UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria

Corso di Laurea Triennale in Ingegneria Informatica

TESI DI LAUREA

Integrazione con i servizi Google nella piattaforma Android

Integration with Google services in the Android platform

RELATORE: Carlo Fantozzi

LUREANDO: Steven Cadalt

A.A. 2010-2011

Indice

Introduzione

Negli ultimi anni si è vista un'espansione esponenziale delle soluzioni smartphone in circolazione. Ormai questi dispositivi riescono a raggiungere le caratteristiche hardware di un pc all'avanguardia di una decina di anni fa, ma concentrate nel palmo di una mano.
Per sua natura uno smartphone è orientato all'uso in mobilità, e quindi molte funzionalità, prima rivolte solo al mondo desktop, ora si sono rivolte anche al mondo mobile; è un esempio la possibilità di gestire la propria casella di posta elettronica in mobilità.
Inoltre funzionalità che fino a poco tempo fa erano relegate a dispositivi dedicati sono state inserite nel telefono cellulare; forse l'esempio più lampante è la navigazione satellitare che ora è prevista in tutti gli smartphone moderni, ma integrata con tutte le altre possibilità che fornisce un dispositivo di questo tipo.
In questo elaborato si illustrerà come funzionano il client di posta elettronica e il sistema di navigazione di Google all'interno di Android, uno dei sistemi operativi mobile più diffusi e in più rapida espansione. Verranno inoltre descritte le modalità di integrazione di queste due feature all'interno di una applicazione sviluppata dal laureando.
Verrà introdotta la piattaforma Android, la sua storia e le caratteristiche del sistema, per poi passare per una trattazione più dettagliata dei servizi che sono stati integrati. Al termine verrà trattato l'aspetto di implementazione delle nuove funzionalità all'interno dell'applicazione.

Capitolo 1

La piattaforma Android

Android non si compone solo di un sistema operativo, ma un vero e proprio ecosistema che comprende, oltre all'OS, un kit comprensivo di tutte le librerie necessarie per lo sviluppo delle applicazioni, e l'Android Market che permette la pubblicazione e la vendita delle applicazioni degli sviluppatori.

Android si basa su Linux. Una sua caratteristica importante è di essere open: Google infatti lo rende disponibile ai produttori di hardware con licenza Open Source Apache License 2.0, senza richiedere in cambio il pagamento di royalties.

1.1 Cenni Storici

Negli ultimi 10 anni si è vista un'espansione considerevole del mercato mobile, complici le nuove intuizioni e strategie di marketing delle aziende del settore, e la concorrenza delle stesse, con conseguente aumento della complessità hardware e software dei dispositivi mobili.

Android nasce in questo clima frizzante dalla mente di Andy Rubin (a sua volta co-fondatore di Danger, società leader in soluzioni software mobile), Rich Miner, Nick Sears e Chris White nel 2003, ma presto la startup viene acquistata da Google nel 2005 che ne fiuta le potenzialità.
Nel 2007 Google, insieme alle principali aziende del mondo della telefonia, fonda l' Open Handset Alliance (OHA) con l'obbiettivo di creare una piattaforma open in grado di essere competitiva nel mercato evitando il peso delle royalties che ne possono frenare lo sviluppo.
Sempre nel 2007 esce la prima versione del Software Development Kit (SDK) di Android, grazie alla quale gli sviluppatori hanno potuto iniziare a toccare con mano la piattaforma e a creare le prime applicazioni.
Verso la fine del 2008 esce il primo dispositivo con sistema operativo Android, l'HTC G1. Nello stesso periodo viene rilasciata anche la versione 1.0 dell'SDK.

HTC Dream (T-Mobile G1) - Il primo smartphone a montare Android

Nei mesi successivi vengono corretti bug e aggiunte delle feature e nell' ottobre 2009 si passa alla versione 2.0.
Il 5 gennaio 2010 viene messo in commercio il primo smartphone con il marchio Google, il Nexus One con versione di Android 2.1.

Durante il 2010 sono stati presentati diversi tablet che utilizzano la piattaforma Android, ma solo nel corso del 2011 viene presentata la versione 3.0, con pieno supporto per le dimensioni ampie di schermo tipiche dei tablet.

Attualmente per gli smartphone la versione più recente è la 2.3 Gingerbread e ormai può competere ad armi pari con iOS di Apple, suo principale concorrente.

Ad oggi la stragrande maggioranza dei maggiori produttori di smarphone hanno in listino device equipaggiati con OS Android. Stiamo parlando di società del calibro di HTC, Samsung, LG, Sony Ericsson.

Sono oltre 250.000 le applicazioni presenti nel Market Android e i download hanno toccato quota 6 miliardi. In un solo anno, le applicazioni sono più che raddoppiate.

Nel luglio 2011 Larry Page, CEO di Google, ha dichiarato che nel mondo vengono attivati 550'000 device Android ogni giorno.

1.2 Caratteristiche

E' nell'interesse di Google, per incentivare l'adozione della piattaforma, fornire agli sviluppatori tutti gli strumenti necessari per la creazione di applicazioni, anche di una certa complessità.
Android non utilizza un proprio linguaggio di programmazione: è stato deciso di adottare Java, linguaggio descritto dalle specifiche di Sun Microsystems del 1995. Con questa scelta lo sviluppatore si trova a maneggiare un linguaggio già noto e collaudato ed inoltre ha a disposizione quasi la totalità delle API di Java 5, permettendo di utilizzare le stesse librerie e classi utilizzate per le applicazioni desktop.

Come detto, Android è open, non può utilizzare quindi la VM (Virtual Machine) di Sun, in quanto l'utilizzo della stessa è soggetto al pagamento di royalties. Per questo motivo e per ottimizzare al massimo l'utilizzo delle risorse su dispositivi con hardware limitato quali i telefoni cellulari, Google ha adottato una propria VM che prende il nome di *Dalvik.*

Il codice Java viene per prima cosa processato, ottenendo dei file .class di bytecode, dopodiché il compilatore crea del codice Dalvik (file con estensione .dex) che viene eseguito dalla Virtual Machine proprietaria.

Anche in Android, come ormai per tutti i sistemi embedded mobile, l'interfaccia grafica (UI) viene definita con un approccio dichiarativo, tramite file XML che permettono di creare l'interfaccia utente in maniera indipendente dal codice e con l'aiuto di tool grafici.

1.3 Lo sviluppo su piattaforma Android

Per lo sviluppo Google mette a disposizione un plugin per l'IDE Eclipse (Integrated Development Environment – ambiente di sviluppo integrato) che consiste in una serie di strumenti per interfacciarsi in maniera semplice con l'SDK e fornisce inoltre un emulatore in grado di simulare il funzionamento delle applicazioni su qualsiasi dispositivo che supporti l'OS, quindi tenendo conto delle diverse dimensioni degli schermi e delle caratteristiche hardware peculiari del singolo smartphone o tablet.

Inoltre lo sviluppo è possibile su tutte le piattaforme tutt'oggi utilizzate, Eclipse è infatti disponibile per Windows, Linux e Mac OSX.

L'applicazione di cui parleremo, per la quale sono state aggiunte le feature di condivisione e di navigazione satellitare, è stata sviluppata per la piattaforma Android.

1.4 Il meccanismo degli Intent

L'*Intent* è uno dei concetti più interessanti introdotti da Google nella piattaforma Android; un Intent è un oggetto (derivato dalla classe omonima) che descrive un'azione da far effettuare ad un componente su un insieme di dati. E' quindi un meccanismo che, tramite lo scambio di oggetti Intent, permette di avviare altri componenti, siano essi della stessa applicazione oppure di applicazioni diverse. In diversi casi infatti un componente non conosce l'oggetto in grado di soddisfare la sua richiesta, in quanto questo dipenderà dall'insieme di elementi installati nel sistema. Ciascun elemento infatti, nell'AndroidManifest dell'applicazione della quale fa parte, avrà descritte le proprie "capacità" attraverso un Intent Filter.

Quindi il dispositivo, in base ad una serie di regole e analizzando gli Intent Filter degli elementi, quale di questi sarà necessario avviare in risposta a un dato oggetto Intent.

Gli Intent si differenziano in due categorie:

- Intent espliciti: sono Intent che avviano un componente già noto e presente nel sistema, rintracciandolo attraverso il nome a lui assegnato.
- Intent impliciti: questi Intent tentano di avviare un componente che abbia una serie di caratteristiche, senza conoscere a priori quale elemento specifico risponderà alle caratteristiche specificate.

Un oggetto Intent contiene le seguenti informazioni:

- ComponentName: è il nome del componente che dovrà essere avviato, e che lo identifica univocamente; si tratta quindi di un parametro usato solo per gli Intent espliciti.
- Action: è una stringa che rappresenta l'azione che si vuole far intraprendere al componente di destinazione. Alcune action più usate sono ACTION_MAIN (per avviare un activity identificata come activity principale) e ACTION_EDIT (un'azione di modifica dei dati passati con l'Intent).

Un elemento dichiara a sua volta le azioni che è in grado di gestire attraverso elementi action nell'AndroidManifest.

- Data: Indica l'indirizzo (URI) e il tipo di dato su cui agire. Il tipo di dato permette di inquadrare meglio l'azione, che altrimenti risulterebbe generica. Per esempio l'azione ACTION_EDIT permette di modificate testo o immagini; se però si indentifica una tipologia di dati con il metodo setType("text") allora ne aumentiamo la descrizione e quindi questo andrà a modificare le priorità degli elementi candidati in seguito all'Intent Resolution, che darà priorità alle azioni che agiscono su del testo.

- Category: permette di suggerire nell'Intent le activity, in base al compito che svolgono all'interno dell'applicazione. Vi sono diversi tipi di categorie; per esempio l'Intent lanciato dalla pressione del tasto fisico Home del dispositivo ha categoria CATEGORY_LAUNCHER. Quindi tutte le applicazioni candidate a poter svolgere l'azione di launcher dovranno specificare all'interno dell' AndroidManifest che supportano questo tipo di categoria e offrono quindi quel tipo di servizio.

- Extras: sono coppie chiave-valore che l'activiy di destinazione può estrarre con il metodo getExtras(). Ad esempio nell'Intent per creare una email andremo a definire con un extra i destinatari che a sua volta verranno recuperati dall'applicazione che in quel momento ha il compito di gestire le email.

- Flags: sono altre impostazioni di vario tipo per caratterizzare l'avvio di activiy; per esempio FLAG_ACTIVITY_SINGLE_TOP permette di non avviare una nuova istanza dell'activity se già creata, ma di riciclare quella già presente.

In fase di Intent Resolution vengono analizzati soltanto i campi Action, Data e Category, mentre Flags e Extras non influenzano la ricerca. Se l'Intent Filter di un elemento coincide con le caratteristiche definite dall'Intent allora l'elemento entra nella lista dei candidati a soddisfare l'azione.

Ora lanciamo per esempio un activity tramite un Intent esplicito:

```
Intent myIntent = new Intent (this, destinationActivity.class);
```

Dove viene indicato esplicitamente il nome dell'activity che dovrà essere avviata.

Se invece vogliamo definire un Intent implicito:

```
Intent helpIntent= new Intent();

helpIntent.setAction(android.intent.action.ACTION_EDIT);

helpIntent.putExtras(message,userText);

helpIntent.setType(text/plain);

startActivity(helpIntent);
```

In questo caso il component destinazione dovrà avere la capacità di editare del testo.

Capitolo 2

Google Gmail

Gmail è un servizio di posta elettronica gratuito gestito da Google.
Il rilascio è avvenuto nel 2004, in uno stato di beta pubblica, dalla quale è uscito solo 5 anni dopo, nel 2009.

Caratteristica importante, oltre alla dimensione di archiviazione disponibile elevata (Gmail al lancio prevedeva 1GB di storage, rivoluzionario per l'epoca, ora prevede 7,5GB, in continua crescita), è la profonda integrazione con gli altri numerosi servizi dell'azienda; E' presente la possibilità di utilizzare la messaggistica istantanea di Google Talk, ed è presente una profonda integrazione anche con Google Calendar, Picasa, Google Maps, Google Docs e Google Reader.
E' proprio grazie questo aspetto, insieme alle prestazioni e alla sua gratuità, che il servizio di posta elettronica di Google è uno tra i più usati al mondo.

2.1 Gmail per Android

Gmail è sostanzialmente una webmail, realizzata il AJAX, ma disponibile anche in versione HTML, meno avida di risorse.
Oltre alla webmail, sono disponibili versioni software di Gmail per le principali piattaforme mobile. In particolare è presente l'applicazione ufficiale Android sviluppata da Google per gestire i messaggi di posta elettronica.
L'applicazione permette di avere su smarthone quasi la totalità delle funzionalità che presenta la versione web; oltre infatti al semplice invio e ricezione di email è presente la visualizzazione dei messaggi in thread (come una sequenza di botta e risposta, come accade in una sessione di Instant Messaging), una gestione completa delle etichette, della posta prioritaria, dello spam.
Inoltre è possibile gestire più account di posta elettronica Gmail.

Android permette alle applicazioni di interagire tra di loro, come già visto, tramite il meccanismo degli Intent. Un Intent potrà essere lanciato in seguito a un operazione eseguita da parte dell'utente o da un evento esterno.
Per creare una email si dovrà creare un Intent implicito all'interno del codice dell'applicazione:

```
Intent emailIntent = new Intent {andoid.content.Intent.ACTION_SEND};
```

Questo tipo di chiamata in realtà non permette solo di creare una email utilizzando l'applicazione di default di Android, bensì è data facoltà all'utente di scegliere quale applicazione per la gestione della posta elettronica utilizzate, proponendogli una lista di applicazioni tra quelle installate che offrono quel tipo di servizio.

A questo punto è possibile definire già a livello di codice il contenuto degli spazi della email; è possibile infatti utilizzare dei metodi della classe Intent sull'oggetto appena creato,

ad esempio per definire i destinatari, l'oggetto della email, il corpo del messaggio e gli eventuali allegati:

```
emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, aEmailList);

emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,"This is the
subject");

emailIntent.putExtra(android.content.Intent.EXTRA_TEXT,"This is the body");

emailIntent.putExtra(Intent.EXTRA_STREAM, resourceURI);
```

Come destinatari viene passato un array di stringhe contenente tutti gli indirizzi email, mentre per allegare una risorsa è necessario passare come parametro l' URI della stessa, che la identifica univocamente all'interno della memoria del dispositivo.
Alla fine viene lanciata l'activity tramite l'Intent creato e il controllo passa dalla nostra applicazione all'activity dell' applicazione che si è scelta per gestire i messaggi di posta elettronica.

```
startActivity(emailIntent);
```

Capitolo 3

Google Maps Navigation

Google Maps Navigation è un servizio di navigazione gestito da Google. Attualmente è disponibile per gli smartphone Android dalla versione 2.0.

3.1 Google Maps

Google Maps Navigation è parte di un progetto più esteso, Google Maps, che gestisce la cartografia stradale e le immagini satellitari di buona parte del territorio mondiale.

Il servizio di navigazione utilizza infatti i dati forniti da Maps.

Google ha inglobato nel 2004 la società Where 2 Technologies e ha utilizzato le conoscenze acquisite per creare un applicazione di cartografia web. Il servizio permette la consultazione delle mappe stradali di buona parte del globo terrestre ed inoltre fornisce immagini aeree e satellitari con un elevato dettaglio. Il suo utilizzo è gratuito per l'uso non commerciale.

La versione web è realizzata facendo uso intensivo di Javascript e permette, oltre alla semplice consultazione delle mappe stradali e satellitari, anche tutta una serie di funzionalità associate: è possibile effettuare il calcolo di percorsi, specificando se ci si vuole spostare in auto, a piedi oppure utilizzando i mezzi di trasporto pubblici. Inoltre Maps è fortemente integrato con gli altri servizi di Google in particolare con il motore di ricerca. E' possibile ricercare punti di interesse come gli uffici pubblici, alberghi, scuole, ristoranti, ecc..
A seguito di numerosi tentativi di reverse-engineering, dovuti alla qualità del servizio, Google ha rilasciato delle API per consentire agli sviluppatori di integrare nativamente Google Maps all'interno dei propri siti web.

3.2 L'applicazione per Android

La web-application Google Maps è un ottimo strumento per organizzare i propri spostamenti e calcolare i tragitti migliori tenendo presenti tutta una serie di variabili come il tipo di mezzo utilizzato, il traffico, l'orario di partenza, la possibilità o meno di evitare autostrade o pedaggi.
Ma uno strumento di questo tipo risulta estremamente più utile se utilizzato in mobilità, affiancandosi alla possibilità di interfacciamento con dell'harware che fornisca le nostre coordinate GPS in ogni momento.
Google ha così deciso di rilasciare nel mercato il primo navigatore satellitare gratuito dove le mappe non risiedono sul device, ma vengono scaricate mano a mano che si procede nel percorso.
Google Maps Navigation ha tutte le funzionalità di un normale programma di navigazione; oltre a ricevere indicazioni curva dopo curva è possibile ricercare punti di interesse, e avere informazioni sul traffico. E' inoltre possibile utilizzare le foto di Street View scattate da

Google ormai in moltissimi paesi del mondo, proponendo al guidatore non una riproduzione della strada che sta percorrendo in un certo istante, bensì una foto reale scattata in precedenza con in primo piano le indicazioni sul percorso da seguire.
In realtà l'applicazione di navigazione è parte dell'applicazione Android Google Maps; infatti da quest'ultima è possibile impostare il tragitto che dobbiamo percorrere, analogamente a come avviene per la versione web-based, ma al termine viene lanciata l'applicazione di navigazione vera e propria.

Navigazione Street View

Come per spedire una email, anche per iniziare la navigazione si può creare un Intent implicito all'interno del codice della propria applicazione:

```
Intent myIntent = new Intent(Intent.ACTION_VIEW, CoordinatesURI);
```

Dove come secondo parametro viene passato un URI che al suo interno contiene le coordinate del punto da raggiungere.

```
startActivity(myIntent);
```

Alla fine viene lanciata l'activity dell'Intent che è stato creato e il controllo passa a Google Maps: l'utente, una volta definite ulteriori variabili come il tipo di mezzo di trasporto da utilizzare, può decidere di visualizzare le indicazioni stradali e successivamente di passare all'applicazione di navigazione vera e propria Google Maps Navigation.
E' da sottolineare che essendo l'Intent di tipo implicito, al momento del lancio dell'activity verranno proposte all'utente tutte le applicazioni che offrono quel tipo di servizio; è

possibile per esempio lanciare il browser web e visualizzare il percorso stradale tramite la versione mobile della web-application di Google Maps.

Capitolo 4

L'applicazione DroidGrill

DroidGrill è un'applicazione realizzata da Steven Cadalt insieme a Marco Gaudino e Giuseppe Bandiera per il corso di Sistemi Embedded del Corso di Laurea triennale in Ingegneria Informatica presso l'Università degli Studi di Padova.

Il software si propone come una soluzione per l'organizzazione di un evento in tutte le sue fasi, dalla scelta della location, alla divisione dei costi, passando per la registrazione delle spese effettuate.

All'applicazione base ottenuta al termine dell'insegnamento sono state aggiunte delle feature aggiuntive integrando i servizi Gmail e Google Maps, come descritto in seguito.

4.1 Funzionalità

DroidGrill si occupa della creazione e gestione di eventi i cui costi vengono in generale divisi dai partecipanti.
Prima di creare un evento è possibile definire una lista delle possibili location per eventuali eventi. Una schermata presenta una lista delle location e permette di rimuoverne o di aggiungerne di nuove. La definizione di una nuova location consente l'inserimento di alcune informazioni oltre al nome: la posizione può essere ricavata scegliendo un punto dalla mappa oppure acquisendola utilizzando il segnale GPS grazie all'hardware del dispositivo; può essere definita un immagine che a sua volta può essere ottenuta scattando una foto al momento, utilizzando la fotocamera del device, oppure scegliendo un'immagine dalla galleria contenente gli scatti già effettuati.

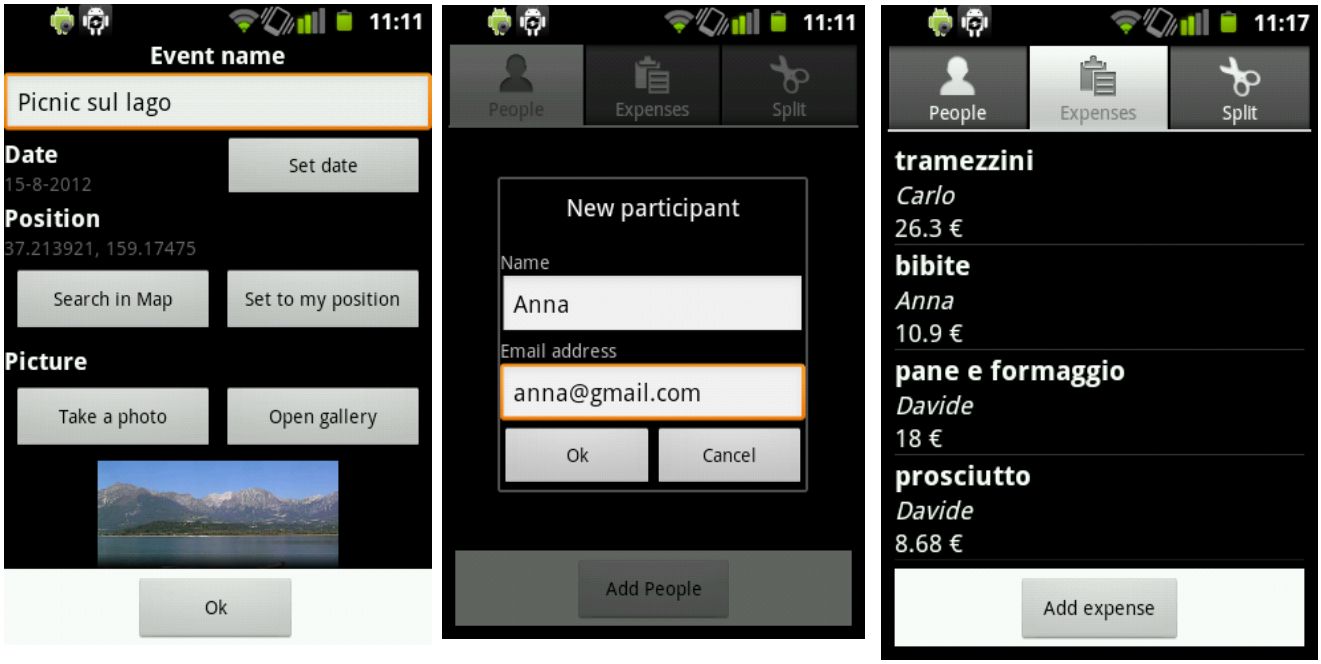
La creazione di un nuovo evento è analoga alla creazione di una location; in questo caso però è possibile definire una data e le informazioni riguardanti immagine e coordinate dell'evento possono essere importate da una delle location definite in precedenza.

Sia per gli eventi che per le location è possibile andare a visualizzare le informazioni salvate. Nel caso avessimo definito anche le coordinate associate all'evento/location, sarà possibile visualizzare su una mappa la posizione salvata.

Andando a selezionare un elemento dalla lista degli eventi viene proposta all'utente un'interfaccia costituita da tre tab.

- Nella prima, People, si possono aggiungere i partecipanti all'evento ed è possibile inserire per ognuno l'indirizzo email.
- La seconda tab, chiamata Expenses, si occupa della raccolta di tutte le spese effettuate per l'evento che stiamo considerando; aggiungendo una nuova spesa è possibile definire, oltre al nome della stessa, anche il suo costo e colui che l'ha sostenuta, andando a scegliere tra i partecipanti definiti nella prima tab.

- Split è la terza tab che permette di visualizzare il bilancio (debito o credito) per ciascun partecipante; questa caratteristica è utile al momento della ridistribuzione dei costi.



DroidGrill – Definizione di un evento, aggiunta di un partecipante e resoconto delle spese

4.2 Database SQLite

DroidGrill si appoggia a SQLite per l'immagazzinamento e gestione dei dati raccolti.

SQlite è una libreria che implementa un DBMS SQL; le sue caratteristiche principali sono l'assenza di dipendenza da librerie (un database SQLite è contenuto in un unico file, la libreria è leggera e pesa meno di 500KB) e la sua licenza è open-source che permette di utilizzare il codice senza restrizioni.

SQLite è ottimizzato per lavorare in sistemi embedded con risorse limitate, quali sono i device che montano OS Android.

La piattaforma mobile di Google offre una serie di librerie che permettono allo sviluppatore di interfacciarsi in modo semplice con SQLite, il che permette di riuscire a realizzare in modo abbastanza semplice applicazioni anche con un certo grado di complessità senza la necessità di conoscere il linguaggio SQL nei minimi dettagli.

Per questi motivi si è scelto di utilizzare questa soluzione per immagazzinare i dati raccolti e organizzarli in tabelle, scelta che si è rivelata azzeccata in quanto ha permesso, senza sforzi particolari, di implementare tutte le successive piccole progressive modifiche alla

struttura e ai nuovi tipi di dati che all'applicazione si è scelto di far gestire, in modo veloce, trasparente e che non intacchi in nessun modo il corretto funzionamento di nessuna altra parte del software.

4.3 Feature aggiunte

E' stato scelto di implementare all'interno dell'applicazione alcune funzionalità, legate fortemente alla tipologia di device sulla quale il software Android viene montato. Gli smartphone moderni possiedono hardware per recuperare la propria posizione dai satelliti GPS ed inoltre possono collegarsi ad Internet attraverso una connessione dati fornita dall' operatore telefonico. Si è deciso quindi di sfruttare queste potenzialità per aggiungere una funzione di condivisione dei dati degli eventi creati e implementare la possibilità di interfacciarsi con il servizio di navigazione satellitare di Google.

4.3.1 Condivisione – Google Gmail

Prima di iniziare a parlare dell'implementazione dell'integrazione della funzionalità di invio di email nell'applicazione DroidGrill è necessario definire quali siano i dati rilevanti relativi ad un evento che sia giusto condividere con gli altri partecipanti allo stesso. Colui che riceverà il messaggio di informazioni dovrebbe poter avere tra le mani tutti i dati relativi alla location dove si svolgerà l'evento, ma anche tutte le informazioni sui partecipanti e sulle spese che hanno effettuato; Nel corpo della email verranno quindi riportati, oltre al nome dell'evento, un link cliccabile che permette di visualizzare via browser web la location scelta, mentre per quanto riguarda i partecipanti sarà utile, oltre a conoscerne la lista, anche mostrarne le spese che ciascuno ha sostenuto e un riassunto dei bilanci degli invitati.

Il codice di seguito riportato andrà inserito all'interno del blocco che gestisce le operazioni da eseguire alla pressione dell'apposito bottone che ha la funzione di inviare agli invitati le info dell'evento.

Come prima operazione è necessario creare un Intent, andando ad indicare il tipo di azione che è necessaria per inviare delle informazioni: l'azione è ACTION_SEND.

```
Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
```

E' giusto ricordare che in fase di Intent Resolution qualsiasi applicazione che nel suo Intent Filter abbia dichiarato di poter risolvere quel tipo di azione sarà candidata tra le possibili applicazioni da lanciare; è il caso dell'applicazione Dropbox oppure della funzionalità di invio file tramite Bluetooth. Si sarebbe potuto creare un Intent esplicito per avviare direttamente l'applicazione Gmail, ma è più utile una funzionalità di condivisione che permetta di scegliere in che modo inviare l'informazione. E' l'utente che dovrà scegliere in base alle proprie necessità che cosa lanciare. Nel nostro caso avvierà Google Gmail.

All'interno del codice è necessario recuperare una lista di tutte le email degli invitati; è da notare che si è scelto un tipo di condivisione dei dati solo tramite posta elettronica e quindi

solo ai partecipanti che al momento dell'inserimento sono stati associati ad un indirizzo email valido, mentre un tipo di condivisione via ad esempio SMS è stata evitata, in quanto, in caso di una lista molto lunga di partecipanti si andrebbe ad avviare una procedura automatica di invio di messaggi SMS ad un elevato numero di contatti, operazione potenzialmente assai costosa.
Gli indirizzi email vengono inseriti in un array `aEmailList` in questo modo:

```
String[] aEmailList = getPeopleMail(cPeopleByEvent);
```

Il metodo `getPeopleMail` che recupera gli indirizzi email dal database è consultabile nell'appendice A.1.

Si va ad inserire l'array, tramite la definizione di un campo Extra, nell'Intent:

```
emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,aEmailList);
```

In modo analogo si vanno ad inserire l'oggetto della email e il corpo del messaggio:

```
emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,"Droidgrill - info
evento");

emailIntent.putExtra(android.content.Intent.EXTRA_TEXT,createBody(id));
```

Il metodo `createBody` si occupa di andare a comporre il messaggio di testo vero e proprio, andando a recuperare i dati dal database; è consultabile nell'appendice A.2.

Se al momento della definizione dell'evento è stata associata un'immagine, allora con il codice seguente è possibile inserirla come allegato:

```
if (getImageUri(id) != null) {

      emailIntent.putExtra(Intent.EXTRA_STREAM, getImageUri(id));

}
```

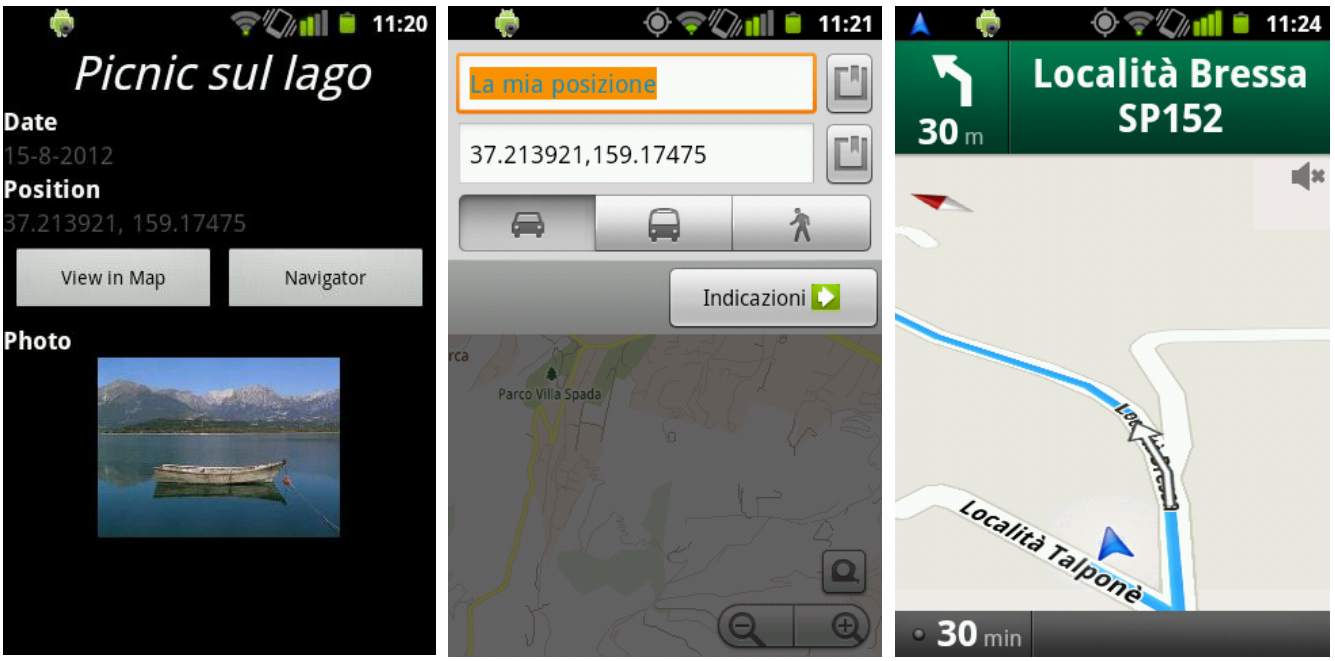
Il metodo `getImageUri` recupera l'Uri dell'immagine salvato nel database; è consultabile nell'appendice A.3

Una volta definiti tutti i campi della email è necessario lanciare l'activity associata all'Intent con

```
startActivity(emailIntent);
```

4.3.2 Navigazione – Google Maps Navigation

Nell'ipotesi di essere in possesso delle coordinate della location scelta per un evento risulta utile da parte dell'utente una funzionalità che lo aiuti a raggiungere tale luogo in modo semplice. Molte sono le applicazioni sviluppate per la piattaforma Android che svolgono questo compito, ma preinstallata nel sistema è presente Google Maps Navigation che oltre a essere gratuita è ben integrata con gli altri servizi di Google.



DroidGrill – Riepilogo dell'evento e inizio navigazione

All'interno del codice di DroidGrill si andrà a definire un Intent implicito:

```
Intent myIntent = new Intent(Intent.ACTION_VIEW,

Uri.parse("http://maps.google.com/?daddr=" + latDegree+ "," + lonDegree));
```

Come secondo parametro viene passato un indirizzo che contiene le coordinate note della location.

Infine basta far partire l'activity associata all'Intent con il codice:

```
startActivity(myIntent);
```

Essendo l'Intent lanciato di tipo implicito, tutte le applicazioni installate nel sistema che dichiarano di poter svolgere l'azione richiesta sono tra le possibili candidate.
Google Maps è una di queste e permette una volta avviata di definire gli ulteriori

parametri relativi al tragitto e di lanciare a sua volta l'applicazione di navigazione vera e propria.

Capitolo 5

Conclusioni e sviluppi futuri

Lo step successivo nello sviluppo dell'applicazione *DroidGrill* sarà l'inserimento della funzionalità di invio, in allegato ad una email, di un file proprietario, all'interno del quale racchiudere tutte le info relative ad una location e che inviato possa essere utilizzato dall'applicazione installata sul device del destinatario per importare i dati della location andando ad aggiungerne una nuova alla relativa lista sul proprio dispositivo.
Uno stadio più avanzato dello sviluppo potrebbe inoltre riguardare la gestione centralizzata di ogni singolo evento, garantendo la sincronizzazione ad ogni aggiornamento effettuato da ciascun partecipante all'evento.

Appendice A

Codice Sorgente

A.1 Metodo per il recupero degli indirizzi email

Il metodo, una vola ottenuto un cursore dal database contenente tutti i dati relativi ai partecipanti all'evento considerato, lo scorre estraendo per ogni riga l'indirizzo email della persona ed inserendolo in un array.

```
// obtained email-addresses of participants

      private String[] getPeopleMail(Cursor people) {

            String[] mails = new String[people.getCount()];

            int i = 0;

            people.moveToFirst();

            while (!people.isAfterLast()) {

                  String newMail = people.getString(people

                              .getColumnIndex(AppDB.peoMail));

                  mails[i] = newMail;

                  i++;

                  people.moveToNext();

            }

            return mails;

      }
```

A.2 Metodo per la costruzione del corpo del messaggio

Per prima cosa viene creato uno `StringBuffer` vuoto, al quale successivamente verranno aggiungi i dati che comporranno il corpo del messaggio.

```
// create a mail body

      private String createBody(int idEvent) {

            StringBuffer body = new StringBuffer();

            db.open();

            String[] columnEve = new String[] { AppDB.eveName,AppDB.evePhoto,

                  AppDB.eveLatitude, AppDB.eveLongitude, AppDB.eveDate};

            cEveInfo = db.getEveInfo(columnEve, id);

            cEveInfo.moveToFirst();
```

Vengono poi recuperati nome e data dell'evento.

```
// event name

            String eveName = cEveInfo.getString(

            cEveInfo.getColumnIndex(AppDB.eveName)).toUpperCase();
```

```
// event date
            long dateTimeStamp = cEveInfo.getLong(cEveInfo
                        .getColumnIndex(AppDB.eveDate));
            Date data = new Date(dateTimeStamp);
            int day = data.getDate();
            int month = data.getMonth() + 1;
            int year = data.getYear();
            String eveDate = ("");
            if (day != 1 && month != 1 && year != 70) {
                  eveDate = (day + "-" + month + "-" + year);
            }
```

Dopo aver ottenuto le coordinate, viene creato un link cliccabile che rimanda a una pagina di Google Maps dove è possibile visualizzare il luogo dell'evento

```
// eve coordinates
            int lat_micro = cEveInfo.getInt(cEveInfo
                        .getColumnIndex(AppDB.eveLatitude));
            int lon_micro = cEveInfo.getInt(cEveInfo
                        .getColumnIndex(AppDB.eveLongitude));
            double latDegree = lat_micro / 1E6;
            double lonDegree = lon_micro / 1E6;


            String latS = cEveInfo.getString(cEveInfo
                        .getColumnIndex(AppDB.eveLatitude));
            String lonS = cEveInfo.getString(cEveInfo
                        .getColumnIndex(AppDB.eveLongitude));
            int latI = Integer.parseInt(latS);
            int lonI = Integer.parseInt(lonS);


            String link = ("");
            if (!(latI == 0 && lonI == 0)) {
                  link = ("http://maps.google.com/maps?q=" + latDegree + ","
                        + lonDegree + "1&ll=" + latDegree + ","
                        + lonDegree + "&z=17");
            }


            db.close();
```

Tutte le informazioni ora vengono raggruppate per formare il campo body; vengono inoltre aggiunti la lista dei partecipanti, le spese e il debito/credito per ciascun invitato.

```
// compose the body
            body.append("=> " + eveName + " <=" + "\n");


            if (!eveDate.equals("")) {
                  body.append("Date: " + eveDate + "\n");
            }


            if (!link.equals("")) {
                  body.append("Google Maps: " + link + "\n");
            }
```

```
// participants
            body.append("\n" + "Participants" + "\n");
            body.append("-------------------------" + "\n");


            cPeopleByEvent.moveToFirst();
            int i = 1;
            while (!cPeopleByEvent.isAfterLast()) {
                  String peo = cPeopleByEvent.getString(cPeopleByEvent
                              .getColumnIndex(AppDB.peoName));
                  Log.i("A PARTICIPANT", i + "." + peo);
                  body.append(i + "." + peo + "\n");
                  i++;
                  cPeopleByEvent.moveToNext();
            }
```

```
// expenses
            cExpInfo = cExpenseByEvent;
            cExpInfo.moveToFirst();
            Log.i("ExpCURSOR LENGTH", "" + cExpInfo.getCount());
            if (cExpInfo.getCount() != 0) {
                  body.append("\n \n" + "Expenses" + "\n");
                  body.append("-------------------------" + "\n");
                  while (!cExpInfo.isAfterLast()) {
                        String expense = cExpInfo.getString(cExpInfo
                                    .getColumnIndex(AppDB.expName));// expense
                        String price = cExpInfo.getString(cExpInfo
                                    .getColumnIndex(AppDB.expPrice));// price
                        String buyer = cExpInfo.getString(cExpInfo
                                    .getColumnIndex(AppDB.expBuyerName));
                        body.append("* " + expense + " => ");
                        body.append("€ " + price + "\n");
                        body.append("    " + buyer + "\n");
                        cExpInfo.moveToNext();
                  }

            }
```

```
// participants and the balances
            body.append("\n \n" + "Balances" + "\n");
            body.append("-------------------------" + "\n");
            cPeopleByEvent.moveToFirst();
            int j = 0;
            while (!cPeopleByEvent.isAfterLast()) {
                  String peo = cPeopleByEvent.getString(cPeopleByEvent
                              .getColumnIndex(AppDB.peoName));
                  body.append("* " + peo + "\n");
                  body.append("    " + "€ " + balances[j] + "\n");
                  j++;
                  cPeopleByEvent.moveToNext();
            }
```

A.3 Metodo per il recupero dell'Uri dell'immagine associata all'evento

```
private Uri getImageUri(int idEve) {
            int theId = idEve;
            db.open();
            String[] columnEve = new String[] { AppDB.evePhoto };
            cEvePhoto = db.getEveInfo(columnEve, theId);
            Log.i("CURSOR LENGTH", "" + cEvePhoto.getCount());
            db.close();
            cEvePhoto.moveToFirst();
            Uri tempUri = Uri.parse(cEvePhoto.getString(cEvePhoto
                        .getColumnIndex(AppDB.evePhoto)));
            String imaUri = cEvePhoto.getString(cEvePhoto
                        .getColumnIndex(AppDB.evePhoto));
            if (imaUri.equals("no photo")) {
                  return null;
            }
            return tempUri;
}
```

Bibliografia

[1] Massimo Carli. *Android, Guida per lo sviluppatore.* Apogeo, 2010.

[2] The Pragmatic Programmers. *Hello, Android: Introducing Google's Mobile Development Platform.* Ed Brunette, 2009.

[3] Android Developers . http://developer.android.com/guide/index.html

[4] Wikipedia. Android. http://en.wikipedia.org/wiki/Android_(operating_system)

[5] Wikipedia. Google Maps. http://en.wikipedia.org/wiki/Google_maps

[6] http://luca-petrosino.blogspot.com

[7] http://mobile.html.it/guide/leggi/212/guida-android/