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

PROGETTAZIONE ED IMPLEMENTAZIONE DI UNA APP PER L'ANALISI DI RUBRICHE TELEFONICHE

Relazione finale in
Algoritmi e Strutture Dati

Relatore:
Prof. Luciano Margara

Presentata da:
Vitrini Luca

Sessione I
Anno Accademico 2014/2015

Indice

Indice 1

Prefazione 3

Capitolo 1 – Smartphone 5

1.1 Cos'è 5

1.2 Sistemi operativi per smartphone 7

1.2.1 Android 8

1.2.2 iOS 9

1.2.3 Windows Phone 9

Capitolo 2 – Ambienti di sviluppo 12

2.1 Android Studio 12

2.2 Visual Studio 14

Capitolo 3 – Analisi e progettazione 17

3.1 Web API 18

3.2 SQL 20

3.3 Metodi API 21

3.4 Installazione server 24

Capitolo 4 – Mockup 25

Capitolo 5 – Codice del progetto 28

Conclusioni 63

Sitografia 64

Prefazione

Negli ultimi anni la grande diffusione di cellulari di ultima generazione ha generato l'interesse di una parte del mondo informatico nella produzione di applicazioni pensate apposta per questi dispositivi. Il costo generalmente ridotto rispetto ad un computer e la portabilità ne hanno sancito il successo, decretando la creazione di sistemi operativi appositi e servizi adeguati alle necessità dell'utenza.
Le opportunità disponibili nell'ambito personal computer, sia per i singoli programmatori che per le grandi aziende, si sono quindi ripresentate in dimensione "ridotta" ma dalle proporzioni eccezionali. Ognuno dispone infatti di un cellulare, dunque ognuno è un potenziale cliente; il numero di funzionalità disponibili all'utente è limitato unicamente dalla fantasia degli sviluppatori.

Questo lavoro di tesi si concentra su una delle funzioni essenziali dei cellulari, la rubrica, la possibilità cioè di memorizzare i dati delle persone che conosciamo, prima fra tutte il loro numero di telefono.
In particolare, l'intento è quello di mostrare come veniamo registrati nella propria rubrica dalle persone a cui affidiamo il numero del nostro dispositivo, un modo simpatico per vedere come siamo considerati dai nostri conoscenti, amici e familiari.
L'applicazione verificherà la presenza nel circuito dell'app delle persone presenti nella rubrica dell'utente, e gli mostrerà i nomi a lui dati dalle suddette persone.

Vediamo ora la struttura secondo cui è organizzato questo elaborato di laurea.

Nel primo capitolo viene trattato, attraverso una breve introduzione, il mondo degli smartphone, con un approfondimento sui sistemi operativi più in voga per questo dispositivo.

Nel secondo capitolo vengono esaminati gli strumenti necessari allo sviluppo, in tutte le sue fasi, di un'applicazione per dispositivi mobile.

Il terzo capitolo si focalizza sulle tecnologie utilizzate nella creazione dell'applicazione, oltre a presentare le metodologie atte ad una buona programmazione in ambito client-server.

Con il quarto capitolo l’attenzione si concentra sul funzionamento dell’applicazione, mostrandone anche gli aspetti grafici attraverso la pratica del mockup.

L’analisi approfondita dell’applicazione viene svolta nel capitolo quinto, dove è presente integralmente il codice dell’applicazione lato client.

Il sesto capitolo si presta alle conclusioni in seno a questa tesi.

Infine sono presentate la bibliografia utile alla scrittura dell’elaborato in questione e i ringraziamenti.

Capitolo 1
Smartphone

1.1 Cos'è

Un telefono cellulare (o semplicemente cellulare o telefonino) è un dispositivo di discrete dimensioni pensato per mettere in comunicazione due entità distanti tra loro. La sua prima ed unica funzione fu quindi inizialmente quella di svolgere chiamate vocali.
Lo sviluppo di differenti tecnologie di comunicazione ha permesso l'accesso a nuove funzioni di utilizzo:

- con il passaggio dal segnale analogico a quello digitale è stato reso possibile l'invio di semplici messaggi di testo chiamati SMS, e successivamente la registrazione e visualizzazione di materiale audio e video;
- il GPRS ha permesso di realizzare il trasferimento dati a commutazione di pacchetto: ciò ha aperto le porte all'utilizzo di Internet su dispositivi mobili, ha permesso di inviare e ricevere e-mail, inviare e fare streaming di materiale audio e video;
- con l'UMTS aumenta la velocità di trasmissione e ciò rende possibile una serie di servizi multimediali di alto livello tra cui le videochiamate, mentre con successivi sviluppi compaiono i primi tivufonini che consentono di visionare il segnale TV su cellulare.

Per poter supportare tutte queste funzionalità i dispositivi mobili si sono trasformati, necessitando di caratteristiche hardware e prestazionali che, negli ultimi anni, sono accomunabili ai personal computer (seppur di fascia bassa).
Questo progresso ha coinvolto ogni singolo aspetto, dalle dimensioni al sistema di comunicazione. Lo sviluppo tecnologico ha infatti permesso col tempo di ridurre drasticamente le dimensioni dei materiali hardware necessari al suo funzionamento, ma l'utilizzo sempre più frequente in ambito social e multimediale, unito ad un cambiamento nel modo di interagire con il dispositivo, passando dall'utilizzo della tastiera al più moderno touchscreen, ha portato ad una significativa ridefinizione

della struttura del corpo stesso del cellulare, che ad oggi risulta piatto ed allungato, con un grande schermo in alta definizione e pochi tasti essenziali nelle fasce laterali.

Figura 1 - Evoluzione Del Cellulare

A questo va aggiunto una serie di elementi, primi fra tutti i sensori, i quali non avrebbero alcun motivo di esistere su di un personal computer ma che risultano qui utili a fini anche videoludici. Misurare i passi compiuti, l'orientamento e l'inclinazione del display, la temperatura, possiamo anche utilizzare il nostro dispositivo come metal detector; sono insomma accessibili funzionalità incredibili, che spesso risultano inutili all'utente medio ma latenti nel dispositivo.
Il termine "cellulare" non basta più, serve un neologismo che rispecchi le reali caratteristiche e finalità con cui vengono utilizzati nell'ambito quotidiano questi apparecchi: **Smartphone**.
A nostra disposizione vengono messi una capacità di calcolo, memoria e di connessione notevole, il tutto adeguatamente supportato da un sistema operativo creato appositamente per i dispositivi mobili. Possiamo accedere a tantissime funzionalità dal momento dell'acquisto,
e aggiungerne di nuove attraverso le cosidette web app, scaricate dai rispettivi market di vendita.
La sua diffusione ha portato il mercato a convogliare in esso le più disparate necessità, cosicché con uno smartphone in tasca si possa avere non solo un ottimo cellulare in grado di svolgere le più elementari operazioni, ma anche una fotocamera capace di scattare ottime foto, un lettore multimediale per ascoltare musica o visionare video, un registratore audio, la possibilità di connettersi alla rete, una console videoludica per svagarsi nel tempo libero, e chi più ne ha più ne metta.

1.2 Sistemi operativi per smartphone

Nell'ambito della telefonia il termine "sistema operativo" compare negli anni '90: la comparsa dei primi touchscreen, dei primi "smartphone", vien di pari passo con la necessità di spostare le capacità prestazionali dei computer sui piccoli dispositivi di uso quotidiano. Sin dai primi anni 2000 diverse aziende tra cui Nokia, Microsoft e Blackberry creano i loro SO proprietari, generando un'efficace interesse del grande pubblico verso i cellulari di ultima generazione.
L'anno della svolta definitiva è il 2007, quando due aziende informatiche fino a quel momento orientate principalmente allo sviluppo di servizi e hardware per personal computer, Google ed Apple, si inseriscono prepotentemente nel settore degli smartphone rilasciando due SO destinati a monopolizzare il mercato negli anni seguenti, rispettivamente Android e iOS. Grazie all' ingresso di queste nuove "leve" la funzione "telefono" diventa progressivamente secondaria, e l'attenzione dell'utente viene indirizzata verso "l'applicazione" (o sempre più frequentemente **app**): ogni sviluppatore crea un suo negozio virtuale online, spesso preinstallato sul dispositivo, rendendo accessibili al fruitore un numero sempre più grande di servizi, dai film alle riviste, dai brani musicali ai libri, passando ovviamente per programmi e giochi. Lo smartphone diventa così uno strumento utilissimo nella vita di tutti i giorni, una tendenza, uno stile di vita, indivisibile da noi perché tramite del nostro rapporto con l'esterno nel suo aspetto social.

Per capire le dimensioni di questo fenomeno basta far riferimento all'anno 2014: il numero di utenti mobile ha superato quello relativo all'utenza computer, sono stati venduti più di 1 miliardo e mezzo di smartphone, di cui più di 1 miliardo sotto il SO Android, che detiene quasi l'80% del mercato. iOS (16%) mantiene il suo range d'utenza, mentre i restanti sistemi operativi, in primi Windows Phone, si dividono le briciole.
Vediamo ora in dettaglio questi tre sistemi operativi, quali sono le loro peculiarità, i loro pregi e difetti.

1.2.1 Android

Figura 2 - Logo Android

Android è un sistema operativo open source inizialmente sviluppato da Android Inc. e successivamente acquistato da Google, rilasciato nella sua prima versione il 23 settembre 2008 e distribuito sotto la licenza libera Apache 2.0.

Progettato per essere utilizzato su di un gran numero di dispositivi mobili diversi, tra cui smartphone, orologi da polso, tablet, automobili ed occhiali, vanta una community particolarmente attiva, data la sua natura open source che consente la modifica e distribuzione del codice sorgente.

La piattaforma è basata sul kernel Linux, usa il database SQLite, la libreria dedicata SGL per la grafica bidimensionale e supporta lo standard OpenGL per la grafica tridimensionale.
Le applicazioni vengono eseguite tramite la Dalvik Virtual Machine che è a tutti gli effetti una Java Virtual Machine adattata ai dispositivi mobili.
Questa macchina virtuale è ottimizzata per sfruttare la poca memoria presente nei dispositivi mobili, consente inoltre di utilizzare diverse istanze della macchina virtuale contemporaneamente e nasconde al sistema operativo sottostante la gestione della memoria e dei thread.
Per programmare in Android non è necessario avere un dispositivo con questo SO installato, è possibile fare uso di emulatori in grado di simulare il comportamento delle app sviluppate.
Le applicazioni Android, dette Java-based, sono caratterizzate dalla presenza di un file dichiarativo, *AndroidManifest.xml*, che descrive dettagliatamente l'app al dispositivo, illustrando le risorse necessarie al suo corretto funzionamento; inoltre, la mancanza di un entry-point, cioè un punto di partenza (come avviene in C con il *main*), consente al programmatore di gestire in maniera indipendente le varie parti del programma e al sistema di ottimizzare le risorse.

1.2.2 iOS

Figura 3 - iOS Logo

Iphone OS (poi diventato iOS) è il sistema operativo creato dall'azienda statunitense Apple per le sue periferiche iPhone, iPad, iPod : la prima versione risale al 29 giugno 2007. Derivazione di Unix, usa un micro-kernel XNU Mach basato su Darwin OS. Il rilascio del primo dispositivo supportato, l'iPhone, è considerato la svolta decisiva nel mercato degli smartphone verso la diffusione e il successo di cui gode il settore al giorno d'oggi.
La struttura chiusa e il modello del sorgente di tipo proprietario, differente dall'approccio open-source di Android, ha definito questi due SO come rivali incontrastati della porzione mobile dell'informatica.
Per lo sviluppo di app iOS è necessario utilizzare un computer Mac, e la sottomissione dell'app nell'Apple Store deve passare attraverso una valutazione di sicurezza.
Questo tutela l'utente finale da programmi eventualmente dannosi o dall'utilizzo problematico, ma ne limita anche la libertà. Il linguaggio utilizzato nella programmazione di app iOS è l'Objective-C.

1.2.3 Windows Phone

Windows Phone è il sistema operativo rilasciato da Windows per dispositivi mobili il 21 ottobre 2010, e introduce la nuova interfaccia chiamata Metro UI che verrà poi allargata all'intero ecosistema Microsoft. Lo stile pulito e innovativo (le cosiddette "piastrelle") e la sicurezza del sistema operativo hanno da subito reso Windows Phone una valida alternativa ad Android e iOS, ma il divario tra lo Store proprietario e quello dei rivali per quantità di app, dovuto principalmente all'ingresso nel mercato ben 4 anni dopo (divario decisamente assottigliato al giorno d'oggi, grazie anche alla partnership con Nokia), ha limitato finora la sua affermazione nel settore degli smartphone, pur restando il terzo SO per diffusione.

Windows Phone è protagonista assoluto nel processo di sincronizzazione dei vari dispositivi Windows, procedimento iniziato con Windows 8 e che vedrà il suo proseguimento con il rilascio di Windows 10.
L'intento è quello di consentire all'utente di connettersi al proprio profilo da qualsiasi piattaforma Windows, che esso sia un computer, uno smartphone, un tablet, un laptop.

Figura 4 - Windows Phone Logo

Il vantaggio che offre ai programmatori, potendo sfruttare l'enorme bagaglio Microsoft, è di poter scrivere codice in molteplici linguaggi (Visual C++, Visual C, HTML5/Javascript) e fare uso di differenti ambienti di sviluppo, primo fra tutti Visual Studio.

Dei tre sistemi operativi presi in esame, è Android il più accessibile in termini di sviluppo e risultati, gettando un occhio anche al bacino d'utenza e la diffusione tra i dispositivi mobili,
oltre alle mie personali conoscenze pregresse.
Questo giustifica l'interesse per lo sviluppo di **Nickname** in ambiente Android.

Capitolo 2
Ambienti di sviluppo

Nella programmazione di una applicazione, indipendentemente dalle sue dimensioni o dal suo scopo, è conveniente fare uso di determinati strumenti che permettano allo sviluppatore di concentrarsi sulla scrittura del codice del programma e sul suo funzionamento, evitando di occuparsi di problemi secondari che risultino alla lunga una semplice perdita di tempo.
Con la diffusione dei personal computer, e ancor più con la moderna distribuzione capillare di dispositivi mobili, chiunque può non solo immaginare un'applicazione e desiderarla installata sul proprio apparecchio, ma può decidere di crearla lui stesso.
Per venir incontro a queste schiere di sviluppatori alle prime armi sono nati, in pochi anni, prodotti inizialmente pensati per gli affiliati al settore ma che sono diventati presto di uso comune anche per l'utente più casual, fino ad essere utilizzati in ambito educativo.
Questi Ambienti di Sviluppo Integrato (Integrated Development Environment, IDE), consistono quindi in un software che, in fase di programmazione, aiuta nella scrittura del codice sorgente di un programma. Nei fatti, l'IDE compila ed interpreta il programma e provvede all'eliminazione dei bug, oltre ad avvisare il programmatore quando questo scrive codice sorgente errato.
I più noti sono Eclipse, Visual Studio, Android Studio, NetBeans.
Per lo sviluppo di **Nickname** ho fatto uso di più ambienti di sviluppo, ognuno utile alla scrittura di una determinata parte di codice: in particolare Android Studio e Visual Studio.
Vediamoli in dettaglio.

2.1 Android Studio

Per la programmazione in Android, e l'applicazione sviluppata non ha fatto eccezione, è fondamentale l'installazione di tre elementi distinti ma il connubio dei quali permette un approccio completo allo scrittura del codice: il JDK, l'Android SDK e un IDE adeguato.
Il Java Development Kit è l'insieme di strumenti utili alla programmazione in linguaggio Java,

il linguaggio principale in ambiente Android; ne deriva che la sua presenza è obbligatoria.
E' liberamente scaricabile dal sito di Oracle.
SDK è una sigla che sta per "Software Development Kit", cioè un kit per lo sviluppo di software; con esso siamo quindi in grado di lanciare, attraverso un'interfaccia, sulla nostra macchina, un'applicazione Android ed eventualmente modificarla.
Esso è inoltre corredato da documentazione e librerie, oltre ad importanti funzionalità di emulazione (AVD Manager) e di comunicazione con dispositivi esterni tramite USB (ADB).
Il più diffuso IDE per lo sviluppo in Android è senza dubbio Eclipse, ambiente multi-linguaggio e multipiattaforma, sviluppato da un consorzio di società informatiche con l'intento di creare un prodotto all-in-one che potesse abbracciare molteplici panorami di sviluppo.
Nell'ultimo anno Google si è tuttavia impegnata nella realizzazione di un ambiente di sviluppo integrato che fosse pensato specificatamente per il sistema operativo Android.
Il risultato di 1 anno e mezzo di beta testing è stato rilasciato nella sua prima forma stabile nel dicembre 2014 con il nome di Android Studio. Caratteristica fondamentale di AS è la praticità di utilizzo; essendo progettato per la programmazione nel solo ambiente Android, snellisce notevolmente alcune azioni e procedimenti che risulterebbero molto più complessi su IDE più generici come Eclipse.

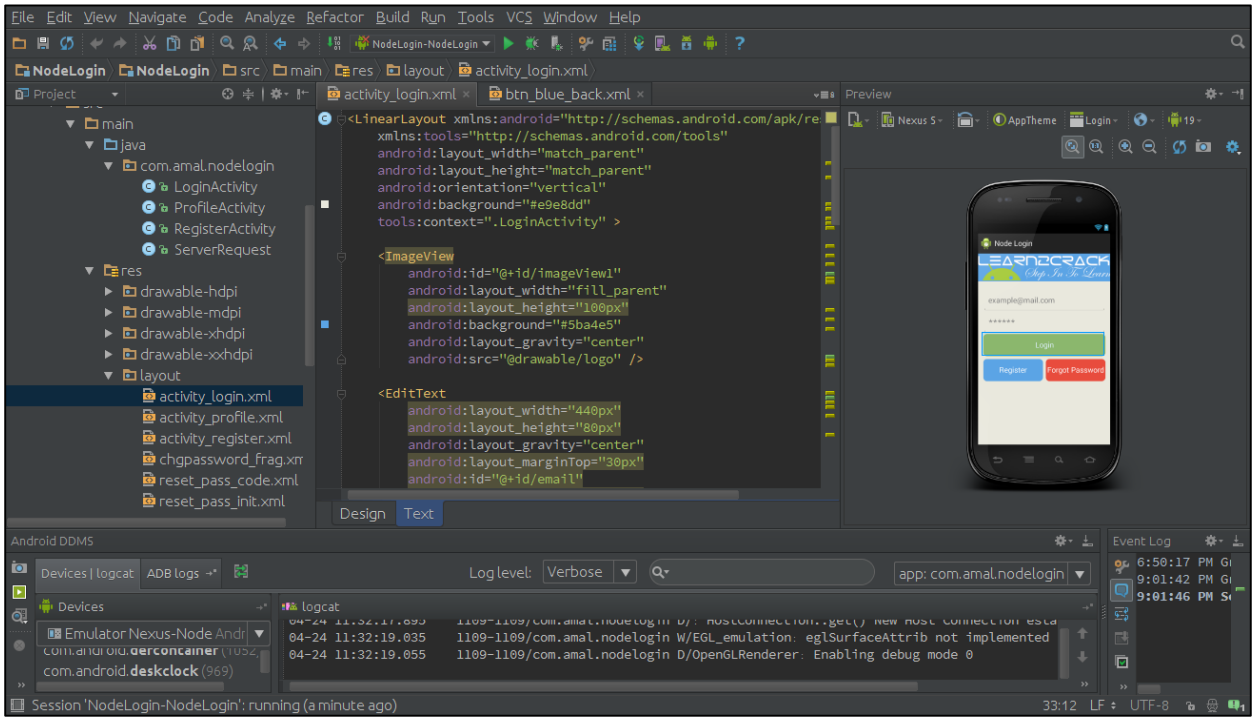


Figura 5 - Android Studio

Inoltre è presente un efficiente sistema di preview dei layout, cosicché l'aspetto dell'interfaccia viene mostrato non in maniera generica ma adattato specificatamente per un dato modello di dispositivo.
Altro elemento degno di nota è Gradle, un prodotto di build automation molto diffuso in ambiente Java; ogni progetto, oltre ai file e cartelle che lo compongono, si vedrà abbinato una serie di file .gradle relativi al progetto nella sua interezza o ai singoli moduli.
La build automation merita di essere trattata in maniera un poco più approfondita. Le fasi di sviluppo di un software sono contornate da una serie di attività accessorie, a volte preliminari, a volte attuate a posteriori, spesso organizzate durante lo sviluppo.
Parliamo di packaging, documentazione del progetto, compilazione dello stesso, esecuzione di test, tutte azioni che un software di build automation rende automatizzate.
I vantaggi in termini di tempo e grattacapi in meno sono notevoli, e Gradle si propone come uno dei tool più innovativi del genere.
Se difatti prodotti come Ant e Maven, tool di riferimento in Java, erano entrambi strutturati secondo file XML e lamentavano quindi una certa rigidità di configurazione, Gradle fa uso di Groovy, un linguaggio dichiarativo in seno a Java, e risulta decisamente più snello dei rivali.
Collegato attraverso un plugin ad Android Studio, è uno strumento potente, esterno al software ma capace di velocizzare sensibilmente il procedimento di sviluppo della nostra app.

2.2 Visual Studio

L’ambiente di sviluppo più completo, multipiattaforma, che supporta molteplici linguaggi, è questo. Un novizio programmatore, approcciando lo sviluppo della sua prima applicazione su pc, dovrà quasi forzatamente fare riferimento a Visual Studio.
Sviluppato da Microsoft, è uno strumento potente, integra la tecnologia IntelliSense (la quale permette di correggere errori sintattici e logici prima della compilazione del codice) e include come ogni altro IDE funzionalità di base come debugger e compilatore: il supporto completo alla piattaforma Microsoft lo rende perfetto per lo sviluppo di applicazioni in Windows Phone e più in generale nell’intero ambiente informatico generato dall’azienda di Redmond.
Il suo grande pregio consiste nel supporto alla programmazione in diversi linguaggi, tra cui C++, C#, ASP .NET, F#, Python (ma non Java): questo lo rende un IDE appetibile per ogni sviluppatore, dato

il suo possibile utilizzo in qualsiasi progetto che richieda la scrittura di codice in uno dei tanti linguaggi supportati, che questo sia una applicazione, un servizio web o un sito web.
Per venire incontro alle esigenze dei molti programmatori che ne fanno uso, Microsoft ha imparato a rilasciare molteplici versioni della stessa edizione, 5 con Visual Studio 2013, Express, Professional, Community, Premium e Ultimate, ognuna con funzionalità sempre più avanzate, così da soddisfare le esigenze di tutti i clienti.

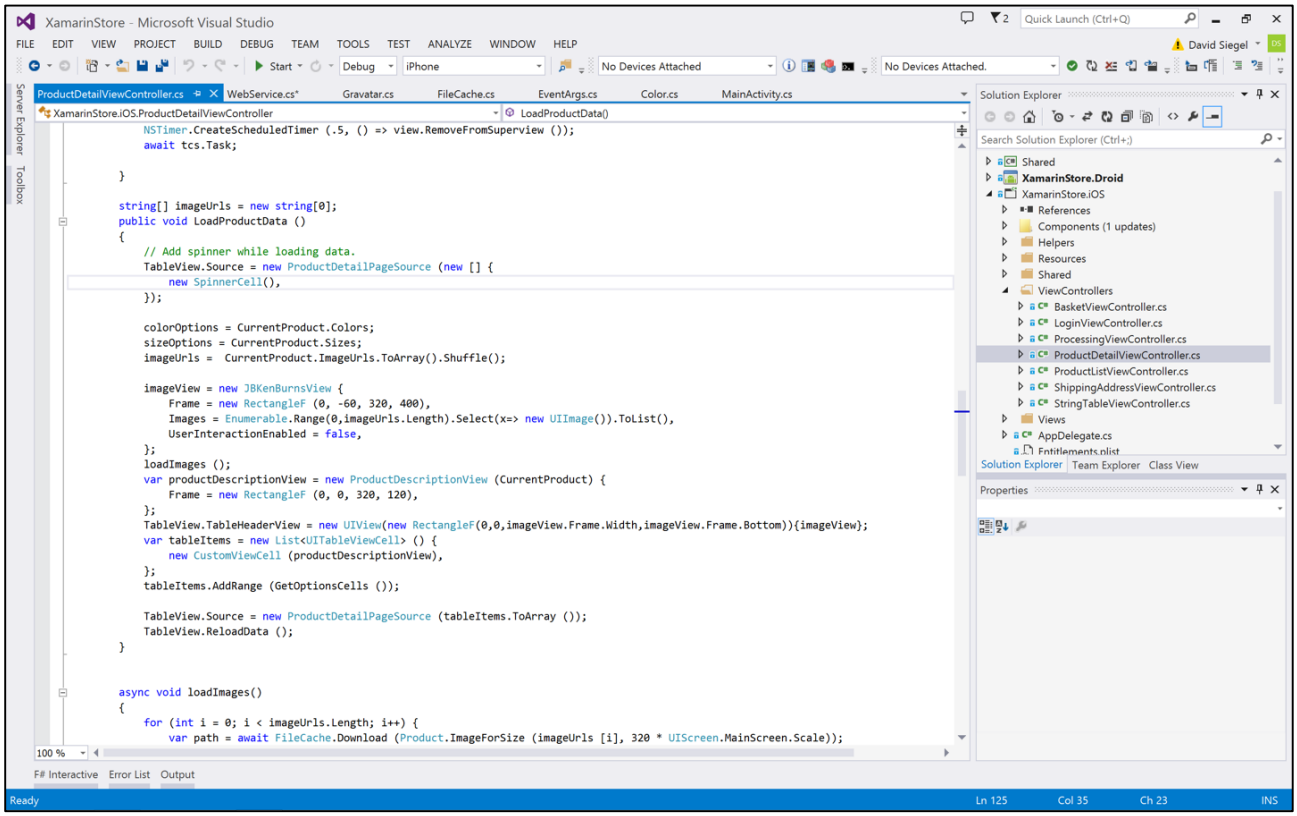


Figura 6 - Visual Studio

Il suo download è gratuito nella versione Community, mentre sono disponibili alcune versioni di edizioni precedenti a quella attuale agli studenti iscritti a Microsoft DreamSpark.
Elemento che contraddistingue Visual Studio è il supporto al framework .NET, di proprietà della stessa Microsoft, cioè l'ambiente per la creazione, la distribuzione e l'esecuzione di tutti gli applicativi .NET, una tecnologia di programmazione ad oggetti molto versatile, nato come diretta concorrente a Java (da qui il mancato supporto al linguaggio open source). Esso è indipendente dalla versione di Windows su cui è installata, ed è stato espressamente progettato per integrarsi con l'ambiente Internet, prediligendo la sicurezza dei dati e la loro integrità. Visual Studio è proprio uno degli strumenti di maggior successo di cui è corredato .NET .
Ho fatto quindi uso di questa potente piattaforma per sviluppare il web service utilizzato nell'app.

Capitolo 3
Analisi e Progettazione

Nell'applicazione sviluppata non è necessaria definire un certo tipo di utente, difatti essa è destinata, nei fatti, a chiunque sia interessato al suo utilizzo.
Dato il fine ben definito dell'app sono presenti un numero limitato di funzionalità di cui si può far uso una volta installata la stessa:

- creare un nuovo account, inserendo il proprio numero di telefono ed una password;
- procedere al login, inserendo le proprie credenziali;

Una volta effettuato il login, all'utente verrà inviato un SMS contenente un codice che dovrà poi inserire per confermare il proprio numero di telefono. Questo perché, successivamente, verrà inviato al server che gestisce l'app l'intero elenco telefonico presente sul telefono in uso.
Tale operazione è necessaria per fornire, in senso opposto al motivo per cui si fa uso dell'app, il nome con cui l'utente ha salvato tutti i conoscenti presenti nella propria rubrica.
L'invio dell'SMS è necessario per evitare che chiunque, anche non possedendo il cellulare di cui viene inserito il numero, possa usufruire liberamente dei servizi che l'app mette a disposizione.
Così ogni nuovo account contribuisce a formare una rete sempre più completa di "nomignoli", aggiungendo quelli da lui dati agli amici e parenti e ricevendo in cambio una lista di termini con cui egli è riconosciuto e memorizzato dagli stessi.
Per rendere possibile il tutto sono necessari 3 componenti fondamentali: un'applicazione installata su di un dispositivo Android, un server che mantenga memorizzati i dati degli utenti e un API, cioè un Application Programming Interface, che mette in comunicazione questi 2 elementi.
L'API utilizzata in questo progetto, in particolare, si basa su di un'architettura software chiamata REST (REpresentational State Transfer), che individua nelle "risorse" l'elemento cardine per la comunicazione tra due elementi, in questo caso client (cioè l'applicazione) e server.
Le due entità si scambiano rappresentazioni di queste risorse in un determinato formato di dati, che esso sia XML, HTML o JSON (quest'ultimo nel caso attuale), senza dover sapere della presenza o meno di firewall, proxy o gateways che ne modifichino i termini.

Per scrivere l'API ho quindi fatto ricorso ad una tecnologia di sviluppo rilasciato da Microsoft, ASP.NET, che verrà trattata nello specifico più avanti.
Iniziamo illustrando più nel dettaglio cosa si identifica con il termine "Web API" per comprendere appieno le tecnologie utilizzate, il loro funzionamento e come il loro connubio consenta di mettere in comunicazione il client, cioè l'applicazione installata sul dispositivo, e il server.

3.1 Web API

L'utilizzo del Web come piattaforma di programmazione è un'idea che si è diffusa intorno al 2000 con l'introduzione dei Web Service, cioè di un meccanismo per l'interoperabilità tra applicazioni basato sulle tecnologie e gli standard definiti per il Web.
I Web service consentono di far interagire due applicazioni indipendentemente dal sistema operativo su cui girano e dal linguaggio di programmazione utilizzato. Tecnicamente il meccanismo di interazione è basato sul protocollo SOAP, su un insieme di regole che definiscono la struttura dei messaggi scambiati dalle applicazioni e tutta una serie di accorgimenti per garantire sicurezza, affidabilità, corretta gestione degli eventi, ecc.
L'approccio è sostanzialmente basato su una definizione di funzione richiamabile via Web.
Un approccio alternativo ai Web service basati su SOAP, che è stato proposto negli stessi anni ma è stato riscoperto soltanto negli ultimi tempi, è l'approccio noto come REST e basato su una serie di principi che un'applicazione deve rispettare.
In sintesi, nella visione RESTful un Web service non definisce una funzione richiamabile da remoto ma mette a disposizione delle risorse su cui è possibile effettuare le classiche operazioni CRUD (Create, Read, Update, Delete) sfruttando i metodi del protocollo HTTP.
Due visioni diverse del Web come piattaforma di elaborazione che sempre più spesso vengono distinte parlando semplicemente di Web service quando si fa riferimento a quelli basati si SOAP e di Web API quando ci ispira al modello REST.
Per riassumere, nei Web Service ogni aspetto messo a disposizione delle entità che comunicano è ben definito, poiché standardizzato, ma consuma nei fatti un maggior numero di risorse e generalmente risulta anche più lento del rivale Web API. Questo infatti è a totale discrezione dello sviluppatore, nulla è definito dall'inizio ed è quindi sempre più ritenuto la scelta vincente in fase di progettazione per le possibilità di configurazione offerte.

Il codice necessario al funzionamento di questa Web API è stato scritto in linguaggio C# facendo uso di ASP.NET, una piattaforma utilizzata prevalentemente nello sviluppo di applicazioni e servizi web che supporta pienamente la modalità REST; questo ha reso il procedimento estremamente semplice, riducendo al minimo il numero di righe di codice.
Per connettere l'API al server basteranno infatti poche righe di codice:

```
using System.Configuration;
using System.Data.SqlClient;

namespace JSONWebAPI
{
    public class DBConnect
    {
        private static SqlConnection NewCon;
        private static string conStr =
ConfigurationManager.ConnectionStrings["ConString"].ConnectionString;

        public static SqlConnection getConnection(){
            NuovaConnessione = new SqlConnection(conStr);
            return NuovaConnessione;
        }
          public DBConnect(){}
    }
}
```

Inoltre, sarà necessario aggiungere nel file web.config, cioè il file di testo utilizzato per configurare il servizio, alcune righe di codice che consentano di individuare il server:

```
<connectionStrings>
    <add name="ConString" connectionString="workstation
id=AndroidAppDb5.mssql.somee.com;packet size=4096;user
id=kreep222_SQLLogin_1;pwd=ma2cer4dr8;data source=AndroidAppDb5.mssql.somee.com;persist
security info=False;initial catalog=AndroidAppDb5 " providerName="System.Data.SqlClient" />
  </connectionStrings>
```

Avremo così connesso il database alla Web API.
Vediamo ora meglio la struttura del database utilizzato.

3.2 SQL

Lo Structured Query Language è un linguaggio standardizzato per database basato sul modello relazionale, e progettato per svolgere alcune fondamentali operazioni, tra cui creare e modificare schemi di database, inserire, modificare ed interrogare dati memorizzati, oltre alla generazione di strutture di controllo e accesso ai dati. Ognuna di queste caratteristiche è definita da un sotto-linguaggio che rende SQL adatto non solo alla creazione di un database, ma anche alla sua corretta gestione. Il suo uso è estremamente semplice e non richiede particolari conoscenze pregresse.
Inizialmente per creare il server ho usufruito di SQL Server, un prodotto sviluppato da Microsoft che per questo motivo si sposava bene con ASP.NET. Dopo aver verificato il suo corretto funzionamento ho spostato l'intero progetto, tolta la parte Android, su un sito esterno, Somee.com, che mi permettesse di accedere sia alla Web API che al server direttamente dal cellulare, senza necessitare di alcun emulatore o di avere il computer a portata di mano, semplicemente con una rete Wi-Fi.
Di seguito le due tabelle necessarie al funzionamento dell'applicazione.
La tabella UTENTI serve, come comprensibile, per memorizzare i singoli utenti, mentre la tabella RUBRICHEUTENTI mantiene in memoria ogni accoppiata nome-numero presente nelle rubriche degli utenti registrati.

```
CREATE TABLE UTENTI(
U_NUMERO VARCHAR(30),
U_PASSWORD VARCHAR(30),
U_CODICECONFERMA VARCHAR(20),
U_CONFERMA VARCHAR(1),
U_ID VARCHAR(10),
);

CREATE TABLE RUBRICHEUTENTI(
R_ID VARCHAR(10),
R_NOME VARCHAR(40),
R_NUMERO VARCHAR(30),
);
```

3.3 Metodi API

Di seguito tutti i metodi messi a disposizione dalla Web API:

CreaNuovoAccount

Parametri d'input: numero e password del nuovo account, oltre all'id che viene trovato con il metodo NuovoId
Descrizione: questo metodo, come intuibile dal nome, consente di creare un nuovo account con cui accedere alle funzionalità dell'app. Per l'utente è necessario inserire il numero di telefono del dispositivo che utilizza ed una password adeguata.
Output: nessuno, poiché il controllo della disponibilità viene svolto da un altro metodo

NuovoId

Parametri d'input: nessuno
Descrizione: il metodo in questione serve per fornire all'app, in fase di creazione dell'account, un ID adeguato. Fa quindi richiesta al server dell'ID di numero più alto in uso nel database, e lo aumenta di 1. Idealmente quindi, con la creazione di sempre più account, il valore di ID salirà gradualmente, con il primo account creato con ID = 1, il secondo con ID = 2 e così via.
Output: l'ID ottenuto dal server + 1

NumeroLibero

Parametri d'input: il numero inserito in fase di creazione dell'account
Descrizione: è necessario controllare che il numero inserito in fase di creazione dell'account sia effettivamente disponibile, questo per evitare che lo stesso numero di telefono venga utilizzato più volte.
Output: se true procede alla creazione dell'account, altrimenti notifica all'utente l'impossibilità di usare quel numero.

Autenticazione

Parametri d'input: il numero e la password inseriti in fase di login
Descrizione: questo metodo controlla che la combinazione di valori inseriti in input sia effettivamente presente nel database
Output: se true procede al login, altrimenti viene notificato all'utente la non correttezza dei valori inseriti

ConfermaConferma

Parametri d'input: il numero inserito in fase di login
Descrizione: al momento del login, è necessario verificare attraverso un codice inviato attraverso SMS se il numero inserito appartiene effettivamente all'utente che sta facendo uso dell'app. Questo ovviamente non deve avvenire ad ogni avvio dell'applicazione, ma solo la prima volta. ConfermaConferma verifica che il valore U_CONFERMA presente nella tabella UTENTI e legato ad ogni account sia settato a 0, e quindi far procedere l'utente alla verifica dell'account, o 1, e quindi procedere direttamente alla schermata finale dell'app
Output: valore booleano che se true lascia procedere alla schermata finale, altrimenti alla verifica dell'account

Codice

Parametri d'input: numero inserito in fase di login
Descrizione: questo metodo viene utilizzato per ottenere dal server il codice di conferma da inviare attraverso SMS all'utente per la verifica dell'account
Output: il codice necessario alla verifica

CodiceConfermaGiusto

Parametri d'input: il codice inserito dall'utente e l'id relativo all'account in uso
Descrizione: il metodo in questione controlla che il codice di conferma inserito dall'utente corrisponda al codice assegnato all'account con id pari a quello in input
Output: se true permette l'avvio del metodo ConfermaAvvenuta, altrimenti viene notificato all'utente la non correttezza del codice inserito

ConfermaAvvenuta

Parametri d'input: l'id dell'account che ha effettuato il login
Descrizione: l'intento di ConfermaAvvenuta è semplicemente quello di procedere all'aggiornamente del valore di U_CONFERMA, che viene quindi settato ad 1. Questo metodo difatti viene chiamato nel caso in cui il codice di conferma inserito dall'utente sia corretto.
Output: nessuno

AggiungiRubricaUtente

Parametri d'input: l'id dell'account, un numero di telefono e il nome corrispondente entrambi presi dalla rubrica del dispositivo utilizzato dall'account in uso
Descrizione: il metodo qui descritto serve per popolare il server con l'intera rubrica dell'utente. Un cursore viene riempito con l'elenco dei contatti del dispositivo in uso, e il metodo viene chiamato iterativamente fino al termine del cursore, andando ad inserire ogni volta nel server un numero di telefono e il corrispondente nominativo
Output: nessuno

TagMap

Parametri d'input: il numero relativo all'account in uso

Descrizione: questo metodo consente di popolare la schermata finale dell'app. Viene richiesto al server l'elenco di nominativi con cui è stato registrato il numero in input; il risultato è espresso attraverso una cloud 3D.

Output: una DataTable contenente i vari nominativi e la frequenza degli stessi nel server

3.4 Installazione server

Per poter testare le funzionalità dell'applicazione mi sono appoggiato ad un sito, Somee.com, che mette a disposizione degli utenti un servizio gratuito di hosting di web services ASP.NET.
Ho potuto quindi non solo installare con successo un server che mi consentisse di gestire il database necessario al corretto funzionamento dell'app, ma dopo aver pubblicato il progetto di Visual Studio su cui avevo precedentemente sviluppato i metodi API e installandone i file all'interno del sito ho creato una piattaforma ideale per il testing dell'applicazione.
Con il web API così attivo e l'app Android installata sul mio dispositivo ho quindi potuto procedere con efficacia al testing dell'applicazione.

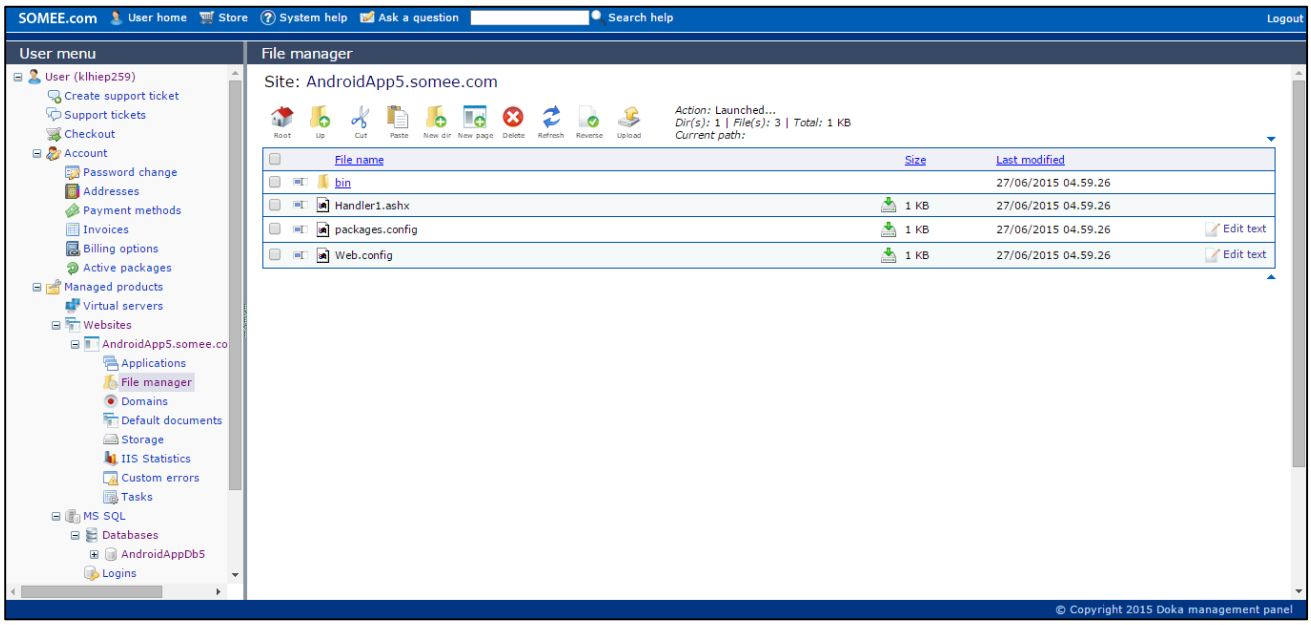
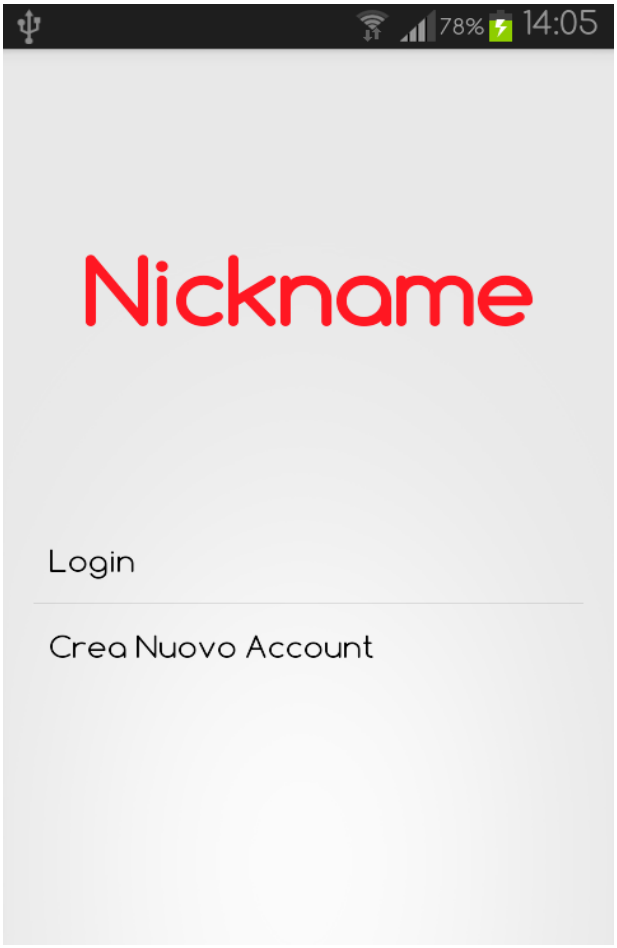
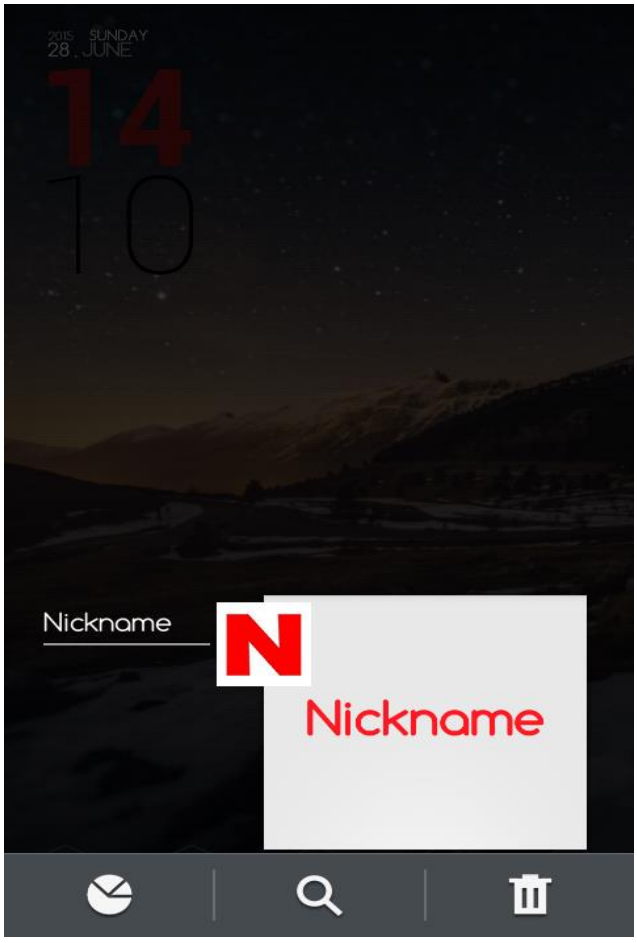


Figura 7 - Somee.com

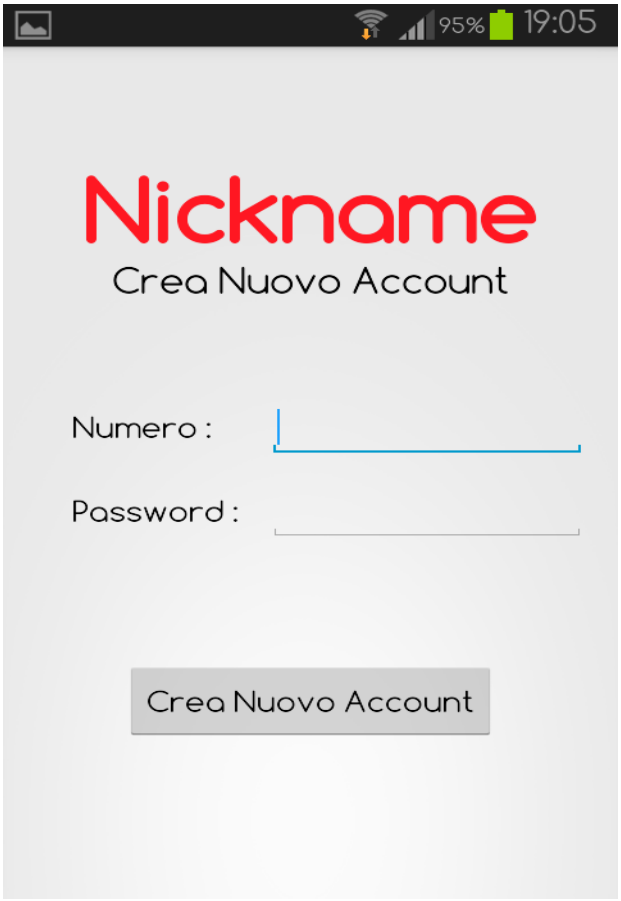
Capitolo 4
Mockup

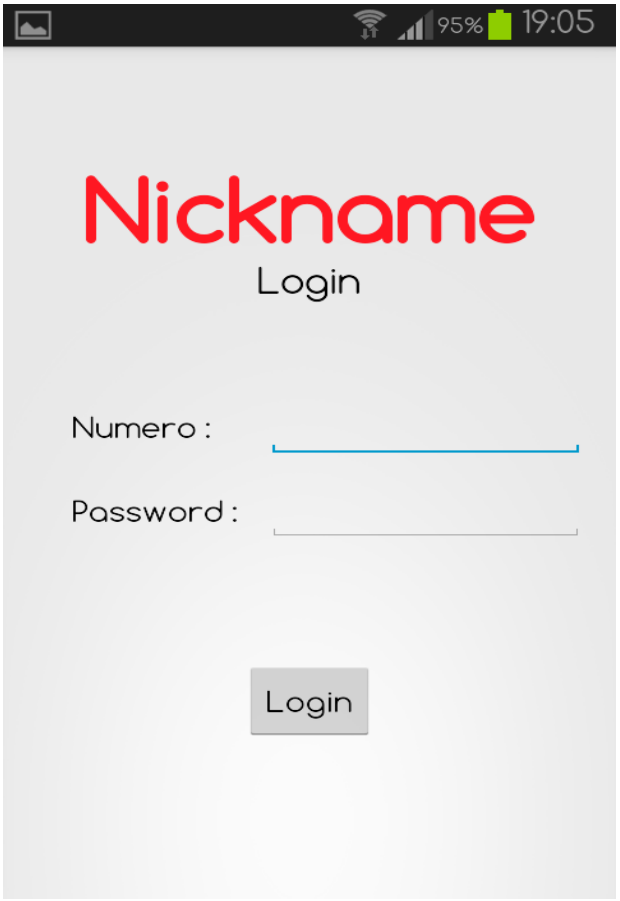
Vediamo ora i vari mockup dell'applicazione sviluppata, così da comprenderne meglio l'utilizzo:





Lanciando l'app, che per semplicità ho chiamato "Nickname", viene visualizzata la home da dove si può accedere alle funzionalità desiderate. Se infatti si è al primo avvio sarà necessario creare un account cliccando sull'apposito riquadro; se invece si avrà già effettuato questa operazione si potrà scegliere l'opzione "Login" che consentirà di procedere nell'applicazione.





Le due funzionalità presenti nella home permettono entrambe l'inserimento di due valori, in particolare il proprio numero di telefono e corrispettiva password, la quale viene scelta in fase di registrazione al servizio.

Dopo aver effettuato il login è possibile procedere alla verifica dell'account.
Al dispositivo in uso viene quindi inviato un SMS contenente il codice di conferma, inserito il quale si procede alla popolazione del server con i contatti presenti sul proprio cellulare.

Una volta terminato questo procedimento, l'applicazione mostra la schermata finale dove saranno visualizzati, attraverso una grafica 3D amovibile a 360 gradi, i nomi utilizzati dai nostri amici e parenti nel momento del salvataggio del nostro numero sul loro dispositivo. Facendo click su ogni nominativo si verrà informati del numero di volte in cui quel dato termine è stato utilizzato per "salvarci" sulla rubrica altrui.

Capitolo 5
Codice del progetto

La gestazione dell'applicazione si è svolta su due binari: la parte relativa alle API, scritta in Visual Studio e già trattata nel capitolo 3; la scrittura dell'applicazione Android, di cui di seguito viene riportato il codice delle varie Activity e Classi utilizzate.
Per la creazione della cloud 3D ho utilizzato un progetto open-source disponibile online consistente in 3 classi: Tag.java, TagCloud.java, TagCloudView.java, presenti per completezza al termine del codice da me scritto.
Ho ritenuto non necessario l'inserimento del codice relativo ai vari file di layout in XML.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.apprubrica.app"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_CONTACTS" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"
android:maxSdkVersion="18" />
    <uses-permission android:name="android.permission.SEND_SMS" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/n"
        android:label="Nickname"
        android:theme="@style/AppTheme">
        <activity
            android:name="com.apprubrica.app.MainActivity"
            android:label="Nickname">
>
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
```

```
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.apprubrica.app.LoginActivity"
            android:label="@string/title_activity_login"
            android:parentActivityName="com.apprubrica.app.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.apprubrica.app.MainActivity" />
        </activity>
        <activity
            android:name=".TagActivity"
            android:label="@string/title_activity_dept"
            android:parentActivityName="com.apprubrica.app.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.apprubrica.app.MainActivity" />
        </activity>
        <activity
            android:name=".CreaUtenteActivity"
            android:label="@string/title_activity_create_user"
            android:parentActivityName="com.apprubrica.app.MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.apprubrica.app.MainActivity" />
        </activity>
        <activity
            android:name="com.apprubrica.app.ConfermaActivity"
            android:label="@string/title_activity_conferma" >
        </activity>
    </application>

</manifest>
```

MainActivity.java

```
public class MainActivity extends Activity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      String listItem []={"Login","Crea Nuovo Account"};
      ListView lvMain=(ListView)findViewById(R.id.lv_main);
      lvMain.setAdapter(new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, listItem));

      lvMain.setOnItemClickListener(new AdapterView.OnItemClickListener() {

         @Override
         public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                           long arg3) {

            if (arg2 == 0) {
```

```
                Intent i = new Intent(MainActivity.this, LoginActivity.class);
                startActivity(i);
            } else if (arg2 == 1) {
                Intent i = new Intent(MainActivity.this,CreateUserActivity.class);
                startActivity(i);
            }
        }
    });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {

    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}
}
```

CreaUtenteActivity.java

```
public class CreaUtenteActivity extends Activity {

    EditText etNumber, etPassword;
    Button btnCreateUser;
    String tu = "stringa";
    String id = "03";
    String number, password;
    Context context;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_create_user);
        context = this;
        etNumber =(EditText) findViewById(R.id.et_numero);
        etPassword = (EditText) findViewById(R.id.et_password);
        btnCreateUser=(Button) findViewById(R.id.btn_createuser);

        btnCreateUser.setOnClickListener(
                new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                number = etNumber.getText().toString();
                password = etPassword.getText().toString();
                new NuovoId().execute();
            }
        });
    }

    public void Controllanumero(){

        new UtenteNew().execute(number);
    }

    public void CreaUtente(){
```

```
      DettagliUtenteTable userDetail = new
DettagliUtenteTable(number,password,id);
      new AsyncCreateUser().execute(userDetail);

   }

   protected class UtenteNew extends AsyncTask<String,JSONObject,Boolean>{

      @Override
      protected Boolean doInBackground(String... params){

         RestAPI api3 = new RestAPI();
         boolean l = false;
         try{

            JSONObject obj = api3.NumeroLibero(params[0]);
            JSONParser parser = new JSONParser();
            l = parser.parseUser(obj);

         } catch (Exception e) {

            Log.d("AsyncUt", e.getMessage());
         }
         return l;
      }

      @Override
      protected void onPostExecute(Boolean result) {

         if (result) {
            Toast.makeText(context, "Numero non disponibile",
Toast.LENGTH_SHORT).show();
         }
         else
         {
            CreaUtente();
         }
      }
   }

   protected class NuovoId extends AsyncTask<Void,JSONObject,String>{

      @Override
      protected String doInBackground(Void... params){

         RestAPI api2 = new RestAPI();
         try{

            JSONObject jsonObj = api2.NuovoId();
            JSONParser parser = new JSONParser();
            tu = parser.parseString(jsonObj);

         } catch (Exception e) {

               Log.d("AsyncNuovoId", e.getMessage());
         }

         return tu;
      }
```

```
        @Override
        protected void onPostExecute(String result) {
            id = tu;
            Controllanumero();
        }
    }

    protected class AsyncCreateUser extends
            AsyncTask<DettagliUtenteTable, Void, Void> {

        @Override
        protected Void doInBackground(DettagliUtenteTable... params) {

            RestAPI api = new RestAPI();
            try {
                api.CreaNuovoAccount(params[0].getNumber(),
                        params[0].getPassword(), params[0].getId());

            } catch (Exception e) {
                // TODO Auto-generated catch block
                Log.d("AsyncCreateUser", e.getMessage());
            }
            return null;
        }

        @Override
        protected void onPostExecute(Void result) {

            Intent i = new Intent(CreaUtenteActivity.this, LoginActivity.class);
            startActivity(i);
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.create_user, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case android.R.id.home:

            NavUtils.navigateUpFromSameTask(this);
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

}
```

LoginActivity.java

```
public class LoginActivity extends Activity {

   EditText etNumero, etPassword;
   Button btnLogin;
   Context context;
   String id2;
   String nonvalid = "non valido";
   String numero;
   String cod;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_login);
      context=this;
      etNumero = (EditText) findViewById(R.id.et_numero);
      etPassword = (EditText) findViewById(R.id.et_password);
      btnLogin = (Button) findViewById(R.id.btn_Login);

      btnLogin.setOnClickListener(new View.OnClickListener() {

         @Override
         public void onClick(View v) {

            numero = etNumero.getText().toString();
            String password = etPassword.getText().toString();
            new Login().execute(numero,password);
         }
      });
   }

   public void PrendiCodice(){

      new CodiceUtente().execute(numero);
   }

   public void InviaSMS(){
      sendSMSMessage();
      Intent i = new Intent(LoginActivity.this,
            ConfermaActivity.class);
      i.putExtra("id2", id2);
      i.putExtra("numero", numero);
      startActivity(i);
   }

   protected void sendSMSMessage() {
      Log.i("Send SMS", "");
      try {
         SmsManager smsManager = SmsManager.getDefault();
         smsManager.sendTextMessage(numero, null,"Il tuo codice di conferma è " +
cod, null, null);
      }
      catch (Exception e) {
         Toast.makeText(getApplicationContext(), "SMS failed, please try again",
Toast.LENGTH_LONG).show();
         e.printStackTrace();
      }
```

```
    }

    protected class CodiceUtente extends AsyncTask<String,JSONObject,String>{

        @Override
        protected String doInBackground(String... params){

            RestAPI api2 = new RestAPI();
            try{
                JSONObject jsonObj6 = api2.Codice(params[0]);
                JSONParser parser = new JSONParser();
                cod = parser.parseString(jsonObj6);

            }catch (Exception e) {
                // TODO Auto-generated catch block
                Log.d("AsyncCod", e.getMessage());
            }
            return cod;
        }

        @Override
        protected void onPostExecute(String result) {

             InviaSMS();
           }
    }

    protected class Login extends AsyncTask<String, JSONObject, String> {

        @Override
        protected String doInBackground(String... params) {

            RestAPI api = new RestAPI();
            try {
                JSONObject jsonObj = api.Autenticazione(params[0],
                      params[1]);
                JSONParser parser = new JSONParser();
                id2 = parser.parseString(jsonObj);

            } catch (Exception e) {
                // TODO Auto-generated catch block
                Log.d("AsyncLogin", e.getMessage());
            }
            return id2;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();

            Toast.makeText(context, "Attendi...", Toast.LENGTH_SHORT).show();
        }

        @Override
        protected void onPostExecute(String result) {

            if(id2.equals(nonvalid)) {

                Toast.makeText(context, "Credenziali non valide
",Toast.LENGTH_SHORT).show();
            }
```

```
            else
            {
                Confermaid();
            }
        }
    }

    public void Confermaid(){

        new Giaconfermato().execute(numero);
    }

    protected class Giaconfermato extends AsyncTask<String,JSONObject,Boolean>{

        @Override
        protected Boolean doInBackground(String... params) {

            RestAPI api3 = new RestAPI();
            boolean var = false;
            try {
                JSONObject obj = api3.ConfermaConferma(params[0]);
                JSONParser parser = new JSONParser();
                var = parser.parseUser(obj);

            } catch (Exception e) {

                Log.d("AsyncLogin", e.getMessage());
            }
            return var;
        }

        @Override
        protected void onPostExecute(Boolean result) {

            if(result){

                Intent i = new Intent(LoginActivity.this,
                        TagActivity.class);
                i.putExtra("numerotag", numero);
                startActivity(i);

            }else{

                PrendiCodice();
            }
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.login, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case android.R.id.home:
            NavUtils.navigateUpFromSameTask(this);
            return true;
        }
```

```
        return super.onOptionsItemSelected(item);
    }
}
```

ConfermaActivity.java

```
public class ConfermaActivity extends Activity {

    Button btnCon;
    String id;
    Context context;
    int a = 0, b = 0;
    String name = "ciao", number = "salve";
    private ProgressDialog progress;
    int codicediconferma;
    EditText codice;
    String numero;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_conferma);
        context = this;
        Intent i = getIntent();
        id = i.getStringExtra("id2");
        numero = i.getStringExtra("numero");
        codice = (EditText) findViewById(R.id.editText);
        btnCon = (Button) findViewById(R.id.buttonconferma);


        btnCon.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {

                String codiceconferma;
                codiceconferma = codice.getText().toString();
                new ConfermaCodice().execute(codiceconferma, id);

            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.menu_conferma, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {

        int id = item.getItemId();
        if (id == R.id.action_settings) {
            return true;
```

```
        }
        return super.onOptionsItemSelected(item);
    }


    protected class ConfermaCodice extends AsyncTask<String,JSONObject,Boolean>{

        @Override
        protected Boolean doInBackground(String... params){

            RestAPI api2 = new RestAPI();
            boolean s = false;
            try{
                JSONObject obj = api2.CodiceConfermaGiusto(params[0],params[1]);
                JSONParser parser = new JSONParser();
                s = parser.parseUser(obj);
            } catch (Exception e) {

            Log.d("AsyncCodice", e.getMessage());
            }
            return s;
        }

        @Override
        protected void onPostExecute(Boolean result) {

            if (result) {
                codicediconferma = 1;
                Confermailcodice();
            }
            else
            {
                codicediconferma = 0;
                Toast.makeText(context, "Codice di conferma non valido
",Toast.LENGTH_SHORT).show();
            }
        }
    }

    public void Confermailcodice()
    {
        new ConfermaAvvenuta().execute(id);
    }

    protected class ConfermaAvvenuta extends AsyncTask<String,Void,Void>{

        @Override
        protected Void doInBackground(String... params){

            RestAPI api3 = new RestAPI();
            try{
                api3.ConfermaAvvenuta(params[0]);

            } catch (Exception e) {

                Log.d("AsyncConfirm", e.getMessage());
            }
            return null;
        }
```

```
        @Override
        protected void onPostExecute(Void result) {

            AggiungiNumeri();
        }
    }

    public void AggiungiNumeri(){
        Rubrica();
    }

    protected class AggiungiNumero extends
            AsyncTask<NuovoUtenteRubricaTable, Void, Void> {

        @Override
        protected Void doInBackground(NuovoUtenteRubricaTable... params) {

            RestAPI api = new RestAPI();
            try {
                api.AggiungiRubricaUtente(params[0].getId(),
                        params[0].getNumber(), params[0].getName());
                b++;
                progress.setProgress(b);
                System.out.println(b);
            } catch (Exception e) {
                // TODO Auto-generated catch block
                Log.d("AsyncConferma", e.getMessage());
            }
            return null;
        }

        @Override
        protected void onPostExecute(Void result) {

            if(b == a)
            {
            Intent i = new Intent(ConfermaActivity.this, TagActivity.class);
            i.putExtra("numerotag", numero );
            startActivity(i);
            }
        }
    }

    public void Rubrica() {

        progress = new ProgressDialog(this,R.style.Theme_MyDialog);
        progress.setMessage("Aggiornando i server...");
        progress.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        progress.setProgress(b);

        progress.show();

        Cursor phones =
getContentResolver().query(ContactsContract.CommonDataKinds.Phone.CONTENT_URI,
null,null,null, null);
        int lunghezzaid = 0;
        while (phones.moveToNext())
        {
            lunghezzaid =
phones.getString(phones.getColumnIndex(ContactsContract.Contacts._ID)).length();
            if( lunghezzaid == 3) {
```

```
                a++;
                name =
phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DIS
PLAY_NAME));
                number =
phones.getString(phones.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUM
BER));
                NuovoUtenteRubricaTable nuovonumber = new
NuovoUtenteRubricaTable(id, number, name);
                new AggiungiNumero().execute(nuovonumber);
            }
        }
        phones.close();
        progress.setMax(a);
    }
}
```

TagActivity.java

```
public class TagActivity extends Activity {

   Context context;
   ArrayList<String> data;
   String numero2;
   int screenWidth,screenHeight;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      this.requestWindowFeature(Window.FEATURE_NO_TITLE);
      getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
            WindowManager.LayoutParams.FLAG_FULLSCREEN);
      DisplayMetrics displaymetrics = new DisplayMetrics();
      this.getWindowManager().getDefaultDisplay().getMetrics(displaymetrics);
      screenWidth = displaymetrics.widthPixels;
      screenHeight = displaymetrics.heightPixels;

      Intent i = getIntent();
      numero2 = i.getStringExtra("numerotag");
      context = this;
      Toast.makeText(this,"Attendi...",Toast.LENGTH_SHORT).show();
      Dai();
   }

   public void Dai(){
      new AsyncLoadDeptDetails().execute(numero2);
   }

   protected class AsyncLoadDeptDetails extends
         AsyncTask<String, JSONObject, ArrayList<TagTable>> {
      ArrayList<TagTable> tags = null;

      @Override
      protected ArrayList<TagTable> doInBackground(String... params) {

         RestAPI api = new RestAPI();
```

```
            try {
               JSONObject jsonObj = api.Tagmap(params[0]);
               System.out.println(jsonObj);
               JSONParser parser = new JSONParser();
               tags = parser.parseDepartment(jsonObj);
               System.out.println(tags);
            } catch (Exception e) {

               e.printStackTrace();
            }
            return tags;
         }

         @Override
         protected void onPostExecute(ArrayList<TagTable> result) {
            List<Tag> tempList = new ArrayList<Tag>();
            for (int i = 0; i < result.size(); i++)
            {
               int sw = result.get(i).getCount();
               String aq = Integer.toString(sw);
               tempList.add(new Tag(result.get(i).getName(),
result.get(i).getCount(),aq));
            }
            mTagCloudView = new TagCloudView(context, screenWidth, screenHeight,
tempList );
            setContentView(mTagCloudView);
            mTagCloudView.requestFocus();
            mTagCloudView.setFocusableInTouchMode(true);
            Toast.makeText(context,"Loading Completed",Toast.LENGTH_SHORT).show();
         }
      }

      @Override
      public boolean onCreateOptionsMenu(Menu menu) {

         getMenuInflater().inflate(R.menu.dept, menu);
         return true;
      }

      @Override
      public boolean onOptionsItemSelected(MenuItem item) {
         switch (item.getItemId()) {
            case android.R.id.home:

               NavUtils.navigateUpFromSameTask(this);
               return true;
         }
         return super.onOptionsItemSelected(item);
      }

      protected void onResume() {
         super.onResume();
      }

      protected void onPause() {
         super.onPause();
      }

      private TagCloudView mTagCloudView;
}
```

JSONParser.java

```
public class JSONParser {

   public JSONParser()
   {
   super();
   }

   public ArrayList<TagTable> parseDepartment(JSONObject object)
   {
      ArrayList<TagTable> arrayList=new ArrayList<TagTable>();
      try {
         JSONArray jsonArray=object.getJSONArray("Value");
         JSONObject jsonObj=null;
         for(int i=0;i<jsonArray.length();i++)
         {
            jsonObj=jsonArray.getJSONObject(i);
            arrayList.add(new TagTable(jsonObj.getInt("count"),
jsonObj.getString("name")));
         }
      } catch (JSONException e) {

         Log.d("JSONParser => parseDep", e.getMessage());
      }
      return arrayList;
   }

   public String parseString(JSONObject object)
   {
      String ret = "stringa";
      try{
         ret = object.getString("Value");
      }catch (JSONException e){
         Log.d("JSONParser => parseStr", e.getMessage());
      }
      return ret;
   }

   public boolean parseUser(JSONObject object)
   {  boolean userAuth=false;
         try {
            userAuth = object.getBoolean("Value");
         } catch (JSONException e) {
            // TODO Auto-generated catch block
            Log.d("JSONParser => parseUser", e.getMessage());
         }
         return userAuth;
   }
}
```

RestAPI.java

```
public class RestAPI {
    private final String urlString =
"http://AndroidApp5.somee.com/Handler1.ashx";

    private static String convertStreamToUTF8String(InputStream stream) throws
IOException {
       String result = "";
       StringBuilder sb = new StringBuilder();
       try {
            InputStreamReader reader = new InputStreamReader(stream, "UTF-8");
            char[] buffer = new char[4096];
            int readedChars = 0;
            while (readedChars != -1) {
                readedChars = reader.read(buffer);
                if (readedChars > 0)
                   sb.append(buffer, 0, readedChars);
            }
            result = sb.toString();
      } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        return result;
    }

    private String load(String contents) throws IOException {
        URL url = new URL(urlString);
        HttpURLConnection conn = (HttpURLConnection)url.openConnection();
        conn.setRequestMethod("POST");
        conn.setConnectTimeout(60000);
        conn.setDoOutput(true);
        conn.setDoInput(true);
        OutputStreamWriter w = new OutputStreamWriter(conn.getOutputStream());
        w.write(contents);
        w.flush();
        InputStream istream = conn.getInputStream();
        String result = convertStreamToUTF8String(istream);
        return result;
    }

    private Object mapObject(Object o) {
      Object finalValue = null;
      if (o.getClass() == String.class) {
         finalValue = o;
      }
      else if (Number.class.isInstance(o)) {
         finalValue = String.valueOf(o);
      } else if (Date.class.isInstance(o)) {
         SimpleDateFormat sdf = new SimpleDateFormat("MM/dd/yyyy hh:mm:ss", new
Locale("en", "USA"));
         finalValue = sdf.format((Date)o);
      }
      else if (Collection.class.isInstance(o)) {
         Collection<?> col = (Collection<?>) o;
         JSONArray jarray = new JSONArray();
         for (Object item : col) {
            jarray.put(mapObject(item));
         }
```

```
            finalValue = jarray;
        } else {
            Map<String, Object> map = new HashMap<String, Object>();
            Method[] methods = o.getClass().getMethods();
            for (Method method : methods) {
                if (method.getDeclaringClass() == o.getClass()
                        && method.getModifiers() == Modifier.PUBLIC
                        && method.getName().startsWith("get")) {
                    String key = method.getName().substring(3);
                    try {
                        Object obj = method.invoke(o, null);
                        Object value = mapObject(obj);
                        map.put(key, value);
                        finalValue = new JSONObject(map);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            }
        }
        return finalValue;
    }

    public JSONObject CreaNuovoAccount(String number,String password,String id)
throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "CreaNuovoAccount");
        p.put("number",mapObject(number));
        p.put("password",mapObject(password));
        p.put("id",mapObject(id));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject Autenticazione(String number,String password) throws
Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "Autenticazione");
        p.put("number",mapObject(number));
        p.put("password",mapObject(password));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject Tagmap(String number) throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
```

```
        o.put("method", "Tagmap");
        p.put("number",mapObject(number));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject NumeroLibero(String number) throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "NumeroLibero");
        p.put("number",mapObject(number));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject AggiungiRubricaUtente(String id,String number,String name)
throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "AggiungiRubricaUtente");
        p.put("id",mapObject(id));
        p.put("number",mapObject(number));
        p.put("name",mapObject(name));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject ConfermaAvvenuta(String id) throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "ConfermaAvvenuta");
        p.put("id",mapObject(id));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject NuovoId() throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "NuovoId");
        o.put("parameters", p);
```

```
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject Codice(String number) throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "Codice");
        p.put("number",mapObject(number));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject ConfermaConferma(String number) throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "ConfermaConferma");
        p.put("number",mapObject(number));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }

    public JSONObject CodiceConfermaGiusto(String codiceconferma,String id)
throws Exception {
        JSONObject result = null;
        JSONObject o = new JSONObject();
        JSONObject p = new JSONObject();
        o.put("interface","RestAPI");
        o.put("method", "CodiceConfermaGiusto");
        p.put("codiceconferma",mapObject(codiceconferma));
        p.put("id",mapObject(id));
        o.put("parameters", p);
        String s = o.toString();
        String r = load(s);
        result = new JSONObject(r);
        return result;
    }
}
```

NuovoUtenteRubricaTable.java

```
public class NuovoUtenteRubricaTable {

    String id, number, name;

    public NuovoUtenteRubricaTable(String id, String number, String name) {

        super();
        this.id = id;
        this.number = number;
        this.name = name;
    }

    public NuovoUtenteRubricaTable() {
        super();
        this.id = null;
        this.number = null;
        this.name = null;
    }

    public String getId() {
        return id;
    }

    public void setId(String id) {
        this.id = id;
    }

    public String getNumber() {
        return number;
    }

    public void setNumber(String number) {
        this.number = number;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

}
```

DettagliUtenteTable.java

```
public class DettagliUtenteTable {

   String number, id, password;

   public DettagliUtenteTable(String number, String password, String id) {

      super();
      this.number = number;
      this.password = password;
      this.id = id;
   }

   public DettagliUtenteTable() {
      super();
      this.number = null;
      this.password = null;
      this.id = null;
   }

   public String getNumber() {
      return number;
   }

   public void setNumber(String number) {
      this.number = number;
   }

   public String getPassword() {
      return password;
   }

   public void setPassword(String password) {
      this.password = password;
   }

   public String getId() {
      return id;
   }

   public void setId(String id) {
      this.id = id;
   }
}
```

TagTable.java

```
public class TagTable {

   int count;
   String name;

   public TagTable(int no, String name) {
      super();
      this.count = no;
      this.name = name;
   }

   public TagTable() {
      super();
      this.count= 0;
      this.name = null;
   }

   public int getCount() {
      return count;
   }
   public void setCount(int count) {
      this.count = count;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
}
```

Tag.java

```
public class Tag implements Comparable<Tag>{
    public Tag() {
        this("", 0f, 0f, 0f, 1.0f, 0, "");
    }
    public Tag(String text, int popularity) {
        this(text, 0f, 0f, 0f, 1.0f, popularity, "");
    }
    public Tag(String text, int popularity, String url) {
        this(text, 0f, 0f, 0f, 1.0f, popularity, url);
    }
    public Tag(String text,float locX, float locY, float locZ) {
        this(text, locX, locY, locZ, 1.0f, DEFAULT_POPULARITY, "");
    }
    public Tag(String text,float locX, float locY, float locZ, float scale) {
        this(text, locX, locY, locZ, scale, DEFAULT_POPULARITY, "");
    }
    public Tag(String text,float locX, float locY, float locZ, float scale, int
popularity,
               String url) {
```

```
        this.text = text;
        this.locX = locX;
        this.locY = locY;
        this.locZ = locZ;

        this.loc2DX = 0;
        this.loc2DY=0;

        this.colorR= 0.5f;
        this.colorG= 0.5f;
        this.colorB= 0.5f;
        this.alpha = 1.0f;

        this.scale = scale;
        this.popularity= popularity;
        this.url = url;
    }

    @Override
    public int compareTo(Tag another) {
        return (int)(another.locZ - locZ);
    }

    public float getLocX() {
        return locX;
    }
    public void setLocX(float locX) {
        this.locX = locX;
    }
    public float getLocY() {
        return locY;
    }
    public void setLocY(float locY) {
        this.locY = locY;
    }
    public float getLocZ() {
        return locZ;
    }
    public void setLocZ(float locZ) {
        this.locZ = locZ;
    }
    public float getScale() {
        return scale;
    }
    public void setScale(float scale) {
        this.scale = scale;
    }
    public String getText() {
        return text;
    }
    public void setText(String text) {
        this.text = text;
    }
    public float getColorR() {
        return colorR;
    }
    public void setColorR(float colorR) {
        this.colorR = colorR;
    }
    public float getColorG() {
        return colorG;
```

```
}
public void setColorG(float colorG) {
    this.colorG = colorG;
}
public float getColorB() {
    return colorB;
}
public void setColorB(float colorB) {
    this.colorB = colorB;
}
public float getAlpha() {
    return alpha;
}
public void setAlpha(float alpha) {
    this.alpha = alpha;
}
public int getPopularity() {
    return popularity;
}
public void setPopularity(int popularity) {
    this.popularity = popularity;
}

public int getTextSize() {
    return textSize;
}
public void setTextSize(int textSize) {
    this.textSize = textSize;
}
public float getLoc2DX() {
    return loc2DX;
}
public void setLoc2DX(float loc2dx) {
    loc2DX = loc2dx;
}
public float getLoc2DY() {
    return loc2DY;
}
public void setLoc2DY(float loc2dy) {
    loc2DY = loc2dy;
}
public int getParamNo() {
    return paramNo;
}
public void setParamNo(int paramNo) {
    this.paramNo = paramNo;
}
public String getUrl() {
    return url;
}
public void setUrl(String url) {
    this.url = url;
}

private String text, url;
private int popularity;  //this is the importance/popularity of the Tag
private int textSize;
private float locX, locY, locZ; //the center of the 3D Tag
private float loc2DX, loc2DY;
private float scale;
private float colorR, colorG, colorB, alpha;
```

```
    private static final int DEFAULT_POPULARITY = 1;
    private int paramNo; //parameter that holds the setting for this Tag
}
```

TagCloud.java

```
public class TagCloud implements Iterable{

    public TagCloud(){
        this(new ArrayList<Tag>());
    }
    public TagCloud(List<Tag> tags){
        this(tags,DEFAULT_RADIUS);
    }
    //Constructor just copies the existing tags in its List
    public TagCloud(List<Tag> tags, int radius){
        this( tags, radius, DEFAULT_COLOR1, DEFAULT_COLOR2, TEXT_SIZE_MIN,
TEXT_SIZE_MAX);
    }
    public TagCloud(List<Tag> tags, int radius,int textSizeMin, int textSizeMax){
        this( tags, radius, DEFAULT_COLOR1, DEFAULT_COLOR2, textSizeMin,
textSizeMax);
    }
    public TagCloud(List<Tag> tags, int radius,float[] tagColor1, float[]
tagColor2){
        this( tags, radius, tagColor1, tagColor2, TEXT_SIZE_MIN, TEXT_SIZE_MAX);
    }

    public TagCloud(List<Tag> tags, int radius, float[] tagColor1, float[]
tagColor2,
                    int textSizeMin, int textSizeMax){
        this.tagCloud=tags;    //Java does the initialization and deep copying
        this.radius = radius;
        this.tagColor1 = tagColor1;
        this.tagColor2 = tagColor2;
        this.textSizeMax = textSizeMax;
        this.textSizeMin = textSizeMin;
    }
    //create method calculates the correct initial location of each tag
    public void create(boolean distrEven){
        this.distrEven =distrEven;
        //calculate and set the location of each Tag
        positionAll(distrEven);
        sineCosine( mAngleX, mAngleY, mAngleZ);
        updateAll();
        //Now, let's calculate and set the color for each tag:
        //first loop through all tags to find the smallest and largest
populariteies
        //largest popularity gets tcolor2, smallest gets tcolor1, the rest in
between
        smallest = 9999;
        largest = 0;
        for (int i=0; i< tagCloud.size(); i++){
            int j = tagCloud.get(i).getPopularity();
            largest = Math.max(largest, j);
            smallest = Math.min(smallest, j);
```

```
        }
        //figuring out and assigning the colors/ textsize
        Tag tempTag;
        for (int i=0; i< tagCloud.size(); i++){
            tempTag = tagCloud.get(i);
            int j = tempTag.getPopularity();
            float percentage =  ( smallest == largest ) ? 1.0f : ((float)j-
smallest) / ((float)largest-smallest);
            float[] tempColor = getColorFromGradient( percentage ); //(rgb Alpha)
            int tempTextSize = getTextSizeGradient( percentage );
            tempTag.setColorR(tempColor[0]);
            tempTag.setColorG(tempColor[1]);
            tempTag.setColorB(tempColor[2]);
            tempTag.setTextSize(tempTextSize);
        }

        this.size= tagCloud.size();
    }

    public void reset(){
        create(distrEven);
    }

    //updates the transparency/scale of all elements
    public void update(){
        // if mAngleX and mAngleY under threshold, skip motion calculations for
performance
        if( Math.abs(mAngleX) > .1 || Math.abs(mAngleY) > .1 ){
            sineCosine( mAngleX, mAngleY, mAngleZ);
            updateAll();
        }
    }

    //if a single tag needed to be added
    public void add(Tag newTag){
        int j = newTag.getPopularity();
        float percentage =  ( smallest == largest ) ? 1.0f : ((float)j-smallest)
/ ((float)largest-smallest);
        float[] tempColor = getColorFromGradient( percentage ); //(rgb Alpha)
        int tempTextSize = getTextSizeGradient( percentage );
        newTag.setColorR(tempColor[0]);
        newTag.setColorG(tempColor[1]);
        newTag.setColorB(tempColor[2]);
        newTag.setTextSize(tempTextSize);
        position(distrEven, newTag);
        //now add the new tag to the tagCloud
        tagCloud.add(newTag);
        this.size= tagCloud.size();
        updateAll();
    }

    //to replace an existing tag with a new one
    //it returns the location of the replacement, if not found=> returns -1
    public int Replace(Tag newTag, String oldTagText){
        int result = -1;
        //let's go over all elements of tagCloud list and see if the oldTagText
exists:
        for (int i=0; i< tagCloud.size(); i++){
            if ( oldTagText.equalsIgnoreCase(tagCloud.get(i).getText())){
                result = i;
                tagCloud.get(i).setPopularity(newTag.getPopularity());
```

```
                tagCloud.get(i).setText(newTag.getText());
                tagCloud.get(i).setUrl(newTag.getUrl());
                int j = newTag.getPopularity();
                float percentage =  ( smallest == largest ) ? 1.0f : ((float)j-
smallest) / ((float)largest-smallest);
                float[] tempColor = getColorFromGradient( percentage ); //(rgb
Alpha)
                int tempTextSize = getTextSizeGradient( percentage );
                tagCloud.get(i).setColorR(tempColor[0]);
                tagCloud.get(i).setColorG(tempColor[1]);
                tagCloud.get(i).setColorB(tempColor[2]);
                tagCloud.get(i).setTextSize(tempTextSize);
                newTag = tagCloud.get(i);
                break;
            }
        }
        return result;
    }

    @Override
    public Iterator iterator() {
        return tagCloud.iterator();
    }

    private void position(boolean distrEven, Tag newTag){
        double phi = 0;
        double theta = 0;
        int max = tagCloud.size();
        //when adding a new tag, just place it at some random location
        //this is in fact why adding too many elements make TagCloud ugly
        //after many add, do one reset to rearrange all tags
        phi = Math.random()*(Math.PI);
        theta = Math.random()*(2 * Math.PI);
        //coordinate conversion:
        newTag.setLocX((int)(radius * Math.cos(theta) * Math.sin(phi)));
        newTag.setLocY((int)(radius * Math.sin(theta) * Math.sin(phi)));
        newTag.setLocZ((int)(radius * Math.cos(phi)));
    }

    private void positionAll(boolean distrEven){
        double phi = 0;
        double theta = 0;
        int max = tagCloud.size();
        //distribute: (disrtEven is used to specify whether distribute random or
even
        for (int i=1; i<max+1; i++){
            if (distrEven){
                phi = Math.acos(-1.0 + (2.0*i -1.0)/max);
                theta = Math.sqrt(max*Math.PI) * phi;
            } else{
                phi = Math.random()*(Math.PI);
                theta = Math.random()*(2 * Math.PI);
            }

            //coordinate conversion:
            tagCloud.get(i-1).setLocX((int)(    (radius * Math.cos(theta) *
Math.sin(phi))
            ));
            tagCloud.get(i-1).setLocY((int)(radius * Math.sin(theta) *
Math.sin(phi)));
            tagCloud.get(i-1).setLocZ((int)(radius * Math.cos(phi)));
```

```
        }
    }

    private void updateAll(){

        //update transparency/scale for all tags:
        int max = tagCloud.size();
        for (int j=0; j<max; j++){
            //There exists two options for this part:
            // multiply positions by a x-rotation matrix
            float rx1 = (tagCloud.get(j).getLocX());
            float ry1 = (tagCloud.get(j).getLocY()) * cos_mAngleX +
                    tagCloud.get(j).getLocZ() * -sin_mAngleX;
            float rz1 = (tagCloud.get(j).getLocY()) * sin_mAngleX +
                    tagCloud.get(j).getLocZ() * cos_mAngleX;
            // multiply new positions by a y-rotation matrix
            float rx2 = rx1 * cos_mAngleY + rz1 * sin_mAngleY;
            float ry2 = ry1;
            float rz2 = rx1 * -sin_mAngleY + rz1 * cos_mAngleY;
            // multiply new positions by a z-rotation matrix
            float rx3 = rx2 * cos_mAngleZ + ry2 * -sin_mAngleZ;
            float ry3 = rx2 * sin_mAngleZ + ry2 * cos_mAngleZ;
            float rz3 = rz2;
            // set arrays to new positions
            tagCloud.get(j).setLocX(rx3);
            tagCloud.get(j).setLocY(ry3);
            tagCloud.get(j).setLocZ(rz3);

            // add perspective
            int diameter = 2 * radius;
            float per = diameter / (diameter+rz3);
            // let's set position, scale, alpha for the tag;
            tagCloud.get(j).setLoc2DX((int)(rx3 * per));
            tagCloud.get(j).setLoc2DY((int)(ry3 * per));
            tagCloud.get(j).setScale(per);
            tagCloud.get(j).setAlpha(per / 2);
        }
        depthSort();
    }

    ///now let's sort all tags in the tagCloud based on their z coordinate
    //this way, when they are finally drawn, upper tags will be drawn on top of
lower tags
    private void depthSort(){
        Collections.sort(tagCloud);
    }

    private float[] getColorFromGradient(float perc){
        float[] tempRGB = new float[4];
        tempRGB[0] = ( perc * ( tagColor1[0] ) ) + ( (1-perc) * ( tagColor2[0] )
);
        tempRGB[1] = ( perc * ( tagColor1[1] ) ) + ( (1-perc) * ( tagColor2[1] )
);
        tempRGB[2] = ( perc * ( tagColor1[2] ) ) + ( (1-perc) * ( tagColor2[2] )
);
        tempRGB[3] = 1;
        return tempRGB;
    }
    private int getTextSizeGradient(float perc){
        int size;
        size = (int)( perc*textSizeMax + (1-perc)*textSizeMin );
```

```
        return size;
    }
    private void sineCosine(float mAngleX,float mAngleY,float mAngleZ) {
        double degToRad = (Math.PI / 180);
        sin_mAngleX= (float) Math.sin( mAngleX * degToRad);
        cos_mAngleX= (float) Math.cos( mAngleX * degToRad);
        sin_mAngleY= (float) Math.sin( mAngleY * degToRad);
        cos_mAngleY= (float) Math.cos( mAngleY * degToRad);
        sin_mAngleZ= (float) Math.sin( mAngleZ * degToRad);
        cos_mAngleZ= (float) Math.cos( mAngleZ * degToRad);
    }
    public int getRadius() {
        return radius;
    }
    public void setRadius(int radius) {
        this.radius = radius;
    }
    public float[] getTagColor1() {
        return tagColor1;
    }
    public void setTagColor1(float[] tagColor) {
        this.tagColor1 = tagColor;
    }
    public float[] getTagColor2() {
        return tagColor2;
    }
    public void setTagColor2(float[] tagColor2) {
        this.tagColor2 = tagColor2;
    }

    public float getRvalue(float[] color) {
        if (color.length>0)
            return color[0];
        else
            return 0;
    }
    public float getGvalue(float[] color) {
        if (color.length>0)
            return color[1];
        else
            return 0;   }
    public float getBvalue(float[] color) {
        if (color.length>0)
            return color[2];
        else
            return 0;   }
    public float getAlphaValue(float[] color) {
        if (color.length >= 4)
            return color[3];
        else
            return 0;
    }
    public float getAngleX() {
        return mAngleX;
    }
    public void setAngleX(float mAngleX) {
        this.mAngleX = mAngleX;
    }
    public float getAngleY() {
        return mAngleY;
    }
```

```
    public void setAngleY(float mAngleY) {
        this.mAngleY = mAngleY;
    }
    public int getSize() {
        return size;
    }

    private List<Tag> tagCloud;
    private int radius;
    private static final int DEFAULT_RADIUS = 3;
    private static final int TEXT_SIZE_MAX = 30 , TEXT_SIZE_MIN= 4;
    private static final float[] DEFAULT_COLOR1= { 0.886f, 0.725f, 0.188f, 1f};
    private static final float[] DEFAULT_COLOR2= { 0.3f, 0.3f,  0.3f, 1f};
    private float[] tagColor1;  //text color 1(rgb Alpha)
    private float[] tagColor2; //text color 2 (rgb Alpha)
    private int textSizeMax, textSizeMin;
    private float
sin_mAngleX,cos_mAngleX,sin_mAngleY,cos_mAngleY,sin_mAngleZ,cos_mAngleZ;
    private float mAngleZ=0;
    private float mAngleX =0;
    private float mAngleY =0;
    private int size=0;
    private int smallest,largest; //used to find spectrum for tag colors
    private boolean distrEven = true; //default is to distribute tags evenly on
the Cloud
}
```

TagCloudView.java

```
public class TagCloudView extends RelativeLayout {
    RelativeLayout navigation_bar;
    TextView mTextView1;
    public TagCloudView(Context mContext, int width, int height, List<Tag>
tagList) {
        this(mContext, width, height, tagList, 6 , 34, 1); //default for min/max
text size
    }
    public TagCloudView(Context mContext, int width, int height, List<Tag>
tagList,
                        int textSizeMin, int textSizeMax, int scrollSpeed) {

        super(mContext);
        this.mContext= mContext;
        this.textSizeMin = textSizeMin;
        this.textSizeMax= textSizeMax;

        tspeed = scrollSpeed;

        //set the center of the sphere on center of our screen:
        centerX = width / 2;
        centerY = height / 2;
        radius = Math.min(centerX * 0.95f , centerY * 0.95f ); //use 95% of
screen
        //since we set tag margins from left of screen, we shift the whole tags
to left so that
        //it looks more realistic and symmetric relative to center of screen in X
```

```
direction
        shiftLeft = (int)(Math.min(centerX * 0.15f , centerY * 0.15f ));

        // initialize the TagCloud from a list of tags
        //Filter() func. screens tagList and ignores Tags with same text (Case
Insensitive)
        mTagCloud = new TagCloud(Filter(tagList), (int) radius ,
                textSizeMin,
                textSizeMax
        );
        float[] tempColor1 = {0.9412f,0.7686f,0.2f,1}; //rgb Alpha
        //{1f,0f,0f,1}  red       {0.3882f,0.21568f,0.0f,1} orange
        //{0.9412f,0.7686f,0.2f,1} light orange
        float[] tempColor2 = {1f,0f,0f,1}; //rgb Alpha
        //{0f,0f,1f,1}  blue      {0.1294f,0.1294f,0.1294f,1} grey
        //{0.9412f,0.7686f,0.2f,1} light orange
        mTagCloud.setTagColor1(tempColor1);//higher color
        mTagCloud.setTagColor2(tempColor2);//lower color
        mTagCloud.setRadius((int) radius);
        mTagCloud.create(true); // to put each Tag at its correct initial
location


        //update the transparency/scale of tags
        mTagCloud.setAngleX(mAngleX);
        mTagCloud.setAngleY(mAngleY);
        mTagCloud.update();

        mTextView = new ArrayList<TextView>();
        mParams = new ArrayList<RelativeLayout.LayoutParams>();
        //Now Draw the 3D objects: for all the tags in the TagCloud
        Iterator it=mTagCloud.iterator();
        Tag tempTag;
        int i=0;

        while (it.hasNext()){
            tempTag= (Tag) it.next();
            tempTag.setParamNo(i); //store the parameter No. related to this tag

            mTextView.add(new TextView(this.mContext));
            mTextView.get(i).setText(tempTag.getText());

            mParams.add(new RelativeLayout.LayoutParams(
                            LayoutParams.WRAP_CONTENT,
                            LayoutParams.WRAP_CONTENT
                    )
            );
            mParams.get(i).addRule(RelativeLayout.ALIGN_PARENT_LEFT);
            mParams.get(i).addRule(RelativeLayout.ALIGN_PARENT_TOP);
            mParams.get(i).setMargins(
                    (int) (centerX - shiftLeft + tempTag.getLoc2DX()),
                    (int) (centerY + tempTag.getLoc2DY()),
                    0,
                    0);
            mTextView.get(i).setLayoutParams(mParams.get(i));

            mTextView.get(i).setSingleLine(true);
            int mergedColor = Color.argb( (int)     (tempTag.getAlpha() * 255),
                    (int)  (tempTag.getColorR() * 255),
                    (int)  (tempTag.getColorG() * 255),
                    (int) (tempTag.getColorB() * 255));
```

```
            mTextView.get(i).setTextColor(mergedColor);
            mTextView.get(i).setTextSize((int) (tempTag.getTextSize() *
tempTag.getScale()));
            addView(mTextView.get(i));

mTextView.get(i).setOnClickListener(OnTagClickListener(tempTag.getUrl()));
            i++;
        }
    }

    @Override
    protected void onDraw(Canvas canvas){
        super.onDraw(canvas);
    }

    public void addTag(Tag newTag){
        mTagCloud.add(newTag);

        int i= mTextView.size();
        newTag.setParamNo(i);

        mTextView.add(new TextView(this.mContext));
        mTextView.get(i).setText(newTag.getText());

        mParams.add(new RelativeLayout.LayoutParams(
                        LayoutParams.WRAP_CONTENT,
                        LayoutParams.WRAP_CONTENT
                )
        );
        mParams.get(i).addRule(RelativeLayout.ALIGN_PARENT_LEFT);
        mParams.get(i).addRule(RelativeLayout.ALIGN_PARENT_TOP);
        mParams.get(i).setMargins(
                (int) (centerX - shiftLeft + newTag.getLoc2DX()),
                (int) (centerY + newTag.getLoc2DY()),
                0,
                0);
        mTextView.get(i).setLayoutParams(mParams.get(i));

        mTextView.get(i).setSingleLine(true);
        int mergedColor = Color.argb( (int)     (newTag.getAlpha() * 255),
                (int)  (newTag.getColorR() * 255),
                (int)  (newTag.getColorG() * 255),
                (int) (newTag.getColorB() * 255));
        mTextView.get(i).setTextColor(mergedColor);
        mTextView.get(i).setTextSize((int) (newTag.getTextSize() *
newTag.getScale()));
        addView(mTextView.get(i));
        mTextView.get(i).setOnClickListener(OnTagClickListener(newTag.getUrl()));
    }

    public boolean Replace(Tag newTag, String oldTagText){
        boolean result=false;
        int j= mTagCloud.Replace(newTag, oldTagText);
        if (j>=0) { //then oldTagText was found and replaced with newTag data
            Iterator it=mTagCloud.iterator();
            Tag tempTag;
            while (it.hasNext()){
                tempTag= (Tag) it.next();
                mParams.get(tempTag.getParamNo()).setMargins(
                        (int) (centerX -shiftLeft+ tempTag.getLoc2DX()),
                        (int) (centerY + tempTag.getLoc2DY()),
```

```
                        0,
                        0);
                mTextView.get(tempTag.getParamNo()).setText(tempTag.getText());
                mTextView.get(tempTag.getParamNo()).setTextSize(
                        (int)(tempTag.getTextSize() * tempTag.getScale()));
                int mergedColor = Color.argb( (int)     (tempTag.getAlpha() *
255),
                        (int)   (tempTag.getColorR() * 255),
                        (int)   (tempTag.getColorG() * 255),
                        (int)  (tempTag.getColorB() * 255));
                mTextView.get(tempTag.getParamNo()).setTextColor(mergedColor);
                mTextView.get(tempTag.getParamNo()).bringToFront();
            }
            result=true;
        }
        return result;
    }

    public void reset(){
        mTagCloud.reset();

        Iterator it=mTagCloud.iterator();
        Tag tempTag;
        while (it.hasNext()){
            tempTag= (Tag) it.next();
            mParams.get(tempTag.getParamNo()).setMargins(
                    (int) (centerX -shiftLeft+ tempTag.getLoc2DX()),
                    (int) (centerY + tempTag.getLoc2DY()),
                    0,
                    0);

mTextView.get(tempTag.getParamNo()).setTextSize((int)(tempTag.getTextSize() *
tempTag.getScale()));
            int mergedColor = Color.argb( (int)     (tempTag.getAlpha() * 255),
                    (int)   (tempTag.getColorR() * 255),
                    (int)   (tempTag.getColorG() * 255),
                    (int)  (tempTag.getColorB() * 255));
            mTextView.get(tempTag.getParamNo()).setTextColor(mergedColor);
            mTextView.get(tempTag.getParamNo()).bringToFront();
        }
    }

    @Override
    public boolean onTrackballEvent(MotionEvent e) {
        float x = e.getX();
        float y = e.getY();

        mAngleX = ( y)*tspeed * TRACKBALL_SCALE_FACTOR;
        mAngleY = (-x)*tspeed * TRACKBALL_SCALE_FACTOR;

        mTagCloud.setAngleX(mAngleX);
        mTagCloud.setAngleY(mAngleY);
        mTagCloud.update();

        Iterator it=mTagCloud.iterator();
        Tag tempTag;
        while (it.hasNext()){
            tempTag= (Tag) it.next();
            mParams.get(tempTag.getParamNo()).setMargins(
                    (int) (centerX -shiftLeft+ tempTag.getLoc2DX()),
                    (int) (centerY + tempTag.getLoc2DY()),
```

```
                    0,
                    0);

mTextView.get(tempTag.getParamNo()).setTextSize((int)(tempTag.getTextSize() *
tempTag.getScale()));
            int mergedColor = Color.argb( (int)    (tempTag.getAlpha() * 255),
                    (int)  (tempTag.getColorR() * 255),
                    (int)  (tempTag.getColorG() * 255),
                    (int) (tempTag.getColorB() * 255));
            mTextView.get(tempTag.getParamNo()).setTextColor(mergedColor);
            mTextView.get(tempTag.getParamNo()).bringToFront();
        }
        return true;
    }

    @Override
    public boolean onTouchEvent(MotionEvent e) {
        float x = e.getX();
        float y = e.getY();

        switch (e.getAction()) {
            case MotionEvent.ACTION_MOVE:
                //rotate elements depending on how far the selection point is
from center of cloud
                float dx = x - centerX;
                float dy = y - centerY;
                mAngleX = ( dy/radius) *tspeed * TOUCH_SCALE_FACTOR;
                mAngleY = (-dx/radius) *tspeed * TOUCH_SCALE_FACTOR;

                mTagCloud.setAngleX(mAngleX);
                mTagCloud.setAngleY(mAngleY);
                mTagCloud.update();

                Iterator it=mTagCloud.iterator();
                Tag tempTag;
                while (it.hasNext()){
                    tempTag= (Tag) it.next();
                    mParams.get(tempTag.getParamNo()).setMargins(
                            (int) (centerX -shiftLeft + tempTag.getLoc2DX()),
                            (int) (centerY + tempTag.getLoc2DY()),
                            0,
                            0);

mTextView.get(tempTag.getParamNo()).setTextSize((int)(tempTag.getTextSize() *
tempTag.getScale()));
                    int mergedColor = Color.argb( (int)    (tempTag.getAlpha() *
255),
                            (int)  (tempTag.getColorR() * 255),
                            (int)  (tempTag.getColorG() * 255),
                            (int) (tempTag.getColorB() * 255));

mTextView.get(tempTag.getParamNo()).setTextColor(mergedColor);
                    mTextView.get(tempTag.getParamNo()).bringToFront();
                }

                break;
      /*case MotionEvent.ACTION_UP:  //now it is clicked!!!!
         dx = x - centerX;
         dy = y - centerY;
         break;*/
        }
```

```
        return true;
    }
    /*String urlMaker(String url){
        if      (  (url.substring(0,7).equalsIgnoreCase("http://"))    ||
                 (url.substring(0,8).equalsIgnoreCase("https://"))
                 )
            return url;
        else
            return "http://"+url;
    }*/

    //the filter function makes sure that there all elements are having unique
Text field:
    List<Tag> Filter(List<Tag> tagList){
        //current implementation is O(n^2) but since the number of tags are not
that many,
        //it is acceptable.
        List<Tag> tempTagList=new ArrayList();
        Iterator itr = tagList.iterator();
        Iterator itrInternal;
        Tag tempTag1, tempTag2;
        //for all elements of TagList
        while (itr.hasNext()){
            tempTag1 = (Tag) (itr.next());
            boolean found = false;
            //go over all elements of temoTagList
            itrInternal = tempTagList.iterator();
            while (itrInternal.hasNext()){
                tempTag2 = (Tag) (itrInternal.next());
                if (tempTag2.getText().equalsIgnoreCase(tempTag1.getText())){
                    found = true;
                    break;
                }
            }
            if (found==false)
                tempTagList.add(tempTag1);
        }
        return tempTagList;
    }

    //for handling the click on the tags
    //onclick open the tag url in a new window. Back button will bring you back
to TagCloud
    View.OnClickListener OnTagClickListener(final String url){
        return new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                //we now have url from main code
                //Uri uri = Uri.parse( urlMaker(url) );
                Toast.makeText(mContext, "Utilizzato " + url + " volte",
Toast.LENGTH_SHORT).show();
                //just open a new intent and set the content to search for the
url
                //mContext.startActivity(new Intent( Intent.ACTION_VIEW, uri ) );
            }
        };
    }
    private final float TOUCH_SCALE_FACTOR = .8f;
    private final float TRACKBALL_SCALE_FACTOR = 10;
    private float tspeed;
```

```
    private TagCloud mTagCloud;
    private float mAngleX =0;
    private float mAngleY =0;
    private float centerX, centerY;
    private float radius;
    private Context mContext;
    private int textSizeMin, textSizeMax;
    private List<TextView> mTextView;
    private List<RelativeLayout.LayoutParams> mParams;
    private int shiftLeft;
}
```

Conclusioni

L’obiettivo di questa tesi è stata sviluppare in tutte le sue parti un’applicazione mobile per la piattaforma Android. L’app risultante è in grado di connettersi online ad un server e di svolgere appieno le varie funzioni richieste.
Il procedimento che ha portato alla sua realizzazione è stato lungo, a tratti travagliato, ma è stato senza dubbio molto educativo: ho imparato a conoscere lo sviluppo di una applicazione mobile sia dal lato client che da lato server, facendo uso di differenti strumenti e linguaggi di programmazione.
L’applicazione sviluppata necessita di alcune migliorie, che potranno e se necessario verranno svolte in futuro, ma il bilancio finale è indubbiamente positivo: ho compreso cosa significhi dover analizzare, progettare e successivamente programmare un’applicazione partendo da zero, ho familiarizzato con tecnologie nuove e approfondimento argomenti già conosciuti; spero di poter procedere così anche in futuro.

Sitografia

Hosting ASP.NET

www.somee.com

ASP.NET, SQL Server e Android

www.tutecentral.com

TagCloud 3D

https://sites.google.com/site/tagindemo/TagCloud

Guida Web API, Differenze Web Service - Web API

www.html.it

SMS

www.twilio.com

JSON

http://json.org/

Consigli, suggerimenti, soluzioni

www.stackoverflow.com

http://programmers.stackexchange.com/

http://programmerguru.com/

Wikipedia

https://it.wikipedia.org/wiki