ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA
CAMPUS DI CESENA
SCUOLA DI SCIENZE
CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

BUSINESS CHAT: SVILUPPO E IMPLEMENTAZIONE

Relazione finale in

Algoritmi e strutture dati

Relatore:
Prof. Luciano Margara

Presentata da:
Stefano Convertino

Sessione III
Anno Accademico 2013/2014

SOMMARIO

INTRODUZIONE

In questa tesi si vuole parlare dello sviluppo e dell'implementazione di un'applicazione per smartphone che abbiamo realizzato.

L'applicazione in questione si chiama BusinessChat, è stata sviluppata per Android e utilizza sia un lato server, sia un lato Client.

BusinessChat è un applicazione che mette in contatto consumatori e gestori di attività.

Uno degli scopi dell'applicazione è quello di permettere al consumatore di comunicare con le attività di business in maniera più immediata, semplice e diretta.

L'altro obbiettivo è quello di permettere al business di utilizzare quest'app sia per fornire una serie di servizi al proprio cliente, sia per avere un canale di pubblicizzazione alternativo per la propria attività. Uno dei metodi ad esempio potrebbe essere quello di mandare in broadcast a tutti i propri contatti messaggi pubblicitari o codici per buoni sconto (al momento della stesura di questa testi questa funzione non è ancora disponibile, ma verrà aggiunta in un breve futuro).

L'utente può ricercare, attraverso l'uso di una mappa, una particolare attività, ed eventualmente mettersi in contatto con questa. Se ciò che cerchiamo è un ristorante, ad esempio, potremmo voler sapere se quella sera sono ancora disponibili dei posti liberi nel proprio locale preferito; nel caso volessimo ordinare delle pizze da asporto dalla nostra pizzeria di fiducia o sapere se sono attrezzati per gestire clienti allergici al glutine, potremmo semplicemente chiederlo attraverso la nostra chat; oppure potremmo ancor più semplicemente voler solo conoscere l'orario di chiusura di un locale.

Tutto questo sarebbe possibile farlo in pochissimi passi grazie a BusinessChat.

Il nostro intento è quello di avvicinare il cliente alle attività commerciali, che queste siano l'edicolante in fondo alla strada o una catena internazionale di vestiti.

Il nostro obbiettivo è di agevolare la comunicazione fra queste due parti,

permettendo di evitare inutili sprechi di tempo o equivoci causati da problemi di comunicazione.

Grazie a BusinessChat entrambe le parti avrebbero agevolazioni: gli utenti potrebbero chiarire i propri dubbi senza dover contattare telefonicamente le attività e le aziende avrebbero finalmente una via di comunicazione semplice, economica ed efficace con i propri clienti.

Per un approccio più funzionale, intendiamo rendere necessaria l'installazione dell'app solo al consumer (come avviene per qualsiasi applicazione mobile). Sarà invece prerogativa del negoziante scegliere di utilizzare o no la nostra applicazione dato che, attraverso un processo di rerouting, esso potrà essere contattato, nel caso avesse deciso di non scaricarla, attraverso i classici canali di comunicazione.

Grazie a questo processo di reroutering, anche le attività commerciali non registrate riceveranno i messaggi dai clienti dato che, grazie alle API di Fousquare, Google e Yelp, riusciremmo ad ottenere in ogni caso i contatti delle attività commerciali, permettendo così all'utente di contattare l'attività che preferisce tramite e-mail, telefono o social.

Successivamente, se lo desiderano, le attività commerciali potranno installare sul proprio dispositivo la nostra applicazione e fare così uso di tutte le sue funzionalità.

1. IL DATABASE

Dato che dobbiamo avere a che fare con le informazioni riguardanti i vari utilizzatori della nostra applicazione, abbiamo ritenuto necessario avere una struttura che ci permettesse di mantenerle memorizzate. Abbiamo quindi deciso di costruire un database che ci permettesse di svolgere questa funzione. Ci siamo quindi soffermati ad analizzare la quantità di dati che dovevamo gestire per poter poi segliere la tecnologia adatta. La struttura che il nostro database dovrebbe assumere è di fatto abbastanza semplice: abbiamo infatti individuato all'inizio due entità basilari che costituirebbero lo schema scheletro del database, ovvero l'entità Utente e l'entità Business; tutto infatti ruota intorno a queste due entità.
Tra le differenti tecnologie a nostra disposizione quindi abbiamo optato per quella di Oracle MySQL, software freeware che permette la gestione di basi di dati relazionali. Ora descriveremo in brave questa tecnologia prima di proseguire.

Oracle MySQL

MySQL o Oracle MySQL è un è un Relational Database Management System, ovvero un software che permette di gestire una base di dati strutturata su un modello di tipo Relazionale.
MySQL generalmente viene utilizzato da moltissime piattaforme web per la gestione di contenuti in modo dinamico ed è spesso scelto sia per la sua accessibilità, dato che è scaricabile da parte di chiunque, sia per la sua malleabilità: esistono infatti numerosissime Fork costruite partendo dalla sua base o sistemi di content managing strutturati nati dall'evoluzione delle sue implementazioni; un esempio di questo può essere MediaWiki, software che gestisce i siti dell'intero progetto Wikipedia e che, come molti altri, è basato su un database MySQL.
Una caratteristica molto importante è il fatto che MySQL di base è sprovvisto di un'interfaccia grafica (GUI) per l'utilizzo da parte dell'utente. Il software infatti è composto da un client a riga di comando e da un server che gestisce i record del database.

Sono quindi stati creati dei software manager che permettono l'interfaccia diretta con il database.
Essi possono essere raggruppati in due categorie:

- Software di Interfaccia al database, cioè applicazioni che consentono di eseguire query e vedere il risultato sullo schermo;

- Software di Gestione del server di database, cioè applicazioni che includono strumenti di amministrazione degli utenti (es. permessi), dei parametri di configurazione, di monitoraggio delle prestazioni e del salvataggio dei dati.

MySQL Workbench

Oracle per rimediare al problema di accessibilità di MySQL, ha quindi sviluppato un software manager proprio chiamato MySQL Workbench.
L'ultima versione realizzata, la 5.2, lo rende un vero strumento visuale, semplice e intuitivo.
Data la versatilità di questo tool, abbiamo deciso di sceglierlo per la gestione del nostro Database. Esso permette di progettare il database, gestire e modellare i dati al suo interno e integra lo sviluppo di query SQL. Permettendo di creare e mantenere il database MySQL all'interno di un unico ambiente, fa si che tutto il processo di gestione del database sia molto semplice e immediato.

PhpMyAdmin

PhpMyAdmin è un'applicazione web scritta in PHP, rilasciata con licenza GPL, che consente di amministrare un database MySQL tramite un qualsiasi browser. L'applicazione è indirizzata sia agli amministratori, sia agli utenti e gestisce i permessi prelevandoli direttamente dal database.

PhpMyAdmin ha i permessi di realizzare un database da zero, creando le tabelle ed eseguendo operazioni di ottimizzazione sulle stesse. Presenta un feedback sulla creazione delle tabelle per evitare eventuali errori. Sono inoltre previste delle funzionalità per l'inserimento dei dati (ovvero l'operazione di popolazione del database, ciò che stiamo cercando), per le query e per il backup dei dati.

L'amministratore ha anche a disposizione un'interfaccia grafica per la gestione degli utenti: l'interfaccia permette l'inserimento di un nuovo utente, la modifica della relativa password e la gestione dei permessi che l'utente ha sul database.

Come programma è molto semplice da configurare e utilizzare. Esso non necessità nemmeno di una reale installazione in quanto è riassumibile in un'insieme di pagine PHP che necessitano solo di essere eseguite.

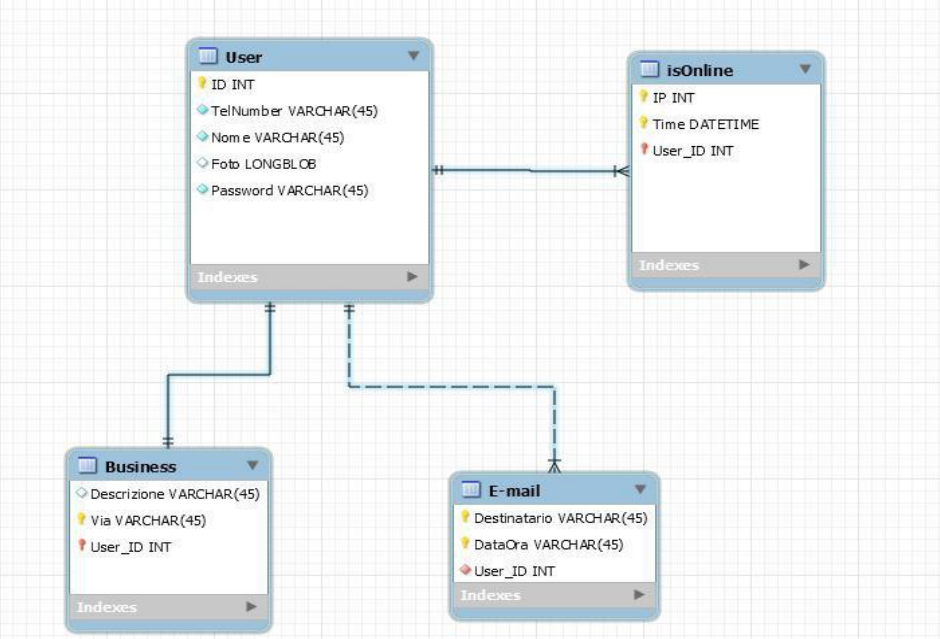
Tutto il codice lato Server è stato usato in un primo momento sulla macchina locale, ciò e stato possibile utilizzando XAMPP, (è una piattaforma software gratuita costituita da Apache HTTP Server, il database MySQL e tutti gli strumenti necessari per utilizzare i linguaggi di programmazione PHP e Perl), successivamente il lato server è stato caricato online su Altervista.

Ora è giunto il momento di parlare della struttura che abbiamo realizzato per i dati all'interno del nostro Server.

La struttura del database

Avevamo precedentemente definito due entità base in un possibile schema scheletro del nostro database relazionale: l'entità Utente e l'entità Business.

Fondamentalmente questo concetto non è sbagliato, ma ad un'analisi più attenta possiamo definire che è troppo riduttivo dato che i dati da memorizzare sono leggermente di più e possono essere gestiti in modo diverso.



Questo appena mostrato è lo schema logico del database che abbiamo realizzato.

Come si può facilmente vedere, abbiamo realizzato quattro elementi principali e ora li analizzeremo uno ad uno.

La tabella User

La tabella User racchiude al suo interno l'insieme di tutti i profili registrati all'interno dell'applicazione, che essi siano di semplici utenti, oppure che siano quelli delle attività. Essa è composta da vari attributi tra cui un ID che funge da chiave primaria della tabella e che da ad ogni profilo un valore univoco con cui viene identificato da parte dell'applicazione. All'interno di questa tabella memorizziamo anche l'attributo Numero di telefono, Nome dell'utente e una password. L'utente una volta registrato può anche Inserire una foto profilo a suo piacimento.

Nel caso l'utente non fosse una persona che vuole semplicemente usufruire dell'app, ma

un'attività che desidera essere "pubblicizzata" all'interno di questo circuito, la tupla presente all'iterno della tabella business relativa a questa azienda sarà legata alla relativa tubla della tabella User.

La tabella Business

La tabella Business ci serve per memorizzare le informazioni aggiuntive a quelle dell'utente nel caso il profilo da noi creato fosse quello di un'azienda.

Questa tabella non presenta un Identificativo come chiave primaria, ma bensì utilizza il campo Via per questo scopo. Esso infatti, accoppiato alla chiave esterna ID presa dal profilo dell'utente, identifica la singola attività a se stante. Abbiamo deciso di identificare le varie attività in base alla via poichè in questo modo nel caso avessimo più attività con lo stesso nome, magari appartenenti alla stessa catena (come ad esempio un supermercato o una serie di negozi di vestiti), non ci saranno problemi di sovrapposizione si potrà contattare il singolo negozio senza sbagliare numero o profilo.

Abbiamo anche inserito tra i vari attributi di questa tabella il campo Descrizione. Questo ci può permettere di inserire una breve descrizione dell'attività nel caso questa non fosse già conosciuta da parte degli utenti.

La tabella Online

Una volta definiti i vari profili, si passa all'altra parte del Database, la tabella Online.

Questa tabella è necessaria per memorizzare i vari accessi effettuati dagli utenti e per permettere di visualizzare nella chat se un'attività al momento è online e contattabile direttamente, oppure se non lo è.

La tabella presenta due specifiche chiavi primarie: la prima è IP, che serve a definire l'indirizzo IP dell'utente al momento della connessione, dato fondamentale per potergli instradare i dati relativi agli altri utenti o per poter permettere le funzionalità di invio e ricevimento dei messaggi nella chat;

l'altra chiave primaria è Time, attributo che memorizza il momento della connessione.

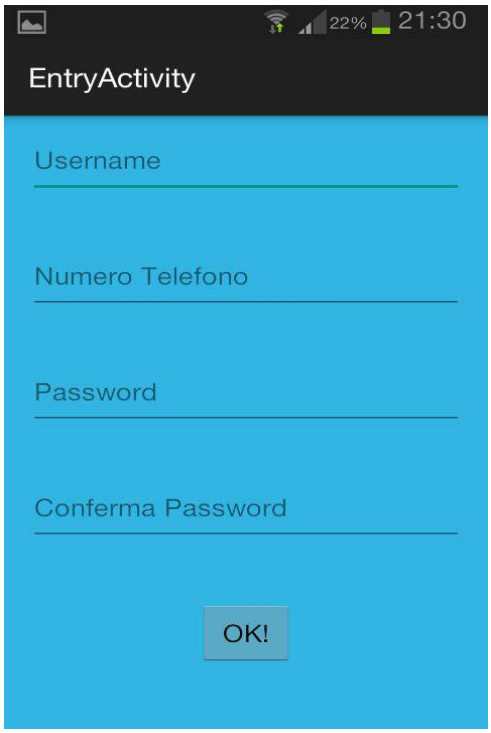
Le due chiavi primarie, associate alla chiave esterna ID, che fa riferimento all'id dell'profilo, permettono di costruire un log delle connessioni del singolo utente.

La tabella E-mail

Infine abbiamo la tabella E-Mail. Questa tabella ci permette di identificare il destinatario e l'ora d'invio delle e-mail che possiamo inviare agli altri utenti o alle attività che non sono registrate all'interno del nostro circuito. I vari attributi che abbiamo infatti sono Destinatario e DataOra, entrambe primary key della tabella. Insieme alla foreign key ID che preleviamo dalla tabella User, ci permettono di memorizzare i dati a noi necessari per poter tenere traccia delle varie e-mail inviate con mittente e destinatario.

SignUpActivity

L'activity per la registrazione (Sign up) è l'activity che ci permette di registrare alla nostra applicazione. Per registrarsi è sufficiente inserire i dati nel form sottostante:




```
      [...]
            String $Stelefono = telefono.getText().toString();
            String $Spassword = password.getText().toString();
            String $ScomparePassword =
confirmPassword.getText().toString();
            String $Susername = username.getText().toString();

           [...]

            String
phpUrl=phpURL+"telefono="+Etelefono+"&password="+Epassword+"&username
="+ Eusername;
```

```
                try {
                    if ($ScomparePassword.compareTo($Spassword)==0) {
                        HttpClient httpClient = new
DefaultHttpClient();
                        HttpResponse httpResponse =
httpClient.execute(new HttpGet(phpUrl));
                        InputStream inputStream =
httpResponse.getEntity().getContent();
                        if (inputStream != null)
                            result =
convertInputStreamToString(inputStream);
                        else
                            result = "Did not work!";
                    }else{

confirmPassword.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);

password.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);
                        result = "You Must use the same Password!";
                    }
                } catch (Exception e) {
                    Log.d("TAG ", "ERRO EM GET HTTP URL:\n" + result
+ "\n" + e);
                    Log.d("TAG", result);
                }
                Log.d("RISULTATO", "GET HTTP URL OK:\n" + result);

[...]
```

l'applicazione leggerà i dati dai form e li passera ad uno script php sul server (al momento il lato server dell'applicazione è su Altervista), lo script leggerà i dati, controllerà che siano validi (ovvero che la chiave primaria non sia già presente ecc...), inserirà il nuovo utente registrato nel database e darà una risposta alla nostra activity.

```
[...]
            if (result.compareTo("OK") == 0 )
            {
                Intent intent = new Intent(EntryActivity.this,
LoginActivity.class);
                startActivity(intent);
            }
            else
            {
                telefono.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);
                password.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);
            }
[...]
```

Nel caso di risposta positiva l'utente verrà reindirizzato alla pagina di login, se la risposta è negativa invece i form di inserimento diventeranno rossi, richiedendo i dati sbagliati.

Il codice lato server è il seguente:

```
[...effettua connessione al database ...]
//echo dirname($databasename). '</br>';
      //scrivo e invio la query
      $query = "SELECT password,TelNumber FROM  `user` WHERE TelNumber =
'$telefono' ";
      $risultato = mysql_query($query);

      //se ci sono problemi mi stampa il problema
                [...ometto controlli...]
            if (mysql_num_rows($risultato) == 0){
                  $output="OK";
                  $sql = "INSERT INTO `user`(`TelNumber`, `Nome`, `Foto`,
`Password`) VALUES ('$telefono','$nome',NULL,'$pass')";
                  // se la query è andata a buon fine
                  if (mysql_query($sql)){
                        echo $output;
                  }else{
                        echo 'Error: ' . $sql . '<br>' . mysql_error($con);
                  }
            }else
                  echo('Telefono già usato');
                  //array_push($response["product"], $product);
            // echoing JSON response

      }
```

Lo script effettua una query, se il risultato è 0, ovvero non ci sono già altri account inseriti con le stessi chiavi inserisco i dati nel database darà come risposta OK all'activity, altrimenti rispondo KO.

LoginActivity

L'activity che ci permette di effettuare il login (Sign In) è la LoginActivity Per effettuare il login è sufficiente inserire nel form sottostante il proprio numero di telefono e la propria password:

Il codice per acquisire i dati ed inviarli allo script lato server è simile a quello dell'activity precedente, una volta premuto il bottone Ok verranno inviati i dati, nel caso di risposta positiva il server risponderà OK e l'applicazione passerà all'activity

successiva, in caso di risposta negativa invece il form diventeranno rossi e chiederanno nuovamente l'inserimento dei dati.

```
  Log.d("TAG", "|"+result+"|");
            if (result.compareTo("OK") == 0 )
            {
                Intent intent = new Intent(LoginActivity.this,
MainActivity.class);
                startActivity(intent);
            }
            else
            {
                telefono.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);
                password.getBackground().setColorFilter(Color.RED,
PorterDuff.Mode.SRC_ATOP);
            }
```

Il codice lato server sarà il successivo :

```
$query = "SELECT * FROM user WHERE TelNumber='$telefono' AND
Password='$pass' ";
      $risultato = mysql_query($query);


             [... controlli...]
            //se la query ha una riga allora esiste la combinazione
nick pass
            if (mysql_num_rows($risultato) == 0){
                  $output="KO";
                  echo ($output);
                  print(json_encode($output));

                  //array_push($response["product"], $product);
            // echoing JSON response
            echo json_encode($output);
      }
            else{
                  $sql = "INSERT INTO isonline(IP, Time, User_ID)
                              VALUES  ('$IP',  'new  DateTime('now')'  ,
'$risultato')";
                  $output="OK";
                  echo ($output);
           // echo json_encode($output);
                  }
      }
```

Il codice effettua una query, nel caso sia presente una riga con la stessa combinazione telefono password, vuol dire che l'utente è presente nel database, quindi l'applicazionione può procedere con il login e noi andremo ad inserire il nostro utente nella tabella del database che viene utilizzata per indicare se un utente è online.

2. I SERVIZI PER LA GEOLOCALIZZAZIONE

MapsActivity

Mappe

Le alternative più utilizzare per visualizzare una mappa su un dispositivo mobile, sono due, OpenMaps e Google Maps, abbiamo scelto di utilizzare la seconda, in quanto è la più diffusa e molte App importanti, tra cui la stessa Foursquare, ne fanno uso, ma per completezza pensiamo sia utile introdurre anche le openStreetMap.

OpenStreetMap

```
« OpenStreetMap è una mappa liberamente modificabile dell'intero pianeta. È fatta da persone come te. OpenStreetMap permette a chiunque sulla Terra di visualizzare, modificare ed utilizzare dati geografici con un approccio collaborativo. »

(Dal sito ufficiale di OpenStreetMap)
```

OpenStreetMap (OSM) è un progetto collaborativo finalizzato a creare mappe a contenuto libero del mondo. Il progetto punta ad una raccolta mondiale di dati geografici, con scopo principale la creazione di mappe e cartografie.

La caratteristica fondamentale dei dati geografici presenti in OSM è che possiedono una licenza libera, la Open Database License. È cioè possibile utilizzarli liberamente per qualsiasi scopo con il solo vincolo di citare la fonte e usare la stessa licenza per eventuali lavori derivati dai dati di OSM. Tutti possono contribuire arricchendo o correggendo i dati. Una alternativa proprietaria che riprende le pratiche di crowdsourcing introdotte dalla comunità di OpenStreetMap è Google Map Maker, nel quale i contributi dell'utente possono venir inseriti su Google Maps a fronte di validazione, impedendone il riutilizzo da parte di terzi.
Penso sia giusto introdurre OpenStreetMap in quanto è una valida alternativa alle Google Maps API , nonostante ciò eviterò di entrare nel dettaglio dell'argomento in OpenStreetMap quanto non è stato usato nel nostro progetto.

Google Maps API

Google Maps è un servizio, oggeto da Google, accessibile dal relativo sito web, che consente la ricerca e la visualizzazione di carte geografiche di buona parte della Terra.
Per i dispositivi mobili che supportano la piattaforma Java (o anche quelli basati sui sistemi operativi Android, iOS, Windows Mobile, Palm OS o Symbian OS) e che dispongono di una connessione a Internet via GPRS, UMTS o HSDPA esiste dal 2006 un'apposita versione Google Maps, detta Google Maps Mobile, che permette di accedere alle mappe e di usufruire del servizio di navigazione gps come un vero e proprio Navigatore satellitare da tali dispositivi.
Google Maps API, ci permette di visualizzare una mappa, sul nostro dispositivo mobile come quella mostrata di seguito:

La parte di codice che ci permette di integrare le Google Maps nel nostro progetto è molto semplice, basta infatti includere un fragent di tipo map all'interno della nostra activity, il codice sarà il seguente:

```
 if (mMap == null) {
            // Try to obtain the map from the SupportMapFragment.
                                        mMap   =   ((SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.map))
                    .getMap();
            // Check if we were successful in obtaining the map.
            if (mMap != null) {
                setUpMap();
                centerMapOnMyLocation();
                mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
                placeMarkers = new Marker[MAX_PLACES];
                updatePlaces();
            }
        }
```

Per evitare di confondersi è importante far notare una cosa: le Google Maps API, permettono solo di visualizzare la mappa sul nostro dispositivo, ma non permettono di visualizzare luoghi di interesse o geolocalizzare la nostra posizione, per fare quello andremo ad aggiungere altre librerie, come mostreremo di seguito.

Geolocalizzazione, Location Listener e locationManager

Implementare il location listener di android ci permette di geolocalizzare il nostro dispositivo, e ci darà la possibilità di inserire all'interno della nostra applicazione 4 importanti funzioni:

1. onProviderDisabled
2. onProviderEnabled
3. onStatusChanged
4. onLocationChanged

La funzione onLocationChanged nel nostro progetto è la più importante. La funzione si attiva ogni volta che il dispositivo cambia la propria posizione, all'interno della funziona onLocationChanged richiamiamo la funzione updatePlaces, che si aggiornerà, cambiando la posizione attuale del nostro device e mostrando i marker, nella zona circostante a noi, rappresentanti i luoghi di interesse.

1.

```
@Override
    public void onLocationChanged(Location location) {
        Log.v("MyMapActivity", "location changed");
        updatePlaces();
    }
```

La funzione updatePlaces, è la seguente :

```
   if (!start){
                                                                locMan     =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
            //get last location
                                                    Location    lastLoc    =
locMan.getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
            lat = lastLoc.getLatitude();
            lng = lastLoc.getLongitude();
            LatLng lastLatLng = new LatLng(lat, lng);
            start=true;
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(lastLatL
ng, 13.0f));
```

```
        }
                                                    placesSearchStr      =
sendRequest(lat,lng,searchText.getText().toString());
       //......Altre funzioni........
  locMan.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,  30000,
10000,         (android.location.LocationListener) this);
```

all'interno della funziona updatePlaces chiameremo il LocationManger, che ci permette di acquisire le coordinate (latitudine e longitudine) del nostro dispositivo, in seguito andremo a memorizzarle in due apposite variabili, aggiorniamo la nostra posizione, nel caso sia richiesto muoviamo la camera che visualizza la mappa, fino a visualizzare la posizione nella quale ci troviamo ed infine chiamiamo la funzione requestLocationUpdates tramite il location manager, che abilità il locationManager ad aggiornare la nostra mappa.

Nel caso si abbia spostato la camera che visualizza la mappa e non si riesca a trovare la propria posizione all'interno della mappa visualizzata sul dispositivo, è possibile individuarla tramite il seguente bottone:

il seguente bottone richiama al click la funzione centerMapOnMyLocatione, la funzione chiamata ci permetterà di spostare la camera relativa alla mappa fino a centrare la nostra posizione sulla mappa.

Luoghi di interesse, venues

Come accennato prima, le Google Maps API ci permettono di visualizzare la mappa, ma non di visualizzare i luoghi di interesse nei nostri dintorni, per farlo abbiamo bisogno di altre librerie. Per la visualizzazione dei luoghi di interesse esistono moltissime librerie,quelle da noi utilizzate sono:

1. Google Places Library
2. Foursquare API
3. Yelp API

Andremo ad analizzare nel dettaglio tutte e 3 le API nel prossimo capitolo, per il momento analizzeremo solo la funzione esplora delle Foursquare API, ovvero la funzione utilizzate in questa activity.
Per ricercare i luoghi di interesse e segnarli sulla mappa abbiamo deciso di usare le api di Foursquare, abbiamo preferito queste rispetto a quelli di Yelp e a quelle di Google perchè dopo averle utilizzate e provate per un po' di tempo, le abbiamo ritenute le più complete, hanno una quantità di luoghi di interesse maggiore di quelle delle altre api, con una quantità di dati superiore rispetto alle altre due, sono ricche di recensioni da parte degli utenti, foto dettagli e dati riguardanti le attività sempre aggiornati.

Foursquare permette agli utenti di effettuare due tipi di ricerche, una richiede l'autenticazione, l'altra no. Nella nostra app abbiamo utilizzato entrambe le ricerche, ma per quello che riguarda questa Activity abbiamo utilizzaato la semplice ricerca senza autenticazione. Andremo a vedere nel dettaglio le api di Foursquare nel prossimo capitolo, per il momento ci limiteremo a guardare la ricerca delle Foursquare API senza chiave di autenticazione.

Foursquare definisce la ricerca di luoghi di interesse nei dintorni di una determinata zona "explore".
Per effettuare la ricerca tramite le Foursquare API basta inviare una richiesta in formato http, tramite AsyncTask dalla nostra activity, la risposta sarà un file di tipo JSON che andremo a leggere di conseguenza.

Il formato della richiesta è il seguente:

```
placesSearchStr = "https://api.foursquare.com/v2/venues/search" +
                "?client_id="+ CLIENT_ID +
                "&client_secret="+ CLIENT_SECRET +
                    "&v=20130815"+ //parametro che indica la versione
delle api di foursquare
                "&ll="+lat+","+ lng +
                "&query="+QueryString.toString();
```

Come si può vedere all'interno della nostra richiesta andremo ad inserire il Client ID e il Client Secret (parametri forniti da foursquare dopo la registrazione della nostra applicazione al loro servizio), la versione delle api che vogliamo utilizzare, la latitudine e la longitudine in cui vogliamo utilizzare la funzione "explore" e possiamo inserire

opzionalmente una stringa di testo, la quale può contenere delle chiavi di ricerca.
La stringa di testo verrà passata tramite la seguente form :

Inserire un parola nella form, la inserirà, automaticamente, nella query, la funzione explorer cercherà i luoghi di interesse vicino a noi che contengono o nel nome, o nei tag la parola da noi usata come chiave di ricerca.

La nostra applicazione provvederà ad inviare una semplice richiesta http tramite AsyncTask:

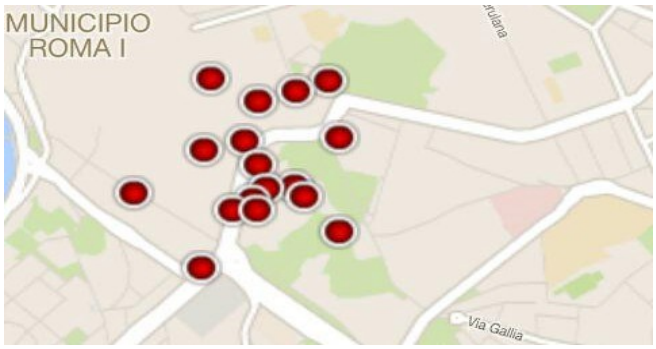
```
 HttpClient placesClient = new DefaultHttpClient();
                try {
                    HttpGet placesGet = new HttpGet(placeSearchURL);
                    HttpResponse placesResponse =
placesClient.execute(placesGet);
                    StatusLine placeSearchStatus =
placesResponse.getStatusLine();
                    if (placeSearchStatus.getStatusCode() == 200) {
                        HttpEntity placesEntity =
placesResponse.getEntity();
                        InputStream placesContent =
placesEntity.getContent();
                        InputStreamReader placesInput = new
```

```
InputStreamReader(placesContent);
                        BufferedReader placesReader = new
BufferedReader(placesInput);
                        String lineIn;
                        while ((lineIn = placesReader.readLine()) !=
null) {
                            placesBuilder.append(lineIn);
                        }
                    }
```

Il metodo sopra descritto salva la stringa di ricerca in una stringa di testo, effettua la richiesta tramite il metodo, nel caso la richiesta risulti positiva, (stato codice 200), il file verrà salvato in una stringa opportuna, che verrà successivamente letta per creare i dei marker.

Marker

All'interno della nostra applicazione, possiamo trovare i luoghi di interesse grazie alle api precedentemente nominate e in seguito localizzarli sulla mappa tramite appositi marker.



Per inserire i marker all'interno del nostro progetto abbiamo dovuto importare le seguenti librerie:

```
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
```

Defininire una variabile di tipo marker a cui andremo ad attribuire l'immagine che vogliamo assegnare ai nostri marker. L'immagine è stata precedentemente importata nel nostro progetto e gli abbiamo attribuito il nome di red_point.

```
  shopIcon = R.drawable.red_point;
```

Successivamente dovremo istanziare i marker nella nostra mappa, la posizione di ogni marker sulla mappa sarà scelta basandosi sulle coordinate del punto di interesse acquisite grazie alle api, ad ogni marker possiamo attribuire un title, il title sarà il nome del luogo puntato dal marker, anche questo acquisito tramite apposite api, ad ogni marker inoltre potremo attribuire un snippet, (una breve descrizione), la nostra descrizione sarà la via del "places".

```
places[p]=new MarkerOptions()
    .position(placeLL)
    .title(placeName)
    .icon(BitmapDescriptorFactory.fromResource(currIcon))
    .snippet(vicinity);
```

I nostri marker verrano creati dal metodo OnPostExecute della nostra classe AsyncTask precedentemente descritta, il metodo è il seguente:

```
JSONObject resultObject = new
JSONObject(result).getJSONObject("response");
               JSONArray placesArray =
resultObject.getJSONArray("venues");
                places = new MarkerOptions[placesArray.length()];
                for (int p=0; p<placesArray.length(); p++) {
                     LatLng placeLL=null;
                    String placeName="";
                    String vicinity="";
                    int currIcon = otherIcon;
                    try{
                       missingValue=false;
                        JSONObject placeObject =
placesArray.getJSONObject(p);
                      JSONObject loc =
placeObject.getJSONObject("location");
                     placeLL = new
LatLng(Double.valueOf(loc.getString("lat")),

Double.valueOf(loc.getString("lng")));
                       currIcon = shopIcon;
                       vicinity =
placeObject.getJSONObject("location").getString("address")+" - "+

placeObject.getJSONObject("location").getString("city")+" ";
                       placeName = placeObject.getString("name");
                       IDHashMap.put(placeName + vicinity,
placeObject.getString("id"));
```

Il metodo prende la stringa Json descritta precedentemente, la scorre tutta, ad ogni

elemento dell'array della stringa è associato un marker. Le coordinate del marker sono quelle dell'elemento della stringa e così anche per il nome e per la via.

Una volta cliccato su un marker si aprirà una finestra contenente delle indicazioni (nome e via), riguardanti il luogo di interesse.

Dato che nella nostra applicazione non andiamo ad inserire un singolo marker, ma una moltitudine di marker, cioè luoghi di interesse nelle nostre vicinanze, non possiamo usare una semplice variabile per salvare il marker, ma dobbiamo usare un array, abbiamo scelto di impostare un limite di elementi a 20.
Ogni volta che ci muoveremo nella mappa l'array viene pulito e vengono inseriti nel luogo richiesto i nuovi marker.

È possibile individuare places che non sono nelle nostre vicinanze tramite un tocco prolungato sulla mappa, il tocco prolungato richiamerà una funzione che cambierà la nostra posizione attuale, con quella in cui è avvenuto il tocco sulla mappa.

```
mMap.setOnMapLongClickListener(new GoogleMap.OnMapLongClickListener()
{
            @Override
            public void onMapLongClick(LatLng arg0) {
                lat = arg0.latitude;
                lng = arg0.longitude;
            }
        });
```

cliccare sulla finestra del marker contenente i dati del luogo di interesse, farà aprire un'altra activity riguardante il places.

```
 mMap.setOnInfoWindowClickListener(new
GoogleMap.OnInfoWindowClickListener() {
            @Override
            public void onInfoWindowClick(Marker marker) {
                Intent intent = new Intent(MapsActivity2.this,
FoursquareActivity.class);

intent.putExtra("ID",IDHashMap.get(marker.getTitle().toString()
+marker.getSnippet().toString()));
                intent.putExtra("queryString", placesSearchStr );
                startActivity(intent);
            }
        });
```

La nuova activity creerà una pagina, generata dinamicamente tramite api, la quale conterrà alcune informazioni e alcune foto riguardanti il luogo di interesse

nel caso le informazione riguardanti il luogo di interesse non vengano trovate dalle api, l' edittext conterrà la stringa "NON PERVENUTO!" (un esempio è l'immagine sopra, l'activity, non ricevendo una risposta riguardante il numero di telefono del Colosseo ha inserito la stringa "NON PERVENUTO!" nell'edittext).

Prima di parlare dell'activity riguardante i luoghi di interesse però dobbiamo fare un passo indietro e tornare all'activity riguardante la mappa.
Per far si che l'activity dei places effettui una ricerca del luogo di interesse è necessario che l'activity della mappa invii l'ID del luogo identificato dal marker.
Fare ciò non è intuitivo come può sembrare infatti il marker non può contenere dati "fantasma" ovvero il marker può contenere solo i dati visualizzati nello snippet, quindi se volessimo inserire l'id del places nel marker saremo costretti a visualizzare l'id nello snippet, fare ciò non sarebbe carino e nemmeno utile in quanto l'id di un places non fornisce un informazione utile per l'utente, ma serve solo all'app al fine di visualizzare il luogo nella prossima activity, come fare per evitare questo inconveniente quindi?

Per risolvere questo inconveniente andremo ad inserire nella nostra activity un hashtable,

```
private HashMap<String, String > IDHashMap = new HashMap<String,
String>();
```

contenente come indice il nome e la via del luogo visualizzato dal marker e come

contenuto l'ID del places indicato, ogni volta che la nostra app andrà ad inserire i marker nella mappa, aggiornerà automaticamente l'hashtable, andando ad inserire al suo interno anche gli id del marker.
Quando un utente clickerà su un marker per visualizzare l'activity del luogo, l'activity mappa andrà a ricercare nella nostra hashtable l'id del places, lo invierà all'activity successiva, la quale farà una richiesta tramite apposite api e visualizzerà infine il luogo ricercato.

La funzione tramite inseriamo l'elemento nella hastable è la seguente:

```
IDHashMap.put(placeName + vicinity, placeObject.getString("id"));
```

La funzione tramite la quale recuperiamo l'id inserito nell'hastable è la seguente:

```
IDHashMap.get(marker.getTitle().toString()
+marker.getSnippet().toString())
```

NOTA: Google Maps ha recentemente aggiunto due funzioni molto interessanti alle proprie API (ancora in fase Beta), non sono indispensabile per la nostra app, ma sono, comunque, un servizio aggiuntivo che può sempre tornare utile.
Per accedere ai nuovi servizi basta clickare su uno dei due tasti in fondo all'activity.

cliccando sul tasto a sinistra si aprirà un navigatore, il quale sarà già impostato con la strada per raggiungere il luogo indicato dal marker:

cliccando sul tasto a destra invece si aprirà una finestra nella quale sarà possibile inserire i dati per impostare il navigatore:

PlacesActivity

La seguente activity, come precedentemente descritto, ci permette di visualizzare i dati relativi ad un luogo di interesse, i dati saranno acquisiti tramite 3 API di seguito descritti, e mostrati all'utente tramite dei semplici Edittext, nel seguente capitolo vedremo nel dettaglio le 3 api utilizzate e il codice riguardante la visualizzazione delle immagini.

Google Places Library

Le Google Places Library consentono, alla tua applicazione, di ricercare i luoghi di interesse in una determinata posizione.

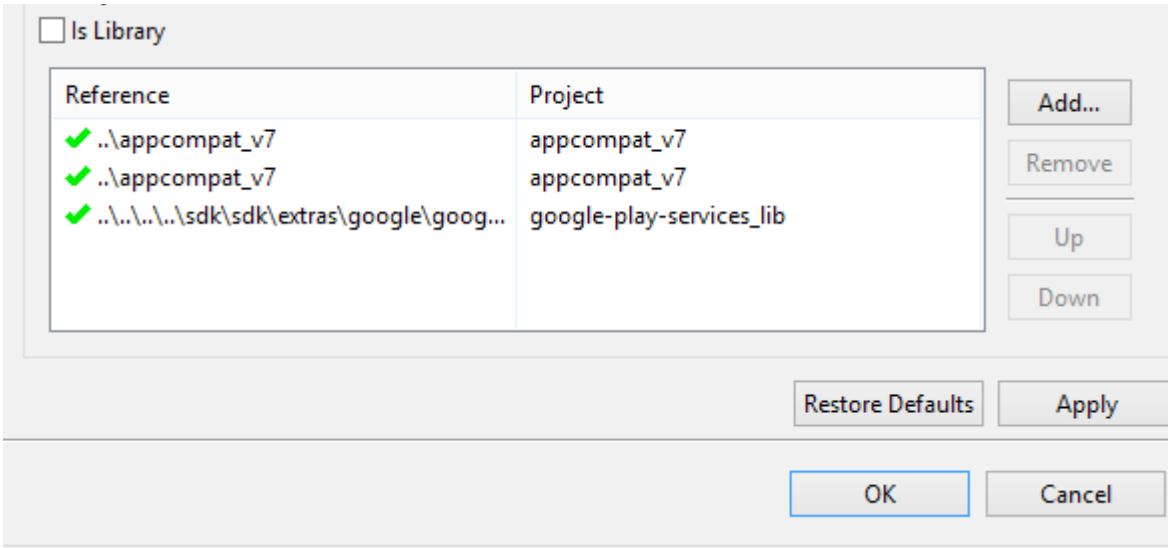
I punti di interesse, vengono chiamati da Google: "places".

Per poter utilizzare le Google places Library, dobbiamo registrare la nostra applicazione sulla console API di Google, richiedere la chiave di autenticazione ed inserirla nella nostra app.

Una volta fatto questo e abilitato il nostro dispositivo ad accedere alle API dalla console di google potremo ricercare i luoghi di interesse grazie a una richiesta in formato http.

La richiesta sarà analoga a quella di foursquare precedentemente descritta con l'unica differenza per i tag di risposta, infatti, dato che Google attribuisce dei tag diversi alla stringa Json in risposta, non possiamo utilizzare l'AsyncTask precedentemente descritto, ma dobbiamo crearne uno ad hoc.

Ovviamente oltre a registrare l'App sulla console di Google dovremo anche scaricare le librerie dal sito di Google ed importare nel nostro progetto le Google Play Library:



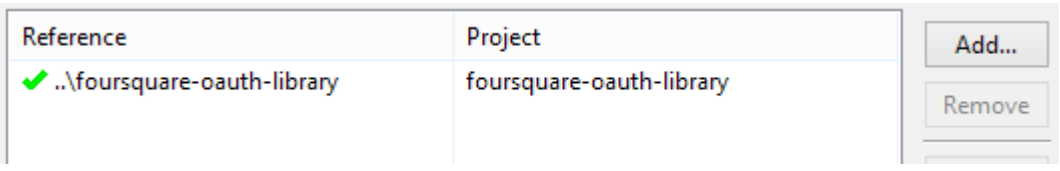
Di seguito un esempio di richiesta tramite Google Places Library:

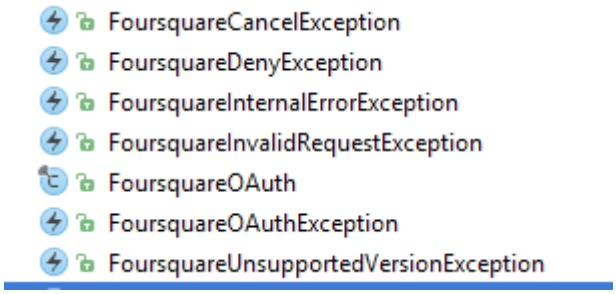
```
String placesSearchStr =
"https://maps.googleapis.com/maps/api/place/nearbysearch/" +
                        "json?location="+lat+","+lng+
                        "&radius=1000&sensor=true" +
                        "&types=food"+
                        "&key=Your_Google_key_here";//ADD KEY
```

Foursquare

Foursquare è una rete sociale basata sulla geolocalizzazione disponibile tramite web e applicazioni per dispositivi mobili. Foursquare è un'applicazione mobile e web che permette agli utenti registrati di condividere la propria posizione con i propri contatti.

Per quanto riguarda le API di Foursquare, analogamente alle api di Google, è stato necessario, in un primo momento, registrare la propria app sul sito di Fourquare, scaricare le librerie dal sito di Foursquare ed importarle nel nostro progetto.





Precedentemente, nella MapActivity, abbiamo utilizzato le API di Foursquare senza autenticazione, in questa Activity invece, ci occorrerà un autenticazione tramite chiave Oauth per effettuare la ricerca tramite ID.
È stato necessario, in un primo momento, ottenere quest'autenticazione, per farlo inviamo inizialmente una richiesta senza autenticazione:

```
https://foursquare.com/oauth2/authenticate
    ?client_id=YOUR_CLIENT_ID
    &response_type=code
    &redirect_uri=YOUR_REGISTERED_REDIRECT_URI
```

la richiesta viene inviata tramite un bottone apposito, la prossima activity mostra due bottoni, uno è il bottone per connettersi con le Foursquare api, l'altro è il bottone per passare alla map activity, è possibile accedere all'applicazione senza connettersi alle api di Foursquare, questo permetterà di visualizzare i luoghi di interesse sulla mappa (questo perchè la richiesta http di quell'activity non richiede autenticazione) ma non di visualizzare l'activity contenente i dettagli dei places (non è possibile farlo senza autenticazione Oauth), tuttavia l'applicazione permette di contattare le attività commerciali già nella propria chatlist.

Se l'utente clicca sul bottone verrà inviata la richiesta tramite l'app di Foursquare, la risposta sarà un token che verrà salvato in un apposita classe di tipo static alla quale sarà possibile accedere dalle prossime activity per inviare le richieste alle api di Foursquare.

```
public String getToken() {
    return token;
}

public void setToken(String token) {
    this.token = token;
}
```

Sarà possibile accedere al token tramite il metodo getToken, per impostarlo invece utilizziamo il metodo setToken.

Yelp API!

Yelp è un sito che recensisce tutto, dai ristoranti ai parrucchieri, dalle carrozzerie ai veterinari, ha però una piccola pecca, in Italia è arrivato da poco.
Le recensioni vengono scritte dagli utenti stessi, è sono consultabili da tutti.
Per quanto riguarda le Yelp API, il lavoro fatto è molto simile a quello di Foursquare, anche per queste API, come per le precedenti di Foursquare è necessaria un'autenticazione tramite chiave Oauth, ottenerla non è stato difficile.
Come per le Api precedenti ci siamo registrati al sito di Yelp! E abbiamo ottenuto i codici per l'utilizzo delle api, una volta fatto questo abbiamo scaricato ed installato il pacchetto contenente le api.
Fra le classi fornite dalle api di Yelp c'è n'è una che permette molto facilmente di richiedere l'autenticazione Oauth, tramite le seguenti righe di codice è possibile ottenere i permessi:

```
 OAuthService service = new
ServiceBuilder().provider(YelpV2API.class).apiKey(CONSUMER_KEY).apiSe
cret(CONSUMER_SECRET).build();
            Token accessToken = new Token(TOKEN, TOKEN_SECRET);
```

Come per le classi precedenti con Yelp dobbiamo fare una richiesta http, di seguito un esempio:

```
OAuthRequest request = new OAuthRequest(Verb.GET,
"http://api.yelp.com/v2/search");
            request.addQuerystringParameter("ll", lat + "," + lng);
            request.addQuerystringParameter("category", category);
```

La risposta sarà una lista contententi i posti di interesse che rispondono ai requisiti da noi inviati in richiesta, di seguito vi mostriamo un esempio:

```
Drowsy Poet at Innerlight
  655 Pensacola Beach Blvd
  Gulf Breezehttp://www.yelp.com/biz/drowsy-poet-at-innerlight-gulf-breeze
30.336641
-87.145103

Peg Leg Pete's
  1010 Fort Pickens Rd
  Pensacola Beachhttp://www.yelp.com/biz/peg-leg-petes-pensacola-beach
30.328065
-87.16436
```

GetPlacePhotos Class

In questa activity, oltre ai dati riguardanti l'attività è possibile visualizzare delle foto del luogo d' interesse, le foto vengono prese direttamente da Foursquare, cliccare sulla foto permetterà di scorrere le foto e visualizzare le successive foto di Foursquare, di seguito forniamo un esempio:

In Android, per visualizzare un immagine è necessario utilizzare un ImageView, cioè un form appositamente studiato per la visualizzazione delle immagini.
Come possiamo ottenere l'immagine da visualizzare però?
È possibile ottenere l'immagine tramite la richiesta fatta a Foursquare.
Nel nostro progetto potremo visualizzare due tipo differenti di immagine, il primo è l'immagine di default, che viene mostrata o nel caso la nostra activity non abbia ricevuto come risposta nessuna immagine da foursquare o nel caso stia ancora caricando le immagini ricevute, quest'immagine è già presente nel nostro progetto e non richiede di essere scaricata.

La seconda tipologia di immagini invece è composta da tutte le immagini che vengono ricevute come risposta dal database di Foursquare in seguito ad una richiesta e che quindi devono essere scaricate e poi visualizzate.
Eviteremo di entrare nel dettaglio del come scaricare un immagine tramite task asincrono, dato che il procedimento è molto simile a quelli visti negli esempi precedenti solo che con un tipo di file diverso, piuttosto preferiamo analizzare il tipo di richiesta da inoltrare a Foursquare.
Per ottenere le immagini da Foursquare dovremmo fare due richieste differenti, la prima conterrà l'id dell'attività, fra i dati relativi all'attività, foursquare darà come risposta anche un array Json contenente gli indirizzi delle immagini da scaricare, mentre nella seconda richiesta, utilizzeremo i link ottenuti dalla prima per ad ottenere le immagini vere e proprie.

Dopo la prima richiesta Foursquare darà come risposta un array contenenti elementi simili al seguente:

```
prefix: "https://irs2.4sqi.net/img/general/",
suffix: "/30345866_Wuc6EH1Yv2ocLImTmEgSrlb2KEr9iapPuEAwcxE0qHQ.jpg",
width: 720,
height: 720,
```

Per creare la stringa di testo a cui andremo a fare la richiesta sarà sufficiente prendere il prefisso, aggiungere la lunghezza inserire il segno "x", aggiungere l'altezza e infine aggiungere il suffisso, di seguito forniamo un esempio:

```
photoResult.add(placeObject.getString("prefix") +
placeObject.getString("width") + "x" +
placeObject.getString("height") + placeObject.getString("suffix"));
```

Ricordiamo però che la nostra app permette la visualizzazione di più immagini nella stessa activity, quindi dovremmo essere in grado di scaricare più immagini, salvarle sul dispositivo e poi permetterne la visualizzazione.

Abbiamo pensato che un procedimento simile potesse generare un grande spreco di memoria, facciamo un esempio per capire meglio il perchè.

Ipotizziamo che l' Utente1 stia cercando un ristorante nei paraggi, quindi tramite la nostra app vorrà consultare le pagine di più ristoranti senza però guardare tutte le loro foto. Se la nostra App scaricasse tutte le foto di un locale nel momento in cui si accede alla pagina di quel locale ci sarebbe un grande spreco di memoria, in quanto non tutte le foto scaricate verrebbero visualizzate. Per evitare questo spreco, appena entriamo nell'activity di un luogo di interesse effettuiamo solo la prima richiesta, (quella in cui otteniamo tutti gli url delle foto), il metodo doInBackground del primo asyncTask andiamo a salvare tutti gli indirizzi delle foto in un arrayList di stringhe, quindi una volta fatto questo visualizzeremo solo la prima immagine tramite un'altro asyncTask, ovvero l'immagine contenuta in posizione 0 del nostro arrayList, nel caso l'utente volesse visualizzare le immagini successive, dovrà fare click sulla foto e questa permetterà all'imageView di visualizzare l'immagine appartenente all'url che si trova nella posizione dell'indice successivo a quello attuale.

```
final int loader = R.drawable.loader;
        ImageLoader imgLoader = new
ImageLoader(getApplicationContext());
        if (img_url!=null) {
            imgLoader.DisplayImage(img_url.get(0), loader, imgVv);
        }
        imgVv.setOnClickListener( new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (img_url!=null) {
                    photoNumber = (photoNumber + 1) % img_url.size();
                    ImageLoader imgLoader = new
ImageLoader(getApplicationContext());
                    imgLoader.DisplayImage(img_url.get(photoNumber),
loader, imgVv);
                }
            }
        });
        Log.d("PhotoInUscita", img_url.size() + " ");
```

La variabile photoNumber mantiene l'indice della foto visualizzata, nel caso photoNumber superasse il numero di elementi dell'arrayList contenente gli url delle foto, l'immagine successiva da visualizzare sarebbe nuovamente la 0, per questo in photoNumber utilizziamo l'operazione modulo.

Il pulsante contatta, permette di contattare direttamente l'attività, nel caso l'attività sia già inserita nel database sarà possibile comunicare direttamente con il business tramite la nostra chat, nel caso invece l'attività non fosse presente nel nostro database il business verrà contattato tramite re-routering, vedremo nei prossimi capitoli i funzionamenti di queste due activity.

3. IL REROUTERING

Come visto nel capitolo precedente, una volta che l'utente avrà scelto un attività potra contattarla, a questo punto ci saranno due possibilità, la prima è che l'attività abbia installato la nostra app, in quel caso si aprirà una chat diretta con il business e sarà possibile comunicare come in una qualsiasi chat, il secondo caso invece è che l'attività non sia registrata nel nostro database e quindi dovremmo essere noi a trovare i loro contatti, (i contatti verrebbero acquisiti grazie alle api che abbiamo mostrato nel capitolo precedente).

La funzionalità di contattare un business che non ha ancora scaricato la nostra app attraverso vie alternative, l'abbiamo chiamata rerouteringg.

Tutta la parte di codice riguardante il rerouteringg lavora lato server ed è stata scritta in linguaggio php, vi sono diverse possibilità per contattare il cliente, vediamole nel dettaglio una a una.

E-mail

L'email non è certamente il metodo più veloce e pratico per rispondere ai messaggi, ma è uno dei più sicuri, infatti per registrarsi alle precedenti app, bisogna obbligatoriamente avere una mail, quindi saremo sempre sicuri di poter raggiungere un business tramite questo metodo.

Per inviare una mail al destinatario abbiamo usato il seguente script in Php

```
$txt = $_GET[testo];
$destinatario = $_GET[destinatario];
      $subject = "BusinessChat";
      $headers =  $UserMail+"\r\n";
      $headers .= $UserMail+"\r\n";
      $headers .= "Content-type: text; charset=iso-8859-1" . "\r\n";
      $headers .= "Organization: BusinessChat" . "\r\n";
      $headers .= "MIME-Version: 1.0" . "\r\n";
      $headers .= "X-Mailer: PHP". phpversion() ."\r\n";

      [... .......]

      $con = mysql_connect($databasehost,$databaseusername,
$databasepassword) or die(mysql_error());
      mysql_select_db($databasename) or die(mysql_error());
```

```
      $sql = "INSERT INTO mail(From, To, Testo, Data)
                  VALUES ('$utente','$destinario','$txt','2015-02-
11')";

      $risultato = mysql_query($sql);

      if (mysql_errno()) {
            header("HTTP/1.1 500 Internal Server Error");
            echo $query.'n';
            echo mysql_error();
      }else{
            //se non ci sono problemi con la query
            echo "mail inviata";
      }
} else echo "problem"; //se non invia la mail
[....]
```

Come si può vedere dal codice riportato qui sopra, lo script prende in input il messaggio del destinatario e il nome, effettua la connessione al database e inserisce nell'entità email chi ha inviato l'email, cosicché, quando il destinatario risponderà all'email, la risposta tornerà sempre ai nostri server e il mittente potrà continuare a comunicare dalla nostra app, evitando di dover accedere alla propria mail.

Una volta registrato il mittente nel database, lo script crea un email, come mittente ci saranno i nostri server, come oggetto il nome della nostra app e come corpo messaggio quello inserito dal mittente, lo script aggiungerà inoltre, in fondo al messaggio poche righe in cui verrà scritto che il messaggio è stato inviato tramite la nostra app.

Una volta fatto questo invierà il messaggio

Social Network

Per quanto riguarda i social network, abbiamo pensato potesse essere bello, permettere alla nostra app di comunicare con le attività commerciali che hanno un account Facebook, Twitter o Google+, ciò però non è sempre possibile, in quanto, spesso, le api non ci forniscono gli account dei social network, abbiamo comunque pensato potesse essere carino implementare questa funzione, anche se il bacino di utenti a cui inviare messaggi tramite social non è molto grande.

Twitter

Partiamo parlando dello sviluppo del rerouting per Twitter.Twitter, rispetto a Facebook e Google, funziona in una maniera un po' particolare, qui infatti non è possibile inviare un messaggio direttamente ad un'altro utente, (la funzione tramite social lo prevede, ma le api no), possiamo contattarlo, però, tramite Tweet, abbiamo implementato questa funzionalità in php, riteniamo però che non sia un buon metodo per permettere ad un client di contattare un business, in quanto il tweet è pubblico e ciò permetterebbe anche ad altri utenti di vedere la conversazione.
Crediamo quindi che sia giusto dare la possibilità all'utente di contattare il business tramite questa opzione, ma mettendo fra le note che il messaggio sarà pubblico e quindi leggibile da tutti.

```
<a href="https://twitter.com/intent/tweet?screen_name=<?php echo $nome; ?>"
      class="twitter-mention-button"
      data-related="<?php echo $nome; ?>">Tweet to @<?php echo $nome; ?></a>
<script>!function(d,s,id)
{var js,fjs=d.getElementsByTagName(s)
[0],p=/^http:/.test(d.location)?'http':'https';if(!d.getElementById(id))
{js=d.createElement(s);js.id=id;js.src=p+'://platform.twitter.com/widgets.js';f
js.parentNode.insertBefore(js,fjs);}}(document, 'script', 'twitter-
wjs');</script>
<!--
<a class="twitter-timeline" href="https://twitter.com/ValpianiA" data-widget-
id="539425638533595136">Tweet di @ValpianiA</a>
<script>!function(d,s,id){var js,fjs=d.getElementsByTagName(s)
[0],p=/^http:/.test(d.location)?'http':'https';if(!d.getElementById(id))
{js=d.createElement(s);js.id=id;js.src=p+"://platform.twitter.com/widgets.js";f
js.parentNode.insertBefore(js,fjs);}}(document,"script","twitter-
wjs");</script>
```

Google +

Per quanto riguarda Google+ è Google stessa a fornirci due script per poter comunicare tramite le proprie api, la comunicazione tramite Google+ può essere comoda , però avviene tramite broswer e non tramite la nostra app, potremo, quindi, inviare il messaggio tramite la nostra app, ma non potremo gestire la conversazione tramite la nostra app.

Abbiamo implementato il reroutering per Google + trami script javascript:

```
<script src="https://apis.google.com/js/platform.js" async defer>
  {lang: 'it', parsetags: 'explicit'}
</script>
[...]
<script type="text/javascript">
      (function() {
       var po = document.createElement('script'); po.type =
'text/javascript'; po.async = true;
       po.src = 'https://apis.google.com/js/client:plusone.js';
       var s = document.getElementsByTagName('script')[0];
s.parentNode.insertBefore(po, s);
     })();

<div class="g-plus" data-action="share"></div>
<script type="text/javascript">gapi.plus.go();</script>
```

Come si può vedere è possibile inviare messaggio tramite broswer a Google+, il testo

del messaggio verrà inserito nella finestra di Google e ci sarà possibile personalizzare l'icona.

C'è un grave problema riscontrato però, tramite Google+ non ci sarà possibile comunicare con persone che non abbiamo negli amici, la situazione sarà simile per Facebook, ma adesso la andremo a vedere nel dettaglio.

Facebook

Il funzionamento del rerouting tramite Facebook è molto simile a quello tramite Google+, di seguito inseriamo il codice:

```
[...]

<script> //codice per sdk
  window.fbAsyncInit = function() {
    FB.init({
      appId      : '1583851268503533',
      xfbml      : true,
      version    : 'v2.2'
    });
  };

  (function(d, s, id){
     var js, fjs = d.getElementsByTagName(s)[0];
     if (d.getElementById(id)) {return;}
     js = d.createElement(s); js.id = id;
     js.src = "//connect.facebook.net/en_US/sdk.js";
     fjs.parentNode.insertBefore(js, fjs);
   }(document, 'script', 'facebook-jssdk'));
</script>

[...]
```

Le api di Facebook ci danno la possibilità di contattare l'utente tramite Broswer, come visto precedentemente però non sarà possibile contattare l'utente se questo non è fra i nosti amici.

Un esempio di finestra di dialogo tramite Facebook sarà la seguente:

SMS

Per quanto riguarda gli SMS sarebbero il metodo migliore per il rerouting del messaggio, nonostante ciò, l'invio di SMS dalla nostra applicazione comporta anche molte difficoltà.
In primo luogo, non tutte le attività danno un numero di telefono e la grande maggioranza delle attività commerciali che forniscono un numero di telefono, danno quello del telefono fisso e non di un telefono mobile. Bisogna quindi, in un primo momento accertarsi se il numero di telefono fornito dalle api è di un telefono mobile o fisso, in un secondo momento poi andremo a inviare il messaggio. L'invio dei messaggi da parte della nostra applicazione tramite server non è una cosa molto semplice, abbiamo pensato di farlo attraverso api di terze parti, al momento però non ci è stato possibile lavorare con queste api, quindi inseriamo il reroutering tramite sms negli sviluppi futuri, al momento però ci limitiamo a dare una breve descrizione riguardante le api più importanti che ci permettono di svolgere questa funzione.

Elenchiamo brevemente alcuni fra i servizi che ci permettono di risolvere questa

funzione:

- Senza costo: Kannel, EarthSMS, XMPP (ipoteticamente questo protocollo permetterebbe anche di inviare sms, dobbiamo ancora conoscere tutte le sue funzioni però).
- A pagamento: Skebby, Vianett.

In futuro valuteremo le api , tenendo in considerazioni i costi e le funzionalità e decideremo come sviluppare al meglio questa funziona, al momento siamo propensi per svilupparla o tramite le api di Kannel o tramite il protocollo XMPP.

4. CHAT

La messaggistica istantanea (IM, Instant Messaging) rappresenta la possibilità, diventata al giorno d'oggi una necessità, di contattare qualcheduno, ovunque egli si trovi, essendo certi che il proprio messaggio verrà recapitato in maniera immediata, facendo uso della rete.

Al giorno d'oggi l'uso degli SMS come veicolo di tale intento è deprecato in favore di app accattivanti e ricche di funzionalità, in grado ormai di portare al destinatario ben più di un messaggio; tra contratti internet vantaggiosi e accesso a wi-fi gratuiti, si è quasi sempre in grado di essere "online". Si può così condividere l'esperienza di tutti i giorni con gruppi di amici attraverso foto, video, registrazioni vocali e molto altro: nella forma pratica, allo stato dell'arte, si è in grado di comunicare con un amico semplicemente interagendo con uno schermo, pur trovandosi in luoghi totalmente diversi, anche all'altro capo del mondo.

I più diffusi protocolli riguardanti l'IM sono:

Open-Source	**Privati**
XMPP	Yahoo! Messenger
SIMPLE	Windows Live Messenger
PRIM	Skype

Ovviamente intorno all'Instant Messaging sono nate numerose realtà, tantissime app che si sono distinte per l'interpretazione più o meno avanzata di un particolare protocollo, o la creazione di uno proprietario che rappresenti al meglio determinate funzionalità. Ad oggi, le soluzioni più popolari sono Hangouts, Whatsapp, Telegram, Line, Viber, ognuna nota per determinate caratteristiche che le distinguono, seppur di poco, dalle concorrenti.

Protocolli

Partiamo analizzando brevemente i protocolli, per poi soffermarci su quello utilizzato da noi:

WhatsApp

Whattsapp API:

Whatsapp usa una versione modificata di XMPP, detta FunXMPP, ovviamente di loro proprietà, e non documentata ufficialmente: per essere precisi, non esiste una documentazione ufficiale/pubblica che illustra il corretto funzionamento di FunXMPP. Ne deriva che si trovano molteplici guide non ufficiali in giro per la rete che tentano, spesso per vie non proprio legali, di spiegare come funziona suddetta variazione al protocollo XMPP(che invece è documentato in ogni sua forma essendo open-source).

Si suppone si tratti di una modifica sostanziale al tradizionale metodo di trasportare termini specifici del linguaggio XML, trasformandoli in singoli byte e diminuendo quindi in maniera significativa i dati I/O da/verso il proprio cellulare.

Pusher:

Qui parliamo di un servizio in grado di gestire funzionalità presenti nelle app "realtime" con poche righe di codice, ed in teoria, quindi, con minore perdita di tempo rispetto ad un lavoro svolto autonomamente. Oltre a questo, consente di mantenere la parte di back-end sui loro server(ovviamente a pagamento) quindi consentirebbe di risparmiare parecchio tempo nella(seppur eventuale) gestione dei dati utenti.

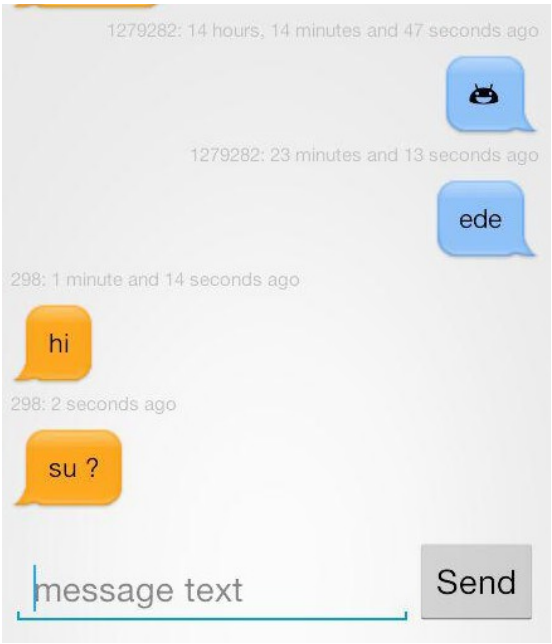
XMPP:

Circa il 60% delle applicazioni IM utilizza il protocollo open-source XMPP (Extensible Messaging and Presence Protocol).
Questo protocollo, aperto ed altamente personalizzabile, consente di collegarsi alla rete Jabber e di fare uso delle sue funzionalità: ogni utente acquisisce quindi un account Jabber, con cui si connette alla rete visualizzando tutti gli altri account con cui ha stretto "amicizia".
XMPP permette inoltre di creare un proprio server, gestisce funzionalità di sicurezza attraverso i protocolli SASL e TLS, e il suo essere uno standard aperto consente a chiunque di modificarne la struttura di base, aggiungendo estensioni o rimuovendo specifiche (Whatsapp stessa fa uso di una versione modificata di XMPP, chiamata FunXMPP), allargandone l'utilizzo anche all'ambito videoludico, file sharing, VoIP e altro ancora.

XMPP è quindi un punto di riferimento per
chiunque voglia approcciarsi all'ambiente mobile chat da sviluppatore. Data la sua natura, non esiste una versione precisa di XMPP che si possa definire "standard",

poiché ognuno, acquisita la giusta maestria, modella il protocollo in base alle sue necessità.
Abbiamo ritenuto che XMPP fosse la scelta adatta a noi, sia per le tante funzionalità che mette a disposizione dei programmatori sia per la grande quantità di documentazione e codice consultabile online, abbiamo quindi sviluppato la nostra chat basandoci su questo protocollo senza inventare nulla di nuovo.

Vediamo di seguito alcune funzioni principali di XMPP:
Per utilizzare una chat è necessario selezionare un utente con il quale si vuole chattare e successivamente aprire una conversazione con quell'utente, per quanto riguarda il selezionamento dell'utente con il quale si vuole chattare, abbiamo già visto nei precedenti capitoli come ricerchiamo con le mappe il luogo con cui ci interessa comunicare, quindi in questa parte non tratteremo nuovamente la ricerca degli utenti all'interno della nostra app, ma parleremo semplicemente delle funzioni necessarie per permettere a un client di comunicare con un business.



Quando andiamo a creare una chat 1-1, XMPP non ci obbliga a chiamare una funzione, questa viene creata automaticamente all'invio del primo messaggio, tuttavia è possibile creare una chat 1-1 senza l'invio di nessun messaggio con la seguente chiamata a funzione:

```
QBPrivateChatManager                    privateChatManager                    =
QBChatService.getInstance().getPrivateChatManager();
privateChatManager.createDialog(opponentId,                                   new
QBEntityCallbackImpl<QBDialog>() {
    @Override
    public void onSuccess(QBDialog dialog, Bundle args) {

    }

    @Override
    public void onError(List<String> errors) {

    }
});
```

Una volta creata la chat con un'altro utente, entrambi saranno abilitati ad inviare messaggi in quella chat, per poter inviare un messaggio si usa la seguente funzione, nella quale andremo ad inserire l'id dell'utente con cui vogliamo comunicare e il corpo del messaggio:

```
QBMessageListener<QBPrivateChat> privateChatMessageListener = new
QBMessageListener<QBPrivateChat>() {

   [.....]
```

```
};

QBPrivateChatManagerListener privateChatManagerListener = new
QBPrivateChatManagerListener() {
    @Override
    public void chatCreated(final QBPrivateChat privateChat, final boolean
createdLocally) {
        if(!createdLocally){
            privateChat.addMessageListener(privateChatMessageListener);
        }
    }
};

QBChatService.getInstance().getPrivateChatManager().addPrivateChatManagerList
ener(privateChatManagerListener);

Integer opponentId = 45;

try {
    QBChatMessage chatMessage = new QBChatMessage();
    chatMessage.setBody("Hi there!");
    chatMessage.setProperty("save_to_history", "1"); // Save a message to
history

    QBPrivateChat privateChat = privateChatManager.getChat(opponentId);
    if (privateChat == null) {
        privateChat = privateChatManager.createChat(opponentId,
privateChatMessageListener);
    }
    privateChat.sendMessage(chatMessage);
} catch (XMPPException e) {

} catch (SmackException.NotConnectedException e) {

}
```

Per ottenere la cronologia di una particolare chat andremo ad utilizzare la seguente chiamata a funzione, dove bisogna specificare l'id della chat e il numero di pagine che vogliamo recuperare da quella chat:

```
QBDialog qbDialog = new QBDialog("53cfc593efa3573ebd000017");

QBRequestGetBuilder requestBuilder = new QBRequestGetBuilder();
requestBuilder.setPagesLimit(100);

QBChatService.getDialogMessages(qbDialog,   customObjectRequestBuilder,   new
QBEntityCallbackImpl<ArrayList<QBChatMessage>>() {
    @Override
    public void onSuccess(ArrayList<QBChatMessage> messages, Bundle args) {

    }

    @Override
    public void onError(List<String> errors) {

    }
});
```

Per eliminare un messaggio dalla propria chat si può usare la seguente funzione (ATTENZIONE: utilizzare questa funzione cancellerà il messaggio dalla propria chat,

ma non da quelle altrui):

```
Set<String> messagesIds = new HashSet<String>() {{
    add("546cc8040eda8f2dd7ee449c"); add("546cc80f0eda8f2dd7ee449d");
}};

QBChatService.deleteMessages(messagesIds, new QBEntityCallbackImpl<Void>() {
    @Override
    public void onSuccess() {

    }

    @Override
    public void onError(List<String> errors) {

    }
});
```

CONCLUSIONI

L'obbiettivo di questa tesi è quello di mostrare le principali funzionalità della nostra applicazione, il loro funzionamento e il codice sviluppato che vi è dietro.

L'applicazione per ora sviluppata non è ancora commercializzabile, vi è ancora molto lavoro da fare prima che questa applicazione diventi competitiva con le sue concorrenti.

A mio parere la grafica non risulta ancora accattivante, l'applicazione non è performante e possiamo implementare molti miglioramenti per quanto riguarda la sicurezza.

Nonostante ciò, sviluppare quest'applicazione mi ha permesso di accumulare molta esperienza e fare pratica con lo sviluppo di applicazioni per android, sia lato client che lato server, mi ha permesso di venire a contatto con una grande varietà di strumenti di lavoro, ho utilizzato diversi linguaggi di programmazione e ide.

Lavorare su quest'applicazione mi ha inoltre permesso di capire l'approccio che bisogna avere quando si sviluppa un'applicazione per mobile partendo da zero, mi ha fatto conoscere cos'è un mock-up e mi ha permesso di fare pratica con la progettazione di applicazioni un po' più complesse, inoltre, oltre alle semplici competenze tecniche, lavorare su un progetto di gruppo mi ha permesso di familiarizzare e capire meglio come funziona il processo per la creazione di un software in un team.

È stata un esperienza interessante, non priva di problemi, che ci hanno stimolato e aiutato ad andare oltre alle nostre conoscenze.

Sviluppi futuri

Come accennato prima, la nostra applicazione ha ancora bisogno di molto lavoro.

In un prima fase pensiamo sia necessario correggere i Bug che si sono presentati, e migliorare le perfomance della nostra applicazione, una volta completato questo passaggio, cercheremo di rendere la grafica più accattivante, magari chiedendo aiuto a qualcuno più esperto di noi in questo ambito.

Dobbiamo anche inserire la possibilità all'utente di registrarsi ed effettuare il login tramite social network, provvederemo a fare questo il prima possibile, anche perché, grazie alle api messe a disposizione dai social network, quali Google, Facebook e Foursquare, è molto semplice aggiungere queste funzionalità.

Una volta completate queste modifiche e dopo aver effettuato i giusti controlli sulla nostra applicazione, provvederemo ad implementare nuove funzionalità.

La prima funzionalità che andremo ad implementare sarà pensata per le aziende, vogliamo dare la possibilità ai nostri utenti business di pubblicizzare la propria attività, lo renderemo possibile grazie a messaggi in broadcast ai propri clienti, permetteremo inoltre alle app degli utenti business di inviare pubblicità, in automatico, ad un utente client, ogni volta che questo si trova in prossimità di quel business, pubblicità di questo tipo permetterebbe al business di pubblicizzare i propri prodotti del giorno o offerte speciali della settimana.

Inoltre vogliamo implementare una lista con le domande più rivolte ad un attività, in modo tale che se determinate domande sono già presenti il client non dovrà chiederle nuovamente.

Pensiamo di creare classifiche riguardanti le attività commerciali più contattate e quelle con i ranking migliori e vogliamo dare la possibilità agli utenti, magari in un futuro, di scrivere recensioni sulle attività.

Sitografia

- http://xmpp.org/
- https://developers.facebook.com/
- https://developer.foursquare.com/
- https://developers.google.com/maps/
- https://developers.google.com/maps/documentation/javascript/places
- http://quickblox.com/
- https://dev.twitter.com/
- https://www.yelp.com/developers/documentation
- http://www.openstreetmap.org/#map=5/51.500/-0.100
- http://www.html.it/
- http://it.wikipedia.org/
- http://php.net/
- http://www.mysql.it/
- https://www.android.com/
- https://www.apachefriends.org/it/index.html
- http://android.hdblog.it/
- http://developer.android.com/index.html
- http://www.anddev.it/
- http://android-developers.blogspot.it/