)*

Il sito internet è dedicato alle agenzie viaggi che, grazie ad una registrazione, possono gestire il proprio profilo e la propria lista viaggi all'interno dell'app.

# CAPITOLO 2 - STRUMENTI UTILIZZATI

In una prima fase della progettazione dell'applicazione bisogna fare delle scelte sulle tecnologie da utilizzare nello sviluppo. Ad esempio, nell'introduzione abbiamo accennato dell'esistenza di diversi sistemi operativi: con quale iniziare? Che strumenti utilizzare per la geo-localizzazione? Dove e come salvare tutte le informazioni? Come far comunicare database e app?
Vediamo una panoramica degli strumenti a disposizione e le decisioni prese.

## 2.1 Sistemi operativi per smartphone

### *2.1.1 Blackberry*

BlackBerry OS è un sistema operativo, sviluppato da Research In Motion (RIM) per la sua linea di smartphone BlackBerry. Il sistema operativo consente di eseguire più processi contemporaneamente (multitasking) e supporta vari dispositivi di input: tra i più famosi si ha la rotellina (trackball) e più recentemente il trackpad e touchscreen. La piattaforma BlackBerry è nota per il suo supporto nativo della mail aziendale attraverso MIDP[2] 1.0 (Mobile Information Device Profile) e, più recentemente, un sottoinsieme di MIDP 2.0 che consente di completare l'attivazione wireless e la sincronizzazione di e-mail, calendario, attività, note e contatti con server mail quali Microsoft Exchange, Lotus Domino o Novell GroupWise. Da 2 anni a oggi la vendita di questi dispositivi è in netto calo.

### *2.1.2 Windows Phone*

Sistema operativo compatto basato sulle API (Application Programming Interface) Win 32 della Microsoft. Questo sistema nasce nel 2003 con il nome di Windows Mobile ed è rilasciato in tre versioni: Windows Mobile 2003 Pocket PC, Windows Mobile 2003 Phone Edition, Windows Mobile 2003 Smartphone, per poi essere migliorato nella versione Windows Mobile 2003 SE grazie al contributo della DELL (azienda statunitense , tra le più importanti al mondo nella produzione di personal computer e di sistemi informatici).

[2] Si tratta di un insieme di API java che si interfacciano con cellulari, cercapersone ecc.

Nonostante Windows Mobile risalga al 2003, i primi tentativi di realizzare un sistema operativo per dispositivi mobili risalgono già al 1996, anno in cui prese vita Windows CE 1.00[3]. L'esperienza acquisita in precedenza con lo sviluppo della piattaforma Windows CE porta a basare Windows Mobile 5.0 sia su tale piattaforma che .NET compact FRAMEWORK 2.0 permettendo di avere una gestione e segnalazione degli errori simile a quella dei sistemi desktop e server di Windows, nonché un incremento della durata delle batterie. Ad oggi l'ultima versione di Windows Mobile è la 6.5 uscita successivamente alla 6.0 e alla 6.1.

Le maggiori novità apportate da questa release, rilasciata ufficialmente nel febbraio 2009 al Mobile Word Congress interessano l'interfaccia grafica, completamente ridisegnata, e un aggiornamento di Internet Explorer.

Nel 2010 viene rilasciato Windows Phone 7 che è completamente differente da tutte le precedenti versioni di Windows Mobile: supporta il multitouch, gli schermi capacitivi, ha una nuova interfaccia grafica molto simile a quella di Zune HD[4], e riunisce in una sola piattaforma i contenuti di Xbox LIVE e Zune. Inoltre consente la gestione gli account di social network quali Facebook e Twitter, e possiede una nuova versione di Internet Explorer basata su Windows Internet Explorer 7 con alcuni elementi della versione 8. Tra le app disponibili si ha una versione mobile di Office 2010, contenente Word, Excel, Powerpoint, OneNote e Sharepoint.

Un ulteriore aggiornamento si ha nel 2011 con Windows Phone 7.5 e con la versione 7.8 rilasciata nel gennaio di quest'anno.

L'ultima versione oggi disponibile è la 8, nata a fine 2012 che consente la condivisione con Windows 8, di kernel, file system, driver, gestione rete, sicurezza, componenti multimediali e grafiche con i relativi benefici come il supporto alle SD.

Windows Mobile può essere considerato una versione in miniatura di Windows: esso infatti fornisce un pieno supporto al multitasking, utilizza un analogo registro di sistema, consente l'installazione di programmi provenienti da diverse fonti e media e gestisce i file in modo molto simile a Windows 9X/NT.

[3] Sistema operativo nato per la gestione dei dispositivi embedded. Con il termine embedded si va ad identificare tutti quei programmati per una determinata applicazione (bancomat, POS, router).

[4] Lettore multimediale prodotto da Microsoft che supporta il touchsrcreen.

### *2.1.3 iOS*

L'IPhone OS (iOS) è il sistema operativo sviluppato da Apple per i suoi dispositivi IPhone, IPod touch e IPad. L'IPhone OS è, per scelta della Apple, funzionante solo sui dispositivi della casa. iOS usa un kernel MACH e deriva dal sistema operativo open source DARWIN[5] rilasciato sempre da Apple. Il kernel Mach condivide la stessa architettura I/O del MAC OS X e permette agli sviluppatori il riutilizzo di ingenti quantità di codice già funzionante per Mac. In questo modo, inoltre, vi è anche la possibilità di espandere, in un futuro, il supporto hardware senza riprogettare l'intero kernel.

L'IPhone OS è essenzialmente costituito da quattro strati (figura 1), dove gli strati Core OS e Core Services offrono i servizi fondamentali mentre gli strati Media e Cocoa Touch forniscono le astrazioni dei servizi dei livelli inferiori.


**Figura 1- Struttura iOs.**

Entrando maggiormente nel dettaglio lo strato Cocoa Touch è composto dai frame-work UIKit e Foundation che mettono a disposizione gli strumenti per la gestione dell'interfaccia grafica, dell'hardware dedicato alla cattura di foto e video, dell'accelerometro e il supporto per la gestione degli eventi.

Lo strato sottostante, Media, offre i servizi per lo sviluppo della grafica, la gestione dell'audio e del video. Tra questi si ha il frame-work OpenGL ES per il disegno 2D e 3D e il frame-work QUARTZ per la realizzazione degli effetti grafici. Lo strato Core Services si occupa dei servizi di rete e della comunicazione con database mentre lo strato Core OS gestisce file system, sicurezza e durata della batteria.

[5] Sistema operativo free di NeXT Computer (Steve Jobs) con kernel XNU derivante da MACH

L’ambiente di sviluppo integrato per IPhone OS è XCode, utilizzato anche per le applicazioni Mac OS X.
L'11 giugno 2012 è stata presentata la sesta versione di iOS che presenta numerose novità: una nuova applicazione delle mappe, altre applicazioni native come Facetime e PassBook, nuove funzioni e lingue per l'assistente vocale Siri, tra cui la lingua italiana, integrazione con Facebook, nuove funzioni di risposta alle chiamate come ad esempio la riposta con sms o la funzione “non disturbare”, novità grafiche, nuovi sistemi di sicurezza.
Il 10 giugno 2013 è stata presentata la settima versione di iOS che offre una nuova grafica e nuove funzioni. Il nuovo layout è stato ridisegnato in modo da essere semplice e intuitivo da utilizzare. L’introduzione di un centro di controllo rende subito disponibili le funzioni più utilizzate. Altro importante punto di rottura col precedente iOS è la rimozione della barra di sblocco presente fin dalla prima release, sostituendola con una schermata di sblocco più semplice e minimalista. Un rinnovamento sostanziale è la totale revisione del multitasking: ne è stata modificata la grafica e le funzionalità. Per gli sviluppatori sono state rilasciate oltre 1500 API. La data esatta di rilascio non è ancora stata comunicata, ma è in attesa per questo autunno.

#### *2.1.3.1 Sviluppo*

Nell’ottobre 2007, Steve Jobs ha annunciato che un SDK (software development kit) sarebbe stato disponibile dal febbraio 2008. L'SDK è stato rilasciato un mese in ritardo rispetto alla data prevista e permette agli sviluppatori di creare applicazioni per iPhone e iPod touch, e di testarle con l’ausilio di una virtual machine. Tuttavia la diffusione di un’applicazione nei dispositivi è possibile solamente dopo aver pagato una tassa di iscrizione all'iOS Developer Program. Come descritto nella sezione precedente l'ambiente di sviluppo per iOS SDK è Xcode.
Gli sviluppatori sono liberi di stabilire il prezzo delle applicazioni che sono distribuite tramite App Store, per le quali riceveranno il 70% del ricavo. Essi possono anche optare per rilasciare l'applicazione gratis.

### *2.1.4 Android*

Android è un sistema operativo di moderna fattura e dalla struttura abbastanza complessa. Come si evince dalla figura 2, Google sotto la supervisione di Open Hanset

Alliance ha attinto, in parte, dal mondo Open Source, con l'obiettivo di realizzare un sistema operativo leggero, robusto e che consenta lo sviluppo di applicazioni in piena libertà senza porre barriere e restrizioni ai progettisti e ai programmatori.

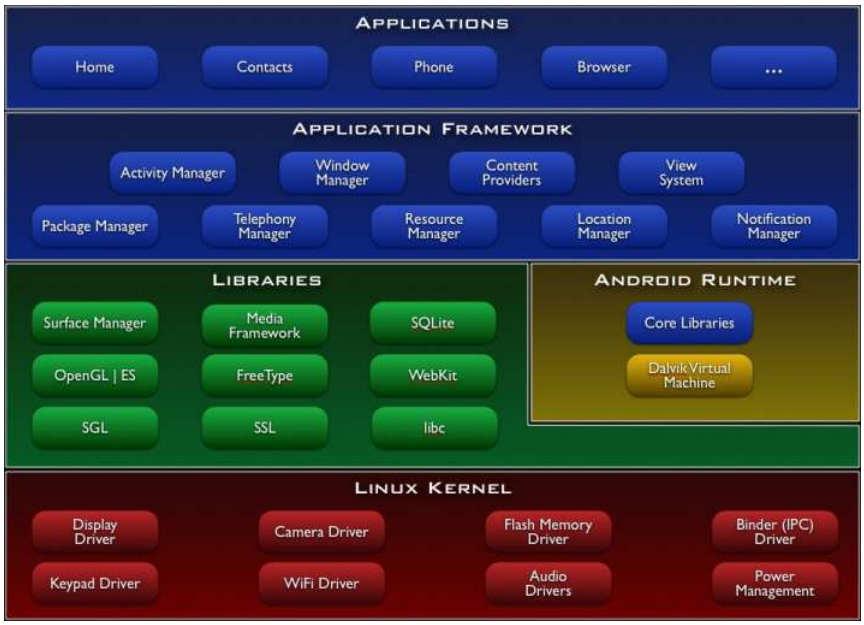



**Figura 2 - Architettura Android**

Queste caratteristiche insieme alle statistiche di vendita (figura 3) dei sistemi operativi per smartphone, all'assenza di costi per lo sviluppo e per la distribuzione sul Play Store ci hanno portato a puntare su questo sistema operativo.

| Italy | 3 m/e Apr 2012 | 3 m/e Apr 2013 | % pt. Change |
|---|---|---|---|
| iOS | 22.2 | 16.6 | -5.6 |
| Android | 47.9 | 66.7 | 18.8 |
| BlackBerry | 5.2 | 2.0 | -3.2 |
| Symbian | 15.8 | 3.5 | -12.3 |
| Windows | 6.7 | 10.5 | 3.8 |
| Other | 2.1 | 0.7 | -1.4 |

**Figura 3 - Statistiche sull'utilizzo dei più importanti s.o. per smartphone (Kantar)**

Osservando la struttura di Android è possibile rendersi conto di come le varie componenti della piattaforma interagiscano tra loro e, in particolare, come i livelli superiori forniscano le astrazioni per i servizi offerti dai livelli inferiori.

#### *2.1.4.1 Struttura*

Vediamo ora più nel dettaglio la struttura.

Kernel

E' lo strato di più basso livello della architettura Android ed è basato su kernel Linux (in figura 4 si vedono le varie versioni). In questo livello avviene la gestione della memoria e di tutti i driver hardware come quello dedicato alla tastiera, allo schermo, al touchpad, al Wi-Fi, al Bluetooth, al controllo dell'audio e così via.

| Android Version | Linux Kernel Version |
|---|---|
| 1.0 | 2.6.25 |
| 1.5 (Cupcake) | 2.6.27 |
| 1.6 (Donut) | 2.6.29 |
| 2.2 (Froyo) | 2.6.32 |
| 2.3 (Gingerbread) | 2.6.35 |
| 3.0 (Honeycomb) | 2.6.36 |
| 4.0.x (Ice Cream Sandwich) | 3.0.1 |
| 4.1./4.2 (Jelly Bean) | 3.0.31 |

**Figura 4 - Versione del kernel nelle varie distribuzioni Android.**

Libraries

Un livello più su troviamo un insieme di librerie C/C++ Open Source, usate dai vari componenti del sistema Android:

- *Media Library*: permette di supportare la riproduzione e la registrazione dei più diffusi formati audio-video (MP3, AAC, AMR, JPG, PNG).
- *LIBWebCore* : un moderno motore per browser web che è la base sia per il browser Android sia per la Web View utilizzabile via codice.
- SGL: gestisce la grafica 2D.
- 3D Libraries : basate su OpenGL ES 1.0 utilizzano, se disponibile, l'acceleratore grafico del dispositivo oppure in sua assenza utilizzano codice altamente performante per la gestione della grafica 3D.
- SQLite: per la gestione dei dati persistenti sul dispositivo.

Android Runtime

Questo livello è formato da due parti:

- *Dalvik Virtual Machine ( DVM )*: la macchina virtuale, chiamata Dalvik, è stata scritta da Dan Bornstein, sviluppatore Google. Si caratterizza per essere particolarmente efficiente in termini di uso della memoria e per il fatto di poter essere eseguita efficientemente in istanze multiple. Tale macchina virtuale non esegue bytecode Java ma un suo formato .dex anche lui ottimizzato per occupare poco spazio in

memoria e opportunamente generato tramite il tool dx a partire dai file .class ottenuti, a loro volta, dalla normale compilazione del codice Java.

- *Core Libraries*: insieme a Dalvik costituiscono il kit di sviluppo delle applicazioni Android. Le Core Libraries sono tutte le librerie di base necessarie quando si implementa un'app.

Application Framework

Nel penultimo strato dell'architettura si trovano i gestori e le applicazioni di base del sistema. Ci sono gestori per le risorse, per le applicazioni installate, per le telefonate, per il file system e per tutti i componenti di cui difficilmente si può fare a meno. I più importanti sono:

- Content Provider: permette alle applicazioni di accedere ai dati di altre applicazioni (ad esempio i contatti di una rubrica telefonica) oppure di condividere i propri dati.
- Notification Manager: permette alle applicazioni di mostrare avvisi personalizzati nella barra di stato.
- Activity Manager: per la gestione del ciclo di vita di una applicazione.

Application

Sullo strato più alto dell'architettura, poggiano gli applicativi destinati all'utente finale. Molti, naturalmente, sono già inclusi con l'installazione di base: il browser ed il player multimediale sono dei facili esempi altri possono essere installati dal Play Store. A questo livello si inseriranno anche le applicazioni sviluppate dai programmatori e quindi anche l'app Viaggi trattata in questo documento.

#### *2.1.4.2 Caratteristiche delle app Android*

Le applicazioni Android sono scritte tutte in linguaggio Java e questo semplifica non di poco la programmazione. Java mette a disposizione, infatti, tante librerie e classi già pronte per l'uso permettendo di concentrarsi prettamente sul problema da risolvere. Tuttavia Android ha come caratteristica principale il fatto che un'applicazione possa usufruire di elementi che fanno parte di un' altra applicazione estendendo, così, il concetto di riuso del codice. Ciò nonostante non è sufficiente la sola conoscenza di java ma anche delle componenti che, integrate tra loro, realizzano una applicazione Android funzionante. Un'applicazione per questo sistema operativo è formata da 4 mattoni fondamentali:

- Activity;

- Service;
- Broadcast receiver;
- Content provider.

***Activity***

Un'activity è composta da un'interfaccia grafica e da alcune funzionalità. Comprende tutto quello che è possibile vedere in una singola schermata video come ad esempio può essere la home dell'app Facebook. Una applicazione Android può avere una o più activity e ognuna di esse deve realizzare il layout tramite una gerarchia di oggetti grafici chiamati View, ogni vista controlla una particolare porzione della finestra e si comporta da padre nei confronti delle altre poste all'interno di tale porzione. Android mette a disposizione degli sviluppatori un numero considerevole di componenti già pronti per la composizione dell'interfaccia di un'activity: bottoni, caselle di testo e liste a scomparsa. Una activity può lanciarne un'altra che non deve per forza appartenere alla stessa applicazione. All'utente sembrerà che le activity, caratterizzate da schermate differenti, appartengano alla stessa applicazione. Per rendere possibile tutto questo, Android colloca le due activity in uno stesso task e fa si che l'utente concepisca il task come l'applicazione. L'organizzazione delle activity all'interno del loro task si realizza tramite una struttura a stack, ogni nuova activity viene posta in cima alla struttura con una operazione di push e al termine del suo ciclo di vita viene rimossa con una operazione di pop. Non è possibile in nessun modo effettuare una riorganizzazione delle activity, questo per far sì che quella che si trova in cima alla pila sia effettivamente in esecuzione e sia quella che interagisce con l'utente. Tutte le activities hanno un proprio ciclo di vita.

Ciclo Di Vita

Un'activity una volta avviata dal sistema può presentarsi essenzialmente in quattro strati:

- **INACTIVE**: Un'activity in questo stato è pronta per essere eseguita. Questo è anche lo stato in cui un'activity si trova quando viene distrutta dal sistema;
- **RUNNING**: Quando un'activity è in questo stato significa che è in cima allo stack (e quindi in ogni istante ci può essere solo un'activity in questo stato). Android non terminerà mai il processo legato a quest'activity, anche in condizioni di pochissime risorse disponibili;

- **PAUSED**: Un'activity si trova in questo stato quando è la penultima dello stack ed è ancora parzialmente visibile. Ovvero quando l'activity principale è una dialog, cioè una finestra in stile popup, o ha un effetto trasparenza. In pratica l'activity in questo stato era quella in cima allo stack prima che venisse lanciata una nuova activity che, però, non ricopre completamente la vecchia;
- **STOPPED**: Una activity in questo stato non è più visibile.

E' importante sapere lo stato in cui l'activity si trova perché Android, in caso di scarse risorse disponibili, può o meno decidere di chiudere una o più activity per liberare risorse. Le activity in stato STOPPED e INACTIVE possono essere eliminate tranquillamente. Le activity che mai cercherà di chiudere sono quelle in stato di RUNNING, mentre quelle che cercherà di chiudere per ultime sono quelle in stato PAUSED.

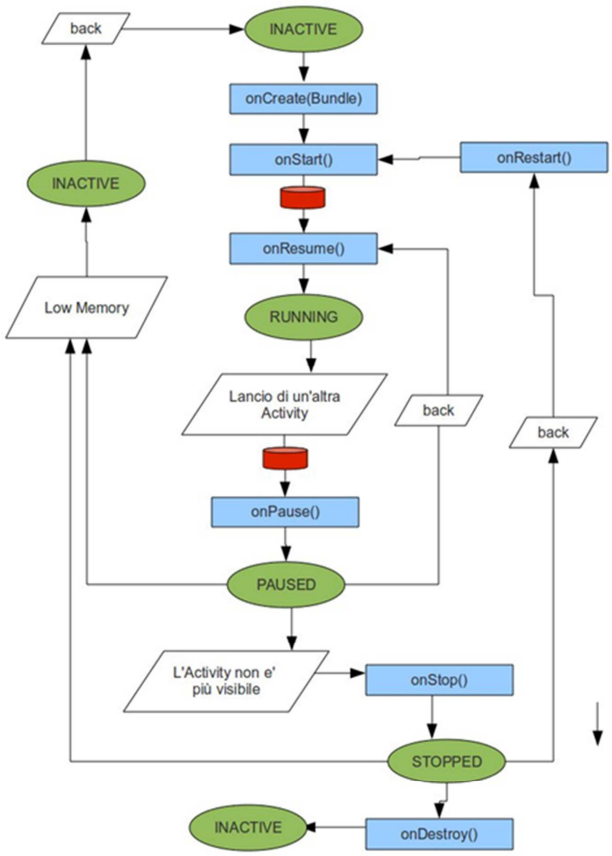



**Figura 5 - Ciclo di vita di un'activity Android**

Quando viene lanciata un'activity per la prima volta, essa parte dallo stato INACTIVE e vengono invocati in successione i metodi onCreate(), onStart() e onResume(). A questo punto l'activity è visibile sullo schermo (stato RUNNING) e l'utente può interagire con essa. L'activity rimane in questo stato fino a quando non avviene una delle due cose:

- l'utente preme back (volendo tornare alla schermata / activity precedente);
- Viene visualizzata un'altra activity, che può voler dire che l'utente ha premuto un bottone per visualizzare un'altra activity (operazione voluta) o magari è in arrivo un telefonata e compare l'activity del telefono (operazione non voluta).

Se l'utente preme back, allora viene invocato il metodo onPause() forzandone la messa in pausa, (stato PAUSED) e quando sarà comparsa l'activity precedente, verrà invocato il metodo onStop() (stato STOPPED). Cosa quasi simile accade se si presenta il secondo caso. Quando la nuova activity sta per essere visualizzata, viene immediatamente chiamato il metodo onPause() (stato PAUSED) e a seconda che la nuova activity, una volta visualizzata, copra o meno tutto lo schermo, viene invocato o meno il metodo onStop() (stato STOPPED).

Nell'immagine 5 raffigurante il ciclo di vita di un'activity sono presenti anche dei cilindri rossi che corrispondono a 2 metodi che consentono rispettivamente di salvare e rispristinare lo stato dell'activity.

Il primo metodo, invocato prima del metodo onPause(), è

- *public void onSaveInstanceState(Bundle bundle)*

che consente di salvare lo stato dell'activity un attimo prima di essere messa in pausa. Un oggetto Bundle appartiene all'SDK di Android e mappa valori String a altri valori di vario tipo. Da notare che a questo instante non si è a conoscenza se l'activity sarà successivamente messa in CLOSED e poi in INACTIVE (e quindi distrutta) quindi è buona norma usare questo metodo per salvare tutti i dati dell'activity che vorremmo successivamente ripristinare.

Il secondo metodo da analizzare (invocato dopo l' onStart() e prima dell' onResume()) è il metodo

- *public void onRestoreInstanceState(Bundle bundle)*

che riceve il Bundle salvato in precedenza che ci consente di ripristinare i valori salvati. Successivamente viene invocato il metodo onResume() e di conseguenza possiamo far ripartire tutte le computazioni come se nulla fosse avvenuto.

***Services***

Un service è un componente Android che non ha un'interfaccia grafica per l'utente ma può operare in background per un periodo indefinito di tempo. Viene usato in quelle situazioni in cui si devono svolgere operazioni onerose che non richiedono iterazione con l'utente (riproduzione di musica, download di dati dalla rete, un processo di installazione).

Un service può essere usato in due modi:

• Può essere indipendente ed eseguire il suo compito sino alla terminazione (naturale o forzata).

• Il service definisce e pubblica un'interfaccia che un client usa per stabilire una connessione per usufruire dei suoi servizi. Più client possono connettersi con lo stesso service.

***Broadcast receiver***

Il broadcast receiver è un componente Android realizzato per implementare un protocollo di scambio messaggi di tipo asincrono[6] tra componenti della stessa applicazione e tra componenti di applicazioni diverse. Una activity o un service possono registrarsi ad uno o più receiver se interessate a un particolare annuncio, identificativo di un cambiamento dello stato del sistema o della stato di un componente e il receiver una volta intercettato il messaggio svolgerà delle azioni per gestire il fenomeno segnalato. Ogni receiver estende la classe base Broadcast Receiver di cui ridefinire il metodo onReceive().

Nel metodo onReceive(), il programmatore deve aver cura di inserire le istruzione da eseguire per la gestione dell'evento mentre la chiamata del metodo avviene in maniera automatica da parte del sistema operativo.

***Content provider***

Un content provider permette ad una applicazione di rendere accessibili da parte di altre applicazioni un insieme di dati memorizzati nel file system o in un database. Un content provider estende la classe base ContentProvider e deve implementare una serie di metodi che permettano la lettura e la scrittura dei dati.

[6] Asincrono perché non si sa quanto tempo passi tra 2 eventi successivi.

#### *2.1.4.3 Intent*

Le activities, i services e i broadcast receiver vengono attivati tramite messaggi di tipo asincrono chiamati intents. Un intent è un oggetto istanza della classe Intent e può essere considerato come una struttura dati passiva che contiene una descrizione astratta delle operazioni da eseguire. In particolare un intent contiene informazioni di interesse per il componente che lo riceve (l'azione da intraprendere e i dati su cui operare) e informazioni di interesse per Android (la categoria del componente che deve gestire l'intent). Esistono essenzialmente due tipologie di Intent: impliciti ed espliciti.

Gli intent di tipo esplicito specificano il componente da attivare tramite il suo nome e vengono usati generalmente per gestire l'invio di messaggi interni all'applicazione, ad esempio quando una activity ne vuole lanciare un'altra. Gli intent impliciti, invece, non prevedono il nome del componente da attivare e lasciano ad Android il compito di trovare il miglior componente da attivare. Questa tipologia di Intent viene utilizzata per usare componenti appartenenti ad altre applicazioni dove, per forza di cose, non si è a conoscenza dei nomi delle varie parti.

Per poter rispondere ad un intent, un componente deve utilizzare uno o più "Intent Filters" per poter comunicare al sistema Android la capacità di gestire un dato intent. Tale registrazione può avvenire sia dinamicamente via codice, sia staticamente attraverso l'AndroidManifest. Nel caso di intent espliciti Android non consulta nessun intent filters e chiama direttamente il componente interessato. La disattivazione dei componenti Android si svolge in maniera analoga alla loro attivazione, un componente lanciando un intent relativo ad un altro componente può deciderne la disattivazione oppure può decidere di disattivarsi da solo al termine delle sue operazioni usando il metodo finish() (activity) o stopSelf() (service).

#### *2.1.4.4 L'interfaccia grafica*

I due elementi fondamentali dell'interfaccia grafica si chiamano View e ViewGroup. I bottoni, i campi di testo, le icone, le checkbox e tutti gli altri oggetti di un'interfaccia grafica sono oggetti View. I ViewGroup, invece, sono dei contenitori che possono mettere insieme e gestire graficamente più oggetti View. I ViewGroup, inoltre, sono a loro volta degli oggetti View in quanto estendono la classe View, e di conseguenza

possono contenere altri ViewGroup. In questo modo è possibile organizzare i componenti sullo schermo secondo uno schema ad albero (figura 6).

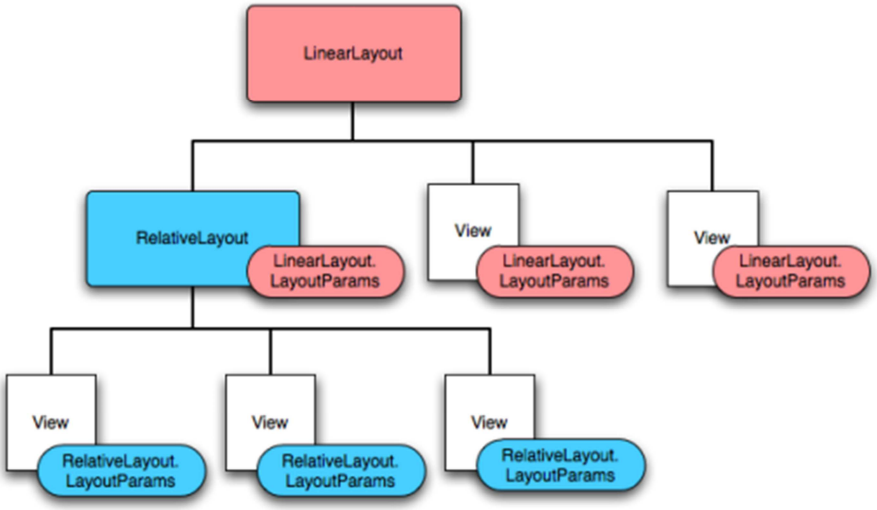



**Figura 6 - Organizzazione View/ViewGroup**

I componenti View estendono tutti la classe baseandroid.view.View. Nella libreria standard di Android ci sono già molti componenti predefiniti (widget), soprattutto nel pacchetto android.widget.

Oltre ai widget di base, che solitamente sono sufficienti, è possibile estendere la classe View e implementare componenti personalizzati. La classe android.view.ViewGroup è una speciale estensione di View. Come accennato in precedenza, e come rappresentato in figura, un ViewGroup è un oggetto che può contenere altre View. Per questo motivo il metodo addView(), che permette proprio di aggiungere una nuova View al gruppo, ha diverse firme in base alle esigenze. ViewGroup è una classe astratta, pertanto non può essere istanziata direttamente. Come nel caso di View, anche qui è possibile realizzare il proprio ViewGroup custom, ma quelli messi a disposizione da Android consentono già di realizzare tutto quello che serve. Queste implementazioni differiscono nella maniera di strutturare i componenti che sono al loro interno: alcuni li mettono uno dopo l'altro, altri li organizzano in una tabella, altri ancora possono essere usati per avere una gestione a schede dello schermo, e così via. Una volta che, combinando oggetti View e ViewGroup, si è ottenuta l'interfaccia utente che si desidera, è necessario che questa venga mostrata sullo schermo. Per far ciò le attività (cioè gli oggetti android.app.Activity) mettono a disposizione un metodo setContentView().

Widget

Con il termine widget (congegno) si indicano quegli oggetti predefiniti dal sistema che consentono per l'interazione con l'utente, come i bottoni, le checkbox, le liste, i campi di testo e così via. I widget estendono tutti (direttamente o indirettamente) la classe View, e sono conservati nel package android.widget. Vediamone una veloce panoramica:

- *android.widget.TextView* : permette di mostrare del testo all'utente;
- *android.widget.EditText*: estende TextView e permette all'utente di modificare il testo mostrato;
- *android.widget.Button* : realizza un bottone che l'utente può premere o cliccare;
- *android.widget.ImageView* : è un componente che permette di mostrare un'immagine;
- *android.widget.ImageButton* : è un bottone con un'immagine;
- *android.widget.CheckBox:* è un componente realizza una casella di spunta (check box, appunto);
- android.widget.AnalogClock: è un componente che mostra all'utente un orologio analogico.

Di widget ce ne sono molti altri, alcuni dei quali li vedremo nella descrizione del progetto.

***Layout***

Con il termine layout (disposizione, impaginazione), in Android, si identificano tutti quei ViewGroup che consentono di organizzare i widget sullo schermo. Vediamo alcuni dei layout predefiniti che offre Android:

- *android.widget.RelativeLayout:* organizza i componenti al suo interno relativamente agli altri. In questo modo una view può essere posizionata a destra di un'altra, nella parte bassa della view group padre, in centro e così via;
- *android.widget.LinearLayout*: un layout utile per disporre più componenti uno di seguito all'altro, sia orizzontalmente sia verticalmente;
- *android.widget.TableLayout* : un layout che permette di sistemare i componenti secondo uno schema a tabella, suddiviso cioè in righe e colonne.

I widget ed i layout illustrati sinora, naturalmente, devono essere combinati in maniera coerente. I layout, in maniera particolare, possono e devono essere annidati l'uno dentro l'altro, finché non si ottiene il design desiderato.

***Come utilizzare widget e layout***

Esistono 2 modi per utilizzare gli oggetti appena citati. Il primo metodo è simile quello adottato da altre librerie Java per le interfacce utente, come AWT e Swing. Questo modo di creare le GUI è potente, ma anche estremamente tedioso. Ogni volta che si deve utilizzare un widget, lo si deve creare, personalizzare ed inserire in un contenitore predisposto in precedenza, il tutto utilizzando il linguaggio Java. Questo porta a scrivere grandi quantità di codice e non è l'approccio migliore. Un secondo metodo consiste nell'utilizzare il linguaggio XML (EXtensible Markup Language) che offre una serie di vantaggi. Grazie all'indentazione si vede chiaramente chi è il contenitore e chi il contenuto, in questo modo si dispone di una visione più chiara della struttura dei vari componenti grafici. Gli attributi di XML, poi, sono molto più semplici ed intuitivi rispetto ai metodi di Java. Con gli attributi è più semplice settare le caratteristiche di ogni singolo componente, come il testo visualizzato, il margine, la gravità. Grazie all'XML è possibile utilizzare anche un semplice editor visuale, che appartiene al plug-in Android Development Tools (ADT) di Eclipse, che consente di gestire la grafica in modo più semplice.

Ovviamente serve qualche componente che si occupi di gestire e in qualche modo di interpretare le risorse XML. Di questo si occupa un Resource Manager che permette l'accesso a risorse non create via codice, siano esse layout, stringhe o oggetti grafici.

#### *2.1.4.5 L'interazione con l'utente*

Nella sezione precedente abbiamo parlato di come gestire il layout ma questi componenti devono in qualche modo interagire con l'utente e "reagire" agli eventi come ad esempio un click (touch), lo scroll ecc.. Tutti i widget di Android dispongono di una serie di metodi di callback, con nomi del tipo onTipoEvento(). Questi metodi vengono richiamati automaticamente ogni volta che il corrispondente evento è riscontrato sul widget.

Prendiamo ad esempio metodo chiamato onClick(). Questo metodo è eseguito automaticamente ogni volta che un oggetto viene toccato dall'utente, attraverso il touch screen del suo dispositivo. Sono molti i metodi di questa categoria, ed ogni widget ha i propri con le sue regole e la sua firma: parametri e valore di ritorno, insomma, cambiano di caso in caso. Ogni volta che si usa un widget, bisogna estenderlo e generare quindi un'altra classe, in modo da poter ridefinire il metodo (o i metodi) di

callback di proprio interesse. Per questo, Android mette a disposizione un meccanismo più semplice e sbrigativo, basato sull'utilizzo dei cosiddetti event listener.
Tutti i widget mettono a disposizione una seconda serie di metodi, questa volta del tipo setOnTipoEventoListener(). Il widget Button, ad esempio, dispone del metodo setOnClickListener(). Attraverso i metodi di questa categoria è possibile registrare al widget degli event listener, cioè delle istanze di speciali classi, realizzate appositamente per ricevere una notifica ogni volta che lo specifico evento accade. Per ciascun differente tipo di event listener esiste un'interfaccia apposita, che lo sviluppatore deve implementare per creare il suo gestore dell'evento. Ad esempio, l'interfaccia da implementare per gli eventi di clic è android.view. View.OnClickListener.
Ciascuna interfaccia, ovviamente, richiede l'implementazione di uno o più metodi.

***Eventi***

Vediamo adesso quali sono gli eventi che, molto probabilmente, realizzando un'applicazione Android sarà necessario gestire:

- *Click*. Il metodo sul widget è setOnClickListener(), e l'interfaccia per il gestore da implementareè View.OnClickListener. Il metodo richiesto dall'interfaccia è onClick(View view), che in parametro riporta il widget che ha subito l'evento;
- *Click lungo*. Un evento che accade quando si clicca su un widget e si mantiene la pressione per qualche istante. Il metodo per registrare l'event listener è setOnLongClickListener(), e l'interfaccia per il gestore è View.OnLongClick Listener. Il metodo da implementare è onLongClick(View view). Il metodo, come nel caso precedente, riceve come parametro un riferimento al widget su cui si è prodotto l'evento. In più, il metodo deve restituire un booleano per segnalare se l'evento è stato completamente gestito (true) oppure no (false);
- *Tocco*. Un evento più generico dei due precedenti: serve per rilevare un tocco qualsiasi su un componente. Il metodo per registrare il listener sul widget è setOnTouchListener(), mentre l'interfaccia per implementarlo è View.OnTouch Listener. L'interfaccia richiede il metodo onTouch(View view, MotionEvent event);
- *Digitazione*. Un evento usato per segnalare la pressione o il rilascio di un tasto della tastiera hardware. Il metodo per registrare il listener sul widget è

setOnKeyListener(), mentre l'interfaccia per implementarlo è View.OnKeyListener. L'interfaccia richiede il metodo onKey(View view, int keyCode, KeyEvent event).

#### *2.1.4.6 Dati persistenti*

Realizzando un'applicazione, con una buona probabilità, sarà necessario salvare dei dati in maniera persistente.

Per questo Android offre diverse possibilità a seconda delle esigenze:

- Le Shared Preferences che consentono di salvare dati primitivi (quindi interi, stringhe, float…) in formato chiave-valore. Android mette a disposizione la classe SharedPreferences che consente proprio la gestione di queste informazioni;
- E' possibile salvare informazioni all'interno della memoria del dispositivo. Di default le informazioni salvate da un'applicazione sono accessibili solo dall'applicazione stessa;
- Un altro metodo è quello di utilizzare la memoria esterna (la classica micro SD);
- SQLite database: Android supporta pienamente questo tipo di database. Ogni base di dati sarà accessibile solo dalle classi dell'applicazione;
- Un ultimo metodo è quello di utilizzare la connessione Internet per comunicare con un database remoto. Per far ciò è possibile utilizzare le classi contenute nei seguenti package: java.net.* , android.net.* .

Nell'applicazione Viaggi avremo bisogno di un database remoto per salvare le informazioni degli utenti, delle agenzie, dei viaggi ma anche di salvare informazioni sul dispositivo. Per i dati locali la modalità che mi è sembrata più adatta consiste nell'utilizzare le Shared Preferences, in quanto i dati necessari in locale all'app sono di tipo semplice.

## 2.2 Parte Web

Anche per quanto riguarda la parte web abbiamo deciso di puntare il più possibile su tecnologie free e open source, ovvero su quelle che non richiedono costi di licenza e il cui codice sorgente è disponibile e consultabile. La scelta delle tecnologie utilizzate è

stata guidata sia dalla loro validità sia dal fatto che esse fossero già in uso presso i clienti.
Vediamo ora nel dettaglio quali sono :

### 2.2.1 PHP

Php(acronimo ricorsivo di "PHP: Hypertext Preprocessor", preprocessore di ipertesti) è un linguaggio di programmazione interpretato, originariamente concepito per la programmazione Web ovvero per la realizzazione di pagine web dinamiche. L'interprete ha una licenza open source e libera (ma incompatibile con la GPL). Attualmente è utilizzato principalmente per sviluppare applicazioni web lato server, ma può essere usato anche per scrivere script a riga di comando o applicazioni stand-alone con interfaccia grafica. PHP riprende per molti versi la sintassi del C, come peraltro fanno molti linguaggi moderni, e del Perl. È un linguaggio debolmente tipizzato e dalla versione 5 migliora il supporto al paradigma di programmazione ad oggetti. PHP è in grado di interfacciarsi a innumerevoli database tra cui MySQL e PostgreSQL, e supporta numerose tecnologie, come XML e SOAP (Simple Object Access Protocol). Nella nostra applicazione verrà utilizzato sia per realizzare l'applicazione web sia per i web services di cui parleremo tra poco.

### 2.2.2 MySQL

Si tratta di un relational database management system (RDBMS) scritto in C/C++, composto da un client con interfaccia a riga di comando e un server, entrambi disponibili sia per sistemi Unix o Unix-like come GNU/Linux che per Windows, anche se prevale un suo utilizzo in ambito Unix. Come già accennato in precedenza, nel database della nostra applicazione saranno memorizzate le informazioni dei clienti, delle agenzie, dei viaggi, del rating ecc.

### 2.2.3 Web Service

Un web service è un componente applicativo. Possiamo definirlo come un sistema software in grado di fornire un servizio ad un'applicazione comunicando attraverso la rete tramite il protocollo HTTP (Hypertext Transfer Protocol). Un web service consente quindi alle applicazioni che vi si collegano di usufruire delle funzioni che mette a

disposizione. Un web service comunica tramite protocolli e standard definiti "aperti" e quindi sempre a disposizione degli sviluppatori.
I principali vantaggi delle tecnologie web sono:

- I web services permettono l'interoperabilita' tra applicazioni scritte in linguaggi diversi( language agnostic);
- Utilizzano nella maggior parte dei casi dati di tipo testuale, quindi più comprensibili e facili da utilizzare per gli sviluppatori, senza tuttavia impedire completamente lo scambio di dati in formato binario;
- Normalmente, essendo basati sul protocollo HTTP, non richiedono modifiche alle regole di sicurezza utilizzate come filtro dai firewall;
- Sono semplici da utilizzare e possono essere combinati l'uno con l'altro (indipendentemente da chi li fornisce e da dove vengono resi disponibili) per formare servizi "integrati" e complessi;
- Permettono di riutilizzare applicazioni già sviluppate;
- Fintanto che l'interfaccia rimane costante, le modifiche effettuate ai servizi rimangono trasparenti;
- I web services sono in grado di pubblicare le loro funzioni e di scambiare dati con il resto del mondo;
- Tutte le informazioni vengono scambiate attraverso protocolli "aperti".

Nell'applicazione per lo scambio di dati ho utilizzato l'XML e JSON (JavaScript Object Notation).

- ***XML***: acronimo di EXtensible Markup Language, fornisce le fondamenta per i Web Services. E' quella tecnologia dei Web services su cui si basano tutte le altre. Similmente a HTML, XML usa tag di programmazione per definire la proprietà dei dati sulle pagine Web e altri documenti. I tag XML, tuttavia, sono molto più versatili perchè possono definire non solo il modo in cui questi dati vengono visti, ma anche il contenuto stesso dei dati. Mentre HTML limita l'utente ad utilizzare tag predefiniti, con XML è possibile definire un numero illimitato di tag o schemi per l'accesso, per la manipolazione e la visualizzazione dei dati. Una delle caratteristiche principali di XML è la possibilità di definire dei tipi di documento utilizzando linguaggi di schema quali DTD (Document Type Definition) e XML Schema;

- ***JSON***: acronimo di JavaScript Object Notation, è un formato adatto allo scambio dei dati in applicazioni client-server. La semplicità di JSON ne ha decretato un rapido utilizzo specialmente nella programmazione in AJAX (Asynchronous JavaScript and XML). Il suo uso tramite JavaScript è particolarmente semplice, infatti l'interprete è in grado di eseguirne il parsing tramite una semplice chiamata alla funzione eval(). Questa sua caratteristica lo ha reso velocemente molto popolare a causa della diffusione della programmazione in JavaScript nel mondo del Web.

Nell'applicazione "Viaggi" i web services saranno utilizzati per effettuare delle interrogazioni sul database come ad esempio "dimmi quali sono i viaggi più vicini", "quali sono le informazioni di questa agenzia?". Più avanti questa parte sarà discussa nel dettaglio.

## 2.3 Social Network

In molte applicazioni è ormai possibile effettuare il login utilizzando i social network e questo comporta a non dover compilare nessun form di registrazione in quanto grazie ai social si hanno tutte le informazioni necessarie.

### *2.3.1 Facebook*

Il nome "Facebook" prende spunto da un elenco con nome e fotografia degli studenti, che le università statunitensi distribuiscono all'inizio dell'anno accademico per aiutare gli iscritti a socializzare tra loro. Facebook, con oltre un miliardo di utenti, è indubbiamente il social più diffuso ed è proprio per questo che nell'applicazione abbiamo deciso di inserire la possibilità di effettuare il login attraverso questo social network. Gli utenti possono accedere al sito previa una registrazione gratuita, durante la quale vengono richiesti dati personali come nome, cognome, data di nascita e indirizzo email. Completata la registrazione, gli utenti possono creare un profilo personale, includere altri utenti nella propria rete sociale, aggiungendoli come amici, e scambiarsi messaggi, anche via chat, incluse le notifiche automatiche quando questi ultimi aggiornano i propri profili.
Inoltre le persone registrate possono fondare e unirsi a gruppi per condividere interessi in comune con altri utenti, organizzati secondo il luogo di lavoro, la scuola,

l'università o altre caratteristiche, condividere contenuti multimediali ed utilizzare varie applicazioni presenti sul sito. Per personalizzare il proprio profilo l'utente può caricare una foto, chiamata immagine del profilo, con la quale può rendersi riconoscibile. Può inoltre fornire ulteriori informazioni, come il comune di nascita e quello di residenza, la scuola frequentata, il proprio datore di lavoro, l'orientamento religioso e quello politico, la propria situazione sentimentale e molte altre.

#### *2.3.1.1 Le applicazioni Facebook*

All'interno di questo social network sono presenti una grande quantità di applicazioni che possono essere usate per condividere ciò che si sta leggendo, giocare e molto altro. Le applicazioni su Facebook, che vengono create da sviluppatori esterni, devono essere conformi alle Normative della Piattaforma Facebook. Anche il login con Facebook su un sito esterno o un app e il famoso like sono sue applicazioni. Effettuare il login con questo social network può essere vantaggioso sia da parte di chi accede, perché evita di compilare form di registrazione, sia da parte di chi gestisce il sito/app perché può accedere a tutta una serie di informazioni supplementari e preferenze dell'utente. Facebook mette a disposizione un SDK per ogni piattaforma (Android, iOS, PHP…) con tutte le funzioni necessarie per lo sviluppo.

Tutti i dati presenti all'interno di Facebook sono rappresentati come entità all'interno di un immenso grafo (open graph – figura 7). Ogni nodo è dotato di un identificativo univoco (es: 100000187279921 è il mio id) e l'insieme dei milioni e milioni di utenti e le loro informazioni formano il grafo completo.

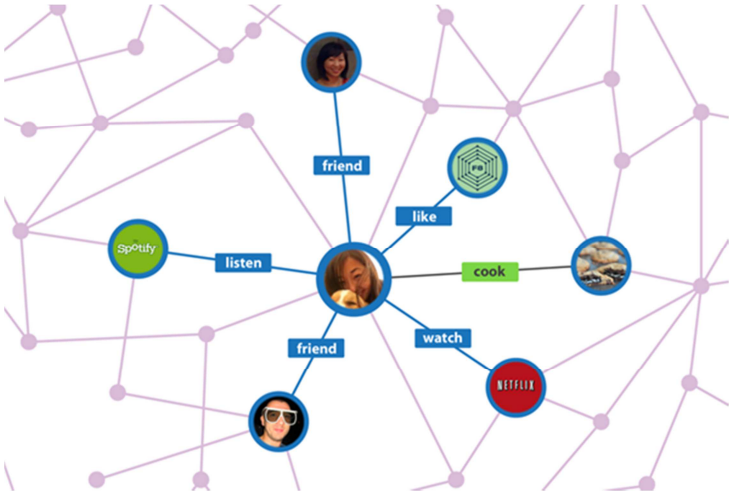
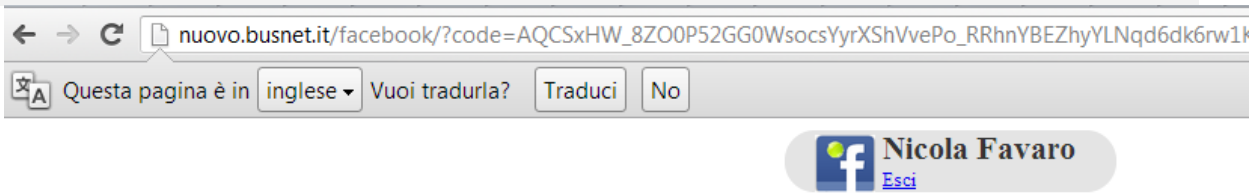



**Figura 7 - Open Graph**

L'accesso a queste informazioni è possibile grazie alla Graph API (figura 8): una collezione di web services che utilizzano JSON come formato di scambio.

*Es:* https://graph.facebook.com/100000187279921




```
Array
(
    [id] => 100000187279921
    [name] => Nicola Favaro
    [first_name] => Nicola
    [last_name] => Favaro
    [link] => https://www.facebook.com/Nicolaf.91
    [username] => Nicolaf.91
    [hometown] => Array
        (
            [id] => 110539155637396
            [name] => Padua, Italy
        )

    [location] => Array
        (
            [id] => 106070729423398
            [name] => Torreglia
        )

    [favorite_teams] => Array
        (
            [0] => Array
                (
                    [id] => 161021483931281
                    [name] => Troppo arrapato di Juventus
                )
```

**Figura 8 - Graph API con login**

Un problema non indifferente quando si parla di dati personali è quello della privacy. Infatti provando a scrivere l'URL qui sopra nel browser con il proprio id non si otterranno tutte le informazioni ma solo quelle pubbliche (figura 9). Nel mio caso:

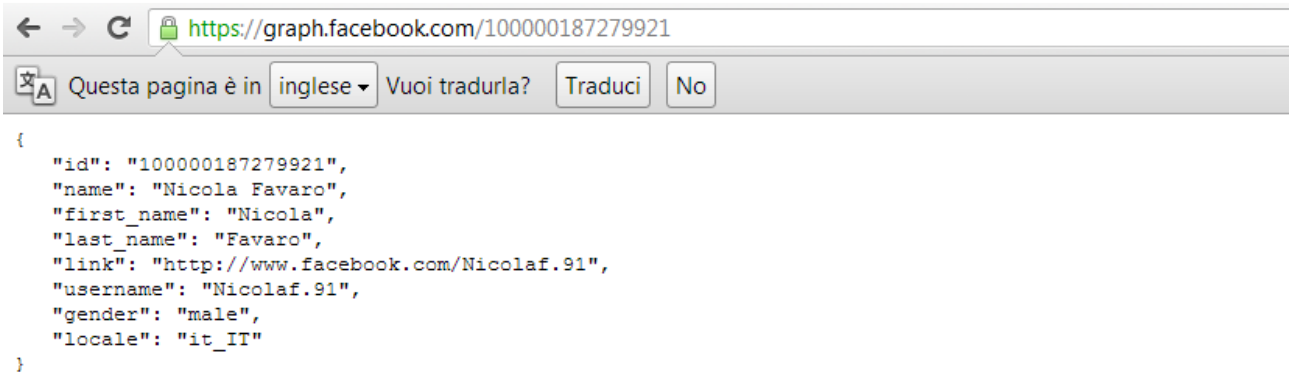



Figura 9 - Graph API senza login

Per avere altre informazioni l'applicazione dovrà richedere dei permessi all'utente attraverso il protocollo OAuth 2.0 (figura 10) il cui utilizzo è facilitato dall'SDK di Facebook.


Figura 10 - Richiesta dei permessi con protocollo OAuth 2.0.

## 2.4 Geolocalizzazione

Come già accennato l'applicazione è location-based ovvero basata sulla geolocalizzazione in quanto propone i viaggi delle agenzie che sono nei dintorni dell'utente.

Analizziamo ora come sia possibile recuperare la posizione dell'utente, conoscere la distanza cliente-agenzia ed avere altre informazioni relative a luoghi geografici.

Per quanto riguarda la posizione dell'utente, come tutti sappiamo, nei nostri dispositivi c'è un GPS (Global Positioning System). Altri strumenti che si possono utilizzare per la localizzazione sono wifi e 3g (network-based). Si tratta di meccanismi di localizzazione

meno accurati, ma comunque affidabili quando è sufficiente conoscere l'area dove si trova l'utente, e non si ha bisogno di calcolare la sua posizione precisa. Android offre un'interfaccia di programmazione che si astrae dal meccanismo di localizzazione utilizzato dal dispositivo. Tale interfaccia è offerta sotto forma di servizio di sistema:

*LocationManagerlocationManager=(LocationManager)getSystemService(LOCATION_SERVICE);*

Ora resta da vedere come sia possibile recuperare la posizione fornita da uno dei provider previsti. Le misurazioni vengono rappresentate mediante oggetti di tipo android.location.Location. che fornisce metodi come *public double getLatitude()* e *public double getLongitude().*

Conoscere le coordinate della nostra posizione ovviamente non ci basta. E' necessario conoscere anche l'indirizzo in cui ci troviamo e magari quanto lontani siamo dall'agenzia viaggi. Per tutto ciò ci vengono in aiuto le API di Google. Per dare un nome alle nostre coordinate ci torna molto utile l'API geocode. Vediamone un esempio:

http://maps.googleapis.com/maps/api/geocode/xml?address=45.2138792,11.770042&sensor=true

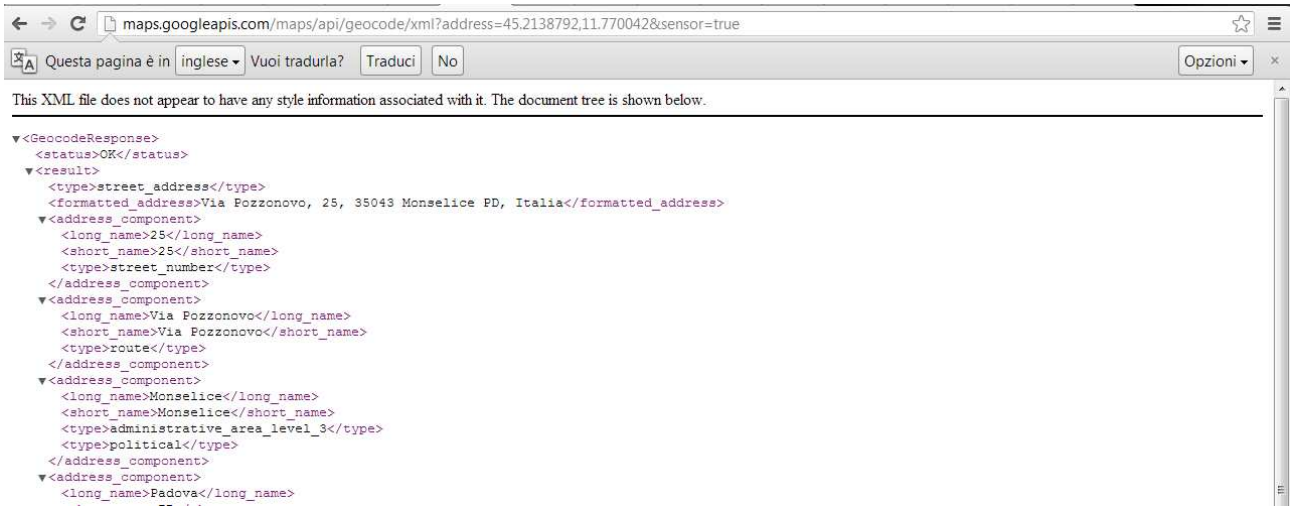



**Figura 11 - API geocode**

Scrivendo sul browser l'URL con le nostre coordinate come parametro riceveremo tutta una serie di informazioni in formato XML che potremo estrarre e utilizzare a nostro piacimento.

Un'altra API che Google ci offre è "direction" che dato un indirizzo di partenza e una destinazione ci restituisce la strada che li collega, la distanza ecc.. Vediamone un esempio:

http://maps.googleapis.com/maps/api/directions/xml?origin=viale%20della%20navigazione%20padova%20PD&destination=torreglia%20pd&sensor=false


**Figura 1 - API directions**

Anche qui attraverso l'analisi del documento è possibile ricavare tutte le informazioni necessarie.

Queste non sono gli unici servizi messi a disposizione da Google come Google non è l'unico a fornire questi servizi. Diretti concorrenti di Google sono Bing e Yahoo. La scelta di utilizzare Google è stata guidata dal fatto che i sistemi Android usino le mappe di Google.

# CAPITOLO 3 - IL PROGETTO

Dopo aver discusso un po' degli strumenti utilizzati, iniziamo a vedere nello specifico il progetto.

## 3.1 L'ambiente di sviluppo

Per sviluppare applicazioni in grado di girare su sistemi Android, è necessario installare sul proprio PC un apposito kit di sviluppo (SDK), che sia completo di emulatore, librerie e documentazione. E' gratuito indipendentemente dal sistema operativo che si usa per lo sviluppo.

Benché Android SDK disponga di script che automatizzino l'installazione delle applicazioni, il lancio dell'emulatore e il debug del codice, lavorare in un ambiente integrato, con ogni opzione a portata di clic, è sicuramente più facile.

Come IDE per lo sviluppo ho scelto Eclipse che è un ambiente di sviluppo integrato multi-linguaggio, multi-piattaforma e open source. Oltre a supportare molto bene il linguaggio Java offre anche un tools per lo sviluppo Android chiamato ADT (Android Development Tools).

### *3.1.1 AVD*

Il kit di sviluppo comprende un emulatore che consente di testare le applicazioni direttamente da PC, prima di installarle su un reale dispositivo Android. AVD sta per Android Virtual Device: dispositivo virtuale Android. Nel nostro PC possiamo creare e configurare quanti dispositivi virtuali vogliamo. È come avere tanti differenti smartphone da utilizzare per i propri test, solo che invece di dispositivi reali si tratta di macchine virtuali, fatte cioè di puro software, da eseguire attraverso l'emulatore. In questo modo è anche possibile avviare contemporaneamente sullo stesso PC due o più dispositivi virtuali. ADT offre un'interfaccia grafica anche per la gestione di questi dispositivi. Una cosa interessante è che gli emulatori sono completamente personalizzabili a seconda delle esigenze: ad esempio è possibile testare la propria applicazione utilizzando varie versioni di Android, o diverse dimensioni dello schermo. In questo modo si può essere sicuri che l'applicazione sarà compatibile con tutti i dispositivi.

La compatibilità quando si parla di sistemi operativi Android non è affatto trascurabile, infatti applicazioni che funzionano benissimo sulla versione 2.2 potrebbero non funzionare sulla versione 4.2.

## 3.2 APK

Le applicazioni Android sono distribuite sotto forma di file APK (Android Package). Al loro interno vengono raccolti gli eseguibili in formato DEX, le eventuali risorse associate e una serie di descrittori (come il manifesto che vedremo tra poco) che delineano il contenuto del pacchetto.

Una volta che il lavoro è stato completato, è possibile esportare il file APK da distribuire nel sistema Android. Per fare ciò, però, è necessario apporre su di esso una firma digitale. In caso contrario, Android non potrà installare l'applicazione. Questo è l'unico vincolo imposto dal sistema. Perché i nostri software siano autorizzati non è necessario pagare, possiamo fare tutto grazie all'aiuto di Eclipse.

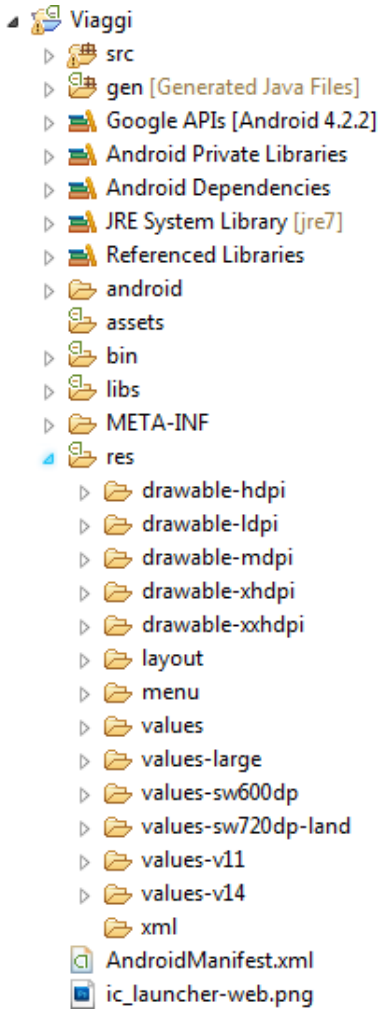



Figura 2 - Struttura progetto

## 3.3 Componenti fondamentali di un progetto Android

Un progetto Android ha una struttura complessa (figura 13).

***AndroidMainifest.xml***

Si tratta di un file XML presente in tutti i progetti Android. Al suo interno è necessario dichiarare i componenti del software. Eclipse, all'atto di creazione del progetto, esegue su di esso alcune configurazioni iniziali. Contiene le proprietà fondamentali del progetto, la descrizione delle activity (nome, label, se è quella che deve essere lanciata per prima ecc.), services, broadcast receiver, il riferimento al logo dell'app e così via.

***Resource***

All'interno dei progetti Android è sempre presente una cartella di nome *res* dotata di una particolare struttura formata dalle tre sotto-directory drawable, layout e values. La prima, drawable, serve per le immagini utilizzate dal software, mentre layout e values ospitano dei speciali file XML utili per definire in

maniera dichiarativa il layout dell'applicazione e i valori utilizzati al suo interno.
Partendo dalla cartella values, al suo interno vengono definiti dei file XML che contengono coppie chiave-valore che potranno essere utilizzate nel progetto. Esistono più tipi di dato associati ad ogni valore: stringhe, colori, misure e dimensioni, array di stringhe e interi e stili. Nella cartella layout invece sono presenti i layout delle activity in formato XML già accennati in precedenza. Per richiamare una risorsa da XML si usa la notazione @tipo/nome .
Per richiamare le risorse da Java invece torna utile il file della cartella /gen R.java: un file auto-generato da Eclipse, abbreviazione di resources. Invocando questa classe, infatti, è possibile richiamare via codice le risorse memorizzate sotto la directory res. Il file R da solo non basta ed è necessaria anche la classe android.content.res.Resources. Un esempio:

```
Resources res = getResources();
String appName = res.getString(R.string.app_name);
```

## 3.4 Database

Vediamo adesso tutte le informazioni che abbiamo deciso di memorizzare nel database.

Agenzia(identificativo, nome, via, CAP, città, provincia, latitudine, longitudine, cellulare, mail ,descrizione)
Viaggio(identificativo, **agenzia**, destinazione, descrizione, continente, data, prezzo, partecipazione, categoria)
Utente(identificativo, nome, cognome, mail, password, città, dataNascita, telefono, isFacebook )
Programma(**viaggio**, giorno, programma, titolo)
VotiViaggi(**utente, viaggio**, voto)
VotiAgenzia(**utente, viaggio**, voto)

I campi sottolineati stanno ad indicare che si tratta di una chiave primaria mentre quelli con in grassetto sono chiavi esterne.

# CAPITOLO 4 APP. VIAGGI

Vediamo ora nel dettaglio tutte le varie Activity dell'app.

## 4.1 Login Activity

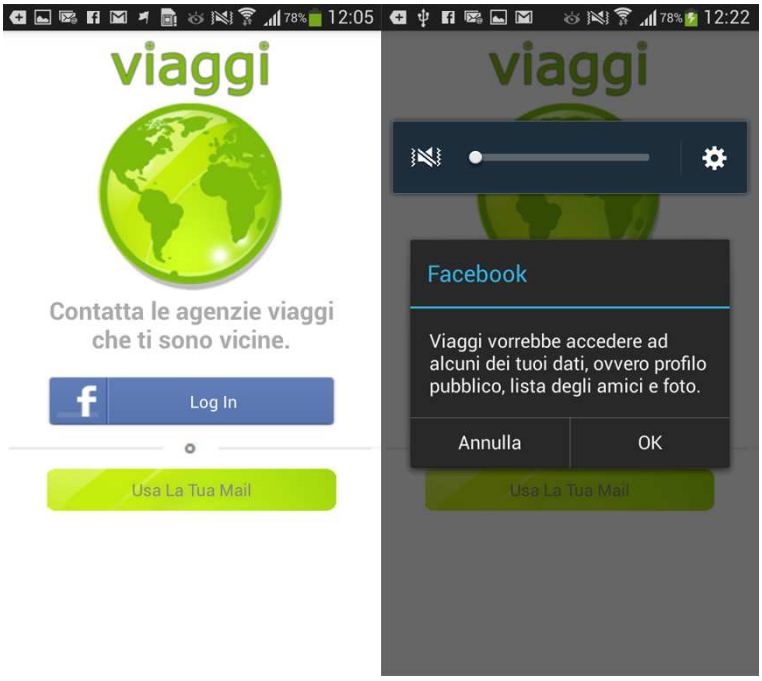



**Figura 3 - Activity per il login: l'applicazione Facebook chiede i permessi.**

In questa activity si effettua il login, necessario per procedere con l'utilizzo dell'applicazione. Come già accennato si può fare il login utilizzando un qualsiasi account di posta o per mezzo di Facebook. Nel caso di login con mail si verrà indirizzati in una seconda attività dove inserire le informazioni. Nel caso di utente non registrato sarà necessario compilare un apposito form di registrazione che utilizzando i web services comunicherà con il database e inserirà l'utente. La parte di login con mail è ancora in fase di sviluppo.

Per quanto riguarda Facebook, per prima cosa, è stato necessario realizzare un'applicazione di Facebook dalla sezione developers del social network. Ogni applicazione è identificata da un id che fa riferimento all'applicazione stessa. Nel file strings.xml ho definito una coppia chiave-valore come segue contenente l'id:

```
<string name="app_id" translatable="false">679083872118329</string>
```

Nel manifesto ho dovuto assegnare all'SDK di Facebook l'identificativo dell'applicazione:

```
<meta-data
    android:name="com.facebook.sdk.ApplicationId"
    android:value="@string/app_id" />
```

Da questo momento "Viaggi" può utilizzare l'applicazione Facebook destinata al login.

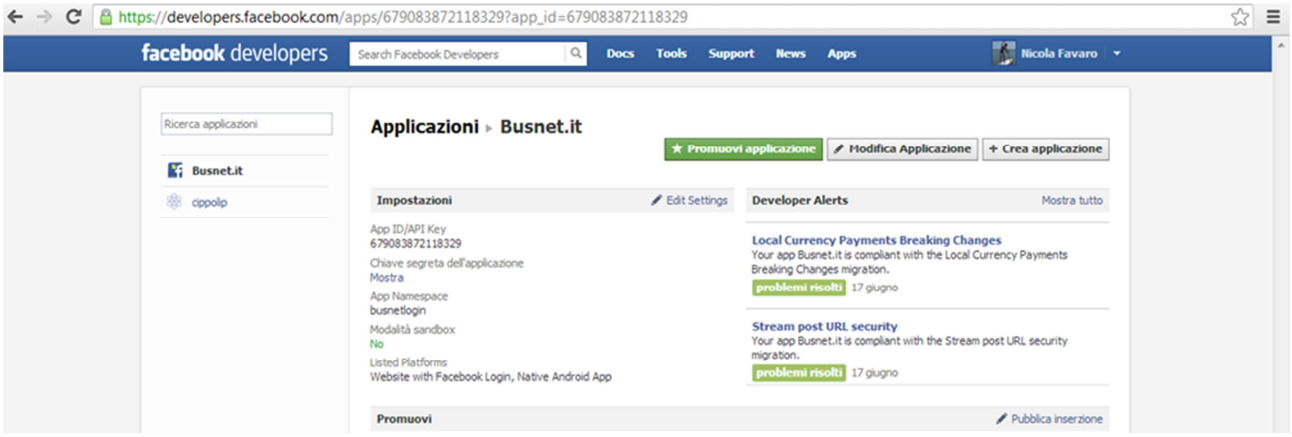



**Figura 15 - L'applicazione Facebook di Viaggi.**

L'SDK di Facebook offre numerose funzionalità tra cui il widget per il login. Per utilizzarlo è necessario aggiungere il seguente namespace nel file XML del layout dell'activity:

```
xmlns:facebook="http://schemas.android.com/apk/res-auto"
```

A questo punto posso utilizzare il mio bottone per l'accesso all'applicazione:

```
<com.facebook.widget.LoginButton
    android:id="@+id/authButton"
    android:layout_width="275dp"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
    android:layout_marginTop="20dp" />
```

Al click del pulsante l'SDK si occupa automaticamente di aprire una nuova sessione. Quello di cui mi sono dovuto occupare io è la richiesta alla Graph API, per accedere alle informazioni dell'utente, e il cambio di activity una volta effettuato il login. Prima ho parlato degli eventi e dei metodi di callback, è necessario gestire in qualche modo l'evento di apertura della sessione che, come già detto, si verifica automaticamente al click del pulsante. Per questo torna molto utile la classe com.facebook.Session.StatusCallback(), sempre di facebook SDK, che al cambio di stato

della sessione esegue automaticamente il metodo call() che può essere sovrascritto in base alle esigenze.

```
67          LoginButton authButton = (LoginButton) findViewById(R.id.authButton);
68
69
70          authButton.setReadPermissions(Arrays
71                  .asList("basic_info", "user_photos"));
72          // session state call back event
73
74          authButton.setSessionStatusCallback(new Session.StatusCallback() {
75
76              @Override
77              public void call(Session session, SessionState state,
78                      Exception exception) {
79
80                  if (session.isOpened()) {
81
82                      Request.executeMeRequestAsync(session,
83                              new Request.GraphUserCallback() {
84                                  @Override
85                                  public void onCompleted(GraphUser user,
86                                          Response response) {
87                                      if (user != null) {
88
89                                          utente.setId(user.getId());
90                                          utente.setCognome(user.getFirstName());
91                                          utente.setDataNascita(user
92                                                  .getBirthday());
93                                          utente.setNome(user.getLastName());
94                                          utente.setMail(user.asMap()
95                                                  .get("email").toString());
96                                          Log.i("utente", utente.toString());
97                                          Log.i("utente", user.toString());
98                                          SharedPreferences pref = getApplicationContext()
99                                                  .getSharedPreferences("MyPref",
100                                                         0); // 0 - for private
101                                                             // mode
102                                         Editor editor = pref.edit();
103                                         editor.putString("id", utente.getId());
104                                         editor.commit();
105
106                                         intent = new Intent(
107                                                 getApplicationContext(),
108                                                 Welcome.class);
109                                         startActivity(intent);
110                                         finish();
111                                         return;
112                                     }
113                                 }
114                             });
115                 }
116
117             }
118         });
119
120     }
```

Esaminiamo il frammento di codice. Alla riga 67 si vede come richiamare l'oggetto LoginButton dall'XML per poi poterlo gestire da Java, successivamente tramite il metodo setReadPermission() si va a indicare quali sono i permessi Facebook richiesti all'utente. In questo caso le informazioni di base e le foto. Tramite il metodo setSessionStatusCallback() vengono definite le azioni da intraprendere al momento dell'apertura della sessione. All'interno del metodo astratto call() ho fatto una richiesta alla Graph API, già menzionata prima, che ritorna le informazioni dell'utente all'interno dell'oggetto user. Queste informazioni vengono inviate utilizzando il metodo HTTP POST ad un web service che controlla se l'utente esiste già. In caso contrario

viene registrato. A questo punto grazie agli intent viene avviata l'attività Welcome.java, che contiene la lista dei viaggi, e l'attività corrente termina.

### 4.2 Lista Viaggi

Questa attività contiene la lista dei viaggi nei dintorni dell'utente, quindi per prima cosa è necessario ricavare la posizione. Ho creato una classe chiamata GPSTracker che si occupa di restituire la posizione dell'utente indipendentemente dal meccanismo disponibile per ricavarla( GPS/ wifi/ 3g). Se risulta impossibile ricavare la posizione un alert si occuperà di guidare l'utente verso il menù del sistema operativo per attivare le funzioni di geolocalizzazione. Questa classe utilizza i metodi e le funzioni della classe LocationManager descritta brevemente in precedenza.

```
47      public Location getLocation() {
48          try {
49              locationManager = (LocationManager) mContext
50                      .getSystemService(LOCATION_SERVICE);
51
52              // getting GPS status
53              isGPSEnabled = locationManager
54                      .isProviderEnabled(LocationManager.GPS_PROVIDER);
55
56              // getting network status
57              isNetworkEnabled = locationManager
58                      .isProviderEnabled(LocationManager.NETWORK_PROVIDER);
59
60              if (!isGPSEnabled && !isNetworkEnabled) {
61                  // no network provider is enabled
62              } else {
63                  this.canGetLocation = true;
64                  // First get location from Network Provider
65                  if (isNetworkEnabled) {
66                      locationManager.requestLocationUpdates(
67                              LocationManager.NETWORK_PROVIDER,
68                              MIN_TIME_BW_UPDATES,
69                              MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
70                      Log.d("Network", "Network");
71                      if (locationManager != null) {
72                          location = locationManager
73                                  .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
74                          if (location != null) {
75                              latitude = location.getLatitude();
76                              longitude = location.getLongitude();
77                          }
78                      }
79                  }
```

```
80                // if GPS Enabled get lat/long using GPS Services
81                if (isGPSEnabled) {
82                    if (location == null) {
83                        locationManager.requestLocationUpdates(
84                                LocationManager.GPS_PROVIDER,
85                                MIN_TIME_BW_UPDATES,
86                                MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
87                        Log.d("GPS Enabled", "GPS Enabled");
88                        if (locationManager != null) {
89                            location = locationManager
90                                    .getLastKnownLocation(LocationManager.GPS_PROVIDER);
91                            if (location != null) {
92                                latitude = location.getLatitude();
93                                longitude = location.getLongitude();
94                            }
95                        }
96                    }
97                }
98            }
99
100       } catch (Exception e) {
101           e.printStackTrace();
102       }
103
104       return location;
105   }
```

Vediamo il metodo getLocation(), il più importante di questa classe. Dopo aver ottenuto l'oggetto locationManager effettuo un controllo su quale provider è in grado di fornirmi il servizio con il metodo isProviderEnabled(TipoProvider). Se il provider è in grado di restituire la posizione il metodo ritorna true e in questo caso la vado a recuperare la posizione con il metodo requestLocationUpdate().

Una volta ottenuta la mia posizione posso effettuare la richiesta al web service per ottenere la lista viaggi.

Grazie ai metodi dell'HTTP, GET e POST, posso passare al servizio web la mia posizione. In questo caso è sufficiente utilizzare il metodo GET. Quindi la richiesta avrà una forma di questo tipo.

```
http://www.busnet.it/appviaggi/?lat=45.408950687771345&lon=11.926710605621352&distanza=30
```

Si noti come dopo l'URL del web service siano presenti 3 parametri che sono quelli necessari a effettuare la query. Il parametro distanza sta ad indicare di prendere i viaggi delle agenzie che si trovano in un raggio di 30 km dalla mia posizione. Ovviamente questo dato può essere modificato dall'utente utilizzando l'interfaccia dell'app. Provando a scrivere l'URL soprastante nel browser si otterrà un insieme di informazioni sui viaggi in formato JSON. Toccherà all'activity della lista dei viaggi effettuare l'analisi di queste informazioni per poi visualizzarle correttamente nell'applicazione.

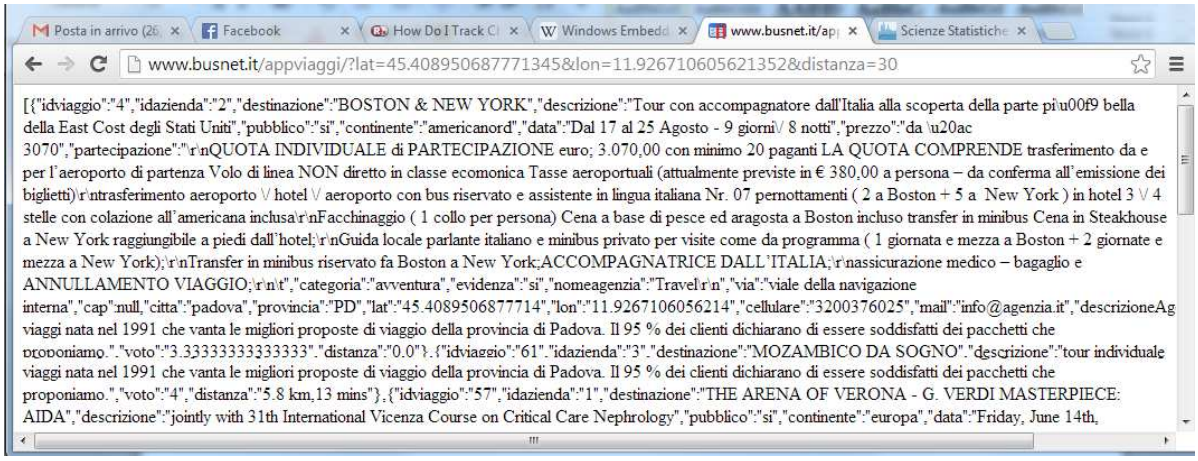



**Figura 16 - Web service con le informazioni sui viaggi.**

Vediamo ora come ottenere i viaggi delle agenzie più vicine. In precedenza abbiamo parlato delle API di Google ed in particolar modo dell'API direction (la distanza non viene offerta in linea d'aria ma in termini di strade). Un primo tentativo è stato quello di utilizzare proprio quest'ultima.

http://maps.googleapis.com/maps/api/directions/xml?origin=viale%20della%20navigazione%20padova%20PD&destination=torreglia%20pd&sensor=false

Scrivendo i due indirizzi o le coordinate, Google offre tutta una serie di informazioni tra cui la distanza tra i due punti. Testando però l'applicazione mi sono accorto che in questo modo ottenevo un sensibile decremento delle prestazioni. Ragionando un po', se nel database ci fossero 50 agenzie sarebbe necessario richiamare 50 volte il servizio direction, quindi effettuare 50 richieste HTTP.

Per questo ho optato per un calcolo in linea d'aria della distanza, ottenibile con la seguente funzione php.

```
47  function getDistance($latitude1, $longitude1, $latitude2,
48  $longitude2) {
49    $earth_radius  = 3960.00;
50    $delta_lat = $latitude2 - $latitude1 ;
51    $delta_lon = $longitude2 - $longitude1 ;
52    $alpha     = $delta_lat/2 ;
53    $beta      = $delta_lon/2 ;
54
55    $a         = sin(deg2rad($alpha)) * sin(deg2rad($alpha)) +
56  cos(deg2rad($latitude1)) * cos(deg2rad($latitude2)) * sin(deg2rad($beta)) *
57  sin(deg2rad($beta)) ;
58    $c         = asin(min(1, sqrt($a)));
59    $distance = 2*$earth_radius * $c;
60    $distance = round($distance, 4);
61    $distance = $distance*1.609344;
62    return $distance;
63
64  }
```

Sostanzialmente il web service va a prendere i dati di tutte le agenzie grazie alla seguente query:

```
126   $query=mysql_query("SELECT idazienda,lat,lon FROM agenzie ");
```

che calcola la distanza, in linea d'aria, dalla posizione dell'utente; se la distanza è minore di quella impostata viene effettuata una nuova query che va a prendere i viaggi dell'agenzia. Solo per le agenzie entro il raggio si utilizza l'API direction per avere informazioni più precise come ad esempio la strada e il tempo per raggiungerle. I viaggi che dovranno essere ritornati all'applicazione (per scelta 10 per volta) sono contenuti in un array associativo. Come già detto il formato di ritorno è JSON. Per convertire i dati nel formato richiesto Php fornisce la funzione json_encode().

```
188   print(strip_tags(json_encode($output)));
```

Una volta ricevuti i viaggi il controllo torna all'app Viaggi che avrà il compito di visualizzarli. I viaggi sono un insieme di informazioni dello stesso tipo visualizzabili come una lista. L'SDK di Android offre una view che fa proprio al caso nostro: ListView. Ogni elemento della lista sarà formato da un'immagine, la destinazione, una descrizione, la categoria, la distanza e il rating. Una lista Android ha bisogno di un Adapter che ho chiamato ViaggiAdapter: si tratta si una classe che estende ArrayAdapter e che si occupa di popolare la lista.

```
169         adapter = new ViaggiAdapter(this, R.layout.viaggiolista, viaggi);
170         headerText = (TextView) findViewById(R.id.raggio);
171         lista.setAdapter(adapter);
172
```

Alla riga 169 viene creato l'oggetto adapter che ha 3 parametri interessanti:

- *this* : è un'oggetto che consente di far riferimento all'activity stessa. In realtà il parametro richiesto dalla classe ViaggiAdapter è di tipo Context ovvero una classe che consente di far riferimento a Activity e Service. Le classi Activity e Service estendono la classe astratta Context;
- *R.layout.viaggiolista* : si tratta dell'identificativo del file XML contenente il layout di un singolo elemento della lista. R è la classe java auto-generata accennata in precedenza;
- *viaggi* : si tratta di una struttura dati LinkedList del package java.util. che contiene tutte le informazioni dei viaggi.

Questo Adapter deve poi essere assegnato alla listaView che ho chiamato “lista”; come si vede alla riga 171.
Ogni volta che il contenuto dell’oggetto viaggi viene modificato in seguito alla chiamata del web service sarà possibile aggiornare il contenuto della ListView.

```
407             adapter.notifyDataSetChanged();
```

Ad una ListView è possibile aggiungere anche un’intestazione (header) e un footer.
Durante l’implementazione di questa activity mi sono trovato di fronte ad un problema di compatibilità tra le varie versioni di Android.; infatti da Honeycomb, ovvero la versione 3 del sistema operativo, in poi non è non è possibile effettuare chiamate HTTP dal thread principale e quindi dall’activity stessa ma è necessario utilizzare dei task separati che vengono eseguiti concorrentemente a quello principale. Io ho utilizzato *android.os.AsyncTask*: una classe astratta che prevede l’override di 3 metodi:

- onPreExecute(): operazioni da eseguire prima dell’esecuzione del task. Io l’ho utilizzato solo per inizializzare mostrare  a video un dialogo di caricamento chiamata ProgressDialog. Questo punto l’ho trovato indispensabile in quanto la richiesta richiede del tempo e l’utente deve essere a conoscenza che l’app sta effettivamente scaricando i dati;

```
285         @Override
286         protected void onPreExecute() {
287             super.onPreExecute();
288 
289             mDialog = new ProgressDialog(Welcome.this);
290             mDialog.setMessage("Please wait...");
291             try {
292                 mDialog.show();
293             } catch (Exception e) {
294 
295             }
296 
297         }
```

- doInBackground(): il task vero e proprio dove ho effettuato le richieste ai web services e effettuato il parsing dei dati in formato JSON per poi inserirli nell’oggetto LinkedList citato in precedenza. Questo è uno dei metodi richiamati all’interno di doInBackground() che si occupa di effettuare la richiesta al servizio web e di salvarne i contenuti;

```
41      private static final String KEY_121 = "http://busnet.it/appviaggi/index.php";
42
43      public static InputStream Connection(String lat, String lon, String raggio) {
44          InputStream is = null;
45          try {
46
47              HttpClient httpclient = new DefaultHttpClient();
48              HttpGet httpget = new HttpGet(KEY_121
49
50              + "?lat=" + lat + "&lon=" + lon + "&distanza=" + raggio
51
52              );
53              Log.i("url", KEY_121
54
55                      + "?lat=" + lat + "&lon=" + lon + "&distanza=" + raggio);
56              HttpResponse response = httpclient.execute(httpget);
57              HttpEntity entity = response.getEntity();
58              is = entity.getContent();
59
60          } catch (Exception e) {
61              Log.e("log_tag", "Error in http connection " + e.toString());
62
63          }
64          return is;
65      }
```

- onPostExecute(): operazioni da eseguire successivamente all'esecuzione del task. In questo metodo ho chiuso il dialogo di caricamento e aggiornato il contenuto della LinkedList che poi l'adapter si occuperà di visualizzare.

```
403             if(viaggi!=null){
404                 viaggi.clear();
405             }
406             viaggi.addAll(viaggi2);
407             lista.setOnItemClickListener(clickListener);
408             adapter.notifyDataSetChanged();
409             if (mDialog != null)
410                 if (mDialog.isShowing()) {
411                     mDialog.dismiss();
412                     mDialog = null;
413
414                 }
415             return;
416         }
```

I dati della lista non devono essere aggiornati solo quando si avvia questa attività ma anche quando cambiano alcune impostazioni che l'utente può modificare come ad esempio il raggio. Per far ciò tornano molto utili le shared preferences accennate prima, infatti è possibile settare un listener in modo tale che ogni volta che cambiano le shared preferences vengano eseguite determinate istruzioni. Nel nostro caso un aggiornamento della lista.

Per prima cosa ho dovuto implementare l'interfaccia OnSharedPreferenceChangeListener()

```
54 public class Welcome extends Activity implements OnClickListener,
55         OnSharedPreferenceChangeListener {
```

e fare l'override del metodo OnSharedPreferenceChanged() che è quello che viene eseguito ogni volta che le shared preferences cambiano. Le istruzioni che stanno all'interno vanno ad eseguire nuovamente l'AsyncTask descritto prima.

```
488    @Override
489    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences,
490            String key) {
491        if(!NetworkServices.networkStatus(this)){
492            return;
493        }
494        if (key.equalsIgnoreCase("mylat") || key.equalsIgnoreCase("mylon"))
495            return;
496
497        new Header().execute(this);
498        new Connect().execute(this);
499        return;
500
```

Vediamo ora come si presenta graficamente questa activity (figura 17). Partendo dall'alto si nota subito la foto profilo di Facebook, che successivamente vedremo come ottenere, e un menù. Sotto c'è l'header dove ho messo la posizione dell'utente il raggio e la possibilità di cambiare il raggio. Ancora più in basso la lista viaggi con tutte le informazioni.

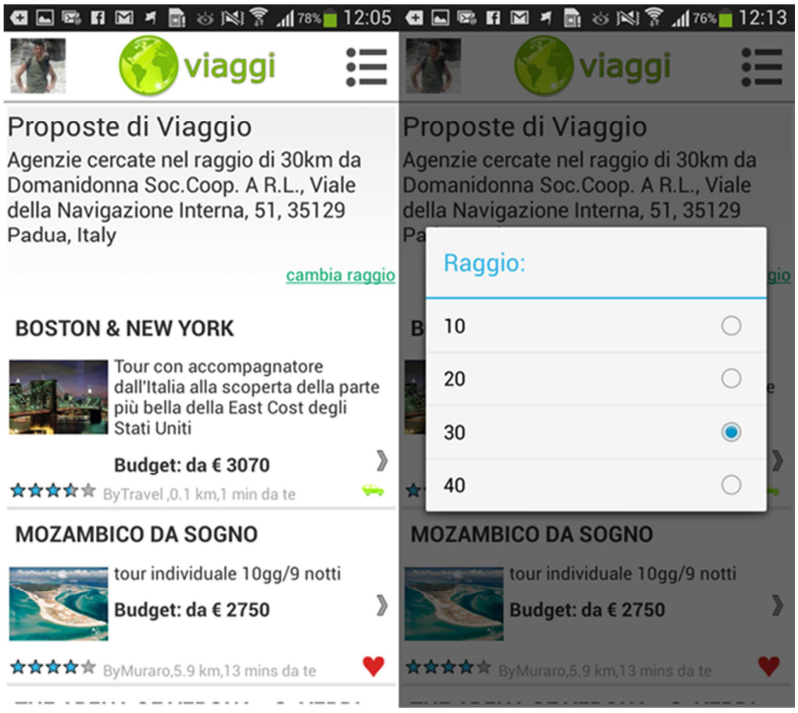



**Figura 17 - Activity lista viaggi e funzione cambia raggio.**

## 4.3 Preferences

Le preferences, che si differenziano dalle shared preferences, costituiscono una schermata in stile impostazioni Android. Per arrivarci dall'applicazione basta cliccare sul menù in alto a destra. Come impostazioni ho previsto un cambio di posizione, se per qualche motivo non voglio la mia posizione attuale, un cambio di raggio, una scelta di categorie di viaggio e per ultimo il pulsante indietro (figura 18).

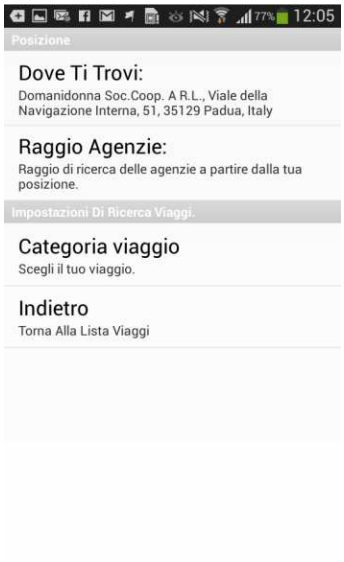



**Figura 18 – Preferences**

Partiamo dalla parte più interessante: il cambio posizione. Ho deciso di supportare l'utente nella scrittura del luogo in modo da evitare errori. Ecco quindi che entrano in gioco le API di auto-completamento di Google Places. Esse in base al testo già inserito forniscono un insieme di luoghi. Per poterle utilizzare è necessario registrarsi come sviluppatore all'interno della console delle API di Google, scegliere il servizio da utilizzare (ovviamente nel nostro caso sarà Google Places Autocomplete) e ottenere l'API-key; senza questa non si avrà accesso a Places.

Anche qui c'è bisogno di una classe che estende ArrrayAdapter, che si occupa di effettuare l'ennesima richiesta ai web services di Google, e di effettuare il parsing della risposta. Nell'URL di richiesta ci sarà anche la key come parametro passato con GET, altrimenti il valore ritornato sarà null.

Questo ovviamente non basta. Nell' SDK di Android esiste un widget che si chiama AutoCompleteTextView al quale è possibile assegnare l'adapter appena citato utilizzando il metodo setAdapter.

Dal file XML:

```
20
21      <AutoCompleteTextView
22          android:id="@+id/citta"
23          android:layout_width="fill_parent"
24          android:layout_height="wrap_content"
25          android:layout_margin="10dp"
26          android:maxLines="1"
27          android:selectAllOnFocus="true"
28          android:singleLine="true" />
29
```

Dall'activity:

```
35          final AutoCompleteTextView autoCompView = (AutoCompleteTextView) findViewById(R.id.citta);
36          autoCompView.setAdapter(new PlacesAutoCompleteAdapter(this,
37                  R.layout.list_item_viaggi));
```

All'interno di questa activity ho previsto la possibilità di riutilizzare anche la posizione del GPS (figura 19).


**Figura 19 - Cambio posizione.**

Le altre cose che si possono modificare all'interno delle preferences sono raggio e categorie di viaggio; analizziamo queste ultime.

Una delle caratteristiche, come già detto, dell'applicazione è il fatto di categorizzare gli utenti in base a quello che vogliono fare durante il viaggio non in base a dove vogliono andare. Le categorie che ci sono sembrate più consone quelle riportate in figura 2.

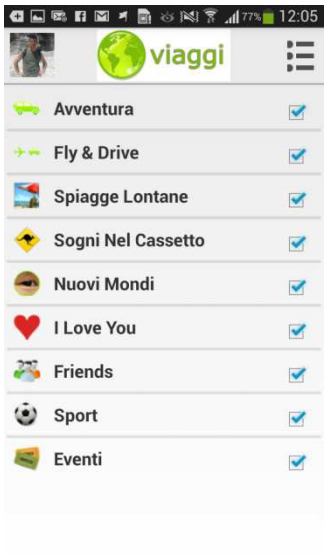



**Figura 20 - Categorie di viaggio.**

Da notare che anche questa è una ListView.

Man mano che si modificano le preferences vengono modificate anche le shared preferences utilizzando l'oggetto Editor.

```
121             SharedPreferences pref = getApplicationContext()
122                     .getSharedPreferences("MyPref", 0);
123             Editor editor = pref.edit();
124             editor.putString("indirizzo", citta);
125             editor.putString("mylat", coord.get(0));
126             editor.putString("mylon", coord.get(1));
127             editor.commit();
```

Per rendere effettive le modifiche è necessario utilizzare il metodo commit() dell'Editor. Ho deciso di eseguirlo all'interno del metodo on destroy() dell'activity preferences per far in modo che alla chiusura della schermata delle preferences l'adapter della lista viaggi si accorga che le Shared Preferences sono cambiate e aggiorni quindi il contenuto della lista.

```
65      @Override
66      public void onDestroy(){
67          super.onDestroy();
68          if(!NetworkServices.networkStatus(this)){
69              finish();
70              return;
71          }
72          editor.commit();
73      }
74
```

### 4.4 Scheda di un viaggio

Cliccando su uno dei viaggi si passa all'activity dei dettagli del viaggio. Avrete di sicuro notato che nella parte alta è presente la foto profilo di Facebook; per ottenerla ci viene in aiuto ancora una volta l'SDK di Facebook che mette a disposizione il seguente widget:

```
<com.facebook.widget.ProfilePictureView
    android:id="@+id/selection_profile_pic"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_marginLeft="5dip"
    android:layout_marginRight="5dip"
    facebook:preset_size="small" />
```

Dall'activity è invece necessario impostare lo user id di Facebook al widget. In breve recupero la view utilizzando nuovamente il file R.java e attraverso il metodo setProfileId(id) setto l'identificativo dell'utente che ha effettuato l'accesso.

```
103         ProfilePictureView picture = (ProfilePictureView) findViewById(R.id.selection_profile_pic);
104         if (iduser != null) {
105 
106             picture.setCropped(true);
107             picture.setProfileId(pref.getString("id", -1 + ""));
108         }
```

Per ottenere tutte le informazioni riguardanti un viaggio è stato necessario recuperare l'id del viaggio cliccato: questo è possibile farlo grazie al passaggio parametri tra activities. In precedenza abbiamo parlato degli Intent che sono oggetti per avviare una nuova attività, essi consentono anche di passare dei parametri usando il metodo putExtra(parametro). Quello qui sotto è il listener di quando si clicca su un viaggio della lista.

```
390             OnItemClickListener clickListener = new OnItemClickListener() {
391 
392                 @Override
393                 public void onItemClick(AdapterView<?> adapter, View view,
394                         int position, long id) {
395                     Intent intent = new Intent(getApplicationContext(),
396                             Viaggio.class);
397                     intent.putExtra("id", ((InfoViaggio) adapter
398                             .getItemAtPosition(position)).getId());
399 
400                     startActivity(intent);
401                 }
402             };
```

Mentre nell'activity appena avviata è possibile recuperare gli extra con questa semplice istruzione.

```
idViaggio = this.getIntent().getStringExtra("id");
```

Recuperato l'identificativo del viaggio adesso è possibile effettuare una richiesta al web service passandogli, con il metodo HTTP GET, l'id del viaggio. L'URL avrà una forma di questo tipo:

http://busnet.it/appviaggi/?filtra=si&id=4

Anche qui una volta ricevuta la risposta è necessario effettuarne il parsing per poter poi maneggiare a piacimento i dati. A questo scopo ho scelto di utilizzare StringBuilder, che è una stringa le cui dimensioni crescono in base al contenuto e nel nostro caso conterrà tutte le info sul viaggio, e un oggetto JsonArray, che è un'oggetto costruito a partire da una stringa in formato JSON da cui è possibile ottenere un JsonObject formato da coppie chiave-valore e utilizzabile come un array associativo.
Ad esempio qui ottengo l'elemento 0 del Jsonarray che è un JsonObject:

```
345                 JSONArray ja;
346                 JSONObject jo;
347                 ja = DbServices.getAll(is);
348                 jo = ja.getJSONObject(0);
```

Per ricavare il contenuto dell'istanza jo posso utilizzare i metodi getTipo("chiave"):

```
368                 // getting all other information
369                 iv.setDestinazione(jo.getString("destinazione"));
370                 iv.setData(jo.getString("data"));
371                 iv.setDescrizione(jo.getString("descrizione"));
372                 iv.setPrezzo(jo.getString("prezzo"));
373                 iv.setVoto(Float.parseFloat(jo.getString("voto")));
374                 iv.setPartecipazione(jo.getString("partecipazione"));
375                 iv.setVotanti(jo.getString("votanti"));
```

Le informazioni saranno poi il contenuto dell'activity e andranno inserite nei vari widget in questo modo:

```
((TextView) findViewById(R.id.destinazione)).setText(iv
        .getDestinazione());
((TextView) findViewById(R.id.durata)).setText(iv.getData());

((TextView) findViewById(R.id.descrizione)).setText(iv
        .getDescrizione());
```

Questa è la grafica dell'activity:

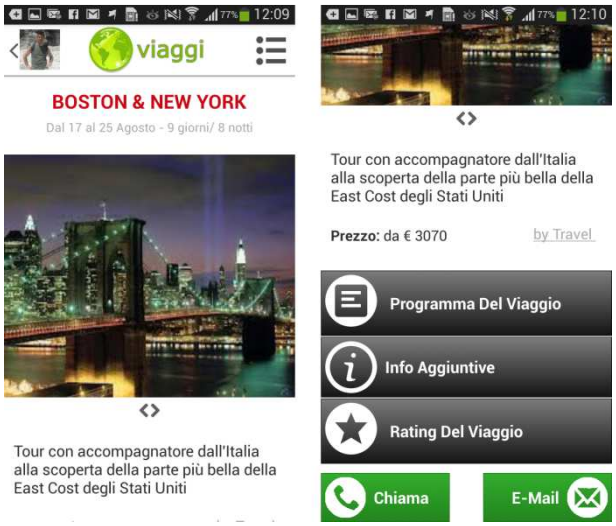



**Figura 21 - Scheda di un viaggio.**

Un aspetto dell'applicazione su cui vorrei soffermarmi è la galleria. Per iniziare vediamo come recuperare una foto. Per scelta di progettazione le immagini non sono nel database ma si trovano nella cartella /fotoviaggi del server. Ogni foto ha come nome l'identificativo del viaggio al quale appartiene (es: 30.jpg), nel caso un viaggio ne abbia più di una all'id viene aggiunto un underscore e un numero progressivo (es: 30_2.jpg). Il metodo loadBitmap(url) si occupa di effettuare una richiesta HTTP al server e di recuperare l'immagine. L'SDK di Android offre la classe Bitmap del package android.graphics. che consente di gestire le immagini. Vediamo un esempio di URL :

http://busnet.it/appviaggi/fotoviaggi/4.jpg

Mentre la classe in questione è la seguente:

```
136     static public Bitmap loadBitmap(String url) {
137         try {
138             HttpGet httpRequest = null;
139             URL bitmapUrl = new URL(url);
140 
141             httpRequest = new HttpGet(bitmapUrl.toURI());
142 
143             HttpClient httpclient = new DefaultHttpClient();
144             HttpResponse response = (HttpResponse) httpclient
145                     .execute(httpRequest);
146 
147             HttpEntity entity = response.getEntity();
148             BufferedHttpEntity bufHttpEntity = new BufferedHttpEntity(entity);
149             InputStream instream = bufHttpEntity.getContent();
150             return BitmapFactory.decodeStream(instream);
151         } catch (URISyntaxException e) {
152             e.printStackTrace();
153             return null;
154         } catch (IOException e) {
155             e.printStackTrace();
156             return null;
157         }
158     }
```

Recuperare le foto è solo il primo passo per implementare una galleria scorrevole; ci occorre anche il widget Gallery.

```
<Gallery
    android:id="@+id/gallery"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="1dp"
    android:layout_marginTop="10dp" />
```

Da Java, dopo aver recuperato le immagini e averle salvate in un ArrayList, è necessario definire una classe che estenda BaseAdapter che si occupa di gestire lo scroll e di decidere quale immagine visualizzare. Vediamone un pezzo:

```
857     public class ImageAdapter extends BaseAdapter {
858         int mGalleryItemBackground;
859         private Context mContext;
860 
861         public ImageAdapter(Context c) {
862             mContext = c;
863             TypedArray a = obtainStyledAttributes(R.styleable.HelloGallery);
864             mGalleryItemBackground = a.getResourceId(
865                     R.styleable.HelloGallery_android_galleryItemBackground, 0);
866             a.recycle();
867         }
868 
869         public int getCount() {
870             return mImageIds.size();
871         }
872 
873         public Object getItem(int position) {
874             return position;
875         }
876 
877         public long getItemId(int position) {
878             return position;
879         }
880 
```

In questa activity si può anche contattare l'agenzia viaggi. E' possibile farlo tramite mail compilando un apposito form. Tutte le informazioni inserite andranno inviate al web service che si preoccupa dell'invio  vero e proprio. Nel form di inserimento (vedi figura 22) è necessario lasciare un messaggio (EditTextView), dire quando e come si vuole essere ricontattati (RadioGroup/RadioButton) e sceglire una data d'incontro (DatePicker).

Quest'ultimo widget, fornito dall'SDK di Android, consente di scegliere agevolmente una data.

```
<DatePicker
    android:id="@+id/date"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

Per recuperare il dato inserito da java invece :

```
((DatePicker) findViewById(R.id.date)).getYear(),
((DatePicker) findViewById(R.id.date)).getMonth(),
((DatePicker) findViewById(R.id.date)).getDayOfMonth())
.toString()));
```

Una volta compilati tutti i campi tramite l'HTTP post verranno inviate tutte le informazioni. La scelta di questo metodo è dovuta al fatto che i dati da spedire potrebbero indurre l'URL a superare la dimensione massima consentita. Per questa operazione mi è tornato utile l'oggetto ArrayList<NameValuePair>, dove NameValuePair è una coppia chiave valore.

```
743             postParameters = new ArrayList<NameValuePair>();
744             postParameters.add(new BasicNameValuePair("utente", pref.getString(
745                     "id", "errore")));
746             postParameters.add(new BasicNameValuePair("agenzia", iv
747                     .getIdAgenzia()));
748             postParameters.add(new BasicNameValuePair("messaggio", messaggio));
749             postParameters.add(new BasicNameValuePair("dataIncontro", new Date(
750                     ((DatePicker) findViewById(R.id.date)).getYear(),
751                     ((DatePicker) findViewById(R.id.date)).getMonth(),
752                     ((DatePicker) findViewById(R.id.date)).getDayOfMonth())
753                     .toString()));
754             postParameters.add(new BasicNameValuePair("disponibile",
755                     radioButtonSelected));
756             postParameters.add(new BasicNameValuePair("modoRicontatto",
757                     ricontatto));
```

L'arrayList postParameters contiene tutte le informazioni del form e, grazie al metodo setEntity(Url) della classe HttpPost, può essere allegato alla richiesta HTTP:

```
HttpPost httppost = new HttpPost(url);
httppost.setEntity(new UrlEncodedFormEntity(postParameters));
response = httpclient.execute(httppost);
HttpEntity entity = response.getEntity();
InputStream is = entity.getContent();
```

Il web service all'URL http://busnet.it/appviaggi/mailContatto.php si occuperà di ricevere le informazioni, di comporre la mail e di spedirla. Il metodo mail() di PHP consente di spedire e-mail. Vediamo un esempio:

```
$mittente = 'From: "Nicola Favaro" <nicolaf.91@hotmail.it> \r\n';
$destinatario = "agenzia@server.com";
$oggetto = "Richeista contatto";
$messaggio = "Contentuto";
mail($destinatario, $oggetto, $messaggio, $mittente);
```

A questo punto starà all'agenzia prendere contatto con il cliente interessato.

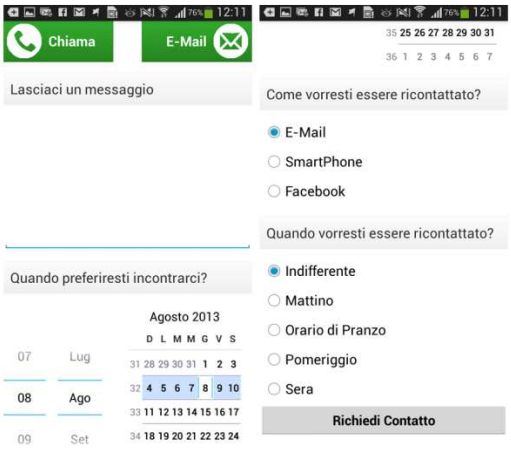



**Figura 22 - Form per l'invio mail.**

L'applicazione consente anche di telefonare all'agenzia e qui entra in gioco l'intent Intent.ACTION_CALL. Attraverso il metodo setData() è necessario aggiungere il numero di telefono da chiamare e il gioco è fatto.

```
Intent callIntent = new Intent(
        Intent.ACTION_CALL);
callIntent.setData(Uri.parse("tel:"
        + iv.getPhone()));
startActivity(callIntent);
```

Essendo questa un'operazione delicata e che comporta dei costi prima che parta la chiamata apparirà una sorta di popup, che si chiama Alert, che chiederà una conferma. E' possibile personalizzare l'alert usando i seguenti metodi:

- setTitle(titolo) : da un titolo alla finestra;
- setIcon(Icona):inserisce l'icona della finestra;
- setMessage(messaggio):setta il messaggio dell'alert;
- I metodi setPositiveButton e setNegativeButton vanno a definire le azioni da intraprendere in caso di conferma oppure di annullamento dell'operazione. Nel mio caso se l'utente conferma parte la chiamata, altrimenti solo l'alert con il metodo dismiss();

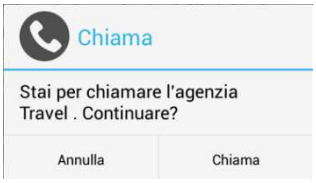



**Figura 23 - Alert per la telefonata.**

```
557                 AlertDialog.Builder alert = new AlertDialog.Builder(
558                         Viaggio.this);
559                 alert.setTitle("Chiama");
560                 alert.setIcon(R.drawable.phone);
561                 alert.setMessage("Stai per chiamare l'agenzia "
562                         + iv.getAgenzia().replace("\n", "")
563                         + ". Continuare?");
564                 alert.setPositiveButton((CharSequence) "Chiama",
565                         new DialogInterface.OnClickListener() {
566 
567                             @Override
568                             public void onClick(DialogInterface dialog,
569                                     int which) {
570 
571                                 try {
572                                     Intent callIntent = new Intent(
573                                             Intent.ACTION_CALL);
574                                     callIntent.setData(Uri.parse("tel:"
575                                             + iv.getPhone()));
576                                     startActivity(callIntent);
577                                 } catch (ActivityNotFoundException e) {
578                                     Log.e("helloandroid dialing example",
579                                             "Call failed", e);
580                                 }
581                             }
582                         });
583                 alert.setNegativeButton((CharSequence) "Annulla",
584                         new DialogInterface.OnClickListener() {
585 
586                             @Override
587                             public void onClick(DialogInterface dialog,
588                                     int which) {
589 
590                                 dialog.dismiss();
591 
592                             }
```

## 4.5 Agenzia viaggi

Dal viaggio è possibile accedere alla scheda dell'agenzia dove è possibile trovare tutta una serie di informazioni: nome, indirizzo, descrizione, mail, telefono, rating. Anche qua è possibile contattare l'agenzia, con le stesse modalità descritte nel paragrafo precedente, e esprimere un giudizio.


**Figura 24 - Scheda agenzia.**

Una cosa interessante di questa activity è la funzione di collegamento con Google Maps per vedere come raggiungere l'agenzia. Cliccando sul pin in alto si va ad avviare un'altra activity a cui vengono passate, sempre tramite il metodo putExtra(), le coordinate dell'agenzia. La nuova activity, che estende MapActivity, preleva (usando gli Intent) le coordinate dell'agenzia e, dalle Shared Preferences, la posizione dell'utente, per poi avviare l'applicazione Maps passandole il punto di partenza e la destinazione.

```
13 public class Mappa extends MapActivity {
14
15     Uri uri;
16
17     @Override
18     public void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         String lonAgenzia=this.getIntent().getStringExtra("lon");
21         String latAgenzia=this.getIntent().getStringExtra("lat");
22
23         SharedPreferences pref = getApplicationContext().getSharedPreferences(
24                 "MyPref", 0);
25         String lat = pref.getString("mylat", null);
26         String lon = pref.getString("mylon", null);
27         uri = Uri.parse("http://maps.google.com/maps?&saddr=" + lat + "," + lon
28                 + "&daddr=" + latAgenzia + "," + lonAgenzia);
29         Intent intent = new Intent(Intent.ACTION_VIEW, uri);
30         startActivity(intent);
31         finish();
32     }
33
34     @Override
35     protected boolean isRouteDisplayed() {
36         return false;
37     }
38
39 }
```

Un'altra funzione è la possibilità di esprimere un giudizio (figura 25). L'SDK di Android mette a disposizione un widget chiamato RatingBar che serve proprio in questi casi.

```
<RatingBar
    android:id="@+id/media"
    style="?android:attr/ratingBarStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_marginBottom="5dip"
    android:layout_marginLeft="10dp"
    android:layout_marginRight="5dp"
    android:layout_marginTop="17dp"
    android:isIndicator="true"
    android:numStars="5" />
```

Questo widget può essere gestito (con il linguaggio Java) utilizzando i metodi setRating() e getRating() che consentono il primo di impostare il voto ed il secondo di leggerlo. Una volta che l'utente ha votato vengono inviati id utente, id agenzia, e voto ad un web service che di occuperà di scrivere nel database queste informazioni. Nel caso

l'utente abbia già votato il suo voto viene solo aggiornato. Questa è la query per inserire un voto nella tabella votiAgenzie:

```
$query=mysql_query(" INSERT INTO voti VALUES (".$_GET['utente'].",".$_GET['id']."
,".$_GET['voto'].");");
```

Per calcolare, invece, la media dei voti basta utilizzare la seguente query:

```
$query4=mysql_query(" SELECT AVG(voto) as media ,COUNT(voto) as quanti FROM
 votiAgenzie as a,viaggi as v WHERE v.idviaggio='".$_GET['id']."' AND a.agenzia=v.idazienda");
```

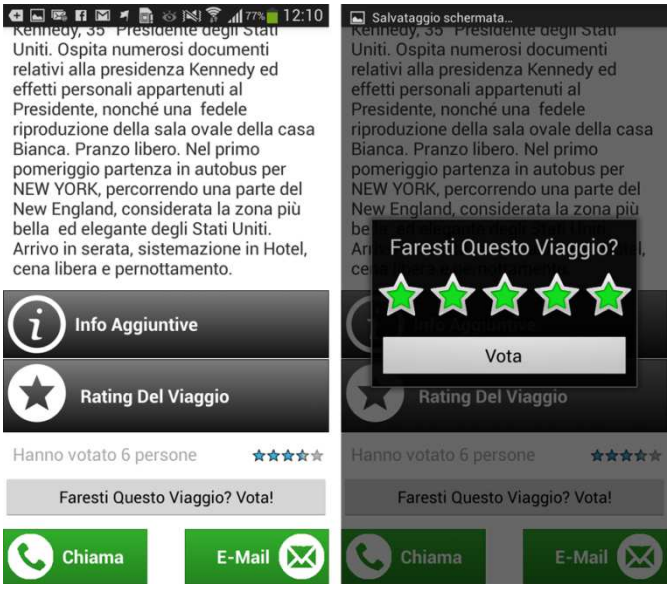



Figura 25 - Rating del viaggio.

## 4.6 I menù

Una cosa essenziale nelle applicazioni è il menù e quindi vediamo come si realizza nelle applicazioni Android. Consideriamo l'activity della lista viaggi. Per questa attività ho previsto 3 voci nel menù: logout, settings, aggiorna (figura 26).

Nella struttura di ogni progetto android, all'interno della cartella /res, c'è una directory /menu che contiene i file XML che descrivono la parte grafica dei vari menù.

```xml
<menu xmlns:android="http://schemas.android.com/apk/res/android" >

    <item
        android:id="@+id/action_settings"
        android:checkable="true"
        android:checked="false"
        android:icon="@android:drawable/ic_menu_preferences"
        android:menuCategory="alternative"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/action_settings"
        android:visible="true"/>
    <item
        android:id="@+id/refresh"
        android:checkable="true"
        android:checked="false"
        android:icon="@drawable/refresh"
        android:menuCategory="alternative"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/refresh"
        android:visible="true"/>
    <item
        android:id="@+id/logout"
        android:checkable="true"
        android:checked="false"
        android:icon="@drawable/facebook"
        android:menuCategory="alternative"
        android:orderInCategory="100"
        android:showAsAction="never"
        android:title="@string/logout"
        android:visible="true"/>

</menu>
```

Ogni Item corrisponde ad un elemento del menù. Ovviamente bisogna anche andare a definire le azioni da intraprendere al click e per questo c'è Java. Innanzitutto per rendere visibile il menù all'interno dell'attività bisogna sovrascrivere il metodo OnCreateOptionsMenu() nel quale è necessario specificare qual è il file XML contente la struttura del menù. Per le funzionalità se ne occupa il metodo onOptionsItemSelected() che in base all'item cliccato intraprende delle azioni:

- se viene premuto l'item delle impostazioni si avvia l'attività settings descritta prima. Questo usando gli Intent;
- se si clicca su logout si torna alla schermata iniziale;
- 'aggiorna' invece va a richiamare l'AsyncTask che ricarica la lista viaggi.

```
200     @Override
201     public boolean onCreateOptionsMenu(Menu menu) {
202         // Inflate the menu; this adds items to the action bar if it is present.
203         getMenuInflater().inflate(R.menu.welcome, menu);
204         return true;
205     }
206
207     @SuppressWarnings("unchecked")
208     public boolean onOptionsItemSelected(MenuItem item) {
209         // Handle item selection
210         Intent intent;
211         switch (item.getItemId()) {
212         case R.id.action_settings:
213             intent = new Intent(getApplicationContext(), Settings.class);
214             startActivity(intent);
215             return true;
216         case R.id.logout:
217
218             Session.getActiveSession().closeAndClearTokenInformation();
219             uiHelper.onDestroy();
220             Session.setActiveSession(null);
221
222             return true;
223         case R.id.refresh:
224             new Header().execute(this);
225             new Connect().execute(this);
226             return true;
227         default:
228             return true;
229         }
230     }
231
```

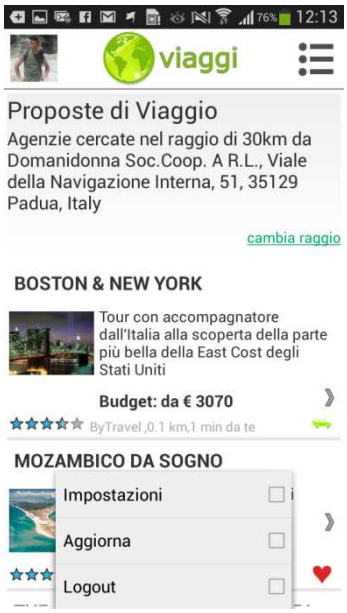



Figura 26 - Menù

## 4.7 Logout

Nel menù è possibile fare il logout dall'applicazione. L'oggetto Session dell'SDK di Facebook consente di gestire la sessione e quindi anche di terminarla. Per far ciò mi sono servito del metodo cloeAndClearTokenInformation(); in aggiunta è necessario svuotare le shared preferences con il metodo clear().

## 4.8 I permessi

Ogni app Android richiede dei permessi per essere installata che corrispondono alle risorse necessarie all'applicazione per poter funzionare. Per poter accedere allo stato della connessione, a internet, per poter chiamare dall'app, e per aver accesso al GPS sono necessari dei permessi che l'utente deve concedere altrimenti l'installazione sarà annullata. Questo è un modo per tutelare l'utente e per metterlo a conoscenza delle azioni che svolge l'applicazione.

I permessi vanno dichiarati nel AndroidManifest.xml altrimenti non sarà possibile accedere a quei servizi e si verificheranno degli errori runtime. Quelli richiesti da "Viaggi" sono i seguenti:

```
11      <uses-permission android:name="android.permission.INTERNET" />
12      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
13      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
14      <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES" />
15      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
16      <uses-permission android:name="android.permission.CALL_PHONE" />
```

## 4.9 Test

Durante le sviluppo dell'app mi sono trovato di fronte a numerose problematiche legate alla compatibilità tra sistemi operativi, dimensioni dello schermo differenti tra i vari dispositivi, attese troppo lunghe nel ricevere una risposta dai web services e nel caricare le immagini.

E' stato necessario effettuare vari test per verificare il corretto funzionamento di Viaggi in ogni situazione. Nelle prime fasi dello sviluppo, usando AVD e device diversi, mi sono accorto che l'applicazione funzionava correttamente nelle versioni di Android precedenti alla 3 mentre si arrestava in quelle più recenti. Dopo diverse ricerche nel web ho capito che dalla versione Honeycomb non è possibile effettuare richieste HTTP dal thread principale ed è necessario usare gli AsyncTask.

Testando l'applicazione, inoltre, mi sono accorto che i tempi di risposta erano troppo lunghi. Per provare a capirne il motivo ho sfruttato un metodo statico della classe java.lang.System che restituisce il timestamp corrente: sottraendo 2 timestamp si riesce a capire il tempo trascorso tra una sequenza di operazioni. Per visualizzare il tempo trascorso ho utilizzato i log che sono delle informazioni (errori e non solo) che l'Ide di Eclipse fornisce durante l'esecuzione.

```
304             long startTime = System.nanoTime();
305             InputStream is = DbServices.Connection(
306                     pref.getString("mylat", "not found"),
307                     pref.getString("mylon", "not found"),
308                     pref.getString("raggio", "not found"));
309             ja = DbServices.getAll(is);
310             if (ja==null){
311                 return null;
312             }
313             Log.i("info", ja.toString() + ja.length());
314             long endTime = System.nanoTime();
315             Log.i("database", (endTime - startTime) / 1000000000 + "");
```

Questo sistema l'ho utilizzato sia per conoscere il tempo di risposta dei web services sia per conoscere il tempo di caricamento delle immagini.

- Le immagini erano troppo grandi: sono stato costretto a ridimensionarle e a modificarne il formato;
- I web services, come già accennato in precedenza, effettuavano troppe richieste HTTP alle API di Google.

L'applicazione deve funzionare in qualsiasi, anche in quelle definite come "scorrette", come l'assenza di una connessione o di un servizio di geolocalizzazione. In questi casi sarà necessario avvisare l'utente del problema riscontrato e possibilmente guidarlo verso una soluzione. Nel caso "Viaggi" si trovi di fronte ad un'assenza di rete viene avvisato l'utente con un Toast, ovvero un messaggio in stile popup che resta in primo piano sullo schermo per qualche secondo, e si apre la scheda delle impostazioni di rete del sistema operativo.

```java
public class NetworkServices {

    static public boolean networkStatus(Context context) {
        ConnectivityManager connectivityManager = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo activeNetInfo = connectivityManager
                .getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
        NetworkInfo activeNetInfo2 = connectivityManager
                .getNetworkInfo(ConnectivityManager.TYPE_WIFI);
        boolean isConnected = activeNetInfo != null
                && activeNetInfo.isConnectedOrConnecting();
        if (!isConnected) {
            isConnected = activeNetInfo2 != null
                    && activeNetInfo2.isConnectedOrConnecting();
        }
        if (isConnected) {
            Log.i("NET", "connected " + isConnected);
            return true;
        } else {
            Log.i("NET", "not connected " + isConnected);

            Intent i = new Intent(Settings.ACTION_WIRELESS_SETTINGS);
            i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            context.startActivity(i);
            Toast.makeText(
                    context,
                    "Controlla la tua connessione per usare l'applicazione --Viaggi--",
                    Toast.LENGTH_LONG).show();
            return false;
        }
    }

}
```

Il metodo networkStatus() si occupa proprio di effettuare questo controllo: ritorna true se c'è connettività, false altrimenti. Il metodo controlla sia se c'è una connessione wifi sia se c'è una rete mobile disponibile.

Un procedimento analogo avviene anche in caso di assenza di un servizio di geolocalizzazione.

Inoltre per verificare l'usabilità dell'applicazione l'ho fatta provare a persone che non ne conoscevano struttura e funzionamento, per capire se riuscivano a muoversi bene al suo interno.

# CAPITOLO 5 - PARTE WEB

La parte web dell'applicazione è dedicata principalmente alle agenzie viaggi ed ha lo scopo di presentare l'app e di descrivere i vantaggi che possono trarre agenzie e turisti accedendo al servizio. In questa sezione le agenzie potranno registrarsi, scegliere il tipo di pacchetto (free, premium, full) ed accedere ad un pannello di controllo dove inserire e gestire tutti i loro viaggi che poi verranno visualizzati dall'app. Anche qui il login può essere fatto sia via Facebook che tramite una mail. Come già detto è stata realizzata in PHP.

Graficamente si presenta così:

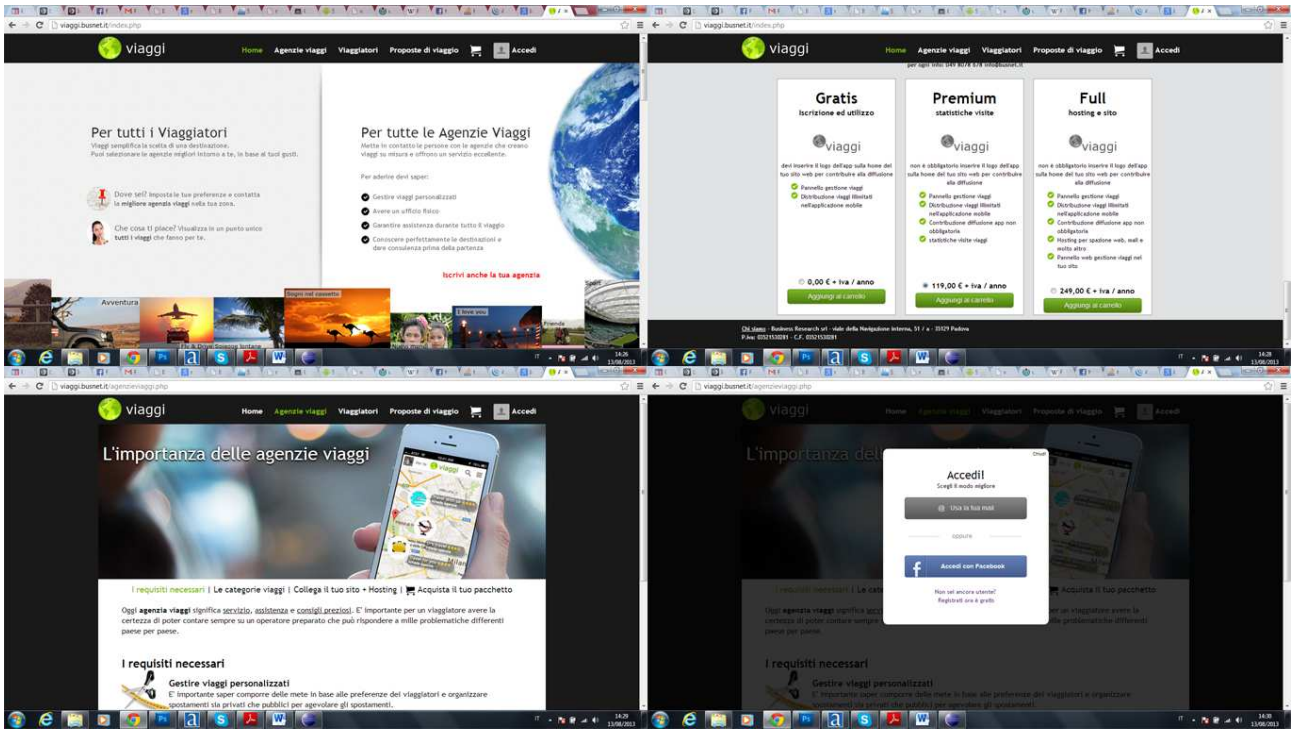



**Figura 27 - Applicazione web di Viaggi.**

Non approfondisco ulteriormente la parte web perché ritengo che l'argomento di interesse sia costituito dall'app smartphone.

# CAPITOLO 6 - LA DIFFUSIONE

Oggi ci sono moltissime applicazioni per Smartphone e la maggior parte non ha un grosso successo anche se effettivamente si presentano molto bene. Ritengo che uno degli aspetti fondamentali sia avere utenza e per questo, insieme al titolare dell'azienda, abbiamo deciso di rilasciare l'app in modalità gratuita per i clienti/turisti e di guadagnare con le agenzie. In precedenza abbiamo parlato di 3 tipi di pacchetti per le agenzie:

- free: le agenzie avranno accesso al pannello di inserimento viaggi ma dovranno contribuire alla diffusione dell'app inserendo un link per il download nel loro sito internet;
- Premium(119 euro/anno): le agenzie avranno accesso al pannello di inserimento, link all'applicazione non obbligatorio e accesso alle statistiche;
- Full: le agenzie avranno accesso al pannello di inserimento, link all'applicazione non obbligatorio, accesso alle statistiche, hosting e reimplementazione del form di inserimento dei viaggi all'interno del loro sito in modo che i viaggi vengano direttamente inseriti anche nel database dell'app.

Quando il numero di agenzie sarà consistente sarà possibile guadagnare anche attraverso il posizionamento all'interno della lista dei viaggi: se un'agenzia paga i suoi viaggi saranno in cima alla lista.

# CAPITOLO 7- CONCLUSIONI E SVILUPPI FUTURI

Nei 2 mesi di stage sono riuscito a realizzare un'applicazione location-based funzionante , implementando tutte le activity e la parte grafica.

Alcune funzioni, però, non sono riuscito a terminarle: l'autenticazione via mail, la visualizzazione per categoria e l'inserimento di alcuni criteri di ricerca dei viaggi, come prezzo e destinazione.

Potrebbe essere interessante, inoltre, introdurre una sorta di diario di bordo, compilabile durante il viaggio in modalità offline, dove è possibile dove è possibile inserire foto, commenti, pensieri e emozioni provate giorno per giorno durante la vacanza. Questo sicuramente resterà al turista come ricordo ma sarà poi condivisibile dall'app e consultabile da altri utenti.

Utilizzando le informazioni (come i like) fornite da Facebook è possibile proporre all'utente dei viaggi che potrebbero essere di suo interesse. Questi meccanismi sono già largamente utilizzati da Youtube e Google. Sicuramente implementare il login con qualche altro social network potrebbe essere un'aggiunta da prendere in considerazione.

Nel caso l'app Viaggi avesse successo, sarà d'obbligo distribuirla anche per iOs e Windows Phone in modo tale da renderla disponibile per tutti coloro in possesso di uno smartphone e che desiderino usufruire del servizio.

# BIBLIOGRAFIA


[1] Andreucci Giacomo, *Applicazioni iOS e Android con Google Maps*, Edizioni FAG, Milano, 2011;

[2] Carli Massimo, *Sviluppare applicazioni per Android*, Edizioni Apogeo, 2011;

[3] *Android Programming, Edizioni Master, 2011;*

[4] *Il marketing con Facebook*, di Dan Zarrella e Alison Zarrella, Tecniche Nuove Editore, Milano 2011;

[5] Emanuele Cisotti e Marco Giannino, *Android, Uso avanzato e personalizzazione,* Edizioni FAG Milano, 2012;

[6] Mark, Dave; LaMarche, Jeff. *Beginning iPhone 3 Development: Exploring the iPhone SDK*. Apress, 2009;

*[7]* Nick Gerakines, *Facebook Application Development;*

[8] Roberto Bruni, Andrea Corradini, Vincenzo Gervasi, *Programmazione in Java,* Apogeo, 2009*.*


# SITOGRAFIA


- http://developer.android.com/index.html Android developer;
- https://developers.facebook.com/docs/ Sezione developer di Facebook;
- https://developers.google.com/maps/?hl=it Sezione developer di Google.