Alma Mater Studiorum - Università di Bologna

CAMPUS DI CESENA

SCUOLA DI SCIENZE

CORSO DI LAUREA IN SCIENZE E TECNOLOGIE INFORMATICHE

APPLICAZIONE SMARTPHONE SOCIAL TIME MACHINE

RELAZIONE FINALE IN MOBILE WEB DESIGN

Relatore

Dott. Mirko Ravaioli

Presentata da

Kseniia Pelykh

Sessione III

Anno Accademico 2013-2014

Sommario

Introduzione....4
1.1 Dispositivi Mobile....5
1.2 Applicazioni Mobile....6
1.2.1 Applicazione nativa....7
1.2.2 Applicazione web....7
1.2.3 Mobile User Interface....7
1.3 Android....8
1.3.1 Il kernel Linux....8
1.3.2 Caratteristiche....10
1.4 Social Media....12
1.4.1 Facebook....13
1.5 Cloud Computing....14
1.5.1 BaaS (Backend as a Service)....18
Progettazione....20
2.1 Analisi delle funzionalità....20
2.2 Progettazione della base di dati....21
2.2.1 Utente....21
2.2.2 Gioco....21
2.2.3 Mossa....22
2.3 Progettazione dell'applicazione lato server....23
2.3.1 Integrazione con Facebook....23
2.3.2 Facebook Login....23
2.3.3 Invio di richieste....24
2.3.4 Recupero della lista degli amici del utente....24
2.3.5 Pubblicazione post....24
2.3.6 Integrazione con Parse....25
2.3.7 Notifiche Push....26
2.3.8 Salvataggio del gioco....26
2.3.9 Recupero della storia....26
2.3.10 Recupero dei giochi attivi....27

2.3.11 Recupero della galleria 27
2.4 Logica lato client....27
2.4.1 Salvataggio delle immagini nella cache....28
2.4.2 Salvataggio dell'account Facebook....28
2.4.3 Interfaccia utente....28
2.5 Diagramma dei casi d'uso 31
Implementazione....34
3.1 Sviluppo in Android....34
3.2 Struttura del progetto....34
3.2.1 Creazione di un nuovo progetto Android....35
3.2.2 Activity e risorse....36
3.2.3 Le risorse....39
3.2.4 AndroidManifest.xml....41
3.3 Material Design....43
3.3.1 Liste e Carte....44
3.3.2 View Shadows....45
3.3.3 Animazioni....45
3.4 Implementazione dell’applicazione....45
3.4.1 Integrazione Facebook SDK per Android....45
3.4.2 Facebook Login per Android....47
3.4.3 Inviare le richieste....49
3.4.4 Inizializzazione Parse SDK....51
3.4.5 Creazione di una nuova sfida....52
3.4.6 Invio delle notifiche Push....54
3.4.7 Recupero dei dati 55
Conclusioni 60
Bibliografia....62

Introduzione

I dispositivi mobile, al giorno d'oggi, rivestono un ruolo sempre più importante tanto nelle aziende quando nella vita privata, permettendo di compiere operazioni e svolgere dei compiti che, fino a qualche anno fa, erano eseguibili solo attraverso un normale PC.

Non è raro oggi incontrare, in un comune bar, un rappresentante che acquisisce gli ordini direttamente su un Pocket PC e li invia direttamente in azienda sfruttando una connessione GPRS o UMTS, riducendo i tempi di lavoro e, di conseguenza, aumentando potenzialmente il volume d'affari.

L'architettura dei dispositivi mobile si differenzia dall'architettura dei PC, per questo motivo è iniziata a diffondersi una nuova generazione di applicazioni: quelle per dispositivi mobile. Caratterizzate da una netta semplificazione rispetto a quelle per i tradizionali dispositivi informatici, queste nuove applicazioni vengono identificate semplicemente come app.
Il mercato della telefonia mobile tra cui smartphone e tablet continua ad evolversi con un ritmo rapido. L'individuale consumo mensile dei dati smartphone aumentata da 150 MB nel 2011 a 2,6 GB nel 2014. La creazione di applicazioni mobile è tra i settori più favorevoli sul mercato per gli sviluppatori IT. Il sistema operativo Android ha conquistato la maggior parte del mercato degli smartphone.
Android è il sistema operativo mobile più diffuso ed essendo open source lo sviluppo nel proprio ambiente è aperto a chiunque; per tal motivo dispone anche di diversi distributori digitali. Il distributore ufficiale è Google Play che nel 2012 ha festeggiato 25 miliardi di applicazioni scaricate in meno di 4 anni (è stato lanciato ufficialmente nel 2008).
Lo sviluppo di mobile app era inizialmente destinato esclusivamente alla produttività individuale e aziendale, project management, e-commerce, posta elettronica, calendario e banche dati. Successivamente, complice la crescente domanda pubblica dovuta alla rapida diffusione dei moderni dispositivi mobile, è stata registrata la rapida espansione in altre aree, come ad esempio giochi per cellulari, scienza applicata, automazione industriale, GPS e acquisti di biglietti.
Oggi esistono centinaia di migliaia di app: giochi e widget di varia natura, consultare riviste

e quotidiani online, ascoltare la radio, fotografare e modificare le foto con particolari effetti grafici, trovare indirizzi e ottenere indicazioni stradali, ricevere informazioni turistiche, prenotare e acquistare biglietti del treno e dell'aereo o direttamente alberghi, seguire ricette e corsi di varia natura, condividere e scambiare informazioni, foto con i propri amici con le app dei principali social network.
La rete sociali è una delle forme più evolute di comunicazione in rete. La rete delle relazioni sociali che ciascuno di noi tesse ogni giorno, in maniera più o meno casuale, nei vari ambiti della nostra vita, si può così "materializzare", organizzare in una "mappa" consultabile, e arricchire di nuovi contatti. Tra i social network più usati al mondo vi sono Facebook e Twitter.
Facebook integra alcune funzionalità sulle sue applicazioni per dispositivi mobile, come la possibilità di caricare contenuti, di ricevere e rispondere ai messaggi, di mandare e ricevere "poke" (cioè richiamare l'attenzione di un utente toccandolo simbolicamente sulla spalla), scrivere sulla bacheca degli utenti o semplicemente la possibilità di navigare sul sito.

1.1 Dispositivi Mobile

Con il passare del tempo, l'evoluzione tecnologica che ha accompagnato lo sviluppo dei normali PC, ha coinvolto i dispositivi così detti "mobili". Evoluzione che li ha trasformati da semplici organizer "da tasca" a veri e propri terminali ricchi di funzionalità, discreta potenza di calcolo ma soprattutto di connettività. Quest'ultima caratteristica li ha resi estremamente versatili soprattutto per applicazione di tipo aziendale e di produttività personale.

Esistono sostanzialmente quattro tipologie di dispositivi mobile, ognuna dotata di caratteristiche particolari da tenere in considerazione quando si progettano le applicazioni:

- pocket PC: sono i così detti "palmari". Tra le principali caratteristiche di questi dispositivi troviamo il display di tipo touch screen, la compattezza, una discreta possibilità di espansione e una discreta potenza di calcolo;
- smartphone: sono dispositivi prettamente telefonici (sia nella forma sia nella

modalità di input). Generalmente sono meno potenti dei Pocket PC. Data la tipologia e il target di questi dispositivi, anche la loro espandibilità attraverso device esterni è limitata;

- tablet PC: sono dispositivi molto simili ai normali Notebook ma dotati di touch screen e di funzionalità di riconoscimento della scrittura;
- UMPC (Ultra Mobile PC): sono una via di mezzo tra il Pocket PC e il Notebook. Più grandi e potenti di un Pocket PC ma più piccoli e meno potenti di un Notebook, sono più orientati verso il mercato consumer che business.

La scelta del device corretto in base alle esigenze specifiche è il primo passo per la realizzazione di un'applicazione funzionale. Tale scelta però, pone dei vincoli che si riflettono sulle applicazioni. Ad esempio, se il nostro obiettivo è di realizzare un'applicazione per la gestione dei dati in un magazzino, la scelta del device non può ricadere su un dispositivo di tipo Smarphone, date le sue caratteristiche sopraelencate. Viceversa, se il target device dell'applicazione è uno Smartphone, bisogna tenerne necessariamente conto nella progettazione delle applicazioni considerando, ad esempio, il display di dimensioni ridotte e la modalità di inserimento dei dati attraverso il tastierino numerico.

1.2 Applicazioni Mobile

Una app per dispositivi mobile si differenzia dalle tradizionali applicazioni sia per il supporto con cui viene usata sia per la concezione che racchiude in sé. Si tratta a tutti gli effetti di un software che per struttura informatica è molto simile a una generica applicazione ma è caratterizzata da una semplificazione ed eliminazione del superfluo, al fine di ottenere leggerezza, essenzialità e velocità. Il nome stesso, di per sé un'abbreviazione, può essere percepito come una semplificazione del nome completo "applicazione" per dare l'idea di un qualcosa di semplice e piccolo. Le app si suddividono in app native e web app, con i casi intermedi o misti che vengono talvolta definiti app ibride.

1.2.1 Applicazione nativa

Consiste in uno strumento informatico che si installa e si utilizza sul proprio dispositivo mobile, vale a dire un insieme di istruzioni informatiche progettate con lo scopo di rendere possibile un servizio o una serie di servizi o strumenti ritenuti utili o desiderabili dall'utente, creata appositamente per uno specifico sistema operativo. L'interazione diretta con le API messe a disposizione dal costruttore del sistema operativo garantirà accesso immediato a tutte le funzionalità del dispositivo oltre a permettere prestazioni ottimali e migliorare sensibilmente l'usabilità. Le app, infatti, vanno ad ampliare le capacità native del dispositivo incluse all'interno del sistema operativo (configurazione di base). Una volta acquistato il dispositivo, sia esso smartphone o tablet, si ha la possibilità di personalizzarlo aggiungendo nuove applicazioni a seconda dei propri gusti ed esigenze.

1.2.2 Applicazione web

Mentre una mobile app è installata fisicamente sul dispositivo dell'utente, una web app è sostanzialmente un collegamento verso un applicativo remoto, scritto in un linguaggio cross-platform come HTML5. Questa soluzione comporta delle importanti conseguenze in termini di funzionamento: il vantaggio principale di una web app consiste nel fatto di non incidere in alcun modo sulle capacità di memoria del dispositivo e sulle sue capacità di calcolo dei dati. Tuttavia, per funzionare, una web app richiede il costante accesso a internet e le sue prestazioni dipendono in modo sensibile dalla velocità della connessione.

1.2.3 Mobile User Interface

Come parte del processo di sviluppo, Mobile User Interface (UI) Design è un altro elemento essenziale per la creazione di applicazioni mobile. Mobile UI considera vincoli e contesti, schermo ed input come gli oggetti principali per progettazione di applicazione mobile. L'utente è al centro di un interazione con il proprio dispositivo. L'input consente agli utenti di manipolare un sistema e l'output del dispositivo permette al sistema di indicare gli effetti della manipolazione degli utenti. I vincoli di progettazione dell'interfaccia utente mobile devono considerare i fattori limitati, come la dimensione dello schermo di un

dispositivo mobile per la mano. Mobile UI indica i segnali di attività dell'utente, come la posizione e la gestione, che possono essere visualizzati dalle interazioni all'interno di un'applicazione mobile. In generale, l'obiettivo di progettazione dell'interfaccia è creare una interfaccia user-friendly e comprensibile. L'interfaccia utente di applicazioni mobili dovrebbe considerare la scarsa attenzione degli utenti, ridurre al minimo le battiture, ed essere task-oriented con un set minimo di funzioni. Questa funzionalità è supportata da piattaforme di applicazioni enterprise mobili o ambienti di sviluppo integrati (IDE).

1.3 Android

Come è già stato detto, android è un sistema operativo per dispositivi mobile (OS) sviluppato da Google Inc. sulla base del kernel Linux. Android è stato progettato principalmente per smartphone e tablet, con interfacce utente specializzate per televisori (Android TV), automobili (Android Auto), orologi da polso (Android Wear), occhiali (Google Glass), e altri.
Android è quasi totalmente Free e Open Source (ad esclusione per esempio dei driver non-liberi inclusi per i produttori di dispositivi) ed è distribuito sotto i termini della licenza Apache 2.0, il quale consente di modificare e distribuire liberamente il codice sorgente.
Inoltre, Android dispone di una vasta comunità di sviluppatori che realizzano applicazioni con l'obiettivo di aumentare le funzionalità dei dispositivi. Queste applicazioni sono scritte soprattutto in linguaggio di programmazione Java.
Nell'ottobre 2012 le applicazioni disponibili presenti sul market ufficiale Android (Google Play) hanno raggiunto le 700.000 unità. Questi fattori hanno permesso ad Android di diventare il sistema operativo più utilizzato in ambito mobile, oltre a rappresentare per le aziende produttrici la migliore scelta in termini di bassi costi, personalizzazione e leggerezza del sistema operativo stesso, senza dover scrivere un proprio sistema operativo da zero.

1.3.1 Il kernel Linux

Android è costituito da un Kernel basato sul kernel Linux 2.6 e 3.x (da Android 4.0 in poi),

con middleware, Librerie e API scritte in C(o C++) e software in esecuzione su un framework di applicazioni che include librerie Java compatibili con librerie basate su Apache Harmony. Android utilizza la Dalvik virtual machine con un compilatore just-in-time per l'esecuzione di Dalvik dex-code (Dalvik Executable), che di solito viene tradotto da codice bytecode Java. La piattaforma hardware principale di Android è l'architettura ARM. L'architettura x86 è supportata grazie al progetto Android x86 e Google TV utilizza una speciale versione x86 di Android.

Il kernel Linux di Android mette a disposizione modifiche all'architettura create da Google al di fuori del ciclo di sviluppo del kernel. Un tipico sistema Android non possiede un X Window System nativo, né supporta il set completo standard di librerie GNU, e nel caso del C++ vi è solo una implementazione parziale delle STL. Tutto ciò rende difficile il porting di applicazioni Linux o librerie per Android. Per semplificare lo sviluppo di applicazioni C/C++ sotto Android si usa SDL che tramite un piccolo Wrapper java permette l'utilizzo della JNIdando un'idea di utilizzo simile a un'applicazione nativa in C/C++.

Le applicazioni Android sono Java-based; in effetti le applicazioni scritte in codice nativo in C/C++ devono essere richiamate dal codice java, tutte le chiamate a sistema fatte in C (o C++) devono chiamare codice virtual machine Java di Android: infatti l'API multimediale SDL sotto Android richiama metodi di Java, questo significa che il codice dell'applicazione C/C++ deve essere inserito all'interno di un progetto Java, il quale produce alla fine un pacchetto Android (APK).

Alcune funzionalità implementate da Google hanno contribuito al kernel Linux, in particolare una funzione di gestione dell'energia denominata wakelocks, che però è stata respinta dagli sviluppatori del kernel mainline, in parte perché gli sviluppatori del kernel hanno ritenuto che Google non manifestasse alcuna intenzione di mantenere il proprio codice. Anche se Google ha annunciato nel mese di aprile 2010 che avrebbero assunto due dipendenti per lavorare con la comunità del kernel Linux, Greg Kroah-Hartman, l'attuale responsabile del kernel di Linux per il ramo stabile, ha dichiarato nel dicembre 2010 che era preoccupato del fatto che Google non sembrava più intenzionata a far includere le modifiche al codice nel ramo principale di Linux. Alcuni sviluppatori di Google Android hanno fatto capire che "il team di Android si è stancato del processo", perché erano una

piccola squadra e avevano molto lavoro urgente da fare su Android

La memoria flash sui dispositivi Android è divisa in diverse partizioni, ad esempio "/system" per il sistema operativo stesso e "/data" per i dati utente e le installazioni delle app. Diversamente, rispetto alle tradizionali distribuzioni GNU/Linux, agli utenti di dispositivi Android non sono disponibili i privilegi di superutente, o root, per l'accesso al sistema operativo e alle sue partizioni, quali "/system" (per le quali l'utente dispone dei permessi di sola lettura). Tuttavia, l'accesso root del dispositivo è quasi sempre possibile: in certi casi tramite richiesta al produttore, in altri sfruttando certe falle di sicurezza di Android. L'accesso root viene utilizzato frequentemente dalla comunità open source, per migliorare le capacità dei loro dispositivi, ma anche da soggetti malintenzionati per installare virus e malware.

1.3.2 Caratteristiche

L'interfaccia utente di Android è basata sul concetto di direct manipulation per cui si utilizzano gli ingressi mono e multi-touch come strisciate, tocchi e pizzichi sullo schermo per manipolare gli oggetti visibili sullo stesso. La risposta all'input dell'utente è stata progettata per essere immediata e tentare di fornire un'interfaccia fluida. Sensori hardware interno come accelerometri, giroscopi e sensori di prossimità sono utilizzati da alcune applicazioni per rispondere alle azioni da parte dell'utente, ad esempio la regolazione dello schermo da verticale a orizzontale a seconda di come il dispositivo è orientato o che consentono all'utente di guidare un veicolo in una corsa virtuale ruotando il dispositivo, simulando il controllo di un volante.
La cosiddetta Homescreen è simile al desktop trovato su Windows ed è la schermata principale che ci si trova appena il device è stato avviato, oppure premendo il tasto Home. L'homescreen di Android è in genere occupata sia dalle icone delle applicazioni che dai widget, da una sorta di gadgets con varie funzioni; ci sono widget che mostrano vari stili di orologi, quelli che mostrano gli ultimi video di YouTube, altri che visualizzano informazioni meteo, quelli relativi alle email. L'homescreen può essere costituita da più pagine tra cui l'utente può scorrere avanti e indietro.
Sempre presente nella parte superiore dello schermo si trova una barra di stato, che mostra

le informazioni sul dispositivo e la sua connettività. Trascinando la barra di stato verso il basso compare una schermata di notifica in cui le applicazioni possono visualizzare notifiche relative ad informazioni importanti o aggiornamenti come ad esempio una e-mail appena ricevuta o un SMS, in modo da non interrompere immediatamente l'utente. Nelle prime versioni di Android tali notifiche potevano essere sfruttate esclusivamente per aprire l'applicazione in questione, ma gli aggiornamenti più recenti hanno fornito maggiori funzionalità, come ad esempio la possibilità di chiamare un numero direttamente dalla notifica della chiamata persa, senza dover aprire l'applicazione telefono. Le notifiche sono persistenti fino alla loro lettura o cancellazione da parte dell'utente.

La piattaforma usa il database SQLite, la libreria dedicata SGL per la grafica bidimensionale (invece del classico server X delle altre distribuzioni linux) e supporta lo standard OpenGL ES 2.0 per la grafica tridimensionale. Le applicazioni vengono eseguite tramite la Dalvik virtual machine, una macchina virtuale adattata per l'uso su dispositivi mobile.

Android è stato progettato principalmente per smartphone e tablet, ma il carattere aperto e personalizzabile del sistema operativo ne permette l'utilizzato anche su altri dispositivi elettronici tra cui portatili, netbook, smartbook, eBook reader, fotocamere e smart TV (Google TV). Il mercato delle "smart things" è cresciuto in maniera notevole in questi ultimi periodi a tal punto da stimolare la creatività delle persone. Un esempio è lo Smartwatch dotato di sistema operativo Android in versione "light", lettori auto CD e DVD, occhiali intelligenti (Google Glass), frigoriferi, sistemi di navigazione satellitare per veicoli, sistemi di automazione per la casa, console di gioco, specchi, telecamere, lettori MP3/MP4 e tapis roulant.

Il logo di Android è stato progettato con la famiglia di caratteri (font) Droid di Ascender Corporation, il verde è il colore del robot che rappresenta il sistema operativo Android. Il colore di stampa è PMS 376C e il colore RGB in valore esadecimale è # A4C639, come specificato dalle linee guida del Marchio di Android. Il carattere personalizzato di Android si chiama Norad. Viene utilizzato solo nel logo di testo.

1.4 Social Media

I social media rappresentano fondamentalmente un cambiamento nel modo in cui la gente apprende, legge e condivide informazioni e contenuti. In essi si verifica una fusione tra sociologia e tecnologia che trasforma il monologo in dialogo e ha luogo una democratizzazione dell'informazione che trasforma le persone da fruitori di contenuti ad editori. Sono diventati molto popolari perché permettono alle persone di utilizzare il web per stabilire relazioni di tipo personale o lavorativo. I social media vengono definiti anche user-generated content o consumer-generated media.

I social media sono diversi dai media industriali come giornali, televisione e cinema. Mentre i social media sono strumenti relativamente a basso costo che permettono a chiunque (anche soggetti privati) di pubblicare ed avere accesso alle informazioni, i media tradizionali richiedono cospicui investimenti finanziari per pubblicare informazioni. Tra le voci di spesa dei media industriali possiamo annoverare ad esempio la pressa tipografica o le autorizzazioni statali. I media industriali sono comunemente definiti "tradizionali", "broadcasting", o "mass media". Una caratteristica che accomuna social media e media industriali è la capacità di ottenere un'audience sia vasta che ridotta; sia il post di un blog che una trasmissione televisiva possono raggiungere milioni di persone oppure nessuno. I parametri che aiutano a descrivere le differenze tra i due tipi di media variano a seconda del tipo di analisi. Alcuni di questi parametri sono:

- bacino d'utenza: sia i social media che i media industriali offrono a ciascuno l'opportunità di ottenere un'audience globale;
- accessibilità: i mezzi di produzione dei media industriali sono generalmente di proprietà privata o statale; gli strumenti dei social media sono disponibili da ciascuno a un costo contenuto o gratuitamente;
- fruibilità: la produzione di mezzi industriali richiede in genere formazione e competenze specialistiche; i social media invece no, o in qualche caso reinventano le competenze, cosicché ciascuno può gestire i mezzi di produzione;
- velocità: il tempo che intercorre tra le informazioni prodotte dai media industriali può essere lungo (giorni, settimane o anche mesi) in confronto al

tempo impiegato dai social media (che hanno la possibilità tecnica di reagire istantaneamente, solo la mancanza di reattività dei partecipanti può comportare ritardi). Poiché ormai anche i media industriali si avvalgono degli strumenti dei social media, questo potrebbe non essere più un tratto distintivo;

- permanenza: una volta creati, i mezzi industriali non possono essere più modificati (una volta stampato e distribuito, l'articolo di una rivista non può più ricevere modifiche), mentre i social media possono essere cambiati quasi istantaneamente mediante commenti e modifiche;

Un'ulteriore distinzione riguarda la responsabilità. I media industriali sono tenuti a rendere conto alla società della qualità dei contenuti e dei risultati delle loro attività in termini di interesse pubblico, responsabilità sociale ed indipendenza editoriale. I social media non hanno altrettante responsabilità in merito alle loro attività editoriali. Da un lato i social media possono sembrare abbastanza liberi da conflitti di interessi, ma d'altro canto il loro valore economico può essere minacciato da fenomeni in ascesa come Public Relations 2.0, network pubblicitari e pubblicità conto terzi.
La comunità dei media è un ibrido interessante. Anche se community-owned, ossia posseduti dalla comunità, alcuni media si avvalgono di professionisti, ed altri di dilettanti. Essi utilizzano sia l'ambito dei social media che quello dei mezzi tradizionali. Nel 2006, Yochai Benkler ha analizzato molte di queste differenze e le loro implicazioni in termini di libertà economica e politica. Benkler, come molti accademici, usa il neologismo network economy o "network information economy" per descrivere le più rilevanti caratteristiche economiche, tecnologiche e sociali di quelli che chiamiamo "social media".

1.4.1 Facebook

Facebook è un servizio di rete sociale lanciato nel febbraio del 2004, posseduto e gestito dalla corporation Facebook, Inc. 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 conta 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 vengono 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 multimedialied 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 (esempio: Città natale*:* Roma) e quello di residenza (esempio: Vive a Torino), la scuola frequentata, il proprio datore di lavoro, l'orientamento religioso e quello politico, la propria situazione sentimentale e molte altre.

1.5 Cloud Computing

In informatica con il termine inglese cloud computing si indica un insieme di tecnologie che permettono, tipicamente sotto forma di un servizio offerto da un provider al cliente, di

memorizzare, archiviare e elaborare dati (tramite CPU o software) grazie all'utilizzo di risorse hardware or software distribuite e virtualizzate in Rete in un'architettura tipica client-server.
È noto come, utilizzando varie tipologie di unità di elaborazione (CPU), memorie di massa fisse o mobili come RAM, dischi rigidi interni o esterni, CD/DVD, chiavi USB eccetera, un computer sia in grado di elaborare, archiviare, recuperare programmi e dati.
Nel caso di computer collegati in rete locale (LAN) o geografica (WAN) la possibilità di elaborazione, archiviazione e recupero può essere estesa ad altri computer e dispositivi remoti dislocati sulla rete stessa.
Sfruttando la tecnologia del cloud computing gli utenti collegati ad un cloud provider possono svolgere tutte queste mansioni, anche tramite un semplice internet browser. Ad esempio, utilizzare software remoti non direttamente installati sul proprio computer e salvare dati su memorie di massa online predisposte dal provider stesso .
Nonostante il termine sia piuttosto vago e sembri essere utilizzato in diversi contesti con significati differenti tra loro, si possono distinguere tre tipologie fondamentali di servizi cloud computing:

- SaaS (Software as a Service) - Consiste nell'utilizzo di programmi installati su un server remoto, cioè fuori dal computer fisico o dalla LAN locale, spesso attraverso un server web. Questo acronimo condivide in parte la filosofia di un termine oggi in disuso, ASP (Application service provider).
- DaaS (Data as a Service) - Con questo servizio vengono messi a disposizione via web solamente i dati ai quali gli utenti possono accedere tramite qualsiasi applicazione come se fossero residenti su un disco locale.
- HaaS (Hardware as a Service) - Con questo servizio l'utente invia dati a un computer che vengono elaborati da computer messi a disposizione e restituiti all'utente iniziale.

Nel caso di funzionalità di memorizzazione in remoto la creazione di una copia di sicurezza (backup) è automatica e l'operatività si trasferisce tutta online mentre i dati sono memorizzati in server farm generalmente localizzate nei Paesi di origine del service provider.

Il cloud computing rende disponibili all'utilizzatore le risorse come se fossero implementate da sistemi (server o periferiche personali) "standard". L'implementazione effettiva delle risorse non è definita in modo dettagliato; anzi l'idea è proprio che l'implementazione sia un insieme eterogeneo e distribuito – the cloud, in inglese nuvola – di risorse le cui caratteristiche non siano note all'utilizzatore.
L'architettura informatica del cloud computing prevede uno o più server reali, generalmente in architettura ad alta affidabilità e fisicamente collocati presso il data center del fornitore del servizio.
I sistemi di cloud computing vengono criticati principalmente per l'esposizione degli utenti a rischi legati a:
1) Sicurezza informatica e privacy degli utenti:

- Utilizzare un servizio di cloud computing per memorizzare dati personali o sensibili, espone l'utente a potenziali problemi di violazione della privacy. I dati personali vengono memorizzati nelle Server Farms di aziende che spesso risiedono in uno stato diverso da quello dell'utente. Il cloud provider, in caso di comportamento scorretto o malevolo, potrebbe accedere ai dati personali per eseguire ricerche di mercato e profilazione degli utenti.
- Con i collegamenti wireless, il rischio sicurezza aumenta e si è maggiormente esposti ai casi di pirateria informatica a causa della minore sicurezza offerta dalle reti senza fili. In presenza di atti illegali, come appropriazione indebita o illegale di dati personali, il danno potrebbe essere molto grave per l'utente, con difficoltà di raggiungere soluzioni giuridiche e/o rimborsi se il fornitore risiede in uno stato diverso da paese dell'utente.
- Nel caso di industrie o aziende, tutti i dati memorizzati nelle memorie esterne sono seriamente esposti a eventuali casi di spionaggio industriale.

2) Problemi internazionali di tipo economico e politico:

- Possono verificarsi quando dati pubblici sono raccolti e conservati in archivi privati, situati in un paese diverso da quelli degli utenti della "nuvola". Produzioni cruciali e di carattere intellettuale insieme a una grande quantità di informazioni personali sono memorizzate in forma di dati digitali in archivi privati centralizzati e

parzialmente accessibili. Nessuna garanzia viene data agli utenti per un libero accesso futuro.

- Altre problematiche sono legate alla localizzazione degli archivi della "nuvola" in alcuni paesi ricchi. Se non regolato da specifiche norme internazionali ciò potrebbe:
  1. aumentare il "digital_divide" tra paesi ricchi e poveri (se l'accesso alle conoscenze memorizzate non sarà liberamente garantita a tutti);
  2. favorire principalmente grandi corporation con «organismi policentrici" e "menti monocentriche" dislocate principalmente nei Paesi della "nuvola", essendo la proprietà immateriale considerata come un fattore strategico per le moderne economie "knowledge-based".

Maggiori sicurezze e garanzie vi sono nel caso in cui il fornitore del servizio appartenga alla stessa nazione/area applicando le medesime leggi/normative sulla privacy e sicurezza del cliente (la legislazione USA o di altre nazioni è molto diversa dall'italiana e diventa impossibile pensare di soddisfare normative nazionali con servizi in cloud di altre nazioni).

3) Continuità del servizio offerto:

- Delegando a un servizio esterno la gestione dei dati e la loro elaborazione l'utente si trova fortemente limitato nel caso in cui i suddetti servizi non siano operativi (out of service). Un eventuale malfunzionamento inoltre colpirebbe un numero molto elevato di persone contemporaneamente dato che questi sono servizi condivisi. Anche se i migliori servizi di cloud computing utilizzano architetture ridondante e personale qualificato al fine di evitare malfunzionamenti dei sistema e ridurre la probabilità di guasti visibili dall'utente finale, non eliminano del tutto il problema. Bisogna anche considerare che tutto si basa sulla possibilità di avere una connessione Internet ad alta velocità sia in download che in upload e che anche nel caso di una interruzione della connessione dovuta al proprio Internet Service Provider/ISP si ha la completa paralisi delle attività.

4) Difficoltà di migrazione dei dati nel caso di un eventuale cambio del gestore dei servizi cloud:

- Non esistendo uno standard definito tra i gestori dei servizi, un eventuale cambio di operatore risulta estremamente complesso. Tutto ciò risulterebbe estremamente

dannoso in caso di fallimento del gestore dei servizi cui ci si è affidati.

1.5.1 BaaS (Backend as a Service)

Backend come un servizio (in inglese "backend as a service" o in sigla BaaS), noto anche come "mobile backend as a service" (MBaaS), è un modello per fornire a sviluppatori di app web o mobile un modo per collegare le loro applicazioni a un backend cloud storage e API (interfacce di programmazione di un'applicazione) esposte da applicazioni backend, fornendo allo stesso tempo funzioni quali la gestione degli utenti, le notifiche push, e l'integrazione con servizi di social networking.
Questi servizi sono forniti attraverso l'uso di kit di sviluppo software personalizzato (SDK) e interfacce di programmazione delle applicazioni (APIs).
Baas è uno sviluppo relativamente recente nel cloud computing, con la maggior parte delle startup Baas risalente dal 2011 in poi. Anche se è un settore piuttosto nascente, le tendenze indicano che questi servizi stanno guadagnando rilevanza e popolarità con i consumatori aziendali. Il mercato globale Baas nel 2012 aveva un valore stimato di 216,5 milioni dollari e una valore previsto entro il 2017 fino a 7,7 miliardi dollari. Alcuni esempi di fornitori di backend di consumo sono Parse (acquisita da Facebook), Apigee, Backendless, Kii, built.io e Firebase. Fornitori di backend aziendali includono Encore.io, Appear Networks, Appcelerator, AnyPresence, Kidozen, Kinvey, e FeedHenry. Syncano è un esempio di un provider di backend che funziona sia con consumatori e clienti aziendali.
Applicazioni web e mobile richiedono un analogo insieme di funzionalità sul backend, comprese le notifiche push, l'integrazione con i social network e il cloud storage. Ognuno di questi servizi ha la propria API che deve essere incorporate singolarmente in un'app, un processo complicato che può richiedere molto tempo per gli sviluppatori di applicazioni. I provider di BaaS formano un ponte tra il frontend di un'applicazione e vari cloud-based backends tramite una API unificata e SDK (pacchetto di sviluppo per applicazioni).

Fornire un metodo costante e coerente per gestire i dati di backend significa che gli sviluppatori non hanno bisogno di sviluppare più volte il proprio backend per ciascuno dei servizi che le loro applicazioni hanno bisogno di accedere, potenzialmente risparmiando tempo e denaro.

Anche se è simile ad altri strumenti di sviluppo di cloud computing, come il ‘software as a service’ (SaaS),’ Infrastructure as a Service’ (IaaS) e ‘Platform as a Service’ (PaaS), Baas si distingue da questi altri servizi poiché riguarda in particolare le esigenze del cloud-computing di sviluppatori di applicazioni web e mobile, fornendo uno strumento unificato per collegare le loro applicazioni ai servizi cloud.\

Ogni provider Baas offre una serie leggermente diversa di strumenti di backend e di risorse. Tra i servizi più comuni offerti ci sono le notifiche push, l'archiviazione e la condivisione di file, l'integrazione con le reti sociali come Facebook e Twitter, servizi di localizzazione, funzioni di messaggistica e di chat, la gestione degli utenti, esecuzione di logica di business e strumenti di analisi di utilizzo. I Fornitori Baas hanno un centro di attenzione molto ampio, fornendo SDK e le APIs che lavorano per lo sviluppo di applicazioni su più piattaforme, come iOS, Android, Blackberry, Windows Phone, HTML5, e altri.

Progettazione

SocialTimeMachine è un'applicazione mobile che permette agli utilizzatori di condividere i propri interessi e passioni con gli amici, sfidandoli e convincendoli a partecipare.
L'obiettivo fondamentale di questo progetto è quello di creare un'applicazione social, ovvero un'applicazione che si concentra sulla creazione e il mantenimento di un collegamento tra gli utenti. Sfruttando la popolarità dei social network, si dà la possibilità agli utenti di partecipare in un gioco sociale. Nello specifico, l'applicazione si integra con Facebook, il social network che, ad oggi, vanta il maggior numero di utenti attivi.
Prima di procedere con lo svilluppo dell'applicazione è necessario effettuare un'accurata analisi delle funzionalità richieste che l'app deve fornire.

2.1 Analisi delle funzionalità

Come è già stato detto prima, l'applicazione è una specie di gioco sociale. Per entrare a far parte occore effettuare login tramite il proprio account di Facebook. Per iniziare la partita bisogna inserire il titolo, la data inizio e data fine, la foto, descrizione e specificare i partecipanti. Il titolo rappresenta il tema del gioco, è deve far capire agli utenti in che cosa consiste la partita. Una volta creato il gioco, gli utenti devono ricevere una notifica, e se anche la data di inizio è già passata, ma non quella di fine, un partecipante può fare la sua mossa. La mossa contiene una foto riguardo il tema del gioco e una breve descrizione. Alla fine gli utenti possono effettuare una votazione.
Analizzando gli obiettivi principali dell'applicazione si possono specificare le seguenti funzionalità:

- possibilità di effettuare login tramite Facebook;
- possibilità di creare una nuova sfida, specificando il titolo, la data d'inizio e di fine;
- possibilità di invitare gli amici a partecipare nelle sfide;

- possibilità di ricevere le notifiche quando un utente viene invitato ad una sfida;
- possibilità di vedere la storia delle pubblicazioni;
- possibilità di vedere tutte le sfide attive;
- possibilità di accettare o rifiutare la partecipazione nel gioco;
- possibilità di inserire la propria mossa, mettendo la foto e la descrizione;
- possibilità di vedere tutte le sfide precedenti;
- possibilità di pubblicare su Facebook;
- possibilità di votare;

2.2 Progettazione della base di dati

Per realizzare il DB è necessario avere una chiara definizione delle informazioni gestite dall'applicazione, quindi capire i concetti fondamentali da tenere in considerazione nella fase di progettazione.

2.2.1 Utente

Anche se viene usato Facebook per fornire i servizi di Login, bisogna comunque salvare le informazioni riguardo l'utente sul DB per migliorare le prestazioni. L'utente dovrebbe avere la possibilità di vedere la storia delle pubblicazioni con l'elenco dei nomi e delle foto. Ogni volta fare una richiesta (HTTP) a Facebook sarebbe troppo costoso. A questo punto, viene introdotta l'entità utente che identifica univocamente l'utente e memorizza i dati che vengono usati più spesso nell'applicazione, ovvero il nome e la foto del profilo.

2.2.2 Gioco

Le informazioni che riguardano il gioco sono: il titolo, che rappresenta in modo chiaro qual'è la tematica del gioco; la data d'inizio per specificare quando la sfida inizia e gli

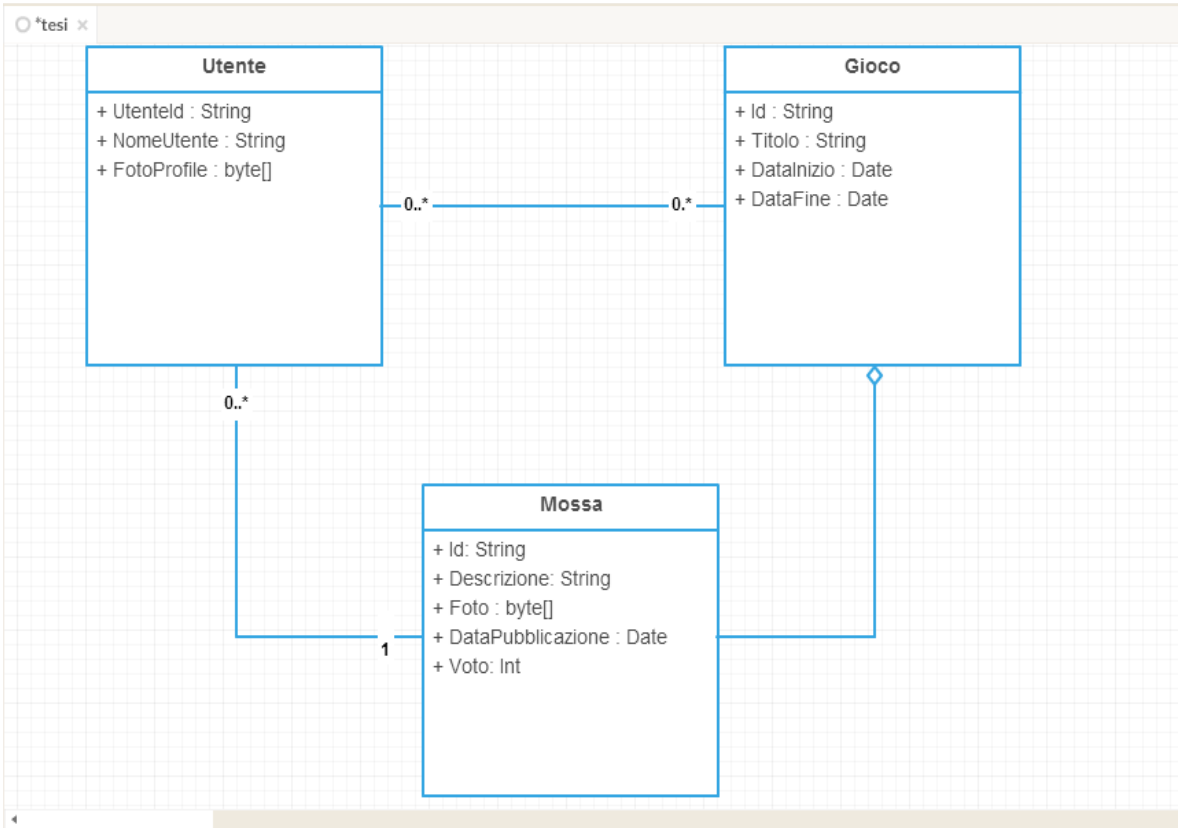
utenti possono iniziare a fare le pubblicazioni; la data di fine, che chiude il gioco e vieta ai partecipanti di fare le pubblicazioni. Il gioco deve essere anche associato alla lista degli utenti, che possono partecipare. Una volta che il giocatore rifiuta la partecipazione, viene cancellato dalla lista.

2.2.3 Mossa

Una volta inserito nel gioco, utente riceverà una notifica push e se il gioco è stato aperto, l'utente può pubblicare la sua mossa. Il partecipante deve inserire la foto, dove fa vedere di aver completato il compito e scrivere una breve descrizione, che possono vedere gli altri utenti. Una volta che il gioco è stato chiuso gli utenti possono votare la mossa che gli è piaciuta di più.

La seguente figura mostra il database è costituito da tre tabelle: Utente, Gioco, Mossa

Figura 1- Schema database



Si vede dal diagramma delle classi che un gioco contiene un elenco degli utenti, ognuno di quali può fare solo una mossa all'interno del gioco.

2.3 Progettazione dell'applicazione lato server.

Per quello che riguarda la logica lato server, gli aspetti principali da considerare sono: l'azione di login ed il salvataggio dei dati riguardanti l'utente; il salvataggio delle mosse (il download delle foto e le descrizioni), le altre informazioni riguardanti il gioco e l' integrazione con Facebook.

2.3.1 Integrazione con Facebook

Al fine di integrare l' applicazione mobile con il social network è necessario creare un' applicazione, che si interfacci con la piattaforma di Facebook. A tal scopo, Facebook mette a disposizione un pannello di controllo per sviluppatori nel quale è possibile creare un' applicazione e configurare alcuni parametri come il server sul quale risiede ed il namespace con il quale può essere riconosciuta. In particolare, l' applicazione Facebook può essere caricata su un server gratuito. Il collegamento tra applicazione mobile e applicazione di Facebook avviene tramite una chiave che si chiama "Key Hash", generata tramite una procedura specifica. Bisogna aggiungere la chiave al profilo di sviluppatore che viene utilizzata per autenticare le interazioni tra l'applicazione e Facebook.
Per la versione di rilascio è necessario creare e impostare il Key Hash di rilascio.
Una volta generata la chiave, l'applicazione mobile è in grado di accedere alle funzionalità messe a disposizione dal social network.

2.3.2 Facebook Login

Facebook fornisce metodi per implementare Facebook Login, il quale permette di ottenere un token per l'accesso API di Facebook per l'utente che utilizza l'applicazione. È possibile

utilizzare questa funzione per costruire il proprio sistema di account o per aggiungere servizi di Facebook per gli account esistenti. SocialTimeMachine utilizza solo Facebook Login per autorizzare le persone che utilizzano l'applicazione.

La prima volta Facebook Login avviene grazie una schermata apposita che richiede anche i permessi necessari per poter utilizzare le funzioni dell'app. Lo stesso accadrà in caso si richiedano i permessi aggiuntivi. Ci sono due tipi di schermate di accesso differenti che possono essere mostrate all'utente.

I permessi richiesti all'utente sono: accesso alle informazioni base del profilo (id profilo, nome, cognome, foto profilo), accesso alla lista degli amici, pubblicazione dei post.

Dopo aver effettuato il login ed aver ricevuto i permessi necessari è possibile utilizzare le funzioni citate in precedenza.

2.3.3 Invio di richieste

Attraverso un'opportuna finestra di dialogo, denominata "Request Dialog", è possibile invitare i propri amici ad utilizzare l'applicazione. A tal fine, Facebook inoltra una notifica all'account selezionato nella Request Dialog. Questa funzionalità viene sfruttata con lo scopo di diffondere l'applicazione nel social network e quindi aumentarne la popolarità

2.3.4 Recupero della lista degli amici del utente

Per poter invitare un amico a partecipare al gioco è necessario ricavare la lista degli amici. Questo rende l'applicazione più interessante e più personale. Ultima versione di Facebook non permette di visualizzare nella lista un amico che non ha ancora installato l'applicazione.

2.3.5 Pubblicazione post

Le funzionalità che permettono di commentare una mossa e di consigliare agli amici di partecipare avvengono tramite la pubblicazione dei post. In questo contesto, un post è

caratterizzato dall'immagine e dalla descrizione della mossa. Per effettuare questa pubblicazione bisogna prima di tutto configurare Open Graph nella App Dashboard. Open Graph permette alle applicazioni di raccontare le storie attraverso l'API fortemente tipizzata.

Le storie hanno i seguenti componenti principali :

- un attore, la persona che pubblica la storia;
- un'azione che l'attore esegue, per esempio: cuocere, correre o leggere;
- un oggetto su cui viene eseguita l'azione: cucinare un pasto, una corsa, leggere un libro;
- un app: l'applicazione su cui la storia è pubblicata, la cui icona si trova anche al fianco della storia stessa.

Facebook fornisce alcuni oggetti e azioni per i casi di uso frequente, e si possono anche creare azioni e oggetti personalizzati per soddisfare le esigenze particolari.

Social Time Machine come azione usa la parola "sfidare" e come oggetto usa la parola "i suoi amici"

2.3.6 Integrazione con Parse

La piattaforma Parse fornisce una soluzione backend completa per l'applicazione. L'obiettivo è quello di eliminare del tutto la necessità di scrivere il codice del server o mantenere i server. Su Parse, si crea uno schema per ciascuna delle applicazioni mobile. Ogni App ha un proprio id applicazione e chiave client, generati da Parse, che si applicano all'installazione. L'account su Parse può essere ospitato da molte applicazioni. Questo è utile anche se si dispone di un'applicazione, dal momento che è possibile distribuire versioni differenti per il test e la produzione.

2.3.7 Notifiche Push

Le notifiche push sono ideali per mantenere gli utenti informati sull'applicazione. È possibile raggiungere l'intera base di utenti in modo rapido ed efficace. Le notifiche push consentono all'applicazione di comunicare all'utente le nuove sfide, anche quando l'utente non sta usando attivamente l'applicazione. Quando l'utente tocca la notifica, viene aperta l'applicazione. Le notifiche saranno trasmesse a tutti i partecipanti.

2.3.8 Salvataggio del gioco

Dopo che l'utente ha inserito tutte le informazioni riguardanti il gioco, il server deve fornire il servizio di salvataggio. L'utente che crea il gioco deve anche creare subito la sua mossa, cosi gli altri possono capire facilmente in che cosa consiste la sfida. La memorizzazione dati su Parse è costruito intorno al ParseObject. Ogni ParseObject contiene coppie chiave-valore di dati compatibili con JSON. Questi dati sono schemaless, il che significa che non è necessario specificare in anticipo quali chiavi esistono su ogni ParseObject. È sufficiente impostare qualsiasi valore-chiave coppie che si desidera, e il backend di Parse lo conserverà.

2.3.9 Recupero della storia

I dati fondamentali che servono all'applicazione per creare la storia di un utente sono rappresentati da una lista delle mosse. Tale lista viene ricavata effettuando una richiesta al cloud, il quale ritorna le informazioni desiderate. In particolare, per ogni mossa, vengono fornite le seguenti informazioni:

- il nome utente;
- la foto profilo dell'utente;
- il titolo del gioco;

- la descrizione;
- la foto;
- la data di pubblicazione;
- la quntità dei voti;

2.3.10 Recupero dei giochi attivi

L’utente avrà la possibilità di vedere tutti i giochi attivi, ovvero i giochi aperti dove è stato invitato, ma ancora non ha partecipato. Una lista dei giochi, ognuna di quali è rappresentata da:

- il titolo;
- la data d’inizio;
- la data di finel

Anche nel caso in cui il giocatore non volesse partecipare deve avere la possibilità di rifiutare la richiesta.

2.3.11 Recupero della galleria

Quando il gioco è finito, l’applicazione deve fornire la possibilità di visualizzare tutte le mosse del gioco. Per questo scopo si usa la galleria delle foto di tutti i partecipanti con la breve descrizione. Il nome di ogni galleria corrisponderà al nome del gioco.

2.4 Logica lato client

Nella logica lato client, gli aspetti principali sono: la gestione della cache, l'utilizzo della gallery del telefono, la memorizzazione dell'ultimo account Facebook che ha effettuato l'accesso all'app.

2.4.1 Salvataggio delle immagini nella cache

L'applicazione utilizza la cache del telefono per una migliore gestione delle immagini. In particolare, ogni qualvolta viene caricata un'immagine, viene verificato se essa è già presente nelle cache e in tal caso viene mostrata immediatamente all'utente. Invece, se l'immagine non è già presente nella cache, viene caricata in quest'ultima e mostrata nell'applicazione.

Inoltre, per una migliore gestione delle risorse, quando l'applicazione termina, vengono cancellate le immagine caricate nella cache. Il motivo principale del salvataggio temporaneo delle immagini è dato dal fatto che se viene richiesta la visualizzazione di una stessa immagine più volte nella stessa sessione, dopo una prima volta non sarà più necessario effettuare un'interrogazione al server per il caricamento della relativa immagine, consentendo un miglioramento delle prestazioni.

2.4.2 Salvataggio dell'account Facebook

Per consentire ad uno stesso utente di utilizzare frequentemente l'applicazione senza dover rifare il login ad ogni accesso, attraverso l'utilizzo delle preferenze, viene memorizzato l'access token, ovvero una stringa generata al momento del login dell'utente che tiene conto anche dei permessi concessi da quest'ultimo all'applicativo.

2.4.3 Interfaccia utente

L'applicazione si presenta con un menù a schede, contenente: My History, Friends' History, All History. Nella parte sottostante viene caricato il contenuto delle singole schede. Nel fondo si presente il pulsante "add", che permette agli utenti di aggiugere un nuovo gioco. Inoltre nella parte superiore del layout viene visualizzato il nome dell'applicazione e quattro icone: la prima per vedere la gallery dei giochi finiti, la seconda per vedere i giochi attivi, la terza per invitare gli amici ad usare l'applicazione e la quarta per effettuare il

logout.

"My History" è la prima scheda che viene mostrata quando si avvia l'applicazione. Permette la visualizzazione di tutte le mosse effettuate dall'utente loggato e contiene i dettagli relativi alla mossa (il nome dell'utente, la foto di profilo, il nome del gioco, la descrizione della mossa e l'elemento principale: la foto).

La scheda "Friends' Histories" visualizza tutte le mosse degli amici dell'utente loggato, e la scheda "All History" visualizza sia le mosse del utente sia dei suoi amici.

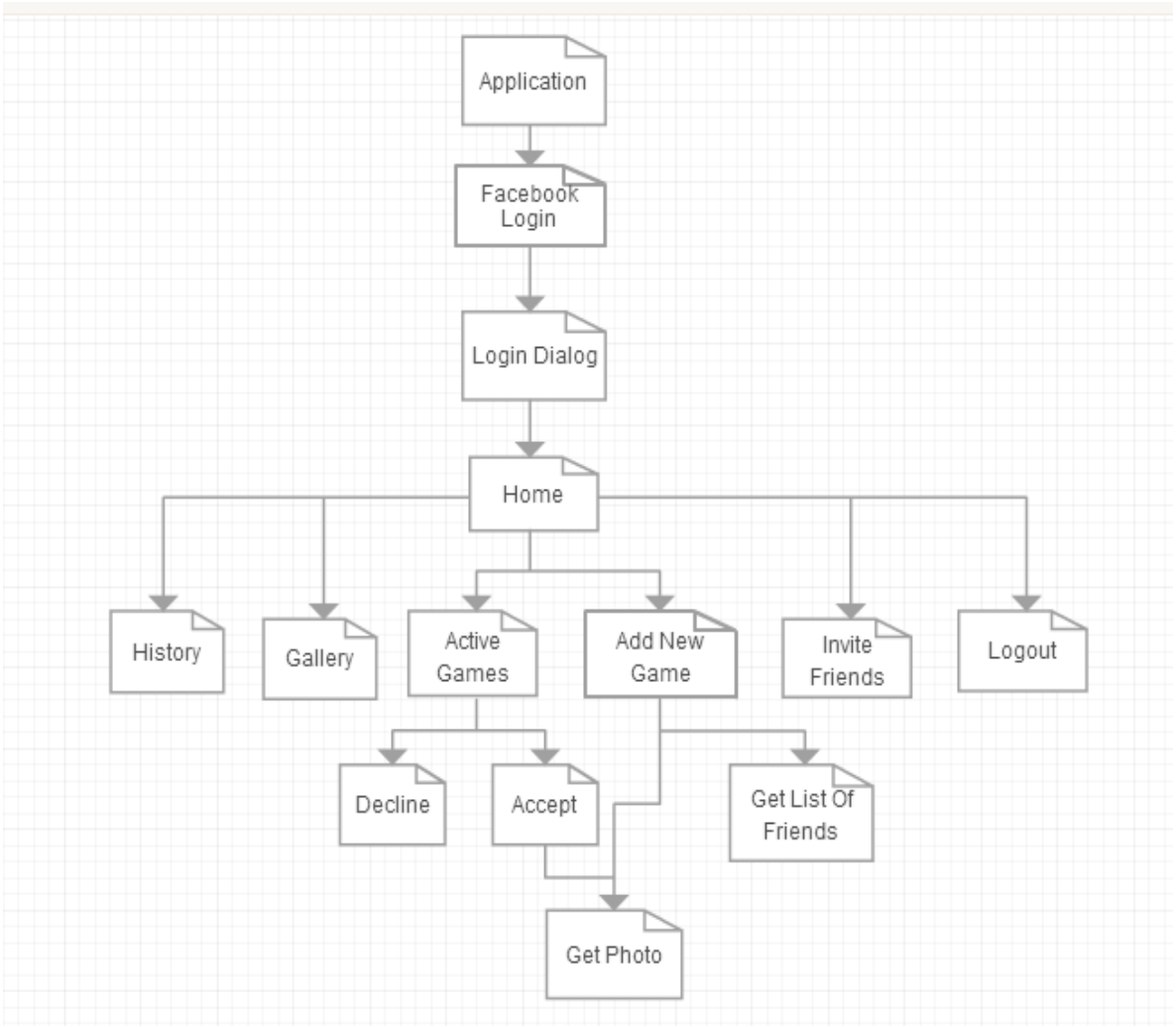
Il pulsante gallery apre la vista di tutti i giochi finiti dimostrando tutte le foto scattate dai partecipanti. Sotto ogni foto si può vedere la breve descrizione ed il pulsante "vote".

Il pulsante "Active Games" visualizza l'elenco di tutti i giochi attivi rapparessentati dal titolo, dalla data d'inizio e dalla data di fine. Sotto ogni gioco ci sono due pulsanti "Accept" e "Decline". Quando utente preme "Accept" viene caricata la form dove si può aggiungere una nuova mossa inserendo la descrizione e la foto.

Quando utente preme il pulsante di aggiungere nuovo gioco viene visualizzata la form che contiene i seguenti campi : titolo, descrizione, data d'inizio, data di fine e foto.

In caso l'utente volesse pubblicare la sua mossa su Facebook, deve selezione casella "POST".

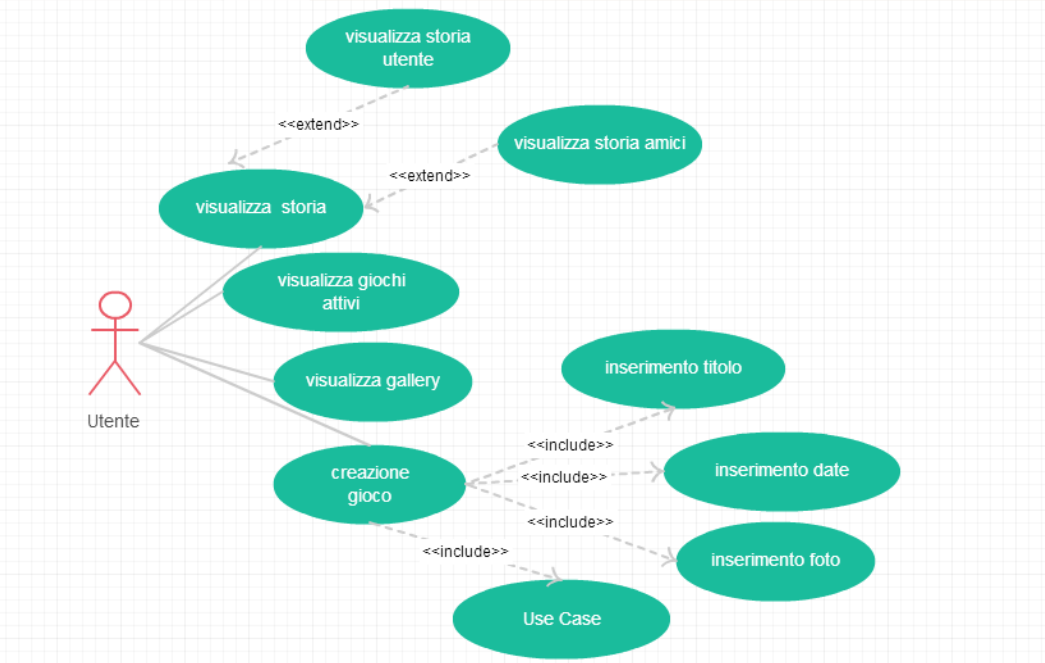
Figura 2- Schema di navigazione



2.5 Diagramma dei casi d'uso

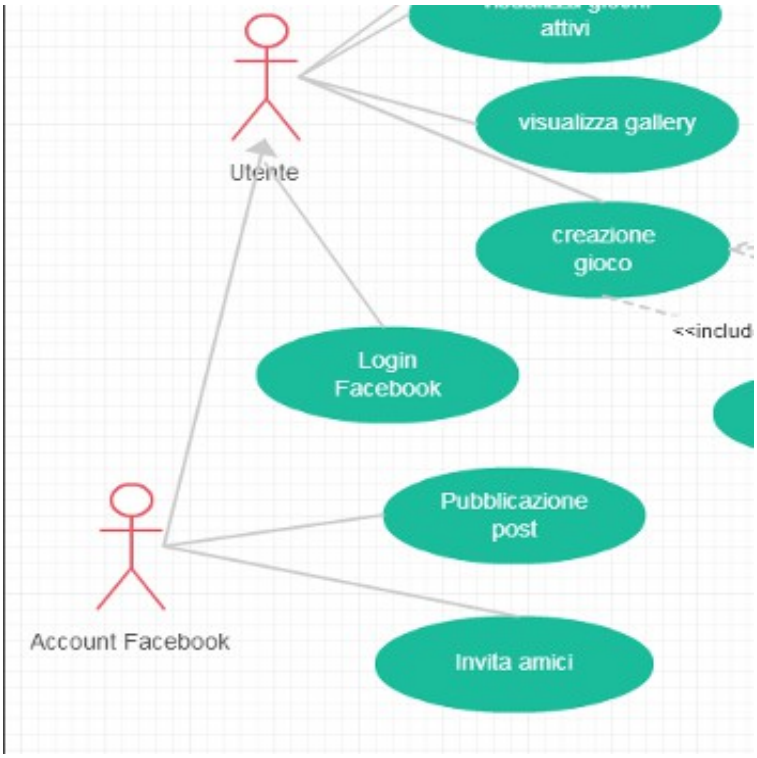
Il diagramma dei casi d'uso permette di mostrare come avviene l' iterazione tra l'attore, in questo caso rappresentato dall'utente, e il sistema, cioè l'applicazione.

Figura 3- Diagramma dei casi d'uso utente



La figura 3 mostra il diagramma casi d'uso e ci fa capire tutte le azioni che può completare un utente. La visualizzazione della storia è suddivisa in tre parti. Utente potrebbe decidere di filtrare la storia sulla base delle proprie preferenze.

Figura 4 - Diagramma dei casi d'uso account



La figura 4 mostra la presenza di una generalizzazione costituita dall'utente che accede all'applicazione dall'account Facebook, ovvero lo stesso utente che ha già effettuato l'accesso al social network tramite l'applicativo. I casi d'uso del diagramma sono tutte le funzionalità messe a disposizione dell'utente: in particolare, quelle riguardanti l'utilizzo del social network sono disponibili per il solo account Facebook.

Implementazione

3.1 Sviluppo in Android

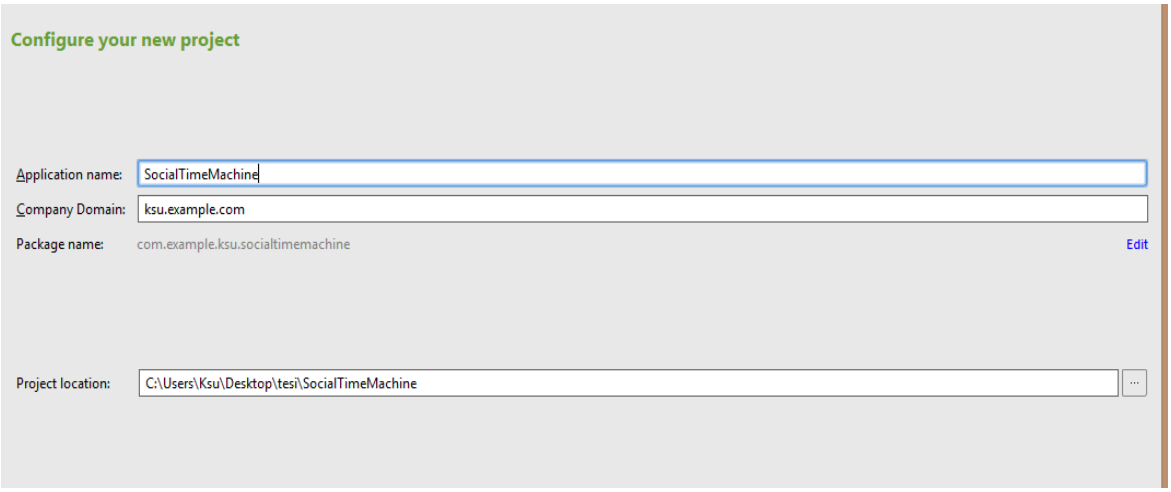
La decisione di sviluppare l'applicazione su Android è stata presa per un paio di particolarità di questo sistema operativo. La prima particolarità è data dal fatto che si tratta di un sistema open source con la possibilità di utilizzare un ambiente di sviluppo come Android Studio, anch'esso open source. La seconda, invece, è data dalla grande diffusione di Android, quindi adottato non da uno specifico device mobile, ma da diversi produttori come Samsung, LG, HTC ed altri.
Inoltre, altri aspetti che hanno portato a questa decisione sono: la possibilità di effettuare i test dell'applicativo direttamente su un dispositivo fisico e la presenza di una grande comunità online: due fattori che aiutano molto allo sviluppatore.
Dunque, un'ultima considerazione è quella che riguarda la pubblicazione dell'applicazione che, in questa piattaforma, risulta molto più semplice, veloce ed economica.

3.2 Struttura del progetto

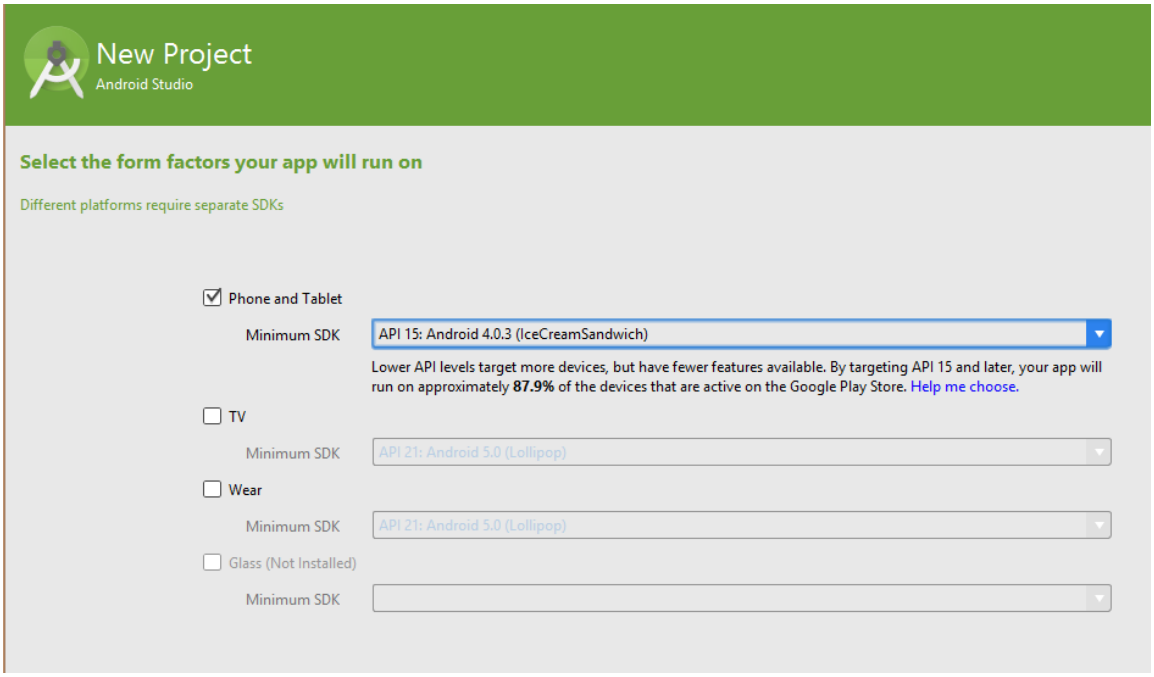
Prima di procedere con l'implementazione bisogna specificare la struttura del progetto. Un applicazione Android è costituita da diversi tipi di componenti che comprendono: codice sorgente Java, documenti XML, icone ed immagini, basi di dati e i file binari di altro tipo. Tutti questi componenti vanno in un qualche modo assemblati fra di loro ed in base della loro tipologia vanno inseriti nei punti specifici del progetto.
Cercheremo adesso di portare un po' di luce sull'intera struttura dell'applicazione, cercando di focalizzarci sugli aspetti fondamentali che riguardano l'implementazione di codice Java e le risorse XML che si possono definire, ed inoltre l'integrazione di questi componenti tra di loro.

3.2.1 Creazione di un nuovo progetto Android

Per creare un nuovo progetto in Android Studio bisogna selezionare `File -> New Project`, quindi si aprirà la finestra del wizard:



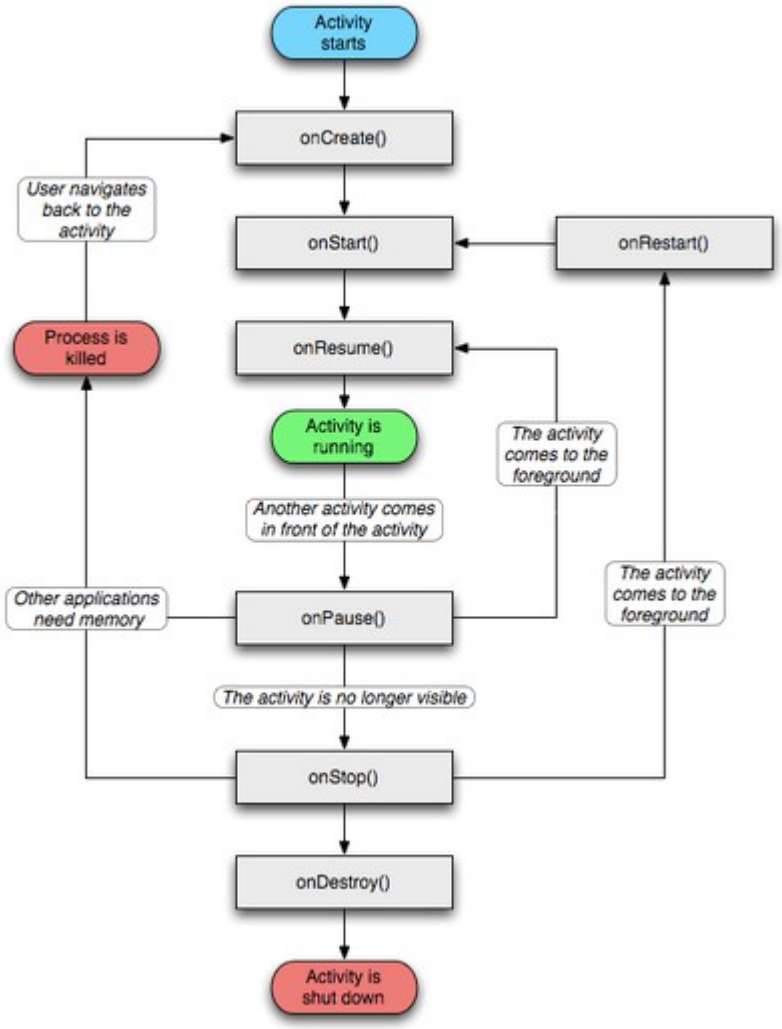
- Application name: questo è il nome dell'applicazione che poi sarà visualizzato nell'emulatore (o dispositivo) accanto all'icona dell'applicazione
- Package name: il nome del package dell'applicazione, deve essere scelto secondo la "reverse domain name notation" (ad es.: `com.example.socialtimemachine`)
- Project location: è il percorso del progetto nel filesystem, quindi la cartella principale nella quale Android Studio creerà tutte le cartelle principali che formano il progetto



Minimum SDK: siccome le API di Android sono in continua evoluzione bisogna specificare la versione del set di API a partire dalle quali l'applicazione che stiamo sviluppando sarà compatibile.

3.2.2 Activity e risorse

L'Activity rappresenta una delle classi fondamentali dell'architettura Android che ha un suo lifecycle (ciclo di vita), gestito dalla piattaforma, ma definito dallo sviluppatore che implementa i suoi metodi.
Nella figura qui sotto viene riassunto il ciclo di vita di un'Activity:

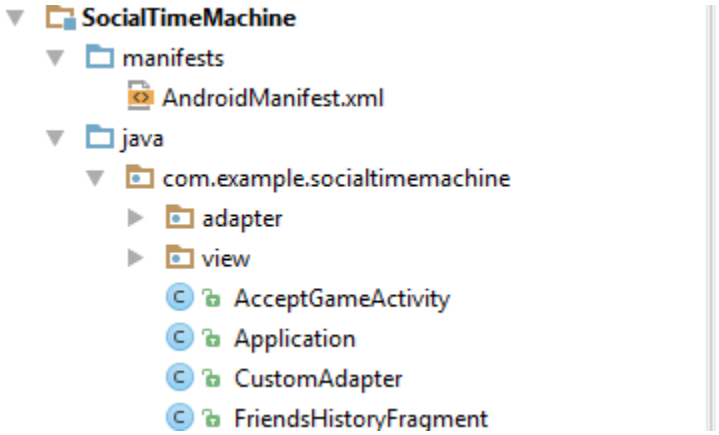


Per il momento è indispensabile capire che un'Activity rappresenta il punto di partenza di un'applicazione Android, dalla quale si sviluppano tutta la logica dell'applicazione con le sue diverse schermate e implementazioni di funzioni legate all'architettura.
Ritornando al Social Time Machine, qui sotto l'implementazione della sua Activity principale**:**

```
public class MainActivity extends FragmentActivity {
      private MainFragment mainFragment;
      @Override
      protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        if (savedInstanceState == null) {
              // Add the fragment on initial activity setup
              mainFragment = new MainFragment();
              getSupportFragmentManager()
              .beginTransaction()
              .add(R.id.container, mainFragment)
              .commit();
        }
        else {
              // Or set the fragment from restored state info
              mainFragment = (MainFragment)getSupportFragmentManager()
              .findFragmentById(R.id.container);
              }
        }
}
```

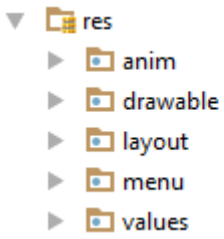
Per quanto riguarda il ciclo di vita, in questo caso è stato ridefinito solo il metodo onCreate(), che sarà il primo metodo invocato dalla piattaforma Android non appena viene lanciata l'applicazione. Fornire l'implementazione del suddetto metodo è essenziale affinchè l'applicazione venga visualizzata nel dispositivo.
Gli altri metodi comunque rivestono un ruolo fondamentale per un corretto funzionamento dell'applicazione.
I sorgenti si inseriscono nella cartella `java/com.example.socialtimemachine/`:



3.2.3 Le risorse

L'ambiente Android riprende il concetto di 'programmazione dichiarativa', dove alcune informazioni (risorse) vengono memorizzate nei file di configurazione (esterni al codice). Ciò offre il vantaggio di non dover ricompilare il codice al fronte di semplici modifiche di configurazione.

La cartella `res/` che contiene le risorse ha la seguente questa struttura:



Una risorsa può essere un'icona (od un altro immagine qualsiasi), un documento XML, oppure un file binario che viene utilizzato nell'applicazione. Comunque per ogni tipo di risorsa, corrisponde una cartella dove posizionare il file.

Specifichiamo quali tipi di risorse sono possibili posizionare nelle diverse sottocartelle:

- drawable-*: Queste cartelle (hdpi, ldpi, mdpi) possono contenere immagini di tipo GIF, PNG e JPEG. Da precisare che ognuna di queste cartelle ha il medesimo ruolo, in quanto Android selezionerà una di queste in base al tipo di display del dispositivo.
- values/: in questa cartella vanno inserite le risorse XML che definiscono le variabili insieme ai loro valori che possono essere stringhe, interi, od altre tipologie di dati. Qui sotto il codice XML contenuto nel file strings.xml:

  ```
  <resources>
       <string name="app_name">Social Time Machine</string>
  </resources>
  ```

  Viene definita semplicemente una stringa “app_name”, che rappresenta il nome dell'applicazione. Questo nome verrà visualizzato accanto all'icona dell’applicazione.
- layout/: contiene documenti XML che definiscono i componenti GUI utilizzati

nell'applicazione. I componenti possono essere, ad esempio, di tipo TableLayout, LinearLayout, ecc.; oppure controlli grafici come Button, EditText, ecc.
Qui sotto il documento activity_main.xml del progetto che è stato creato:

```
<FrameLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:id="@+id/container"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:background="@color/background_list"
      tools:context="com.example.socialtimemachine.MainActivity"
      tools:ignore="MergeRootFrame" />
```

ed i contenuti del corrispondente fragment_main.xml:

```
<?xml version = "1.0" encoding = "utf-8"?>
      <LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="horizontal"
      android:gravity="center"
      android:paddingBottom="@dimen/activity_vertical_margin"
      android:paddingLeft="@dimen/activity_horizontal_margin"
      android:paddingRight="@dimen/activity_horizontal_margin"
      android:paddingTop="@dimen/activity_vertical_margin"
      android:background="@color/background_list"
tools:context="com.example.socialtimemachine.MainActivity$Placehol
derFragment" >
<com.facebook.widget.LoginButton
       android:id="@+id/authButton"
       android:layout_width="wrap_content"
```

```
      android:layout_height="wrap_content" />
</LinearLayout>
```

In poche parole, è stato definito un layout che contiene un Frame che visualizza il pulsante di login per l'utente. La definizione della GUI è un aspetto molto importante che può essere facilitato utilizzando l'editor del plugin ADT che genera l'XML in automatico. (La GUI può essere definita anche direttamente nel codice senza ricorrere all'XML.)
Da una parte ci sono le risorse XML che definiscono alcune parti dell'applicazione (come la GUI oppure dei parametri e valori), mentre dall'altra parte c'è il codice Java all'interno di un'Activity che deve utilizzare queste risorse.
Per utilizzare le risorse esistono alcuni metodi implementati nella classe Activity. Ad esempio analizziamo il codice di MainActivity :

```
setContentView(R.layout.activity_main);
```

Il metodo setContentView() setta il layout grafico da visualizzare sullo schermo del dispositivo. Il metodo riceve un intero che identifica la risorsa XML dove è definito il layout grafico. Quindi nel codice sopracitato, la costante R.layout.activity_main identifica questa risorsa.
Un altro metodo molto importante è findFragmentById(R.id.container) che, in questo caso, ricerca un'elemento in base all'ID definito nella risorsa XML tramite questo tag: `android:id="@+id/container"`

3.2.4 AndroidManifest.xml

Tutte le informazioni specificate nella creazione del progetto vengono memorizzati nel file AndroidManifest.xml. Questo documento viene generato in automatico dall'ADT una volta terminata la creazione del progetto. In poche parole, questo file descrive al sistema operativo, l'applicazione che dovrà eseguire (non in termini di interfaccia grafica), raccogliendo le caratteristiche fondamentali come: il nome, l'icona associata, ecc. Questo file di configurazione si posizione nella root della cartella del progetto.
AndroidManifest.xml ha la seguente struttura:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
 package="com.example.socialtimemachine"
 android:versionCode="1"
 android:versionName="1.0" >
 <uses-sdk
   android:minSdkVersion="21"
   android:targetSdkVersion="21" />
 <uses-permission android:name="android.permission.INTERNET" />
 <application
   android:name="com.example.socialtimemachine.Application"
   android:allowBackup="true"
   android:icon="@drawable/ic_launcher"
   android:label="@string/app_name"
   android:theme="@style/AppTheme" >
   <meta-data
android:name="com.facebook.sdk.ApplicationId"
  android:value="@string/facebook_app_id" />
   <activity
     android:name="com.example.socialtimemachine.MainActivity"
     android:label="@string/app_name" >
     <intent-filter>
       <action android:name="android.intent.action.MAIN" />
       <category
android:name="android.intent.category.LAUNCHER" />
     </intent-filter>
   </activity>
 </application>
```

```
</manifest>
```

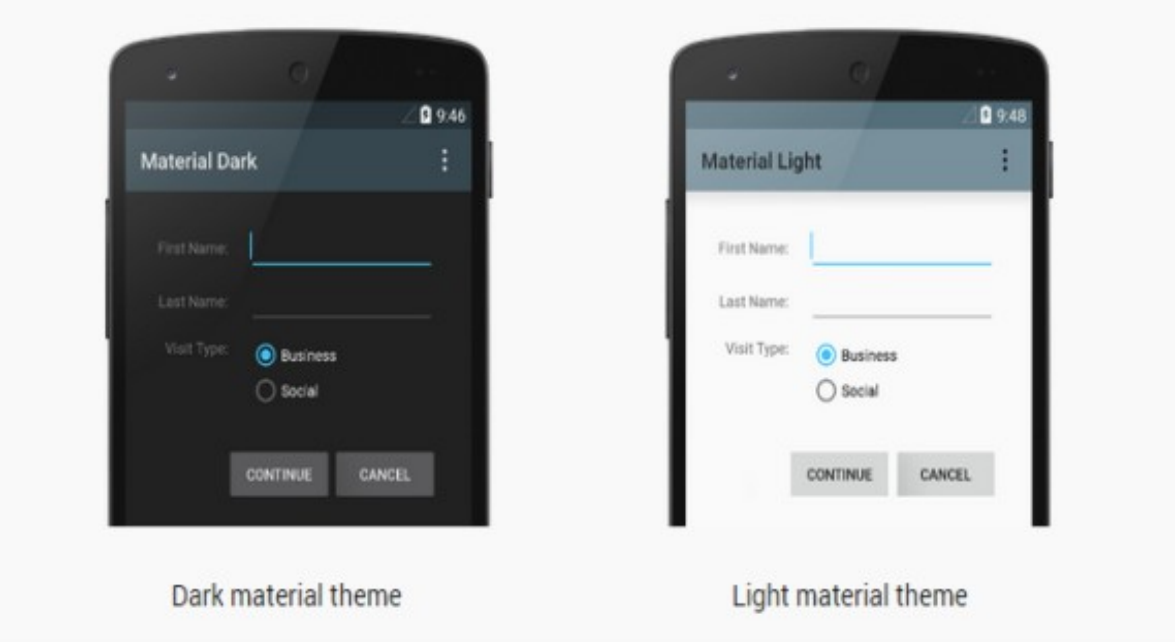
Scendere nei dettagli di tutti i parametri che si possono definire in suddetto file richiede approfondire altri concetti (come l'implementazione di servizi in background) oppure definire i permessi necessari per l'accesso alle funzionalità del dispositivo (come l'accesso alla foto camera).

3.3 Material Design

Nell'app si utilizza il material design come template.
Il Material Design è una guida completa per il design di visualizzazione, movimento ed iterazione attraverso piattaforme e dispositivi. Android supporta le applicazioni di material design. Per usarlo nelle applicazioni bisogna seguire le linee guida definite nella specifica di progettazione di material design ed utilizzare nuovi componenti e funzionalità disponibili per Android 5.0 (livello di API 21) e versioni successive.
Android fornisce i seguenti elementi per costruire applicazioni di material design:

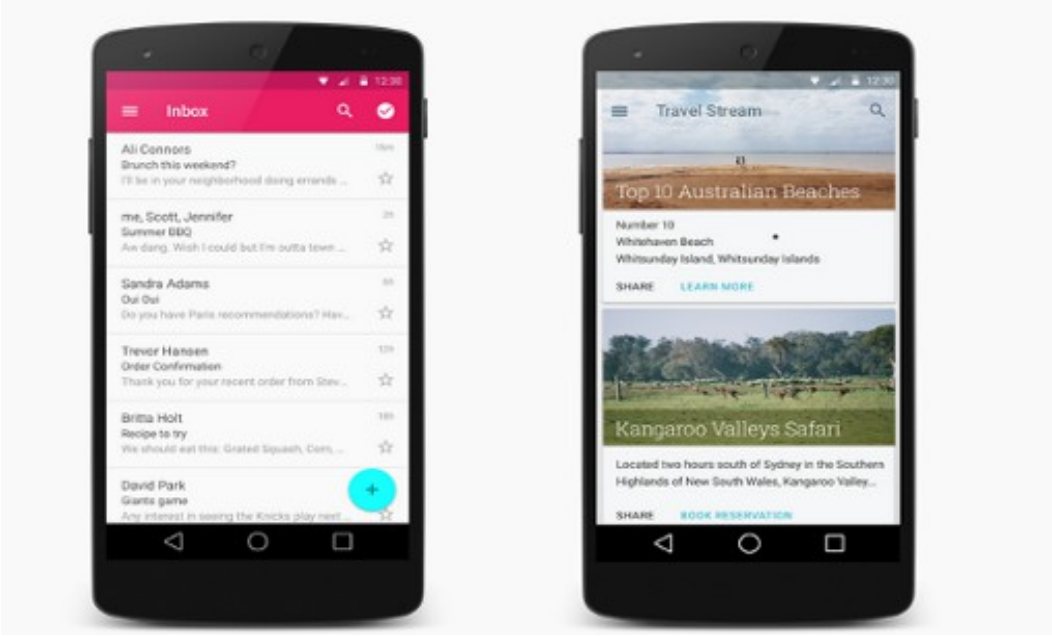
- nuove material theme;
- nuovi widget per le viste complesse;
- nuove API per le ombre e animazioni.3.3.1 Material Theme

Material Theme offre un nuovo stile per un'applicazione. I widget di sistema consentono l'impostazione della loro tavolozza di colori e animazioni predefinite per il touch feedback e activity transit.



3.3.1 Liste e Carte

Android implementa due nuovi widget per la visualizzazione di liste e carte con stili e animazioni di material design:



3.3.2 View Shadows

In aggiunta alle proprietà X e Y, le viste in Android ora hanno una proprietà Z. Questa nuova proprietà, che rappresenta l'elevazione di una vista, determina:

- la dimensione dell'ombra: le viste con i valori più elevati di Z hanno ombre più grandi;
- l'ordine di disegno: le viste con valori Z alti appaiono in cima ad altre viste.

3.3.3 Animazioni

Le nuove API di animazione consentono di creare animazioni personalizzate per il touch feedback per i controlli dell'interfaccia utente, cambiamenti di stato di visualizzazione e activity transactions.
Queste API consentono di:

- rispondere al tocco delle viste con animazioni di touch feedback;
- nascondere e mostrare le viste con animazioni circular reveal;
- passare da una attività ad un'altra con animazioni di activity transition;
- creare animazioni più naturali con curved motion;
- animare i cambiamenti in una o più proprietà della vista con animazioni view state change;
- mostra animazioni in state list drawable tra i cambiamenti dello stato di visualizzazione.

Le animazioni di touch feedback vengono realizzate in diverse viste standard come i pulsanti. Le nuove API consentono di personalizzare le animazioni e aggiungerle alle view.

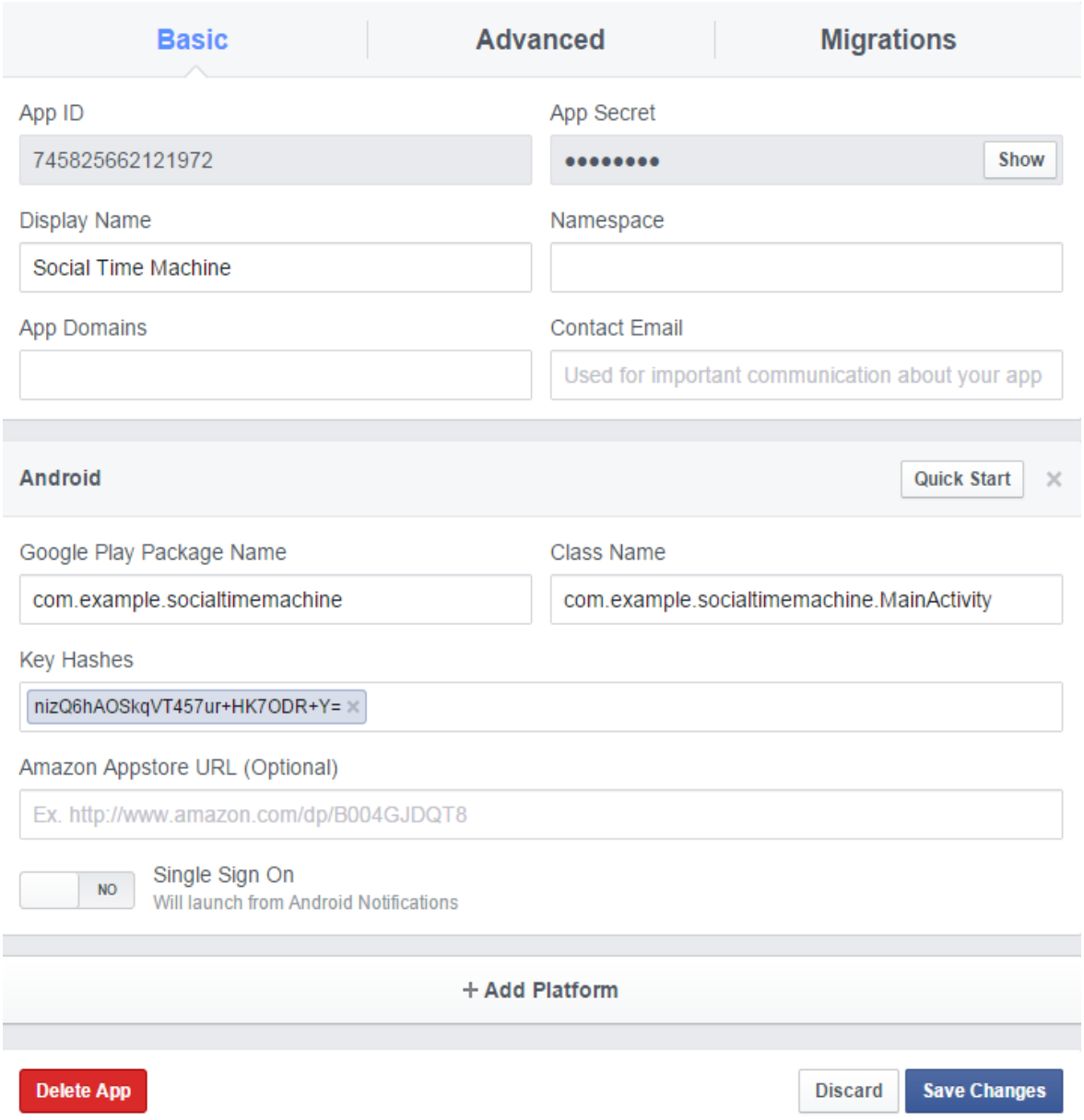
3.4 Implementazione dell’applicazione.

3.4.1 Integrazione Facebook SDK per Android

Per poter sviluppare usando Facebook SDK per Android bisogna registrarsi su facebook come sviluppatore e aggiungere la libreria al progetto. Facebook SDK per Android utilizza

l'applicazione Facebook per le funzionalità di Login e Sharing. Se si sta utilizzando un dispositivo vero, anziché l'emulatore, bisogna installare l'applicazione Facebook da Google Play. Nella pagina di Facebook per gli sviluppatori bisogna inserire le impostazioni principali. Nella figura sotto sono riportate i dati principali di configurazione dell'app:

Figura 5 - Impostazione dell'app sul Facebook



- app ID: l'id dell'applicazione, generata da Facebook, che viene inserita nel AndroidManifest.xml;

- display Name: corrisponde al nome dell'applicazione;
- google Package Name: il nome del pacchetto;
- class Name: l'activity principale dell'applicazione che viene lanciata all'inizio.

3.4.2 Facebook Login per Android

Il login del social network avviene tramite un'apposita finestra di login generata da Facebook.
Il Facebook SDK fornisce una vista LoginButton che viene utilizzata per implementare la funzione di Login. La classe LoginButton mantiene lo stato della sessione. Insieme al pulsante di accesso, si consiglia di controllare gli altri componenti dell'interfaccia utente a seconda se l'utente è autenticato o meno.
Facebook SDK include due classi: UiLifecycleHelper e Session.StatusCallback per gestire i cambiamenti dello stato. Nel fragment in cui si mostra la funzionalità di Login è stata creata un'istanza della classe UiLifecycleHelper. Session.StatusCallback viene informato di qualsiasi cambiamento dello stato di sessione. Il fragment fornisce i metodi pubblici UiLifecycleHelper che rispecchiano i metodi del ciclo di vita. Questi metodi sono utilizzati nella creazione, l'apertura, il salvataggio e il ripristino di una sessione Facebook.
L'implementazione Session.StatusCallback può sovrascrivere il metodo call() per rispondere ai cambiamenti di stato di sessione e aggiornare l'interfaccia utente di conseguenza.

```
private static final String TAG = "MainFragment";
private UiLifecycleHelper uiHelper;
     private Session.StatusCallback callback = new
     Session.StatusCallback() {
@Override
public void call(Session session, SessionState state, Exception
exception) {
onSessionStateChange(session, state, exception);
}
};
```

```
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
uiHelper = new UiLifecycleHelper(getActivity(), callback);
uiHelper.onCreate(savedInstanceState);
}
@Override
public View onCreateView(LayoutInflater inflater,
ViewGroup container,
Bundle savedInstanceState) {
View view = inflater.inflate(R.layout.fragment_main, container,
false);
LoginButton authButton = (LoginButton)
view.findViewById(R.id.authButton);
authButton.setFragment(this);
authButton.setReadPermissions(Arrays.asList("user_friends"));
return view;
}
private void onSessionStateChange(Session session, SessionState
state, Exception exception) {
if (state.isOpened()) {
Log.i(TAG, "Logged in...");
Intent intent = new Intent(getActivity(), HomeActivity.class);
startActivity(intent);
} else if (state.isClosed()) {
     Log.i(TAG, "Logged out...");
     }
}
private void onSessionStateChange(Session session, SessionState
state, Exception exception) {
if (state.isOpened()) {
     Log.i(TAG, "Logged in...");
     Intent intent = new Intent(getActivity(),
HomeActivity.class);
```

```
        startActivity(intent);
} else if (state.isClosed()) {
        Log.i(TAG, "Logged out...");
        }
}
```

3.4.3 Inviare le richieste

Facebook SDK fornisce i metodi per l'integrazione con Facebook, tra cui l'invio delle richieste che l'utente utilizza per invitare i suoi amici ad usare l'applicazione. La maschera delle richieste fornisce le funzionalità di base di Facebook con poche righe di codice. Non è necessario costruire finestre di dialogo native, eseguire chiamate alle API o gestire le risposte. La maschera delle richieste è implementata tramite la classe WebDialog e contiene un costruttore WebDialog.RequestsDialogBuilder. Al costruttore si passano i parametri aggiuntivi che descrivono la richiesta e poi viene chiamato il metodo show() per visualizzione. Prima di tutto bisogna aggiungere un pulsante nella action bar della home che visualizza la maschera di dialogo.
Il pulsante nel menu action bar per inviare le richieste è definito come segue:

```
<item
        android:id="@+id/invite_friends"
        android:icon="@drawable/invite_friends"
        android:title="Invite friends"
        android:showAsAction="always">
</item>
```

La voce di menu è gestita tramite il metodo onOptionsItemSelected

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
        //Handle presses on the action bar items
        switch (item.getItemId()) {
        case R.id.invite_friends:
        sendRequestDialog();
        return true;
```

```
      }
}
```

Nella classe HomeActivity è definito il metodo sendRequestDialog():

```
private void sendRequestDialog() {
      Bundle params = new Bundle();
      params.putString("message", "Invite you to try the STM");
      WebDialog requestsDialog = (
      new WebDialog.RequestsDialogBuilder(HomeActivity.this,
      Session.getActiveSession(),
      params))
      .setOnCompleteListener(new WebDialog.OnCompleteListener() {
      @Override
      public void onComplete(Bundle values,
      FacebookException error) {
            if (error != null) {
                  if (error instanceof
FacebookOperationCanceledException) {
      Toast.makeText(HomeActivity.this.getApplicationContext(),
      "Request cancelled",
      Toast.LENGTH_SHORT).show();
} else {
      Toast.makeText(HomeActivity.this.getApplicationContext(),
      "Network Error",
      Toast.LENGTH_SHORT).show();
      }
} else {
      final String requestId = values.getString("request");
      if (requestId != null {
      Toast.makeText(HomeActivity.this.getApplicationContext(),
      "Request sent",
      Toast.LENGTH_SHORT).show();
} else {
```

```
Toast.makeText(HomeActivity.this.getApplicationContext(),
      "Request cancelled",
      Toast.LENGTH_SHORT).show();
                  }
            }
      }
})
      .build();
      requestsDialog.show();
}
```

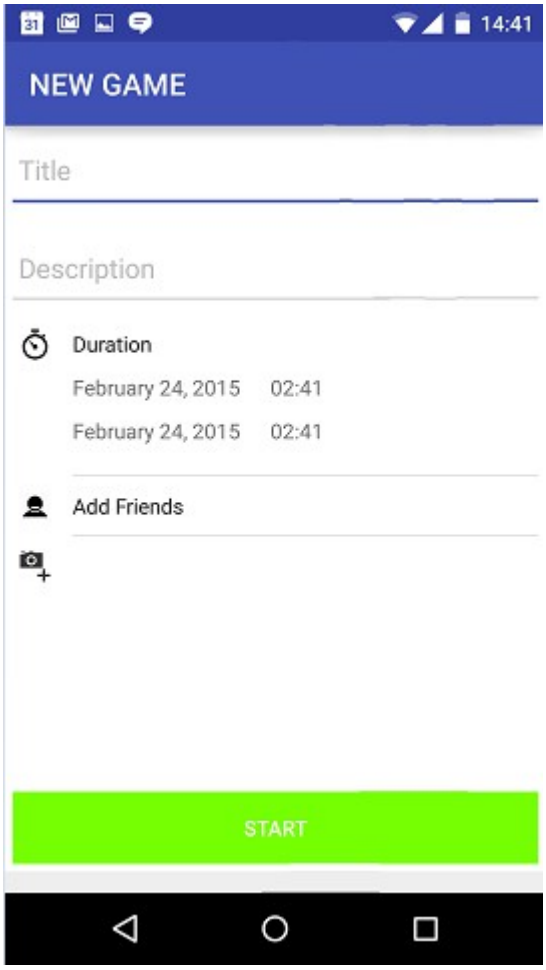
3.4.4 Inizializzazione Parse SDK

Su Parse si crea un'app per ciascuna delle applicazioni mobili. Ogni app ha un proprio id applicazione e client key che si applica durante l'installazione di SDK.

```
public class Application extends android.app.Application {
      public Application(){}
      @Override
      public void onCreate() {
            super.onCreate();
            // Initialize the Parse SDK.
            Parse.initialize(this,
      "CblPQNXB5bztS0zjzox1vPPb8mRCiOorvNQMD3Jb",
      "fcqLiSWLa2JVHMW0esKZP3ewkAJm0jYPEjhlYVmg");
      // Specify an Activity to handle all pushes by default.
      PushService.setDefaultPushCallback(this, MainActivity.class);
      }
}
```

3.4.5 Creazione di una nuova sfida

La form di inserimento dati consente all'utente di creare una nuova sfida aggiungendo i dati principali riguardanti il gioco: titolo, data di inizio, data di fine e i partecipanti.
La figura sottostante dimostra la form di creazione:



La creazione di un nuovo oggetto con Parse SDK avviene tramite associazione chiave-valore. Per salvare il gioco non è neccesario definire una classe apposita chiamata “Game”. L’applicazione Parse crea questa classe in automatico.

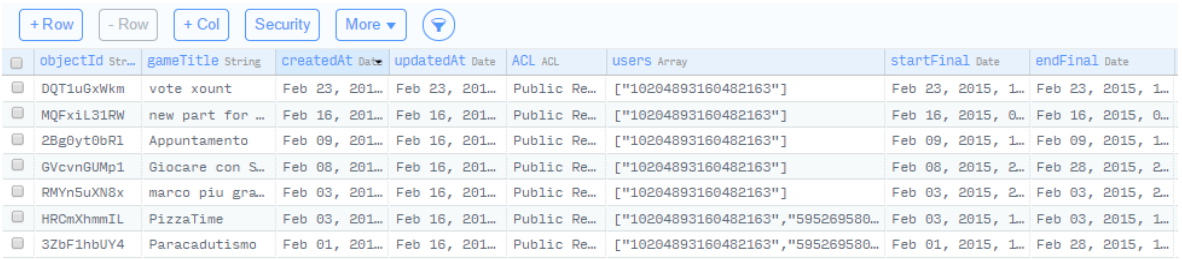
```
ParseObject newgame = new ParseObject("Game");
newgame.put("gameTitle", gameTitle.getText().toString());
newgame.put("startFinal", startFinal);
newgame.put("endFinal", endFinal);
```

```
newgame.addAll("users", usersIds);
newgame.saveInBackground();
```

Inoltre l'applicazione deve creare subito la mossa dell'utente con la foto e la descrizione:

```
ParseObject newpart = new ParseObject("Part");
Bitmap bitmap = ((BitmapDrawable)
gameImage.getDrawable()).getBitmap();
ByteArrayOutputStream stream = new ByteArrayOutputStream();
bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);
ParseFile file = new ParseFile("picturePath.png",
stream.toByteArray());
newpart.put("image", file);
newpart.put("gameParent",newgame);
newpart.put("description", gameDescription.getText().toString());
newpart.put("user", userId);
newpart.put("vote", 0);
newpart.saveInBackground();
```

Per verificare il salvataggio dei dati, Parse predispone un'interfaccia web con tabelle che corrispondono alle classi definite nell'applicazione:



objectId Str…	gameTitle String	createdAt Date	updatedAt Date	ACL ACL	users Array	startFinal Date	endFinal Date
DQT1uGxWkm	vote xount	Feb 23, 201…	Feb 23, 201…	Public Re…	["10204893160482163"]	Feb 23, 2015, 1…	Feb 23, 2015, 1…
MQFxiL31RW	new part for …	Feb 16, 201…	Feb 16, 201…	Public Re…	["10204893160482163"]	Feb 16, 2015, 0…	Feb 16, 2015, 0…
2Bg0yt0bRl	Appuntamento	Feb 09, 201…	Feb 16, 201…	Public Re…	["10204893160482163"]	Feb 09, 2015, 1…	Feb 09, 2015, 1…
GVcvnGUMp1	Giocare con S…	Feb 08, 201…	Feb 16, 201…	Public Re…	["10204893160482163"]	Feb 08, 2015, 2…	Feb 28, 2015, 2…
RMYnSuXN8x	marco piu gra…	Feb 03, 201…	Feb 16, 201…	Public Re…	["10204893160482163"]	Feb 03, 2015, 2…	Feb 03, 2015, 2…
HRCmXhmmIL	PizzaTime	Feb 03, 201…	Feb 16, 201…	Public Re…	["10204893160482163","595269580…	Feb 03, 2015, 1…	Feb 03, 2015, 1…
3ZbF1hbUY4	Paracadutismo	Feb 01, 201…	Feb 16, 201…	Public Re…	["10204893160482163","595269580…	Feb 01, 2015, 1…	Feb 28, 2015, 1…

Ci sono alcuni campi che non devono essere definiti esplicitamente. Questi campi vengono forniti automaticamente per ogni oggetto salvato nella piattaforma Parse:

- objectId: un identificatore univoco;
- createdAt e updatedAt: rappresentano rispettivamente il tempo in cui ogni oggetto è stato creato e modificato nel cloud.

  Ognuno di questi campi viene valorizzato dal Parse SDK e non viene instanziato sul

ParseObject fino a quando un'operazione di salvataggio non sarà terminata con successo.

3.4.6 Invio delle notifiche Push

La piattaforma Parse fornisce la funzionalità di invio delle notifiche push utilizzando Google Cloud Messaging (GCM). Tramite quest'ultimo si invia una notifica push ai dispositivi Android. Ci sono diversi tipi di informazioni che Parse gestisce in automatico:

- ID registrazione: l'ID di registrazione GCM identifica in modo univoco una coppia di app-device;
- ID mittente: l'ID GCM mittente è un numero pubblico che identifica il mittente di una notifica push;
- chiave API: la chiave GCM API è un'informazione segreta che permette ad un server di inviare le notifiche push a un ID di registrazione.

Il Parse Android SDK sceglie una configurazione di default evitando allo sviluppatore le gestione dell'ID registrazione, dell'ID mittente e delle chiavi API. In particolare, l'SDK è responsabile della registrazione automatica di un ID per inviare le notifiche. Inoltre Parse creerà un legame tra l'ID registrazione e il campo deviceToken dell'app.
Ogni dispositivo registrato usando Parse SDK viene salvato in un oggetto Installation, nel quale sono memorizzati tutti i dati necessari per inviare le notifiche push. In Android gli oggetti Installation sono messi a disposizione dalla classe ParseInstallation: una sottoclasse di ParseObject. Quest'ultima utilizza la medesima API per l'archiviazione e il recupero dei dati. Per accedere all'oggetto d'installazione dell'utente corrente è richiesto l'utilizzo del metodo ParseInstallation.getCurrentInstallation().
La prima volta che l'utente effettua il login viene salvata un'istanza dell'oggetto ParseInstallation inserendo lo userId,

```
@Override
public void onCompleted(GraphUser user, Response response) {
if (user != null) {
      userId = user.getId();
      ParseInstallation installation =
      ParseInstallation.getCurrentInstallation();
```

```
        installation.put("userId", userId);
        installation.saveInBackground();
        }
}
```

Dopo che l'utente ha salvato il gioco, tutti i partecipanti riceveranno una notifica push che avviserà l'inizio del gioco.
Per ogni utente inserito deve essere eseguita la seguente operazione:

```
ParseQuery pushQuery = ParseInstallation.getQuery();
pushQuery.whereEqualTo("userId", userId);
ParsePush push = new ParsePush();
push.setQuery(pushQuery);
push.setMessage("You've received a new challange");
push.sendInBackground();
```

3.4.7 Recupero dei dati

Quando si avviano i componenti di un'applicazione Android, che non ha altri componenti in esecuzione, il sistema operativo avvierà un nuovo processo con un singolo thread di esecuzione. Di default, tutti i componenti della stessa applicazione vengono eseguiti nello stesso processo chiamato thread "principale". Tuttavia, è possibile eseguire diversi componenti dell'applicazione in processi separati potendo creare dei thread aggiuntivi per ogni processo. Il recupero dei dati viene eseguito usando un nuovo thread asincrono (un thread che non blocca quello principale).
Per dimostrare un esempio di recupero dati consideriamo quello relativo alla lista delle mosse: nella HistoryActivity viene eseguito il task che recupera i dati in background e inizializza l'adapter.

```
public View onCreateView(LayoutInflater inflater, ViewGroup
container,Bundle savedInstanceState){
        ViewGroup rootView = (ViewGroup)
        inflater.inflate(R.layout.tab_item, container, false);
```

```
      new HistoryTask().execute(2);
      return rootView;
   }

public class HistoryTask extends AsyncTask<Integer, Void, Void> {
      ListView listview;
      ProgressDialog mProgressDialog;
      CustomAdapter adapter;
      private String userId;
      @Override
      protected void onPreExecute() {
      super.onPreExecute();
      mProgressDialog = new ProgressDialog(getActivity());
      mProgressDialog.setTitle("History");
      mProgressDialog.setMessage("Loading...");
      mProgressDialog.setIndeterminate(false);
      mProgressDialog.show();
}

@Override
protected Void doInBackground(Integer... params) {
      Session session = Session.getActiveSession();
      if (session != null) {
      Request request = Request.newMeRequest(session, new
Request.GraphUserCallback() {
@Override
public void onCompleted(GraphUser user, Response response) {
if (user != null) {
                userId = user.getId();
           }
      }
});
Request.executeAndWait(request);
```

```
}
adapter = new CustomAdapter(getActivity(), params[0], userId);
return null;
}

@Override
protected void onPostExecute(Void result) {
listview = (ListView)getActivity().findViewById(R.id.game_list);
listview.setAdapter(adapter);
});
           mProgressDialog.dismiss();
     }
}
```

Adapter a sua volta si connette al cloud di Parse, quindi recupera tutte le mosse che corrispondono alla ricerca effettuata dall'utente:

```
public CustomAdapter(Context context, final int type, final String
userId) {
      // Use the QueryFactory to construct a PQA that will only
show
super(context, new ParseQueryAdapter.QueryFactory<ParseObject>() {
  public ParseQuery create() {
              ParseQuery query = null;
              if (type == GET_ALL_GAMES) {
                  query = new ParseQuery("Part");
              }
              else if (type == GET_MY_GAMES){
                  query = new ParseQuery("Part")
                   .whereEqualTo("user", userId);
              } else {
                   query = new ParseQuery("Part")
                        .whereNotEqualTo("user", userId);
```

```
                }
                query.include("gameParent");
                return query;
            }
        });
}
```

La pagina principale sarà visualizzata all'utente nel seguente modo:

Conclusioni

Grazie alla realizzazione di questo progetto ho potuto migliorare le mie conoscenze sul cloud computing e programmazione mobile, in particolare sulla piattaforma Android.

Inoltre, il progetto mi ha consentito di scoprire un'innovazione che si sta affermando proprio nel periodo più recente e che potrebbe avere sviluppi futuri molto interessanti, ovvero l'applicazione di Social Gaming che nel corso degli ultimi anni ha attirato tanto attenzione. Social Gaming è importante anche dal punto di vista economico perché è un mercato in rapida espansione che può offrire ottime opportunità per interagire con gli utenti.

La programmazione per dispositivi mobili è molto diversa dalla programmazione per i dispositivi fissi: le risorse di un dispositivo mobile sono limitate, quindi è necessario rispettare le operazioni svolte, facendo attenzione alle prestazioni.

I maggiori sviluppi futuri per l'applicazione potrebbero riguardare il miglioramento della logica del gioco: aggiunta della tipologia che specifica il tipo della sfida; video che potrebbe rendere l'applicazione più divertente; timer su status bar, cosi che l'utente possa avere la possibilità di gestire meglio il tempo per completare la sfida, anche le notifiche push prima della scadenza del tempo.

Concludendo, "Social Time Machine" nello suo stato attuale è ancora agli inizi, ma in futuro potrebbe diventare un'applicazione usata da molte persone.

Bibliografia

- Android Tutorial
  https://developer.android.com/training/index.html
- Facebook Developers
  https://developers.facebook.com/
- Parse. Android Guide
  https://parse.com/docs/android_guide
- If you love something, let it go mobile: Mobile marketing and mobile social media 4x4
  Andreas Kaplan
- Sviluppare applicazioni per Android
  Carli Massimo
- App native contro web app: quali sono meglio?
  Silvio Gulizia
- Fundamentals of Grid Computing: Theory, Algorithms and Technologies
  Frédéric Magoulès