UNIVERSITÀ DEGLI STUDI DI PARMA
Dipartimento di Matematcia e Informatica

Corso di Laurea in Informatica

Realizzazione del Supporto a Google+ nella Piattaforma AMUSE

Relatore: **Chiar.mo Prof. Federico Bergenti**
Candidato: **Luca Alberici**

Anno Accademico 2014/2015

A mia mamma Daniela, mio papà Renzo e mia sorella Giulia.

Ringraziamenti

Prima di addentrarsi negli argomenti che sono stati sviluppati nel lavoro di tesi è doveroso fare un ringraziamento ad alcune persone. Un grazie particolare va al prof. Federico Bergenti che oltre ad avere avuto il ruolo di relatore è sempre stato disponibile e i suoi consigli sono stati preziosi per portare a termine questo lavoro. Inoltre intendo ringraziare tutti i professori che in questi anni mi hanno aiutato nel percorso di studi intrapreso facendo nascere in me un interesse sempre più crescente per le materie del corso di laurea. Un altro grazie intendo rivolgerlo a tutti i miei compagni di corso, in particolare intendo ringraziare Riccardo Zangrandi che mi è sempre stato vicino negli anni di studi e che mi ha sempre aiutato in caso di bisogno.

Indice

Capitolo 1

Introduzione a Facebook, Google+ e AMUSE

1.1 Introduzione

In questo capitolo introduttivo andremo a spiegare: che cosa sono i social network che abbiamo utilizzato nella nostra applicazione, cosa sono le API in generale e come si utilizzano (con un approfondimento per quelle di Google+) e cos'è il server AMUSE. La sezione che segue avrà il compito di dare una panoramica generale delle componenti dell'app dando alcuni nozioni di base al lettore.

1.2 AMUSE

AMUSE (Agent-based Multi-User Social Environment, disponibile all'indirizzo `jade.tilab.com/amuse`) è una piattaforma software che facilita lo sviluppo di applicazioni social distribuite, le quali portano gli utenti a cooperare o a competere per la realizzazione di obiettivi comuni o privati. Per questo motivo, lo scopo principale di AMUSE è la realizzazione di giochi multiplayer on-line. La piattaforma AMUSE è un sistema basato su JADE (Java Agent DEvelopment Framework, `jade.tilab.com`) e, per la gestione dei componenti e delle comunicazioni, adopera le soluzioni già implementate WADE e JADE:

- JADE: è un framwork implementato completamente in Java. Il suo scopo è quello di semplificare l'implementazione di sistemi multi-agente. JADE fornisce una semplice ma potente task execution, model, comunicazione peer-to-peer tra agenti basata sullo scambio di messaggi

asincroni, un servizio di pagine gialle e molte altre caratteristiche avanzate che facilitano lo sviluppo di un sistema distribuito. JADE è un software distribuito da Telecom Italia, in open source secondo i termini e le condizioni della licenza LGPL;

- WADE:(Workflows and Agents Development Environment, disponibile all'indirizzo `jade.tilab.com/wadeproject/`): è una piattaforma software basata su JADE, la quale fornisce supporto per l'esecuzione di task definiti secondo la metafora del workflow. WADE offre un servizio di sviluppo, chiamato WOLF, che facilita la creazione di applicazioni WADE-based. WOLF è un plug-in di Eclipse che permette di sfruttare tutte le funzionalità dell'IDE Eclipse in aggiunta alle potenzialità offerte da WADE. WADE è un software gratuito ed è distribuito da Telecom Italia, in open source secondo i termini e le condizioni della licenza LGPL.

1.2.1 Il Social Gaming

L'industria dei videogiochi online svolge un ruolo significativo nella nostra società, infatti è stata recentemente potenziata dalla rapida diffusione dei giochi per smartphone. La caratteristiche dei nuovi dispositivi cellulari, tra cui la connettività e l'avvento dei novel game permette una nuova e duratura sinergia tra le industrie di apparecchi mobili e videogiochi. Adottiamo una delle tante definizioni di gioco come segue: il gioco è l'interazione formalizzata che si verifica quando i giocatori seguono delle regole e accumulano esperienza sul loro sistema di gioco. Una volta appurato il fatto che l'attività di gioco sia di natura sociale, dobbiamo discutere di come i cosiddetti giochi sociali online, o giochi di società, si differenzino dagli altri tipi di giochi. Questa discussione ci ha portato a identificare le caratteristiche principali dei giochi sociali che qualsiasi piattaforma di social gaming come AMUSE deve avere. Un approccio primitivo è quello di etichettare come sociale, tutti i giochi che utilizzano un social network. Qualsiasi gioco collegato, ad esempio, a Facebook, è un gioco social, ma purtroppo questo non basta per capire le caratteristiche di una piattaforma di gioco sociale. Un approccio meno primitivo individua molte dimensioni del social gaming. Qui ci limitiamo a una caratterizzazione triplice:

- Synchronous vs. asynchronous player interaction: le interazioni avvengono simultaneamente in tempo reale ma in tempi diversi,o come in un gioco a turni;

- Symmetrical vs. asymmetrical relationship formation: forma un rapporto richiedendo l'input da entrambi i giocatori o possono essere formati rapporti unilateralmente da un singolo giocatore;
- Strong tie vs. loose tie relationship evolution: le relazioni tendono a diventare profonde e di lunga durata o sono di tipologia transitoria.

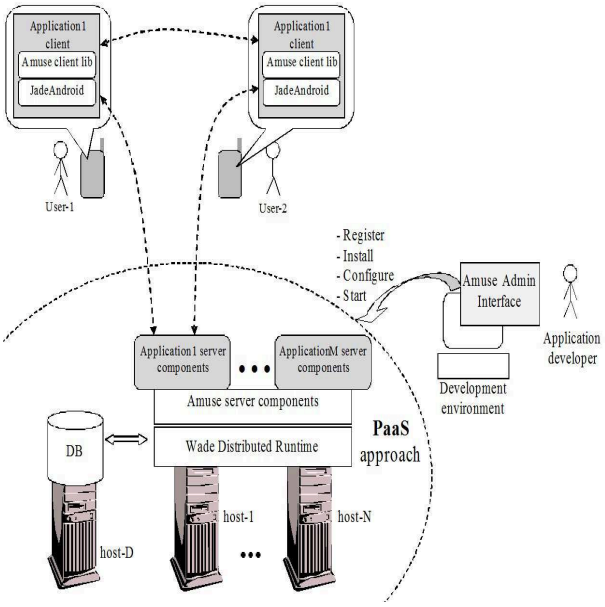
In questo capitolo entriamo nello specifico e e prendiamo come esempio i giochi di società. Sincrono vs. interazione asincrona: i giochi di tipo MMO (Massive Multi-player Online games) sono sincroni, in tempo reale. Mentre i giochi sociali occasionali sono asincroni con interazioni che si verificano a volte in modo scollegato. Tuttavia, anche gli MMO presentano anche importanti funzioni asincrone come i messaggi in-game e qualche gioco basato su Facebook ha caratteristiche sincrone come una chat. I giochi attuali di società, infatti, tendono ad offrire un mix di sincrono e interazioni asincrone.

L'idea del gameplay sincrono è intuitivamente semplice: i giocatori interagiscono in tempo reale e non a turno. Esempi delle interazioni sociali sincrone includono chat di testo, video chat, e elementi di gioco come le battaglie. Le interazioni sincrone possono variare da due giocatori a grandi gruppi. La tipologia di gioco asincrono potrebbe far pensare a qualcosa di più lento e meno intrigante rispetto a quelli sincroni, invece possono essere altrettanto coinvolgenti, ad esempio, basta pensare di giocare a scacchi con un amico a distanza. I giochi sociali asincroni sono disponibili in differenti tipologie di base, queste sono alcune delle più comuni:

- Turn-based shared games;
- Turn-based challenge games;
- Score-based challenge games;
- Open-world asynchronous games.

Architettura di AMUSE

La seguente figura mostra una panoramica dell'architettura di AMUSE: tutte le applicazioni AMUSE-based comprendono la logica necessaria per il lato client insieme a un'eventuale interfaccia grafica. Le applicazioni client fanno uso della libreria AMUSE client library per interagire con altri client e con il server. Nella versione corrente (AMUSE 1.0 rilasciata il 10/12/2014) sono supportati solo le applicazione Android-base Nella maggior parte dei casi le applicazioni AMUSE-based hanno bisogno di una specifica logica che deve essere eseguita nel lato server.



Queste applicazioni, nel terminologia di AMUSE, sono chiamate applicazioni server based. Per l'esecuzioni di queste logiche la piattaforma AMUSE fornisce un ambiente PaaS (Platform as a Service): le applicazioni non sono a conoscenza dei dettagli relativi all'hardware, sistemi operativi e altro riguardanti il server. D'altra parte il sistema è distribuito su un ambiente cloud e AMUSE ha il compito di assicurarsi che ogni lavoro abbia le risorse necessarie per il suo corretto funzionamento. AMUSE mette a disposizione la AMUSE Application Administration Interface, con la quale è possibile registrare, installare, configurare e avviare le specifiche logiche server-side delle varie applicazioni sviluppate. Una volta lanciate i client possono connettersi e sfruttare i servizi messi a disposizione. Nei casi in cui le applicazioni non necessitino di una logica a lato server, in AMUSE ci riferiamo ad esse come applicazioni client-only. Per avviare questo tipo di applicazioni, basta una registrazione sulla AMUSE Application Administration Interface.

Funzionalità di AMUSE

Le funzionalità di AMUSE si dividono in Core e in Gaming. Le Core, al contrario delle Gaming, non sono solo orientate allo sviluppo di giochi e possono essere usate in generiche applicazioni multi utente. Le funzioni Core le troviamo nel package com.amuse.client.features.core, mentre le Gaming, nel package com.amuse.client.features.gaming. Le principali sono:

- Gestione Applicazioni: questa funzione offre la possibilità di registrare, installare, configurare e attivare le applicazioni sviluppate, le quali so-

no disponibili presso la AMUSE Application Administration Interface accessibile via Web;

- Gestione Utenti: questa funzione si occupa della registrazione e della autenticazione degli utenti e supporta in parte minimale la gestione dei profili utenti. Le API client che si occupano di questo sono contenute nella classe com.amuse.client.features.core.UserManagementFeature;

- Sincronizzazione del Clock: questa funzione si incarica della sincronizzazione degli eventi, in modo tale che la stessa applicazione su terminali differenti esegua contemporaneamente un'azione (questa funzione ha una particolare importanza per i giochi real-time, dove un match inizia, gestisce eventi e finisce allo stesso tempo per tutti i giocatori). Scambio di messaggi di testo: questa funzione offre la possibilità di mandare e ricevere messaggi di testo da o verso altri utenti con la stessa applicazione. Questo servizio è accessibile attraverso le API contenute nella classe com.amuse.client.features.core.TextMessageFeature. Nel caso in cui il ricevente non sia connesso nel momento in cui il messaggio è stato inviato, il messaggio sarà automaticamente salvato dell'applicazione e sarà consegnato non appena il ricevente si sarà connesso;

- Gestione delle pipe Peer-to-peer: questa funzione offre la possibilità di stabilire una connessione diretta tra i client, i quali potranno in seguito scambiarsi specifici oggetti. Le API sono accessibili tramite l'interfaccia com.amuse.client.features.core.PipeManagementFeature;

- Coordinazione match Peer-to-peer: questa funzione offre la possibilità di organizzare match one-to-one in modo persistente: se un utente si disconnette durante il match, potrà riconnettersi in un momento successivo. La partecipazione al match utilizza la metafora dell'invito: l'organizzatore manda una richiesta all'avversario in modo esplicito specificando il suo nome o chiede alla piattaforma AMUSE di trovargli un avversario casuale. Una volta accettato l'invito da parte dell'avversario, il match ha inizio. Le applicazioni basate su questo approccio non richiedono nessuna logica da parte del server, e possono essere sviluppate come applicazioni client-only. Le API contenute nella classe com.amuse.client.features.gaming.MatchCoordinationFeature offrono questo servizio;

- Coordinamento centralizzato del match: questo servizio si occupa dell'organizzazione di match che comprendono due o più giocatori, i quali possono entrare o uscire dalla partita, mentre è ancora in svolgimento.

La partecipazione al match utilizza la metafora dei tavoli: l'organizzatore del match crea un tavolo vuoto, dove gli avversari possono sedersi per unirsi alla partita. Non appena il numero minimo di giocatori è stato raggiunto il match ha inizio. I tavoli sono disposti in stanze, dove un giocatore deve accedere prima di sedersi in un qualsiasi tavolo. Le applicazioni basate su questo approccio richiedono una specifica logica che governi il lato server, infatti queste applicazioni vengono chiamate server-based. La logica deve essere implementata estendendo la classe com.amuse.agents.gra.GamesRoomAgent la quale contiene i meccanismi base per la gestione dei tavoli all'interno delle stanze e per la coordinazione dei match giocati sui vari tavoli.

1.3 Social Network

Con l'espressione social network si identifica un servizio informatico on line che permette la realizzazione di reti sociali virtuali. Si tratta di siti Web che consentono agli utenti di condividere contenuti testuali, immagini, video e audio e di interagire tra loro. Secondo gli studiosi Boyd-Ellison, si possono definire siti di social network i servizi Web che permettono:

- la creazione di un profilo pubblico o privato all'interno di un sistema vincolato;
- l'articolazione di una lista di contatti;
- la possibilità di scorrere la lista di amici dei propri contatti.

Generalmente, prevedono una registrazione mediante la creazione di un profilo personale protetto da password partendo da informazioni come il proprio nome, cognome, indirizzo email fino ad arrivare agli interessi e alle passione che abbiamo. Inoltre si ha la possibilità di effettuare ricerche nel database della struttura informatica per localizzare altri utenti e organizzarli in gruppi e liste di contatti. Un importante sviluppo delle reti sociali è rappresentato dalla possibilità di creare da parte di chiunque ne abbia le competenze (sviluppatori con linguaggi solitamente proprietari) applicazioni orientate alla comunità degli iscritti. Tale famiglia di applicazioni, beneficiano della rete di contatti e delle informazioni individuali degli iscritti (es. Facebook, Google+, Twitter) per facilitare operazioni quali registrazioni e accessi a servizi esterni al social network.

1.3.1 Facebook

Facebook è un servizio di rete sociale lanciato nel febbraio 2004, posseduto e gestito dalla corporazione Facebook, Inc., basato su una piattaforma software scritta in vari linguaggi di programmazione. Il sito, fondato a Harvard negli Stati Uniti da Mark Zuckerberg e dai suoi compagni di università Eduardo Saverin, Dustin Moskovitz e Chris Hughes, era originariamente stato progettato esclusivamente per gli studenti dell'Università di Harvard, ma fu presto aperto anche agli studenti di altre scuole della zona di Boston, della Ivy League e della Stanford University. Successivamente fu aperto anche agli studenti delle scuole superiori e poi a chiunque dichiarasse di avere più di 13 anni di età. Da allora Facebook raggiunse un enorme successo: secondo Alexa dal giugno 2013 è diventato il sito più visitato al mondo, superando Google; ha cambiato profondamente molti aspetti legati alla socializzazione e all'interazione tra individui, sia sul piano privato che quello economico e commerciale. È disponibile in oltre 70 lingue e nell'ottobre 2012 contava circa 1 miliardo di utenti attivi, che effettuano l'accesso almeno una volta al mese, classificandosi come primo servizio di rete sociale per numero di utenti attivi. Il nome Facebook prende spunto da un elenco con nome e fotografia degli studenti, che alcune università statunitensi distribuiscono all'inizio dell'anno accademico per aiutare gli iscritti a socializzare tra loro. Gli utenti possono accedere al sito previa una registrazione gratuita, durante la quale vengano richiesti dati personali come nome, cognome, data di nascita e indirizzo email. Il sito chiarisce che l'inserimento obbligatorio della data di nascita serve esclusivamente per favorire una maggiore autenticità e consentire l'accesso ai vari contenuti in base all'età. Completata la registrazione, gli utenti possono creare un profilo personale, includere altri utenti nella propria rete sociale, aggiungendoli come amici, e scambiarsi messaggi, anche via chat, incluse le notifiche automatiche quando questi aggiornano i propri profili. Inoltre gli utenti possono fondare e unirsi a gruppi per condividere interessi in comune con altri utenti, organizzati secondo il luogo di lavoro, la scuola, l'università o altre caratteristiche, condividere contenuti multimediali ed utilizzare varie applicazioni presenti sul sito. Per personalizzare il proprio profilo l'utente può caricare una foto, chiamata immagine del profilo, con la quale può rendersi riconoscibile. Può inoltre fornire ulteriori informazioni, come il comune di nascita (es: Città natale: Roma) e quello di residenza (es: Vive a Torino), la scuola frequentata, il proprio datore di lavoro, l'orientamento religioso e quello politico, la propria situazione sentimentale e molte altro.

1.3.2 Google+

Google+ è una rete sociale gratuita creata da Google Inc. nel 2011. Il servizio è stato lanciato il 28 giugno 2011, in fase test e solo su invito. Il giorno seguente, gli utenti esistenti sono stati autorizzati a invitare gli amici al servizio per creare il proprio account, ma questo è stato rapidamente sospeso il giorno successivo a causa delle eccessive richieste. L'undici agosto 2011 viene annunciato il lancio dei social games. Il venti settembre 2011 apre la beta pubblica, le iscrizioni diventano quindi libere non è quindi più necessario l'invito. Google+ include importanti novità rispetto ad altre reti sociali più affermate, introducendo nuovi contenuti multimediali, infatti offre la possibilità di avviare sessioni audio e video, tramite ad esempio i videoritrovi (chiamati hangouts), stanze virtuali dove è possibile condividere video e parlare allo stesso tempo con tutti i componenti presenti all'interno, tramite microfono e webcam. Sempre tramite la chat gli utenti hanno la possibilità di scambiarsi file. Google+ presenta delle varianti e delle nuove funzionalità che lo contraddistinguono da altri social network. Il sistema dei contatti è organizzato e suddiviso in cerchie (circles) liberamente creabili e modificabili dall'utente. In modo predefinito sono già presenti cerchie denominate amici, conoscenti, lavoro, famiglia, persone che seguo. Tale sistema mira a realizzare un buon livello di privacy. Agendo sulle impostazioni è quindi possibile limitare la diffusione dei dati personali, di qualsiasi notizia o pubblicazione, alle varie cerchie. Un'altra importante funzione di Google+ è detta Spunti (Sparks). Questa funzione permette di creare dei feed semplicemente dopo aver inserito in un box l'argomento interessato. Si creerà un flusso di contenuti inerenti all'argomento scelto, che saranno poi condivisibili con i nostri amici. Google+, come Facebook e Twitter, dispone di un sistema di identificazione delle persone famose, visibile tramite un'icona con un visto accanto al nome del profilo. L'interfaccia di Google+ è di semplice utilizzo e permette anche ai meno esperti di avere un controllo totale del proprio profilo e quindi anche delle persone che possono accedervi e/o mettersi in contatto con l'utente.

1.4 Le API di Google+

Acronimo di Application Programming Interface (in italiano traducibile come Interfaccia di programmazione di un'applicazione), le API sono degli strumenti di programmazione che le maggiori software house e industrie del mondo informatico come Microsoft, Google e Facebook mettono a disposizione degli sviluppatori per facilitare il loro compito nella realizzazione di appli-

cazioni di vario genere. Le API possono assumere diverse forme: possono essere delle librerie di funzioni che permettono al programmatore di interagire con un programma o una piattaforma software o semplicemente una serie di chiamate a parti di un programma che uno sviluppatore può utilizzare per abbreviare il suo lavoro. Le API, quindi, sono delle interfacce grafiche che sviluppatori e programmatori terzi possono utilizzare per espandere le funzionalità di programmi, applicazioni e piattaforme di vario genere (software e non solo). Rappresentano, quindi, l'interfaccia aperta attraverso la quale interagire con programmi (o parti di essi) altrimenti inaccessibili. Utilizzando un'API, un programmatore può far interagire due programmi (o due piattaforme, o un programma e una piattaforma) altrimenti tra loro icompatibili. Utilizzando, quindi, degli artifici di programmazione, si possono estendere le funzionalità di un programma ben oltre le reali intenzioni dello sviluppatore o della sfotware house che l'ha realizzato. Un esempio pratico sono le API delle Google Maps. La casa di Mountain View mette queste API (così come le application programming interface di moltissimi altri suoi servizi Web) a disposizione di tutti gli sviluppatori che le volessero utilizzare per un loro programma o piattaforma Web. Sfruttando le API, ad esempio, è possibile utilizzare il servizio di cartografia digitale di Google per realizzare delle mappe personalizzate; oppure integrarle in siti web per servizi di ricerca georeferenziati; o ancora utilizzarle all'interno di applicazioni per smartphone e tablet. Le API di Facebook, invece, danno la possibilità agli sviluppatori di utilizzare alcune delle funzionalità del social network più famoso al mondo per realizzare delle applicazioni da utilizzare poi nella piattaforma del social network stesso. L'utilizzo di librerie solitamente accessibili esclusivamente al team di sviluppo di Facebook, facilita notevolmente la programmazione di nuove app e funzioni (basta pensare alle decine di giochi presenti nel social network, ad esempio) che arricchiscono l'esperienza degli iscritti di Facebook. Le API di Google+ sono l'interfaccia di programmazione per Google+. Le API si possono utilizzare per integrare le app o i siti Web con Google+. Consente agli utenti di collegarsi tra loro per il massimo coinvolgimento, utilizzando le funzioni di Google+ dall'interno della tua applicazione. Molte chiamate all'API richiedono che l'utente della tua applicazione assegni le autorizzazioni d'accesso ai propri dati. Google utilizza il protocollo OAuth 2.0 `http://oauth.net/2/` per consentire alle applicazioni autorizzate di accedere ai dati dell'utente. Uno dei maggiori vantaggi dei Google Play Services è l'accesso uniforme alle API dei servizi Google. Andiamo a vedere le fasi di interazione con un servizio remoto e la loro contestualizzazione nel ciclo di vita di un'Activity.

Classi e Interfacce da utilizzare

Le classi che utilizzeremo apparterranno alla libreria dei GooglePlayServices, più precisamente al package com.google.android.gms.common. La classe principale è GoogleApiClient. La vedremo presente anche nell'Activity come membro privato, e rimarrà il punto di riferimento, durante l'esecuzione, per la gestione delle connessioni riuscite, mancate e sospese. Con GoogleApiClient impartiremo comandi di connessione e disconnessione ai Google Play Services mediante i metodi connect() e disconnect(). Gli esiti di queste operazioni perverranno all'Activity veicolati da due interfacce, entrambe definite internamente alla classe GoogleApiClient:

- ConnectionCallbacks: in questo caso avremo a disposizioni i seguenti metodi: onConnected, onConnectionSuspended. Il primo viene invocato dal sistema come conferma dell'avvenuta connessione. Da questo momento in poi si potrà invocare qualunque funzionalità presso il servizio remoto. Al contrario, la onConnectionSuspended, notificherà un'eventuale imprevista disconnessione dal servizio;

- OnConnectionFailedListener: fornisce il metodo: onConnectionFailed. Il codice che lo implementa dovrebbe occuparsi di notificare all'utente le difficoltà occorse ed intraprendere eventuali operazioni di recupero.

Altro punto importante è la coesistenza delle operazioni da svolgere, con il ciclo di vita dell'Activity. Un'applicazione deve essere correttamente integrata nel sistema operativo Android, e affinchè ciò sia avvenga è necessario innanzitutto che essa collabori alla corretta gestione delle risorse. A tal fine, è importante scegliere di utilizzare il codice più appropriato da inserire nei metodi del ciclo di vita. Visto che vogliamo realizzare connessioni ai servizi Google, è importante che la connessione e la disconnessione avvengano, rispettivamente, quando l'utente inizia ad utilizzare l'Activity e quando smette di farlo. A tal proposito, le coppie di metodi del ciclo di vita utilizzate di solito per individuare questo lasso di tempo sono onResume-onPause (fase di interazione viva tra utente e UI) e onStart-onStop (fase di visibilità dell'interfaccia). Affinchè si possa interagire con le API di Google è necessario che i Google Play Services siano disponibili.

Capitolo 2

Login Tramite Facebook alla Piattaforma AMUSE

2.1 Introduzione

Fornitami un'applicazione che implementa la connessione tramite Facebook ad AMUSE, mi è stato chiesto di aggiungere il Login tramite Google+. In un primo momento ho studiato il comportamento dell'applicazione, per poi poterci applicare quello che avevo appreso nello studio del Login tramite Google+, come descritto nel sito di Google Developers (disponibile all'indirizzo `https://developers.google.com/+/mobile/android/people`). La difficoltà principale è stata il riuscire ad adattare il Login di Google+ ad un Fragment. Un Fragment non è altro che una parte dell'Activity principale, è utile per suddividere il lavoro in più sottoinsiemi e rendere il codice più chiaro. Il problema principale che ho riscontrato, riguardava un pezzo di codice ausiliario che andava aggiunto nella main activity per far si che il tutto andasse a buon fine (rivedremo meglio il tutto nella sezione relativa al Login con Google+). Proprio come era stato fatto per Facebook ho passato a una seconda activity la foto profilo e il nome dell'utente di Google+ e da li, tramite un tasto, ho permesso l'accesso all'ultima activity, in cui posso accedere agli amici ripartiti nelle varie categorie, rispettivamente: AMUSE, Facebook e Google+. Ognuna di queste categorie di amici viene gestita tramite un Fragment apposito, che si riferirà ad una main activity in cui vengono gestite le trasizioni da un Fragment all'altro. Una volta ottenuti gli amici li ho resi selezionabili e quindi ho fatto si che potessero essere invitati. Per una spiegazione più esaustiva dei Login si rimanda alla lettura dei capitoli successivi.

2.2 Login tramite Facebook

Una volta avvita l'applicazione ci viene presentata subito la scelta del login che possiamo effetuare, infatti possiamo scegliere di loggarci tramite AMUSE oppure tramite Facebook. Ovviamente se scegliamo di collegarci tramite AMUSE e non siamo ancora registrati, ci verrà chiesto di farlo tramite uno username e una password. A questo punto, focalizziamoci sul login tramite Facebook, andandone a spiegare il comportamento sia a livello superficiale, sia a livello implementativo.

2.2.1 Quello che vede l'utente

Al click del pulsante relativo al login di Facebook compare una barra di caricamento e si viene immediatamente proiettati in un altra pagina in cui, dopo qualche secondo di caricamento, compariranno alcuni attributi propri dell'utente che si è autenticato, come: la foto profilo e il nome. Oltre agli attributi dell'utente è presente anche un pulsante che ci permette di passare ad un'altra sezione dell'applicazione dedita alla visualizzazione degli amici disponibili per essere sfidati. A questo punto clicchiamo sul pulsante `Play with friends` e giungeremo nella pagina conclusiva dell'app in cui avremo tre sezioni, a cui potremo accedere scorrendo da destra a sinistra e viceversa. La prima sezione è riservata agli amici di AMUSE, la seconda è riservata agli amici di Facebook e la terza agli amici di Google+. Ovviamente se all'inizio avremo fatto l'accesso con Facebook ci verranno presentati solo gli amici di Facebook disponibili, mentre nelle sezioni relative ad AMUSE e Google+ saranno presenti i bottoni di accesso, che una volta effettuato, riporteranno anche gli amici di AMUSE e Google+. Per poter inviare gli inviti, basta cliccare sugli amici (da un minimo di un amico ad un massimo di 4 amici) e premere il pulsante `invite`. Gli amici selezionati nella lista verranno evidenziati con una spunta.

2.2.2 Implementazione

In questa sezione andremo ad analizzare più da vicino l'applicazione che mi è stata fornita prima dell'aggiunta del login di Google+.

Registrazione dell'App

Prima di tutto, è necessario creare una Facebook App . Quindi, aprire il browser Web e recarsi sul sito `https://developers.facebook.com/`, dopodichè è possibile effettuare il login a Facebook, oppure, se sei uno sviluppa-

tore, puoi registrare l'applicazione cliccando su My App e poi su Register accettare le politiche.

Struttura dell'App

Di seguito viene riportato, sotto forma di elenco, le varie parti di cui si compone la App:

- MainActivity: È l'Activity lanciata all'inizio del applicazione , contiene il modulo per l'accesso ad AMUSE e il Facebook LoginButton Fragment;
- MyApp: Questa classe estende l'applicazione e contiene i metodi necessari per il login nella piattaforma AMUSE;
- FacebookLoginFragment: Questo Fragment contiene i metodi necessari per effettuare il Login con Facebook;
- WelcomeActivity: Questa Activity mostra le informazioni dell'utente e permette di vedere gli amici invitabili con il tasto Play with Friends;
- FriendsActivity: Questa Activity è divisa in tre Fragment che gestiscono gli amici: Amuse-FriendsFragment, FacebookFriendsFragment e GoogleFriendsFragment;
- AmuseFriendsFragment: Questo Fragment mostra gli amici AMUSE come una lista, è possibile aggiungere gli amici e giocare con loro;
- FacebookFriendsFragment: Questo Fragment mostra gli amici Facebook che sono loggati in questa app con Facebook;
- GoogleFriendsFragment: Implementato da me e verrà approfondito nei paragrafi successivi.

Approfondimento del codice

In questa sezione andiamo ad analizzare le parti più importanti del codice, al fine di dare una traccia degli step principali dell'implementazione dell'App che ci è stata fornita.

- `Setup del progetto`: Per usare la Facebook SDK è necessario selezionare le API 9: Android 2.3 o superiori. Per prima cosa va aggiunto nel file string.xml una nuova stringa dal nome `facebook_app_id` che contiene il tuo Facebook App ID, che potrai trovare nella dashboard della tua App. Dopodichè va inserito nel file Manifest il tag dell'applicazione:

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

Poi va aggiunta la FacebookActivity al manifest;

```
..
<activity android:name="com.facebook.FacebookActivity"
android:theme="@android:style/Theme.Translucent.
    NoTitleBar"
android:label="@string/app_name" />
..
```

- `Creazione e gestione del bottone di Login`: Il bottone di Login va aggiunto al layout file, immettendo il seguente codice:

```
<com.facebook.login.widget.LoginButton
android:id="@+id/facebook_login_button"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_gravity="center_horizontal"
android:padding="15dp"/>
```

Dopo aver creato una nuova classe che estende la classe Fragment, al suo interno va chiamato il FacebookSdk.initialize per inizializzare l'SDK e quindi poter chiamare il CallbackManager.Factory.create per creare un gestore di callback per gestire le risposte agli accessi effettuati. Ecco un esempio di aggiunta della callback in un Fragment:

```
private CallbackManager callbackManager;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
FacebookSdk.sdkInitialize(getActivity());
callbackManager = CallbackManager.Factory.create();
}
```

Ora andremo a vedere le parti di codice più importanti all'interno della OnCreateView e della onActivity Result per la gestione del Login:

1. Inizializzazione del bottone di Login:

```
loginButton = (LoginButton)view.findViewById(R.id.
   facebook_login_button);
loginButton.setReadPermissions("user_friends");
loginButton.setFragment(this);
```

2. Registro della Callback:

```
loginButton.registerCallback(callbackManager,
          new FacebookCallback<LoginResult>() {
    @Override
    public void onSuccess(LoginResult loginResult)
   {
    GraphRequest request = GraphRequest.
   newMeRequest(
          loginResult.getAccessToken(),
                new GraphRequest.
   GraphJSONObjectCallback() {
                    @Override
                    public void onCompleted(
   JSONObject object,
                                GraphResponse response) {
                String id = object.getString("id")+"
   $FACEBOOK";
              ((MainActivity)getActivity()).
   facebookLogin(id,
                 AccessToken.getCurrentAccessToken().
   getToken());
                    }
                }
              });
```

Vediamo da questo frammento di codice come venga chiamata la callback sul pulsante di login e venga gestito il successo della transazione tramite la onSuccess. Dopodichè avremo l'esecuzione della callback relativa al Graph di Facebook e in caso di completamento (onCompleted), verrà creato un ID con l'Id restituito da Graph e gli verrà aggiunto `"$FACEBOOK"` per permettere ad AMUSE il riconoscimento del Login attraverso Facebook. Come ultimo passaggio viene richiamata la funzione facebookLogin (MainActivity) che ci permetterà di passare ad AMUSE l'ID modificato e l'access tocken che verrà usato come password.

3. onActivityResult: Infine, nel onActivityResult, vengono trasmessi i risultati del Login al callbackManager, oggetto creato nella onCreate:

```
@Override
protected void onActivityResult(int requestCode,
    int resultCode,
Intent data) {
super.onActivityResult(requestCode, resultCode,
    data);
callbackManager.onActivityResult(requestCode,
    resultCode, data);
}
```

- `Aggiungere il bottone di Login alla MainActivity` : Va creata l'Activity in cui si vuole aggiungere il bottone di Login a Facebook e aggiungere al file di layout Android il seguente codice:

```
<fragment
android:name="com.amuse.facebooktutorial.
    FacebookLoginFragment"
android:id="@+id/facebook_fragment"
android:layout_width="match_parent"
android:layout_height="wrap_content" />
```

Per prima cosa il dominio di Facebook deve essere registrato in AMUSE, per cui va aggiunto il seguente codice nella onCreate dell' Activity:

```
private DefaultLoginManager loginManager;
public void onCreate(Bundle savedInstanceState) {
..
setContentView(R.layout.main_activity_layout);
AmuseUtils.registerDomain("FACEBOOK");
loginManager = new DefaultLoginManager(this);
..
}
```

Adesso va creato il metodo facebookLogin di cui abbiamo parlato prima:

```
public void facebookLogin(String username, String
    password) {
loginManager.loginDone(new LoginInfo(username, password
    , true));
```

```
}
```

Un account di Facebook deve sempre effettuare il login in AMUSE come un nuovo account.

- `FriendsActivity`: è necessario creare un layout contenente una ListView, che mostrerà gli amici, e un tasto, che verrà utilizzato per invitare uno o più di questi.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/
   android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical" >
<ListView
android:id="@+id/facebook_friends_listview"
android:layout_width="match_parent"
android:layout_height="wrap_content"/>
<Button
android:id="@+id/friends_invite_button"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:padding="10dp"
android:text="Invite" >
</Button>
</LinearLayout>
```

Ora va creata un'Activity in cui dovremmo riinizializzare alcuni componenti come:

  - FacebookSDK e la CallbackManager;
  - Una lista di profili Facebook in cui mettere gli amici;
  - Una lista di stringhe in cui mettere gli amici.

Il passaggio successivo è quello di implementare una List adapter. Per farlo, va creato un nuovo layout che conterrà l'immagine del profilo e il nome dell'amico.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/
    android"
android:orientation="horizontal"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:padding="5dp">
<com.facebook.login.widget.ProfilePictureView
android:id="@+id/facebook_friend_image"
android:layout_width="30dp"
android:layout_height="30dp" />
<TextView
android:id="@+id/facebook_friend_name"
android:layout_width="wrap_content"
android:layout_height="match_parent"
android:layout_gravity="center_vertical"
android:layout_marginRight="56dp"
android:paddingLeft="5dp"
android:textSize="24sp"
android:textColor="@android:color/black">
</TextView>
</LinearLayout>
```

Adesso va creata all'interno dell'Activity una nuova classe che estenda BaseAdapter:

```
public class FacebookFriendsAdapter extends BaseAdapter
    {
@Override
public View getView(int position, View convertView,
    ViewGroup
parent) {
LayoutInflater inflater = (LayoutInflater) getActivity
    ()
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
convertView = inflater.inflate(R.layout.
    facebook_friends_row,
null);
TextView name = (TextView)
convertView.findViewById(R.id.facebook_friend_name);
ProfilePictureView pictureView =
(ProfilePictureView)
convertView.findViewById(R.id.facebook_friend_image);
pictureView
```

```
.setProfileId(facebookFriendsList.get(position).getId()
    );
name.setText(facebookFriendsList.get(position).getName
    ());
return convertView;
}
@Override
public long getItemId(int position) {
return position;
}
@Override
public Object getItem(int position) {
return facebookFriendsList.get(position);
}
@Override
public int getCount() {
return facebookFriendsList.size();
}
}
```

Con l'inflater possiamo riempire ogni casella della lista creata in precedenza ogni view. Ora è possibile inizializzare la view e settare l'adapter all' interno del metodo onResume() dell'Activity.

```
public void onResume() {
super.onResume();
facebookFriendsListView =
(ListView) view.findViewById(R.id.
    facebook_friends_listview);
inviteButton = (Button)
view.findViewById(R.id.friends_invite_button);
friendsAdapter = new FacebookFriendsAdapter();
facebookFriendsListView.setAdapter(friendsAdapter);
...
updateFriendsList();
}
```

Capitolo 3

Login Tramite Google+ alla Piattaforma AMUSE

3.1 Introduzione

A questo punto inizia la descrizione della parte innovativa del progetto. Si è iniziato facendo vari esperimenti sull'uso del login di Google+, per poter capire come poter arrivare, tramite l'utilizzo delle API, all'access tocken e all'ID relativo a un account Google+. Nei vari tentativi ho anche implementato un piccolo echo server per poter studiare più da vicino lo scambio di dati client server. Nell'implementazione ho seguito il più fedelmente possibile la struttura dell'applicazione che mi era stata fornita, andando ad aggiungere, ove opportuno, le parti necessarie per consentire l'accesso tramite Google+.

3.2 Attivazione dell'applicazione

Per eseguire l'autenticazione e comunicare con le API di Google+, per prima cosa va registrato il certificato pubblico del tuo file .apk con firma digitale nella Console delle API di Google:

1. Nella Console delle API di Google, crea un progetto API con il nome della tua app;

2. Nel riquadro Services (Servizi), attiva Google+ API (API di Google+) e qualsiasi altra API richiesta dalla tua app;

3. Nel riquadro API Access (Accesso alle API), crea un ID client OAuth 2.0 facendo clic su Create an OAuth 2.0 Client ID (Crea un ID client OAuth 2.0);

4. Digita il nome di un prodotto nella finestra di dialogo visualizzata, quindi fai clic su Next (Avanti);

5. Scegli Installed application (App installata) come Application type (Tipo app), quindi seleziona Android come tipo;

6. Nel campo Package name (Nome pacchetto), inserisci il nome pacchetto della tua app di Android. Il nome del pacchetto utilizzato dall'app campione è il seguente:
`amuse.android.pplus`;

7. In un terminale, esegui l'utility Keytool per ricavare l'ID digitale SHA-1 del certificato:

```
keytool -exportcert -alias androiddebugkey -keystore <
    path-to-debug-or-production-keystore> -list -v}
```

*

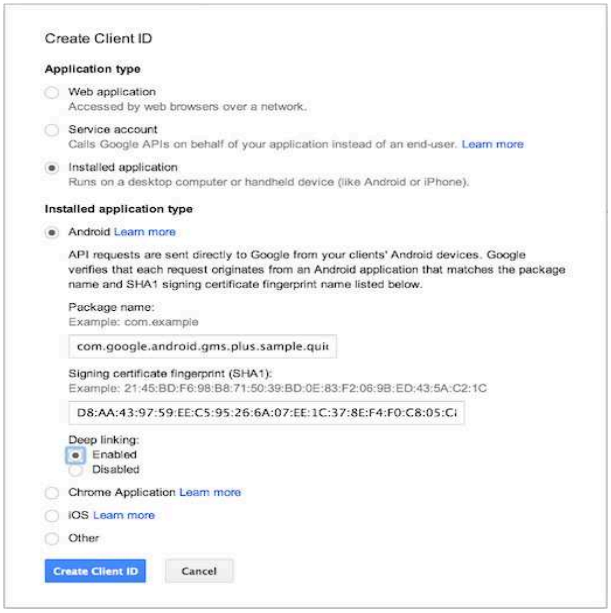
8. Il keytool suggerisce di inserire una password per il keystore.
La password predefinita per il keystore di debug è android. Quindi, il keytool stampa l'ID digitale nella shell. Ad esempio:

```
Certificate fingerprint:
SHA1: DA:39:A3:EE:5E:6B:4B:0D:32:55:BF:EF:95:60:18:90:
    AF:D8:07:09
```

9. Incolla l'ID digitale SHA-1 nel campo signing certificate fingerprint (ID digitale certificato di firma) e fai clic su Create client ID (Crea ID client);

10. Per attivare i post interattivi, attiva l'opzione Deep Linking. Ecco come si presenta la schermata per creazione dell' ID:

*Nota. Per Eclipse, il keystore di debug si trova in genere in `/.android/debug.keystore`.



3.3 Login tramite Google+ ad AMUSE

Come abbiamo visto per Facebook, dovremo aggiungere il pulsante nel layout del Fragment, per poi andare ad implementare il login nel Fragment stesso. Questo, poi, si andrà ad allacciare alla MainActivity. Andiamo ad analizzare i vari passaggi che mi hanno permesso di realizzare il Login.

3.3.1 Aggiunta del pulsante di Login

Ho creato un nuovo Layout apposito per il pulsante di , l'ho quindi inserito in un file `.xml` che ho chiamato `google_login_layout`:

```
<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk
    /res/android"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">



       <com.google.android.gms.common.SignInButton
           android:id="@+id/google_login_button"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"/>


    <Button
        android:id="@+id/btn_sign_out"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/btn_logout_from_google"
        android:visibility="gone"
        android:layout_marginBottom="10dp"/>


</LinearLayout>
```

All'interno ho inserito anche il bottone di sign-out che compare subito dopo l'avvenuta connessione, al posto del pulsante di sign-in.

3.3.2 Creazione del Fragment di Login con Google+

Ora andiamo a creare un Fragment relativo alla Main Activity, e ne andremo a studiare le componenti più significative. Prima però, per gestire l'accesso a Google+, ho aggiunto al già presente layout realativo alla Main Activity, la seguente parte di codice immediatamente dopo a quello relativo al Fragment di Facebook: `main_activity_layout`:

```
...
  </LinearLayout>

            <fragment android:name="com.amuse.
   facebooktutorial.FacebookLoginFragment"
                android:id="@+id/facebook_fragment"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"/>

            <fragment android:name="com.amuse.
   facebooktutorial.GoogleLoginFragment"
                android:id="@+id/google_fragment"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"/>
```

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

Descrizione delle componenti

Giunti a questo punto creiamo la onCreateView all'interno del Fragment. Qui è fondamentale inizializzare l'oggetto `mGoogleApiClient` (che avrà tipo GoogleApiClient) che ci permettrà di comunicare con le API di Google e inoltre, inseriamo i Listener dei bottoni di sign-in e sign-out.

```
public class GoogleLoginFragment extends Fragment
   implements
ConnectionCallbacks, OnConnectionFailedListener {

...

// Google client to interact with Google API
private GoogleApiClient mGoogleApiClient;

...

/**
* A flag indicating that a PendingIntent is in progress and
    prevents us
* from starting further intents.
*/
private boolean mIntentInProgress;

...

private SignInButton btnSignIn;
private Button btnSignOut;


@Override
  public View onCreateView(LayoutInflater inflater,
   ViewGroup container,
       Bundle savedInstanceState) {
    // TODO Auto-generated method stub
   View view = inflater.inflate(R.layout.
   google_login_layout, container, false);
```

```
    btnSignIn = (SignInButton) view.findViewById(R.id.
google_login_button);
    btnSignOut = (Button) view.findViewById(R.id.
btn_sign_out);


    mGoogleApiClient = new GoogleApiClient.Builder(view.
getContext())
    .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this).addApi(Plus.API)
    .addScope(Plus.SCOPE_PLUS_LOGIN).build();

    btnSignIn.setOnClickListener(new View.OnClickListener()
   {

          @Override
          public void onClick(View v) {

              signInWithGplus();
            //mGoogleApiClient.connect();

          }
      });

    btnSignOut.setOnClickListener(new View.OnClickListener
() {

          @Override
          public void onClick(View v) {

            signOutFromGplus();

          }
      });

   return view;

  }
```

In base allo stato di connessione `mGoogleApiClient` chiama alcuni listener, che dovremo andare ad aggiungere al nostro codice, questi sono:

- onConnectionFailed: Viene chiamato nel caso in cui la richiesta di connessione ai Google Play Services non vada a buon fine;

- onConnectionSuspended: Viene invocato nel caso in cui ci sia un'interruzione durante la connessione;
- resolveSignInError: questo metodo mi permette di riolvere eventuali errori durante il sign-in;
- onConnected: Fa parte dell'interfaccia ConnectionCallbacks e serve a gestire la connessione ai Google Play Services in modo appropriato. Il metodo onConnected viene chiamato a connessione eseguita, come suggerisce anche il nome, e serve a mettere a disposizione degli sviluppatori una callback per gestire l'avvenuta connessione con i servizi di Google; All'interno di questo metodo ho recuperato le informazioni di base dell'utente connesso, come: email, foto, ID e accesstocken. In seguito passerò l'access tocken e l'ID alla funzione `googleLogin`, che ho creato nella MainActivity; essa farà in modo che le credenziali arrivino ad AMUSE, che le gestirà in modo tale da ultimare l'accesso.

```
/**
* Method to resolve any signin errors
* */
private void resolveSignInError() {
if (mConnectionResult.hasResolution()) {
    try {
        mIntentInProgress = true;
        mConnectionResult.startResolutionForResult(
   getActivity(), RC_SIGN_IN);
    } catch (SendIntentException e) {
        mIntentInProgress = false;
        mGoogleApiClient.connect();
    }
}
}

@Override
public void onConnectionFailed(ConnectionResult result)
    {
Log.e("errore", result.toString());

if (!result.hasResolution()) {
    GooglePlayServicesUtil.getErrorDialog(result.
   getErrorCode(), getActivity(),
            0).show();
    return;
}
```

```
if (!mIntentInProgress) {
    // Store the ConnectionResult for later usage
    mConnectionResult = result;

    if (mSignInClicked) {
        // The user has already clicked 'sign-in' so we
    attempt to
        // resolve all
        // errors until the user is signed in, or they
   cancel.
        resolveSignInError();
    }
}

}

...

@Override
public void onConnected(Bundle Bundle) {
mSignInClicked = false;
Toast.makeText(getActivity(), "User is connected!",
   Toast.LENGTH_LONG).show();


try {
    if (Plus.PeopleApi.getCurrentPerson(
   mGoogleApiClient) != null) {
        Person currentPerson = Plus.PeopleApi
                .getCurrentPerson(mGoogleApiClient);

        String email = Plus.AccountApi.getAccountName(
   mGoogleApiClient);
        String id = currentPerson.getId()+"$GOOGLE";
        String token = new RetrieveTokenTask().execute(
   email).toString();

        Log.i("token",token);
        Log.i("id",id);

    ((MainActivity)getActivity()).googleLogin(id, token
   );
```

```
    } else {
        Toast.makeText(getActivity(),
                "Person information is null", Toast.
  LENGTH_LONG).show();
    }
} catch (Exception e) {
    e.printStackTrace();
}

updateUI(true);
}

@Override
public void onConnectionSuspended(int arg0) {
mGoogleApiClient.connect();
updateUI(false);
}
```

Per poter ottenere il tocken utilizziamo un altro thread attraverso RetrieveTokenTask che estende AsyncTask e a cui passo la e-mail:

```
private class RetrieveTokenTask extends AsyncTask<
  String, String, String> {

@Override
protected String doInBackground(String... params) {
    String accountName = params[0];
    String scopes = "oauth2:profile email";
    String token = null;
    try {
        token = GoogleAuthUtil.getToken(getActivity().
  getApplicationContext(), accountName, scopes);
        Log.e("token", token);
    } catch (IOException e) {
        Log.e(TAG, e.getMessage());
    } catch (UserRecoverableAuthException e) {
        startActivityForResult(e.getIntent(),
  REQ_SIGN_IN_REQUIRED);
    } catch (GoogleAuthException e) {
        Log.e(TAG, e.getMessage());
    }
    return token;
```

```
}
}
```

- onActivityResult: Questo medoto mi ha creato qualche problema, poichè, per far si che andasse a buon fine il Login, andava aggiunto un altro metodo onActivityResult anche nella Main Activity che mi permettesse di poter arrivare alla onActivityResult all'interno del Fragment. Questo è dovuto al fatto che la resolveSignInError ritorni il suo risultato all'Activity principale e che, quindi, di fatto, non venga mai invocata la onActivityResult all'interno del GoogleLoginFragment. Di seguito riporto entrambi i metodi:
  - onActivityResult nella Main Activity:

```
@Override
  public void onActivityResult(int requestCode, int
    resultCode, Intent data) {

      if (requestCode == GoogleLoginFragment.
  RC_SIGN_IN) {
        GoogleLoginFragment fragment = (
  GoogleLoginFragment) getFragmentManager()
                  .findFragmentById(R.id.
  google_fragment);
          fragment.onActivityResult(requestCode,
  resultCode, data);
      } else {
          super.onActivityResult(requestCode,
  resultCode, data);
      }
  }
```

  - onActivityResult nel Fragment:

```
@Override
public void onActivityResult(int requestCode, int
   responseCode,
    Intent intent) {
if (requestCode == RC_SIGN_IN) {
    if (responseCode !=  Activity.RESULT_OK) {
        mSignInClicked = false;
    }
```

```
        mIntentInProgress = false;

        if (!mGoogleApiClient.isConnecting()) {
            mGoogleApiClient.connect();
        }
    }
}
```

- signInWithGplus: E' il metodo che andremo ad invocare una volta cliccato il bottone di Login e che ci permetterà di loggarci.

```
/**
* Sign-in into google
* */
private void signInWithGplus() {
if (!mGoogleApiClient.isConnecting()) {
    mSignInClicked = true;
    resolveSignInError();
}
}
```

3.4 Visualizzazione delle specifiche dell'utente

A questo punto siamo in possesso delle credenziali dell'utente Google+ e andiamo a modificare la pagina di WelcomeActivity in modo che ci venga presentata la foto e il nome dell'utente. Questa Activity era gia presente nel codice fornitomi, quindi ho dovuto far in modo che venissero presentate le credenziali di Facebook, nel caso dell'acceso tramite Facebook e le credenziali di Google+ nel caso di acceso con quest'ultimo.

3.4.1 Modifiche welcome activity layout

Modifichiamo la `welcome_activity_layout` per aggiungere al layout anche l' immagine profilo e il nome utente di Google+:

```
<ImageView
        android:id="@+id/imgProfilePic"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />
```

```
<TextView
    android:id="@+id/txtName"
    android:layout_width="match_parent"
    android:layout_height="20dp"
    android:layout_marginTop="20dp"
    android:gravity="center" />
```

3.4.2 Modifiche WelcomeActivity

Accediamo a questo nuovo layout tramite un intent che lanciamo nella Main Activity in seguito alla riuscita dell'esecuzione delle funzioni, che invochiamo dall'Activity MyApp, e che ci permettono di interagire con il server AMUSE. Quello che permette il funzionamento di questo meccanismo è la funzione (di cui abbiamo parlato anche nei capitoli precedenti) `googleLogin` che va a modificare il LogniManager. Ecco qui di seguito il codice in cui ho aggiunto anche la registrazione del dominio Google ad AMUSE:

```
AmuseUtils.registerDomain("GOOGLE");

...

// Connect to the Amuse server
    ((MyApp) getApplication()).connect(loginManager, new
  Callback<Void>() {
      @Override
      public void onSuccess(Void v) {
        Log.i(MyApp.TAG, "Connect SUCCESS !!!!!!");
        // Connection to the Amuse server OK
        ((MyApp) getApplication()).initGamesRoomFeature();
        MainActivity.this.runOnUiThread(new Runnable() {
          @Override
          public void run() {
            Intent i = new Intent(MainActivity.this,
  WelcomeActivity.class);
            startActivity(i);

            MainActivity.this.finish();
          }
        });
      }

 ...
```

```
public void googleLogin(String username, String password)
  {
    loginManager.loginDone(new LoginInfo(username, password
  , true));
  }
```

Quello che ho aggiunto io nella WelcomeActivity è la seguente:

- Nella onCreate ho inizializzato l'oggetto GoogleApiClient, imgProfilePic e txtName:

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

    ...

    imgProfilePic = (ImageView) findViewById(R.id.
  imgProfilePic);
        txtName = (TextView) findViewById(R.id.txtName)
  ;


        mGoogleApiClient = new GoogleApiClient.Builder(
  this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this).addApi(
  Plus.API)
        .addScope(Plus.SCOPE_PLUS_LOGIN).build();
  }
```

- Nella onConnected ho recuperato lo username da AMUSE e, tramite un if, vado a verificare se l'accesso sia stato fatto da Google, dopodichè, tramite l'utilizzo delle API, metto all'interno di username e personPhotoUrl rispettivamente il nome dell'utente loggato e l'Url della sua foto profilo:

```
@Override
  public void onConnected(Bundle connectionHint) {
    // TODO Auto-generated method stub

    String username = ((MyApp) getApplication()).
   getAmuseClient().getUsername();

    if (AmuseUtils.getDomain(username).compareTo("
   GOOGLE") == 0){
    Person currentPerson = Plus.PeopleApi.
   getCurrentPerson(mGoogleApiClient);

    username = currentPerson.getDisplayName();
         String personPhotoUrl = currentPerson.getImage
   ().getUrl();



          txtName.setText(username);



      ...


    }
    updateUI(true);

  }
```

Per recuperare la foto profilo ho impostato la dimensione della foto andando ad aggiungerla nella stringa personPhotoUrl, per poi creare un nuovo oggetto LoadProfileImage a cui passo imgProfilePic (per dirgli dove mandare la foto) e lancio la execute passandogli personPhotoUrl. LoadProfileImage estende AsyncTask e,lanciando un nuovo thread, ottiene la foto profilo:

```
@Override
  public void onConnected(Bundle connectionHint) {
```

```
    ...

            personPhotoUrl = personPhotoUrl.substring(0,
                    personPhotoUrl.length() - 2)
                    + PROFILE_PIC_SIZE;

            new LoadProfileImage(imgProfilePic).execute(
    personPhotoUrl);


    ...


   }

    ...

private class LoadProfileImage extends AsyncTask<String
    , Void, Bitmap> {
            ImageView bmImage;

            public LoadProfileImage(ImageView bmImage) {
                this.bmImage = bmImage;
            }

            protected Bitmap doInBackground(String...
    urls) {
                String urldisplay = urls[0];
                Bitmap mIcon11 = null;
                try {
                    InputStream in = new java.net.URL(
    urldisplay).openStream();
                    mIcon11 = BitmapFactory.decodeStream(
    in);
                } catch (Exception e) {
                    Log.e("Error", e.getMessage());
                    e.printStackTrace();
                }
                return mIcon11;
            }

            protected void onPostExecute(Bitmap result) {
```

```
                bmImage.setImageBitmap(result);
            }
        }
```

- Per far si che vengano visualizzate solo le informazioni del social network con cui ci si è collegati ho aggiunto la funzione updateUI, che invocata con parametro true o false, mi permette di nascondere o rendere visibili le informazioni:

```
private void updateUI(boolean isSignedIn) {
        if (isSignedIn) {
          pictureView.setVisibility(View.GONE);
          profileCredentials.setVisibility(View.GONE);
        } else {
          pictureView.setVisibility(View.VISIBLE);
          profileCredentials.setVisibility(View.VISIBLE
  );
        }
    }
```

3.5 Gestione degli amici e degli inviti

Quello che ho dovuto fare in questa parte del progetto è stato creare un nuovo Fragment che facesse riferimento all'Activity FriendsActivity e che mi permettesse di effettuare l'accesso a Google+ nel caso l'utente si fosse connesso all'inizio con Facebook o AMUSE e che, una volta loggato, mi presenti una lista di tutti gli utenti che hanno installato la mia applicazione e che io possa invitare a giocare.

3.5.1 Creazione layout google friends layout

In questo layout ho inserito: la lista degli amici, il pulsante di sign-in, il pulsante per invitare gli amici e il pulsante `get_friends` che mi occorrerà per ottenere la lista degli amici dopo aver effettuato il login.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/
   android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical">

    <ListView
        android:id="@+id/person_list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />


     <com.google.android.gms.common.SignInButton
        android:id="@+id/google_login_button"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20dp"/>

    <Button
        android:id="@+id/friends_invite_button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="10dp"
        android:text="Invite" >
    </Button>

    <Button
        android:id="@+id/get_friends"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="10dp"
        android:text="GET_FRIENDS" >
    </Button>
</LinearLayout>
```

3.5.2 Creazione Fragment GoogleFriendsFragment

Nell'applicazione che mi è stata fornita era già stata implementata l'Activity principale che gestiva le transazioni tra gli amici di AMUSE, Facebook e Google+. A questo punto, aggiungendo il Fragment GoogleFriendsFragment, è stato implementata anche la sezione riservata agli amici di Google+ che era mancante. Il Fragment da me creato, è gestito insieme agli altri, nella Activity principale (FriendsActivity). Andiamo a vedere nel dettaglio le principali operazioni effettuate:

- Nella onCreateView le parti di codice più importanti sono:

1. Istanzio mGoogleApiClient in modo leggermente diverso da come avevamo visto in precedenza, infatti aggiungiamo l'oggetto options di tipo PlusOptions che andiamo a passare a addApi e ci servirà per recuperare gli amici:

```
PlusOptions options = PlusOptions.builder().
    addActivityTypes().build();
mGoogleApiClient = new GoogleApiClient.Builder(view
    .getContext())
            .addConnectionCallbacks(this)
            .addOnConnectionFailedListener(this)
            .addApi(Plus.API, options)
            .addScope(Plus.SCOPE_PLUS_LOGIN)
            .build();
```

2. Dichiaro e inizializzo tutti gli elementi che mi occorrono per poter mettere all'interno di una lista l'elenco degli amici che l'utente potrà invitare:

```
private ArrayAdapter mListAdapter;
private ListView mPersonListView;
private ArrayList<String> mListItems;

  ...

mListItems = new ArrayList<String>();
mListAdapter = new ArrayAdapter<String>(getActivity
    (),android.R.layout.simple_list_item_1,
    mListItems);
mPersonListView = (ListView) view.findViewById(R.id
    .person_list);
```

3. Adesso andiamo ad utilizzare un Listener per rendere la lista di amici sensibile al tocco:

```
mPersonListView.setOnItemClickListener(new
    OnItemClickListener() {

   @Override
   public void onItemClick(AdapterView<?> parent,
    View view, int position, long id) {

     //ImageView friendGoogleSelect = (ImageView)
    view.findViewById(R.id.google_friend_selected);
```

```
    String usernameFriend = mListItems.get(position
  ).toString()+"$GOOGLE";

    invitedFriends.add(usernameFriend);

    Toast.makeText(getActivity(), "Friends add to
  List", Toast.LENGTH_SHORT).show();

  }
});
```

4. Infine inseriamo un meccanismo all'interno del tasto Invite Friends, in modo tale da eliminare i gli utenti che si vogliano sfidare e che siano stati selezionati più volte. Viene anche gestito il numero di sfidanti che l'utente può invitare, infatti potranno essere più di uno ma meno di 4:

```
inviteButton = (Button) view.findViewById(R.id.
  friends_invite_button);


inviteButton.setOnClickListener(new OnClickListener
  () {
  @Override
  public void onClick(View v) {

    Toast.makeText(getActivity(), "Eliminate
  duplicate Friends", Toast.LENGTH_SHORT).show();

    for (int i =0 ;i < invitedFriends.size()-1; i++
    ){

      String temp = invitedFriends.get(i).intern();

      for (int j =i+1 ;j < invitedFriends.size(); j
  ++ ){

        String temp1= invitedFriends.get(j).intern
  ();

        //la j parte dal basso??
```

```
        if(temp == invitedFriends.get(j).intern() ){

          invitedFriends.remove(j);

        }
        }
      }

      if (invitedFriends.size() > 0 && invitedFriends
    .size() < 4) {
        Toast.makeText(getActivity(), "Friends
    Invited", Toast.LENGTH_SHORT).show();
        //IMPLEMENT CLIENT-BASED OR SERVER-BASED
    INVITE
      }
      else {
        Toast.makeText(getActivity(), "Select a min
    of 1 player and a max of 4", Toast.LENGTH_LONG).
    show();
      }

      //Toast.makeText(getActivity(), "Friends
    Invited", Toast.LENGTH_SHORT).show();
    }
});
```

- Nella onConnected verrà inizializzato l'elemento mPersonListView che sarà la lista che poi verrà rappresentata nel layout e sarà riempita con gli utenti ritornati dalla loadConnected. Quest'ultima restituirà gli utenti connessi con la nostra stessa app. La mPersonListView sarà completata solo dopo l'esecuzione della onResult per effetto della Callback:

```
@Override
public void onConnected(Bundle connectionHint) {

  mPersonListView.setAdapter(mListAdapter);
    Plus.PeopleApi.loadConnected(mGoogleApiClient, null
  ).setResultCallback(this);

    btnSignIn.setVisibility(View.INVISIBLE);
    getFriends.setVisibility(View.INVISIBLE);
    inviteButton.setVisibility(View.VISIBLE);
```

```
}
```

- All'interno della onResult abbiamo il riempimento della lista degli amici tramite mListItems e la ListAdapter che verrà aggiornata:

```
@Override
public void onResult(LoadPeopleResult peopleData) {
switch (peopleData.getStatus().getStatusCode()) {
    case CommonStatusCodes.SUCCESS:
        mListItems.clear();
        PersonBuffer personBuffer = peopleData.
  getPersonBuffer();
        inviteButton.setVisibility(View.VISIBLE);
        try {
            int count = personBuffer.getCount();
            for (int i = 0; i < count; i++) {
                mListItems.add(personBuffer.get(i).
  getDisplayName());
            }
        } finally {
            personBuffer.close();
        }

        mListAdapter.notifyDataSetChanged();
        break;

    case CommonStatusCodes.SIGN_IN_REQUIRED:
        mGoogleApiClient.disconnect();
        mGoogleApiClient.connect();
        break;

    default:
        Log.e(TAG, "Error when listing people: " +
  peopleData.getStatus());
        break;
}
}
```

3.6 Gestione dell'accesso in AMUSE

Per quanto riguarda la gestione della parte server sono andato a modificare due file all'interno di AMUSE: la classe AuthenticationService e ho creato la

classe AccountDAOGooglePlus. Per quanto riguarda AuthenticationService, queste sono le modifiche che ho apportato:

```
...

* INIZIO MODIFICHE DI LUCA ALBERICI
  */

      else if (domain.compareTo("GOOGLE") == 0) {
        AccountDAOGooglePlus daoGooglePlus = new
   AccountDAOGooglePlus();

        //Check if the id coincide with the id associated
   at the access token
        if (daoGooglePlus.verifyAccount(username, password)
   ){
          HibernateUtils.beginTransaction();
          AccountDAO dao = new AccountDAOHibernate(
   HibernateUtils.getSession());
          AccountPojo acnt = dao.getAccount(username);
          if (acnt == null) {
            acnt = new AccountPojo();
            acnt.setUsername(username);
            //Come password inserisco una sottostringa dell
   'access token
            //perche' quest'ultimo e' troppo lungo da
   inserire intero.
            acnt.setPassword(password.substring(0, 30));
            dao.makePersistent(acnt);
            // Account creation OK
            myLogger.log(Logger.INFO, "
   AuthenticationService - NEW Account successfully created.
   Username = "+username);
            return username;
          }
          else {
            return username;
          }
        }
        else {
          throw new JADESecurityException("Invalid Google
   account");
        }
      }
```

```
            /*
  * FINE MODIFICHE DI LUCA ALBERICI
  */

    ...

    /*
  * INIZIO MODIFICHE DI LUCA ALBERICI
  */

      else if (domain.compareTo("GOOGLE") == 0) {
        AccountDAOGooglePlus daoGooglePlus = new
   AccountDAOGooglePlus();
        //Check if the id coincide with the id associated
   at the access token
        if (daoGooglePlus.verifyAccount(username, password)
   ){
          return username;
        }
        else {
          throw new JADESecurityException("Invalid␣
   GooglePlus␣account");
        }

      } else {
        throw new JADESecurityException("Username␣"+
   username+"␣does␣not␣contain␣a␣registered␣domain");
      }
  }
      /*
  * FINE MODIFICHE DI LUCA ALBERICI
  */

  ...
```

Mentre questa è la classe AccountDAOGooglePlus che ho creato:

```
/*
 *   FILE CREATO DA LUCA ALBERICI
 */

public class AccountDAOGooglePlus{
```

```
public AccountDAOGooglePlus() {
  // TODO Auto-generated constructor stub
}

public boolean verifyAccount(String username, String
accessToken) throws JADESecurityException {
  String id = username.replace("$GOOGLE", "");
  String urlS = "https://www.googleapis.com/auth/userinfo
.profile"+accessToken;
  try {
    URL url = new URL(urlS);
    HttpURLConnection con = (HttpURLConnection) url.
openConnection();
    con.setRequestMethod("GET");

    int responseCode = con.getResponseCode();
    if (responseCode != 200)
      throw new JADESecurityException("Connection problem
with GPlus Server");
    else{
      BufferedReader in = new BufferedReader(
              new InputStreamReader(con.getInputStream())
);
      StringBuffer response = new StringBuffer();
      String inputLine;
      while ((inputLine = in.readLine()) != null) {
        response.append(inputLine);
      }
      in.close();
      JSONObject object = new JSONObject(response.
toString());
      String idRequest = object.getString("id");
      if (id.compareTo(idRequest) == 0)
        return true;
    }
    } catch (MalformedURLException e) {
      e.printStackTrace();
    } catch (ProtocolException e) {
      e.printStackTrace();
    } catch (IOException e) {
      e.printStackTrace();
    } catch (JSONException e) {
      e.printStackTrace();
```

```
        }

      return false;
    }
}
```

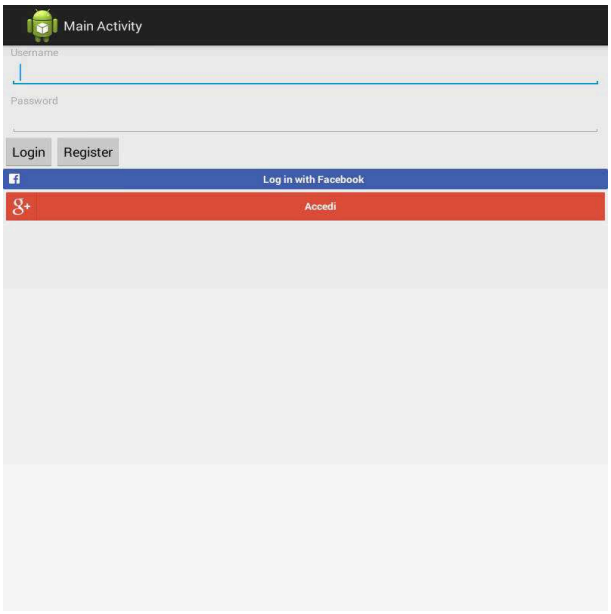


Figura 3.1: Pagina di avvio contenente i tre tipi di login

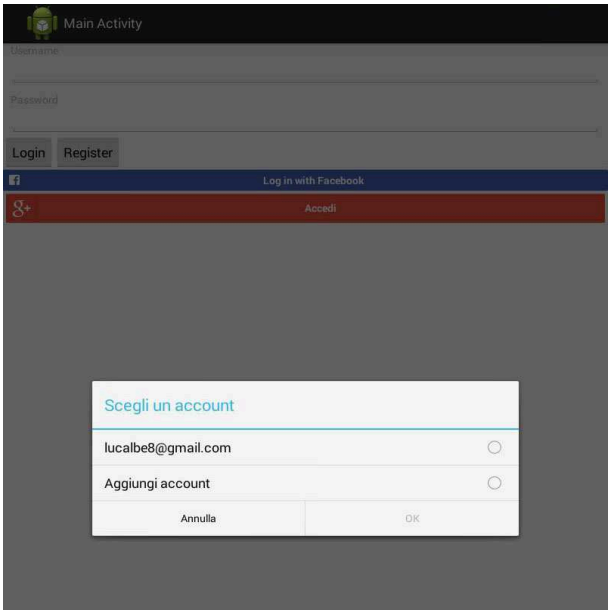


Figura 3.2: Selezione account con cui si vuole accedere a Google+

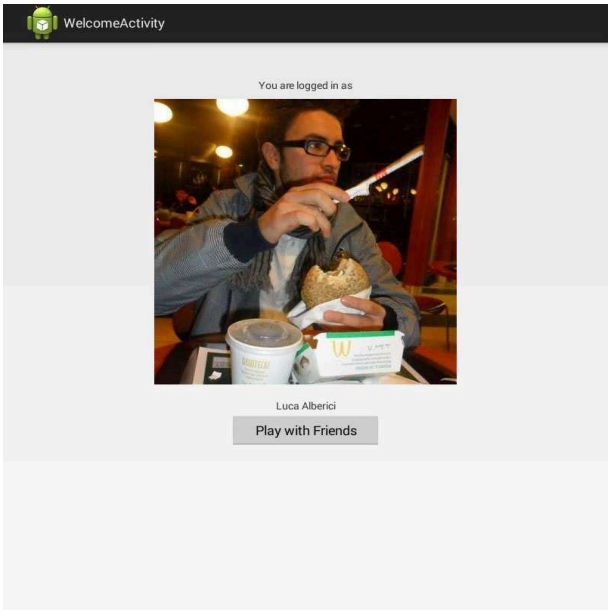


Figura 3.3: Pagina di welcome in cui vengono visualizzate la foto utente e il nome

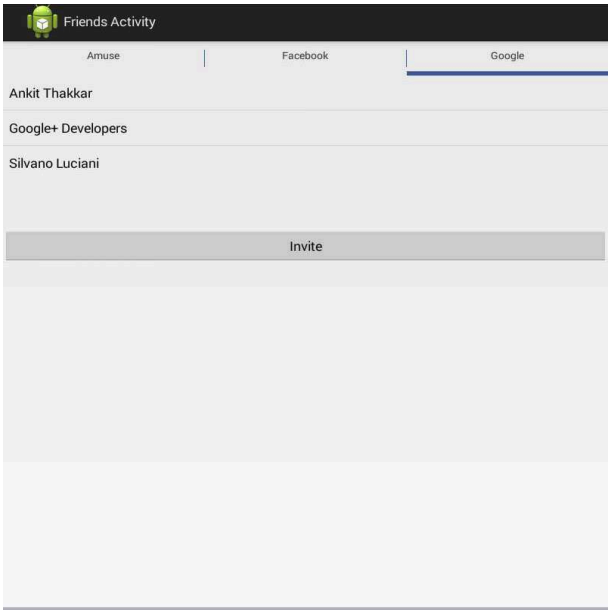


Figura 3.4: Visualizzazione e possibilità di invito a giocare degli amici

Capitolo 4

Conclusioni

In questa tesi, è stato affrontato il problema dell'integrazione del login tramite Google+ nella piattaforma AMUSE. Nel primo capitolo si è parlato a livello generale di tutti gli elementi utilizzati per poter realizzare il progetto. E' stata introdotta la piattaforma AMUSE descrivendola in modo generale, per poi approfondire sia la parte architetturale che quella funzionale, effettuando anche una panoramica sul Social Gaming. Si è poi passati alla descrizione del Social Network, dandone prima un'accezione generale, per poi parlare più dettagliatamente sia di Facebook che di Google+. Per quanto riguarda Google+, si è aggiunta anche una descrizione delle API che sono state utilizzate da interfaccia al suddetto Social. Nel secondo capitolo è stata fatta una panoramica dell'applicazione che mi è stata fornita e che sono andato a modificare. Per farlo è stato effettuato uno studio delle componenti più importanti dell'applicazione, andando ad analizzare le fasi in cui essa si articolava e le parti di codice più significative. Nel terzo capitolo sono state spiegate passo a passo tutte le modifiche apportate al codice già esistente e le varie integrazioni. Nella parte conclusiva si è descritta l'aggiunta della parte di codice modificata relativa ad AMUSE, in modo da permettere alla piattaforma di interfacciarsi con Google+. Le figure collocate alla fine del capitolo sono gli screenshot relativi alla mia applicazione durante un login tramite Google+.

Bibliografia

[1] Giovanni Caire
AMUSE Tutorial
`http://jade.tilab.com/amuseproject/amuse-tutorial/`

[2] Giovanni Caire
AMUSE Startup Guide
`http://jade.tilab.com/amuse/doc/Amuse-Startup-Guide.doc`

[3] Wikipedia
Servizio di rete sociale
`http://it.wikipedia.org/wiki/Servizio_di_rete_sociale`

[4] Google Developers
Sito internet dedicato agli sviluppatori Google
`https://developers.google.com/+/mobile/android/people`