Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Progettazione e sviluppo di un sistema per la gestione dei pagamenti su piattaforme mobili Android

Relatore:
Chiar.mo Prof.
LUCIANO BONONI

Presentata da:
LORENZO BALLIN

Correlatore:
Chiar.mo Dott.
LUCA BEDOGNI

Sessione II
Anno Accademico 2013-2014

'Il valore di un'idea sta nel metterla in pratica.'
(Thomas Alva Edison)

Introduzione

Negli ultimi anni abbiamo assistito ad una crescita senza eguali nella produzione e nello sviluppo di dispositivi mobili, che probabilmente non subirá alcuna flessione neanche negli anni avvenire.
Il rapporto di *Ericsson Mobility Report* sulla *Networked Society* stima, infatti, che nel 2018 gli smartphone ed i tablet venduti nel mondo saranno 4 miliardi con un consumo di dati in mobilitá di 14mila petabyte.[10]
Uno dei punti cardini di questo spaventoso incremento é la possibilitá dei dispositivi di garantire un accesso alla rete costante, scambiando dati a velocitá sempre maggiori.
Il mondo della comunicazione, trascina indirettamente con sé anche la crescita sostanziale e tangibile nello sviluppo di applicativi software per dispositivi mobili.
Le applicazioni scaricate nel 2013 sono circa 102 miliardi (stima Gartner).[14]

É innegabile, quindi, che il mio lavoro di tesi sia stato fortemente influenzato da questi dati.
Il processo di realizzazione di questo lavoro nasce con la ricerca di un'intuizione che potesse emergere come novitá nell'oceano di possibilitá offerte dal mercato degli applicativi per smartphone.
Il risultato finale di questa ricerca ha prodotto una conclusione apparentemente ambiziosa: sostituire il vecchio concetto di 'portamonete' con una versione piú pratica, innovativa ed in accordo con l'attuale direzione delle tecnologie moderne.
In questo lavoro di tesi mi sono occupato, quindi, di progettare e sviluppare un software che fornisse un'interfaccia semplice, intuitiva e funzionalmente completa per la gestione dei pagamenti tramite cellulare.

Per conseguire questo obiettivo sono state implementate funzioni e un'interfaccia lato client, oltre che un importante insieme di strumenti lato server.

Il seguente documento di tesi é diviso in diversi capitoli:

- Nel primo capitolo vengono esposte le caratteristiche e l'architettura del sistema operativo Android (utilizzato nella fase di sviluppo).
  Per completezza del trattato vengono successivamente descritti e presentati i sistemi operativi concorrenti ad Android: iOS, sistema operativo proprietario della Apple Inc. e Windows Phone, astro nascente nel panorama dei sistemi operativi per piattaforme mobili di proprietá della Microsoft Inc.

- Il capitolo successivo affronta particolari tematiche inerenti la sicurezza ed la diffusione del sistema operativo Android, fornendo esempi,spunti e casi di studio tratti da articoli scientifici di spessore e rilevanza.

- Nel terzo capitolo vengono illustrate le tecnologie integrate nel progetto: il servizio PayPal, leader mondiale nei pagamenti online e sistema di pagamento scelto nello sviluppo del progetto e GoogleCloudMessaging, che verrá utilizzato in fase di sviluppo per implementare un adeguato servizio di notifiche all'utente.

- Il penultimo capitolo descrive nei minimi particolari l'architettura, le funzionalitá e l'interfaccia dell'applicazione.Vengono anche esposte le scelte progettuali prese sul tema fondamentale della sicurezza.

- In 'Conclusioni Sviluppi futuri' viene riassunto lo stato dell'opera ed illustrato come il progetto possa essere esteso e migliorato.

Indice

Capitolo 1

Il sistema operativo Android

1.1 Architettura

Android é un sistema operativo progettato nel 2003 e inizialmente sviluppato da *Android Inc.* una societá composta da Andy Rubin (co-fondatore di Danger), Rich Miner (co-fondatore di Danger e di Wildfire Communications), Nick Sears (vicepresidente di T-Mobile), e Chris White (principale autore dell'interfaccia grafica di Web TV), nata con lo scopo di creare dispositivi cellulari piú consapevoli della posizione e delle preferenze del loro proprietario.

Successivamente, nell'Agosto del 2005, la societá fu acquistata da *Google Inc.* con l'obiettivo dichiarato di 'creare un vero e proprio successo, in modo da migliorare l'esperienza mobile per gli utenti. [15]
Ad oggi il sistema operativo Android risulta essere uno dei maggiori sistemi operativi presenti sul mercato mondiale presente in oltre 255 milioni di dispositivi grazie, sopratutto, alla sua natura open-source.

Android adotta, infatti, una politica di licenza *Apache*. Questa tipologia di licenza di sviluppo consente agli utenti di usare il software per ogni scopo, di distribuirlo, modificarlo e di distribuire versioni modificate del software.[4]

Da un punto di vista piú tecnico e architetturale il sistema operativo Android é facilmente rappresentabile come una struttura a livelli:

1. Kernel di Linux
2. Librerie e ambiente di runtime
3. Framework
4. Applicazioni

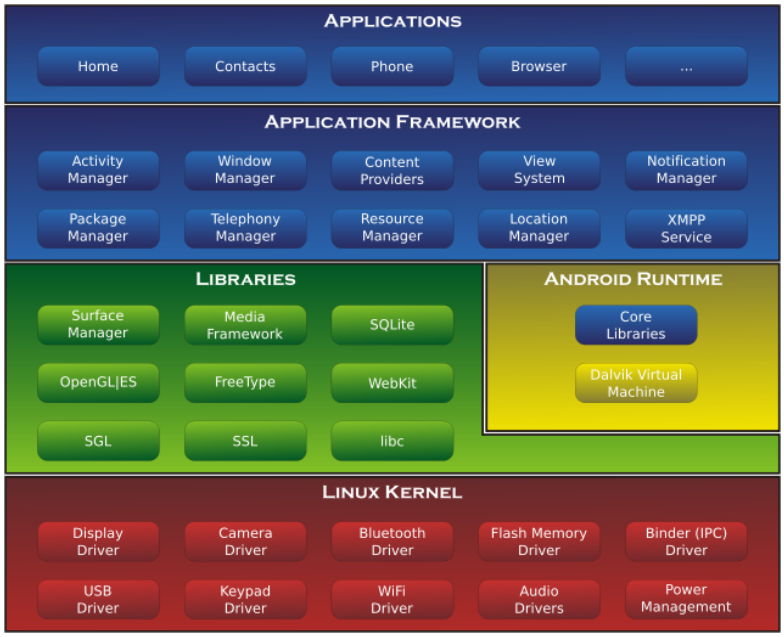

Figura 1.1: Architettura sistema Android

Il kernel di Linux costituisce il livello piú basso e permette l'astrazione da tutto l'hardware sottostante(wi-fi, bluetooth, GPS, fotocamera, touch-screen...)
Il kernel di Linux di Android é un sistema multi-utente nel quale ogni applicazione é identificata come un utente differente. Il sistema infatti associa un ID per ogni applicazione e imposta i permessi dei file dell'applicazione stessa in modo che soltanto quell'ID possa averne accesso.

Inoltre, ogni applicazione sul telefono viene lanciata in un processo Linux a sé stante all'interno della propria istanza della JVM garantendo, difatti, la stabilitá del dispositivo.
Il sistema permette alle applicazioni di comunicare tra loro condividendo lo stesso user ID e la stessa JVM in modo da preservare la coerenza delle risorse di sistema.

Il quarto livello é composto dalle librerie fondamentali relative ad un insieme di progetti Opensource come, ad esempio, OpenGL ES (per la gestione della grafica 2D e 3D), SQLite (per il supporto al database).
Un'altra libreria fondamentale presente in questo livello é la cosiddetta Surface Manager (SM);questo indispensabile componente nasce con lo scopo di coordinare le diverse finestre che le applicazioni vogliono visualizzare sullo schermo gestendo le view di ogni applicazione.

Questo livello comprende anche un particolare insieme di librerie (dette core libraries) che insieme ad una macchina virtuale (VM) costituiscono la piattaforma di sviluppo per Android.
Si noti che la macchina virtuale di Android é una versione particolare della Java Virtual Machine (chiamata Dalvik), progettata e sviluppata appositamente per adattarsi ad hardware particolari come quelli degli smartphone, in principio non altamente performanti.
(Ad oggi, in realtá lo sviluppo e la ricerca continua, producono periferiche sempre piú evolute e performanti.)

Al penultimo livello sono presenti i gestori e le applicazioni di base del sistema, come gestori per le risorse, per le applicazioni installate, per le telefonate, il file system e altro ancora.

Al livello piú alto risiedono le applicazioni utente scritte in linguaggio Java e

che girano ognuna nella propria JVM. Android dispone di una vasta comunitá di sviluppatori che realizzano applicazioni con l'obiettivo di aumentare le funzionalitá dei dispositivi. Le funzionalitá base del sistema, come per esempio il telefono, non sono altro che applicazioni utente.

1.2 Panoramica versioni

Android ha un rapido ciclo di rilascio, vengono presentate nuove versioni ogni sei-nove mesi.
La politica della casa californiana predilige aggiornamenti di natura in genere incrementale, apportando miglioramenti del software a intervalli regolari, piuttosto che revisioni complete del sistema ogni due o tre anni.
Tra una major release e l'altra, ovviamente, vengono comunque messi a disposizione rilasci intermedi per risolvere problemi di sicurezza e altri bug del software.
La maggior parte dei dispositivi Android é in grado di ricevere gli aggiornamenti in modalitá *OTA* (over the air), ovvero senza necessitá di un collegamento concreto ad un PC.

Dalla versione 1.0 (denominata Apple Pie) del 2008 si sono susseguite numerose release garantendo sempre la retrocompatibilitá con le precedenti.
Una panoramica dello storico delle versioni é la seguente:

- Android 1.0 **A**pple Pie o **A**lpha [23 settembre del 2008]
- Android 1.1 **B**anana Bread o **B**eta [9 febbraio 2009]
- Android 1.5 **C**upCake [13 aprile 2009]
- Android 1.6 **D**onut [16 settembre 2009]
- Android 2.0/2.1 **E**clair [27 ottobre 2009]
- Android 2.2 **F**royo [20 maggio 2010]

- Android 2.3 **G**inger Bread [7 dicembre 2010]
- Android 3.0/3.1/3.2 **H**oneycomb [gennaio del 2011]
- Android 4.0 **I**ce Cream Sandwich [19 ottobre 2011]
- Android 4.1 **J**elly Bean [13 novembre 2012]
- Android 4.4 **K**it Kat [3 settembre 2013]
- Android 5.0 **L**ollipop [Non ancora realizzato, solo annunciato]

É curioso notare come il nome di ogni versione segua un ordine alfabetico.

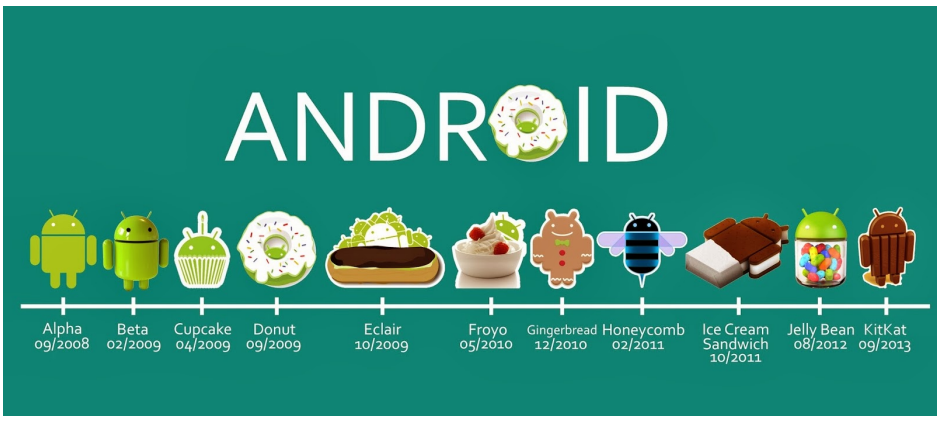

Figura 1.2: Versioni Android dal 2008 al 2013

1.3 Concetti fondamentali

Questo capitolo vuole presentare i componenti architetturali utili alla comprensione del ciclo di vita di un'applicazione Android e gli strumenti a disposizione del programmatore in fase di sviluppo.
Di seguito non verrano descritte le fasi di realizzazione di un'applicazione, ne tanto meno i principi della programmazione orientata agli oggetti (principio cardine del linguaggio di programmazione JAVA), anche se di vitale importanza per chiunque si volesse accingere allo sviluppo di tali applicazioni, in quanto questi argomenti esulano dall'argomento di questa tesi.

I componenti fondamentali di un'applicazione Android sono:

- Activity
- Service
- Content Provider
- Broadcast Receiver
- Manifest

1.3.1 Activity

Ogni schermata dell'applicazione é detta *Activity*, ed ha lo scopo principale di fornire un'interfaccia tra l'utente e l'applicazione.
Un'applicazione puó avere, e nella maggior parte dei casi effettivamente ha, piú di un'Activity ognuna indipendente dalle altre.
Un'esempio di Activity per un'applicazione di gestione degli SMS, potrá ad esempio avere un'Activity per la lettura dei messaggi ricevuti ed una per la creazione di un nuovo messaggio

Ognuna di esse viene creata come sottoclasse della super classe Activity, la quale definisce una serie di eventi che governano il ciclo di vita di un'Activity.
Questi eventi sono, ad esempio:

- onCreate() - chiamato quando l'activity viene lanciata per la prima volta
- onStart() - chiamato quando l'activity diventa visibile all'utente
- onResume() - chiamato quando l'activity inizia ad interagire con l'utente

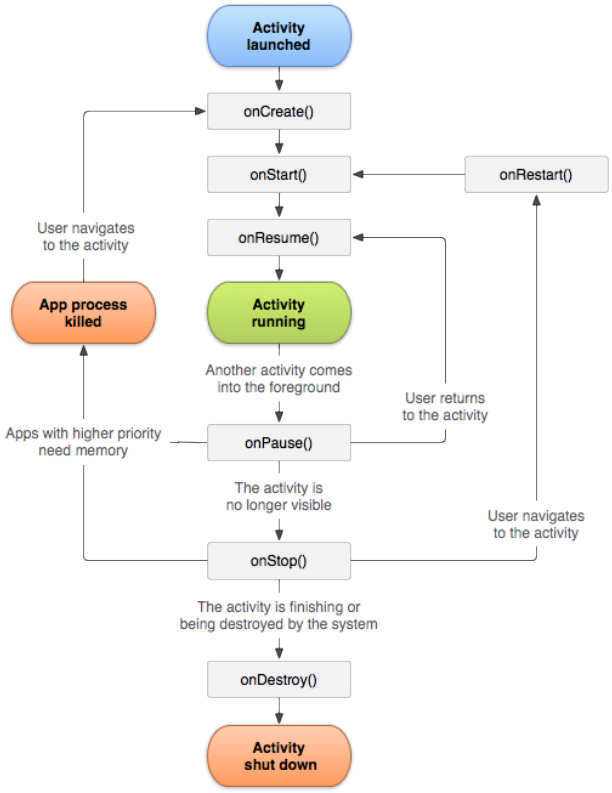

Figura 1.3: Ciclo di vita di un'Activity

- onPause() - chiamato quando l'attuale activity viene messa in pausa e un'Activity precedente viene ripristinata
- onStop() - chiamato quando l'activity non é piú visibile all'utente
- onDestroy() - chiamato prima che l'activity venga distrutta (manualmente o dal sistema operativo per liberare memoria)
- onRestart() - chiamato dopo che l'activity era stata stoppata e quando é pronta ad essere ripristinata

1.3.2 Service

Un **service** é un componente utile per svolgere lavori di durata lunga (o addirittura indeterminata) in background rispetto all'Activity.

Un Service, infatti, non offre alcuna interfaccia utente, ed effettua operazioni in maniera completamente trasparente per l'utente.
Classici esempi di service sono quelli relativi all'esecuzione di musica o al download di dati mentre l'utente usa altre app sul dispositivo.

Esistono due tipi di service, a seconda del modo in cui vengono eseguiti: startedService e BoundService.

Gli **startedService** vengono eseguiti in background indipendentemente, anche se la componente che li ha avviati viene terminata.
Generalmente non offrono interazione con il chiamante e proseguono finché non vengono interrotti o si auto-interrompono con i metodi predisposti.
Questo tipo di service é da prediligere per operazioni con una loro finalitá indipendente dallo stato delle altre applicazioni come aggiornamenti dati, scaricamento di file o immagini, sincronizzazione remota verso server esterni, ecc...

I **BoundService** sono simili ai service Started con l'eccezione che consentono al componente chiamante di interagire e ricevere risultati.
I service Bound hanno senso solo se qualche altra componente vi si collega e vengono interrotti nel momento in cui non vi sono piú client ad essi collegati.

Un service viene creato come sottoclasse della classe Service ereditando da essa una serie di eventi che ne governano il ciclo di vita :

- onCreate() - Metodo usato per inizializzare la classe (sia per startedService che per BoundService)

- onStartCommand() - Metodo usato per attivare il Service da un altro componente, qui vengono svolti i compiti deputati al Service(solo per StartedService)

- onBind() - Corrispondente del metodo onStartCommand() per i BoundService che restituisce un'interfaccia di tipo IBinder, che rappresenta un protocollo di comunicazione con il service a cui abbiamo effettuato un'associazione

- onUnbind() - Metodo chiamato quando il service viene scollegato (nota:un servizio é distrutto dal sistema se e solo se tutti i componenti associati ad esso sono stati scollegati)

- onDestroy() - Metodo chiamato quando il service viene distrutto

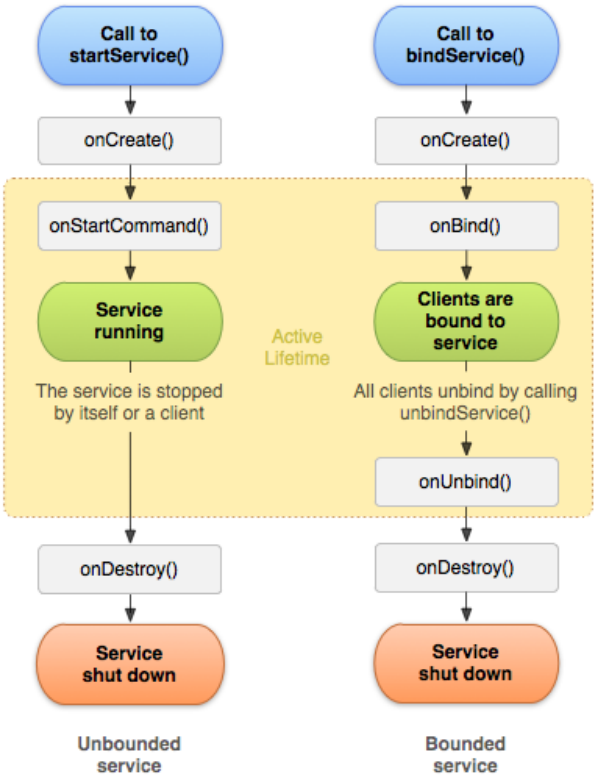

Figura 1.4: Ciclo di vita di un Service

1.3.3 Content Provider

Un content provider é il componente che permette a diverse applicazioni Android di condividere dati.
Grazie a questo componente l'applicazione é in grado di gestire un insieme di dati (memorizzati nel file system, in un database SQLite, ecc..).
Il content provider permette inoltre ad altre app di interagire con i dati che gestisce: ad esempio, esiste un content provider delegato alla gestione delle informazioni relative ai contatti dell'utente che puó essere interrogato da altre app.
Il suo funzionamento puó essere comparato a quello di una directory condivisa.

1.3.4 Intents Broadcast e Broadcast Receiver

Un broadcast receiver é un componente che si attiva in risposta ai messaggi inviati in broadcast. Messaggi di broadcast sono ad esempio :lo spegnimento dello schermo, un basso livello di batteria o la ricezione di un messaggio, una chiamata in entrata.
Questi specifici messaggi sono chiamati *Intents Broadcast*.

Gli Intents Broadcast sono un particolare tipo di Intent che é possibile spedire attraverso il metodo sendBroadcast(). Vengono utilizzati per notificare alle applicazioni del sistema che sono in ascolto, determinati eventi, in modo che possano reagire.
Un broadcast receiver permette all'applicazione di restare in ascolto di eventuali messaggi tramite l'utilizzo del metodo onReceive().

1.3.5 Android Manifest

Android Manifest é un file XML che contiene informazioni dettagliate riguardo l'applicazione come ad esempio il nome del package, la versione dell'applicazione, la versione minima del SDK, l'icona dell'applicazione sul

dispositivo, il nome dell'applicazione, il nome delle Activity presenti nell'applicazione.
Nel file vengono inoltre specificati i permessi che l'applicazione richiede per la sua esecuzione.
Alcuni esempi di permessi sono:

READ_CONTACTS: leggere i contatti dell'utente dalla rubrica.
INTERNET: accedere ed utilizzare la connessione Internet.
WRITE_EXTERNAL_STORAGE: scrivere sulla memoria esterna.
ACCESS_FINE_LOCATION: utilizzare il sistema di localizzazione GPS.

1.4 Struttura di un progetto Android

Un progetto di un applicativo Android richiede tre componenti fondamentali per poter funzionare:

- Il file AndroidManifest.xml, descritto precedentemente.
- La cartella 'src' che contiene il codice sorgente
- La cartella 'resource' che contiene tutte le risorse necessarie all'applicazione per funzionare correttamente.

Una risorsa puó essere un'immagine, un documento XML, oppure un file binario.I file vengono automaticamente disposti, in base al tipo di risorsa, in una cartella specifica.

Ad esempio i file immagine (GIF, PNG, JPEG) vengono salvati nelle directory *drawable-xxhdpi, drawable-xhdpi, drawable-hdpi, drawable-ldpi, drawable-mdpi* in base alla densitá schermo del dispositivo.
I file xml che definiscono variabili e valori (che possono essere stringhe, interi

e tante altre tipologie di dati), invece, vengono memorizzati nella directory *values*.
Un'altra directory importante é *layout*: essa contiene documenti XML che definiscono i componenti GUI (layout oppure strumenti grafici come Button, EditText etc..) che vengono utilizzati nell'applicazione.

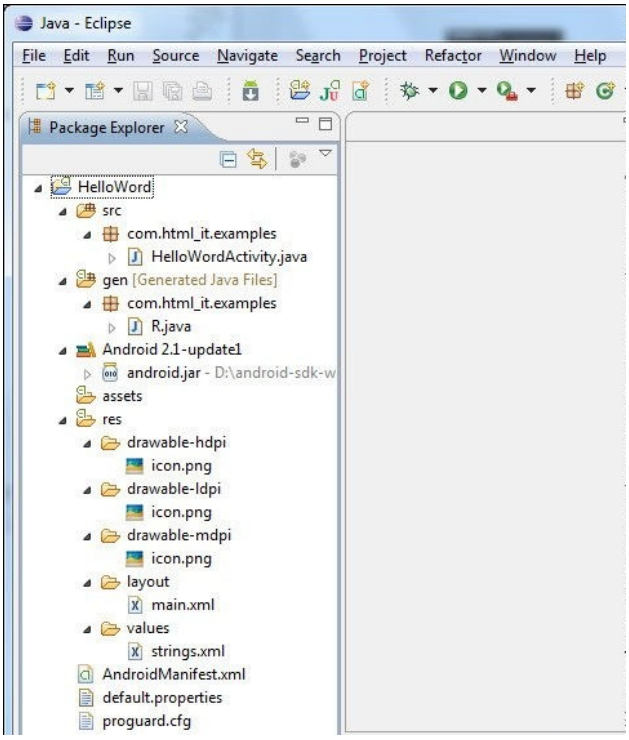

Figura 1.5: Struttura di un'applicazione Android

1.5 Altri sistemi operativi

In questa sezione verranno descritti i principali sistemi operativi per piattaforme mobili presenti sul mercato;
Lo scopo é quello di presentare le principali alternative al sistema scelto nello sviluppo di questo progetto senza un approfondimento dettagliato in quanto estraneo agli obiettivi di questo documento.

Nello specifico verranno presentati i sistemi piú diffusi ed utilizzati, oltre ad Android al momento della stesura di questa tesi.
Secondo i dati pubblicati da IDC(International Data Corporation) relativo al secondo quarto del 2014, infatti, piú del 90% del mercato mondiale degli smartphone utilizza: Android, iOS, Windows Phone.[9]

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
Q2 2014	84.7%	11.7%	2.5%	0.5%	0.7%
Q2 2013	79.6%	13.0%	3.4%	2.8%	1.2%
Q2 2012	69.3%	16.6%	3.1%	4.9%	6.1%
Q2 2011	36.1%	18.3%	1.2%	13.6%	30.8%

Source: IDC, 2014 Q2

Figura 1.6: Diffusione sistemi operativi 2014-Q2, Fonte IDC

1.5.1 iOS

iOS (precedentemente iPhone OS) é un sistema operativo sviluppato da *Apple Inc.* per iPhone, iPod touch e iPad.
iOS, come il suo 'fratello maggiore' Mac OS X, é una derivazione di UNIX e usa un micro-kernel XNU Mach basato sul sistema operativo Dar-win OS.
XNU Mach é basato sull'unione del micro-kernel Mach e del kernel FreeBSD.
Il micro-kernel Mach é stato sviluppato dalla Carnegie Mellon University tra il 1985 e il 1994 ed é stato uno dei primi micro-kernel realizzati.
FreeBSD é stato sviluppato a Berkeley dal 1993 ed é uno dei piú diffusi in ambiti di ricerca, server, in apparecchi embedded.
In particolare le funzioni primitive e i servizi fondamentali del kernel XNU (gestore della memoria, della comunicazione tra processi e del sistema input/output, ecc...) si basano su micro-kernel Mach, mentre altre operazioni (es:gestione utenti e i permessi, stack di rete, ecc...) sfruttano FreeBSD.
Il risultato é un sistema operativo performante e leggero (meno di mezzo gigabyte):perfetto per la piattaforma dei dispositivi mobili Apple.
Attualmente é in circolazione la versione iOS7, ma é stata annunciata per

l'autunno 2014 la versione 8. L'architettura del sistema iOS puó essere schematizzata come un sistema a quattro livelli:

1. Core OS layer: livello piú basso, contiene funzioni per input/output, gestione processi, gestione della memoria, ecc...
2. Core Services layer: servizi proprietarie Apple fondamentali del sistema operativo.
3. Media layer : strumenti per la gestione dell'audio, gestione del video, grafica 2D e 3D, ecc..
4. Cocoa Touch layer :contiene gli strumenti necessari per la creazione di applicazioni iOS, tra i quali:
    - UIKit framework per l'interazione con l'utente: la visualizzazione, gestione del multitouch, ecc..
    - Foundation framework:gestione dei dati, ecc...
    - Address Book UI Framework
    - Game Kit Framework
    - Map Kit Framework
    - Message UI Framework
    - ...

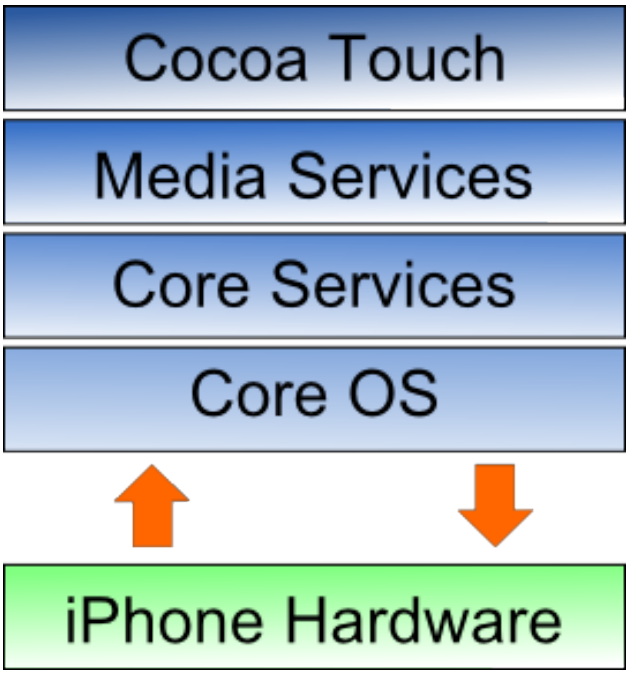

Figura 1.7: Architettura iOS

1.5.2 Windows phone

Windows Phone é un sistema operativo proprietario sviluppato da *Microsoft Inc.* finalizzato all'utilizzo su dispostivi mobili.
Seppur di recente sviluppo (rilasciato nel 2010) il sistema operativo di Microsoft vanta giá numerosi utenti ed é in costante crescita, grazie anche alla partnership con la casa finlandese Nokia (nota casa produttrice di dispositivi mobili).

La schermata principale, chiamata Schermata Start, é sicuramente una delle caratteristiche che lo contraddistingue a prima vista dagli altri sistemi operativi; essa é, infatti, composta dalle cosidette *Live Tiles* (letteralmente piastrelle dinamiche) che sono collegamenti ad applicazioni, contatti, pagine internet a forma quadrangolare che l'utente puó aggiungere, modificare, spostare o eliminare liberamente.
Le tiles appaiono dinamiche grazie ad effetti grafici e all'aggiornamento in

tempo reale (per esempio, la tile di una e-mail mostrerá il numero di messaggi non letti).

Da un punto di vista piú tecnico i sistemi Windows per dispositivi mobili hanno subito un'importante upgrade con la versione 8.
L'attuale Windows Phone 8 sostituisce l'architettura basata sul sistema operativo basato su Windows CE(Windows Embedded Compact), sviluppato esclusivamente per sistemi embedded, con il kernel Windows NT, lo stesso dei sistemi desktop.
Questo importante cambiamento porta numerosi vantaggi permettendo ai dispostivi mobili di supportare processori multi-core, la crittografia, i dispositivi di storage removibile con schede microSD, oltre all'utilizzo di linguaggi di programmazione standard come il C++.
Inoltre la nuova versione utilizza multitasking, permettendo agli sviluppatori di creare applicazioni che lavorano in background e riprendono immediatamente. L'altra faccia della medaglia di un cambiamento cosí profondo, peró, porta problemi di retrocompatibilitá: gli attuali dispositivi Windows Phone 7, infatti, non possono implementare o aggiornarsi alla versione successiva.Inoltre le nuove applicazioni create specificatamente per la versione 8 non saranno disponibili per quelle precedenti.
Gli sviluppatori potranno comunque rendere le applicazioni disponibili per entrambe le piattaforme tramite l'SDK appropriato.

Attualmente Windows Phone 8 é supportato e implementato in numerosi dispostivi delle case Nokia e Huawei, e anche in qualche smartphone marcato Samsung.

Capitolo 2

Related works

2.1 Android: non solo smartphone

Fino a questo punto abbiamo trattato e discusso del sistema operativo Android sempre nell'ottica di smartphone e al piú di tablet.
In realtá il lavoro svolto dalla casa madre Google é assai piú esteso e capillare, tanto che ad oggi il loro sistema operativo sembra essere sempre piú onnipresente Come giá detto uno dei punti chiave del successo planetario Android risiede nel fatto che il sistema operativo viene rilasciato con licenza Apache.Questo ha permesso agli utenti, piú o meno esperti, di poterlo adattare per i propri interessi e per specifici scopi.
Gli sviluppi e le innovazioni piú sorprendenti nascono, peró, dalla stessa casa madre, che non smette mai di immettere nel mercato nuovi prodotti, servizi o gadget che supportano la tecnologia Android fornendo, cosí, costantemente nuovi piattaforme per gli utenti e spronando indirettamente gli sviluppatori a creare nuovi applicativi sempre piú sofisticati.
Di seguito verrá presentata una rassegna delle tecnologie piú moderne e degli attuali settori di interesse scientifico.

2.1.1 Android Wear

Le tecnologie indossabili sono sono la prossima grande frontiera per le cosiddette smart-tecnology, e il 18 Marzo 2014 Google ha annunciato la sua partecipazione presentando Android Wear. Come é facile evincere dal nome si tratta di un'iniziativa che utilizza il sistema operativo Android e un pacchetto di sviluppo software dedicato per entrare a pieno regime nel mondo di queste nuove tecnologie. Gli accordi finanziari presi da Google con le case produttrici Motorola e LG per la produzioni di periferiche adatte sembra essere il segnale di un concreto segnale della direzione dei campi di utilizzo del sistema operativo nei prossimi anni.

2.1.2 Android Tv

Android tv é una piccola periferica hardware dotato di microprocessore e di sistema operativo Android che aggiunge varie funzioni ad un qualsiasi TV dotato di presa HDMI. In gergo informatico questa tipologia di periferica che una volta collegata ad un personal computer, un televisore o qualsiasi interfaccia elettronica viene detta 'dongle'. L'idea alla base di questo progetto é quella di portare il web ed i suoi contenuti sulla TV, tramite il sistema operativo Android, ma opportunamente modificato nell'interfaccia, dato che il tipo di mezzo é profondamente diverso da uno smartphone o un tablet.

2.1.3 Google Glass

Nell'estate del 2012 Google ha annunciato una delle innovazioni tecnologiche piú controverse degli ultimi anni: i Google Glass. Il Project Glass, questo é il nome ufficiale del progetto, ha l'obbiettivo di mettere in produzione su larga scala un paio di occhiali, basati su una versione specifica del sistema operativo Android, che permetterá a chi li indossa di avere davanti ai propri occhi, con una proiezione sulla retina, una realtá piú completa, una alla realtá aumentata.

Per realtá aumentata si intende la percezione sensoriale umana arricchita da informazioni, di solito manipolate attraverso la tecnologia elettronica, che non sono normalmente percepibili attraverso i soli cinque sensi. Lo sviluppo del progetto renderá possibile visionare, informazioni direttamente dalla rete riguardanti le nostre amicizie sui vari social network, riguardo la realtá che ci circonda come previsioni meteo eventuali informazioni su mezzi pubblici o traffico, indicazioni stradali grazie ai servizi google maps, e tante altre informazioni.
Una tecnologia cosí pervasiva ha prodotto numerose accesse discussioni e controversie sopratutto riguardanti tematiche di violazione della privacy, tanto da spingere sette garanti della privacy da tutto il mondo a chiedere importanti delucidazioni su lo sviluppo e l'utilizzo finale di questa periferica. Tra le preoccupazioni di Scott Hutchinson e gli altri firmatari di questa richiesta sembra esserci la possbilitá di un utilizzo mirato anche al riconoscimento facciale degli individui, senza un precedente loro assenso.
La risposta di Google non si é fatta attendere dichiarando che saranno vitate applicazioni di quel tipo nei loro sistemi.

2.1.4 Android Auto

Uno rapporto pubblicato dalla Texas AM ha dimostrato che il tempo passato in auto ogni anno varia dalle 400 alle 500 ore, ovvero almeno 2 ore e mezza al giorno.[3]
Non sorprenderá dunque che una recente attivitá di ricerca della casa di Cupertino si sia occupata dello sviluppo di un sistema, basato su Android, che si potesse integrare nel sistema di un'autovettura.
Il risultato ha portato allo sviluppo di una piattaforma pensata da Google e 40 partner per sincronizzare il proprio smartphone Android con lo schermo al centro della plancia.
Tutto ruota intorno al navigatore: il sistema di navigazione basato su Google Maps é stato riprogrammato per funzionare con i comandi vocali e rispon-

dere a richieste aggiuntive come gli orari di apertura di un locale verso cui navighiamo, la ricerca di punti di interesse o del prossimo appuntamento in calendario.

2.1.5 Casi di studio

Di seguito vengono presentati alcuni studi per ribadire l'estrema versatilitá del sistema operativo Android.
Ho scelto di descrivere due studi che dimostrano come l'utilizzo del sistema operativo della casa di Cupertino possa essere utilizzato ed implementato in progetti e tematiche del tutto sorprendenti.
Il primo trattato analizzato, infatti, dimostra un come l'utilizzo della tecnologia Google Glass possa essere applicata in ambito medico; il secondo, invece, descrive l'applicazione dello stesso sistema operativo unitamente ad una piattaforma hardware apposita nel campo del monitoraggio delle acquacolture.

Immunochromatographic Diagnostic Test Analysis Using Google Glass

I ricercatori Steve Feng, Romain Caire, Bingen Cortazar, Mehmet Turan, Andrew Wong, e Aydogan Ozcan con la collaborazione di:Electrical Engineering Department, Bioengineering Department, California NanoSystems Institute and Department of Surgery, David Geffen School of Medicinee University of California, nel loro studio *'Immunochromatographic Diagnostic Test Analysis Using Google Glass'* sono riusciti a dimostrare che é possibile utilizzare gli occhiali Google Glass per eseguire a lettura e l'analisi qualitativa dei cosiddetti 'test di diagnosi rapida' (RDT).[12]
I test per la diagnosi rapida rappresentano un modo semplice, veloce ed affidabile per la diagnosi tempestiva di molte condizioni cliniche. La caratteristica dei test é appunto, la rapiditá, infatti, al contrario degli altri esami

di laboratorio che richiedono fino a diversi giorni, i risultati si hanno al massimo dopo un'ora. Questo é particolarmente importante nei casi di sindromi che possono aggravarsi molto rapidamente, ed é anche vantaggioso per tutte quelle analisi che prevedono controlli frequenti. I ricercatori hanno utilizzando un'applicazione appositamente scritta ed una serie di RDT con sopra stampato un codice QR in grado di contenere i valori dell'analisi.
I codici vengono letti utilizzando la fotocamera integrata negli occhiali, trasmessi in formato digitale ad un server remoto dove vengono elaborati automaticamente.
I risultati della diagnosi vengono successivamente ritrasmessi ai Google Glass oltre che ad un server centrale che fornisce una rassegna basata anche sulla geolocalizzazione dell'analisi effettuata.

Gli autori di questa ricerca concludono affermando che la possibilitá di avere un monitoraggio in tempo-spazio reali delle varie malattie e condizioni mediche offerta da questa applicazione, potrebbe essere uno strumento molto prezioso per l'epidemiologia, applicazioni mobili sanitarie e di telemedicina.

Design of Remote Monitoring System for Aquaculture Cages Based on 3G Networks and ARM-Android Embedded System

La ricerca condotta da Yanle Wang, Changsong Qi e Hongjun Pan del College of Mathematics, Physics and Information, Zhejiang Ocean University, intitolata *'Design of Remote Monitoring System for Aquaculture Cages Based on 3G Networks and ARM-Android Embedded System'* porta un altro interessante esempio della versatilitá del sistema operativo Android.[29]
Nel loro lavoro gli studiosi hanno descritto l'implementazione di un sistema per il monitoraggio ed il controllo di zone di acquacoltura, ovvero di quelle zone adibite all'allevamento di organismi acquatici come pesci, crostacei, molluschi, ecc...
Utilizzando la piattaforma hardware Samsung ARM11 S3C6410 supportato dal sistema operativo Android, sia come terminale per la rilevazione dei dati

nella gabbia di acquacoltura (come ad esempio temperatura, sale e concentrazione di ossigeno nell'acqua di mare fino la velocitá e all'intesitá delle onde sulla superficie dell'acqua), sia come terminale portatile per il controllo dei dati rilevati a distanza.
I dati raccolti vengono, infatti, inviati ad una postazione mobile sfruttando la rete 3G.
I risultati portano i ricercatori ad affermare che l'interfaccia da loro creata non si limita all'utilizzo solo alle gabbie di acquacoltura, ma anche per esempio, al trasporto prodotti acquatici.
L'uso di questo sistema invece dei sistemi manuali puó portare risparmi e benefici nella produttivitá.

2.2 Tematiche di sicurezza in Android

Chiunque si occupi di sicurezza informatica ha il compito di tenersi costantemente aggiornato sulle trasformazioni ed innovazioni che l'introduzione di nuove tecnologie portano con se.
La rivoluzione tecnologica a cui stiamo assistendo in questi anni porta con se una panoramica di interessantissimi benefici tangibili ai piú, ma anche un insieme di problematiche spesso non visibili a molti.
In quest'ottica risulta difficile pensare che la diffusione planetaria e cosí capillare di smartphone contenenti molto spesso una mole importante di dati sensibili, o quantomeno privati, (numeri di telefono, contatti e-mail, sms, numero di carte di credito, ecc...) non porti con se anche lo svilupparsi di una vastissima comunitá di sviluppatori di codice maligno.

Il pensiero comune di un qualsiasi utilizzatore medio di smartphone é associare il concetto di malware o di programmi dannosi solamente ad una piattaforma desktop.Questo é il punto focale su cui fanno leva gli sviluppatori di codice maligno.

In realtá ogni sistema operativo che sia abbastanza diffuso avrá un numero piú o meno consistente di malware conosciuti.Tra questi non fa eccezione Android.
Esistono aziende ed enti, nel campo della sicurezza informatica, corrisposti proprio alla classificazione e alla nomenclatura di questi software maligni.
Tra questi spicca, per i temi di questo lavoro, il Malware Genome Project che colleziona e classifica i malware conosciuti per il sistema operativo Android.
Ad oggi il progetto conta piú di 1200 elementi dannosi classificati, numero che tende a crescere in modo allarmante: il 91, 3% dei nuovi malware riguarda il sistema operativo Android.(stima F-Secure)[11].

Seppur questi dati possano, a prima vista, suscitare una certa sorpresa, analizzando con piú attenzione la tecnologia sottostante risulta piú facile trovarne una spiegazione ragionevole.
In particolare, rispetto alla infrastruttura di un Personal Computer, uno smartphone risulta una realtá molto piú attraente per un cosidetto 'pirata informatico'.
Innanzitutto perché, come giá detto, uno smartphone generalmente contiene una mole di dati confidenziali sicuramente di gran lunga piú interessanti rispetto a quanto potrebbe essere conservato in un pc ad uso domestico.
Un altro fattore d'interesse importante per un cracker é la differenza di utilizzo di uno smartphone rispetto ad un computer: i cellulari infatti nascono con lo scopo di rimanere accesi e rintracciabili per anche 24 ore continuative, un lasso di tempo enorme per affondare un attacco.
In aggiunta, in questo periodo di tempo il device mobile risulta costantemente connesso alla rete;risulta, quindi, superfluo spiegare il vantaggio che questo porta ad un pirata informatico.
Se queste motivazioni non fossero sufficienti a convincere il lettore, si puó aggiungere l'estrema facilitá di recupero ed installazione di nuove app in un dispositivo mettendo a disposizione dei malware un terreno fertile di facile accesso.

2.2.1 Vulnerabilitá in Android

Il principio su cui si basa lo sviluppo di ogni 'buon' software maligno é quello di sfruttare falle di sicurezza e/o errori di programmazione nel sistema operativo:le cosiddette vulnerabilitá.
Su piattaforme mobili, esistono diverse categorie di minacce, a seconda del livello che contiene la vulnerabilitá sfruttata;queste possono trovarsi dal livello del sistema operativo, a quello di comunicazione di rete, dalle applicazioni alle impostazioni utente, ecc...

Root vulnerability

Esistono particolari malware che nascono con lo scopo di ottenere l'accesso al dispostivo con permessi di root, ovvero con i permessi e la possibilitá di fare qualunque cosa sul proprio smartphone.
Un esempio molto diffuso di utilizzo di malware di questo tipo é la necessitá di eliminare le personalizzazioni che le compagnie telefoniche spesso inseririscono vendendo gli smartphone in abbonamento. Esiste un lungo elenco di software che permettono di accedere al device come amministratore (in gergo 'rooting'), che differiscono sia per versione del sistema operativo che per il provider installato.

Vulnerabilitá del sistema

Le vulnerabilitá del sistema sfruttano gli errori presenti nei vari livelli dell'architettura del sistema operativo Android: kernel, Library Layer, Framework, ecc... Questa tipologia di vulnerabilitá viene combattuta con nuovi aggiornamenti o realease del sistema operativo da parte della casa madre Google.

Vulnerabilitá di applicazione

Questa tipologia di vulnerabilitá riguarda tutte quelle falle ottenute dalla presenza di errori nel codice delle applicazioni.
Gran parte delle app presenti per Android sono, infatti, sviluppate da programmatori amatoriali, non sempre attenti all'implementazione di codice sicuro.
Le tipologie di errori che é possibile riscontrare nelle applicazioni presenti sia nei market ufficiali che non, sono numerosissime e in un certo senso inutili da elencare.
Ovviamente, peró, si puó tracciare un elenco degli errori piú comuni nello sviluppo di un applicativo per Android.

- Connessioni non sicure: utilizzo improprio o assente di protocolli di sicurezza idonei come SSL/TLS
- Archiviazione non sicura: storage di dati senza l'utilizzo di algoritmi di hashing o di crittografia
- Log non sicuro: la visualizzazione del valore di alcune variabili tramite il log risulta molto utile in fase di debug, ma estremamente pericoloso una volta pubblicata l'app.Lo sviluppatore deve effettuare una vera e propria pulizia del codice prima del rilascio.

2.2.2 Sistemi di sicurezza in Android

Sandbox

Per contrastare soprattutto vulnerabilitá di applicazione e di rooting Android ha funzioni integrate nel sistema operativo che riducono significativamente la frequenza e l'impatto dei problemi di sicurezza dell'applicazione.
Tra queste é di particolare rilevanza il concetto di 'isolamento' delle app installate sul dispositivo (il cosidetto 'Sandboxing').
Ad ogni applicazione che gira sul dispositivo viene assegnato un ID univoco e costante durante tutto il suo ciclo di vita all'interno del dispositi-

vo;un'applicazione viene vista come un utente Linux a tutti gli effetti.
Questo principio permette innanzitutto ad Android di limitare i potenziali danni causati da difetti in una determinata applicazione.
Inoltre, proprio come un utente Linux, ogni applicazione dovrá richiedere permessi specifici per eseguire determinate azioni come ad esempio eseguire operazioni sull'interfaccia utente, memorizzare dati nella memoria interna o esterna, creare database, ecc...
In nessun caso peró potrá influenzare il comportamento delle altre applicazioni, né avere accesso ai loro dati.
Per poter accedere a risorse esterne alla loro sandbox (come ad esempio la rubrica telefonica, o la fotocamera), gli sviluppatori dovranno specificare, in maniera statica e persistente, nell'applicazione di quali risorse si ha la necessitá.
Queste dichiarazioni sono salvate in un file nel formato XML chiamato AndroidManifest.xml.
Il file conterrá quindi un elenco di attivitá, servizi, permessi, provider e altri componenti di cui l'applicazione necessita. Al momento Android mette a disposizione 130 permessi differenti.

Google Bouncer

In risposta alla sostanziale crescita della diffusione di malware sulla piattaforma Android, Google ha introdotto a Febbraio 2012 uno strumento di scansione automatica delle applicazioni pubblicate, denominato *Bouncer*.
La scansione di una nuova applicazione avviene seguendo 3 fasi:

1. analisi statica del codice per ricercare minacce conosciute
2. esecuzione del software in un emulatore virtuale, con annesso controllo di comportamenti sospetti
3. controllo dei dati immessi dai nuovi sviluppatori in fase di registrazione per evitare iscrizioni con dati falsi

Il sistema, ancora in fase primordiale, ha suscitato da subito l'attenzione di molti ricercatori in ambito di sicurezza informatica.
Una prima ricerca di spessore é quella presentata da Jon Oberheide e Charlie Miller con il nome *'Dissecting the Android Bouncer'* nel corso della *Summer-Con 2012* (tenutasi a New York City).[17]
La loro ricerca ha messo in luce alcuni importanti aspetti dell'architettura e sulle fasi di analisi del sistema.
Di particolare rilevanza sono anche gli studi effettuati da Nicholas J.Percoco e Sean Schultenel nel loro trattato ***Adventures in BouncerLand*** - *Failures of Automated Malware Detection within Mobile Application Markets* in cui hanno evidenziato difetti, falle e limiti del sistema.[23]

2.2.3 Casi di studio

Dissecting the Android Bouncer

La ricerca presentata da Jon Oberheide e Charlie Miller con il nome *'Dissecting the Android Bouncer'* ha permesso di mettere luce sull'architettura e sulle fasi di esecuzione nell'analisi di Bouncer.[17]
Nello specifico i risultati hanno portato le seguenti informazioni:

- L'analisi delle applicazioni avviene pochi minuti dopo l'invio della richiesta di pubblicazione e prima della pubblicazione effettiva
- L'analisi prevede una fase preliminare di controllostatico del file.apk alla ricerca di pattern sospetti
- Se il pacchetto supera l'analisi precedente, viene operata un'analisi dinamica eseguendo l'applicazione per cinque minuti circa in un emulatore configurato in modo tale che il software testato sia indotto a credere di essere eseguito su un dispositivo reale
- Per testare piú codice possibile viene simulata l'interazione con l'interfaccia utente dell'applicazione tramite tap casuali.

- L'emulatore ha un account Google associato (il cui indirizzo é miles.karlson@gmail.com)
- Nella rubrica di sistema sono stati inseriti dei contatti e sono presenti foto e file di testo all'interno delle memorie di massa alcuni dei quali con nomi di probabile interesse per un'applicazione malevola (come ad esempio 'password.txt').
- L'infrastruttura di Bouncer consente l'accesso delle applicazioni testate ad Internet da blocchi di rete prefissati
- É stato anche identificato il blocco di rete usato per la revisione manuale delle applicazioni

Adventures in BouncerLand - Failures of Automated Malware Detection within Mobile Application Markets

Gli studi di Nicholas J.Percoco e Sean Schultenel propongono un approccio mirato all'evasione dell'architettura del sistema Bouncer.[23]
Nello specifico i ricercatori hanno realizzato un'applicazione del tutto legittima, ma contenente malware appositamente inseriti che si attivavano con una frequenza di 15 minuti .
Per evitare il rilevamento automatico da parte del sistema gli autori hanno semplicemente aggiunto un banale costrutto condizionale che disabilita le funzionalitá malevole nel caso in cui l'indirizzo IP corrente appartenga allo spazio utilizzato da Bouncer(precedentemente rilevato e identificato con indirizzi 74.125.0.0/16 e 209.85.128.0/17). Per ovviare all'eventuale controllo manuale gli sviluppatori hanno codificato alcune delle funzionalitá (tra cui ovviamente anche quelle maligne) in JavaScript e fatto in modo che il programma caricasse dinamicamente tale codice da un server su Internet.
Superati i primi due ostacoli, Percoco e Schultenel si sono spinti oltre: hanno cioé proposto una versione aggiornata dell'applicazione richiedendo autorizzazioni aggiuntive mascherando questa richiesta con funzioni innocque che le sfruttassero.

Sia la prima immediata scansione di Bouncer che la seconda, avvenuta a distanza di qualche settimana, non hanno rilevato l'intenzione maliziosa del codice.
Neanche la successiva rimozione del controllo relativo all'indirizzo IP ha portato al risultato aspettato.
Solo riducendo l'intervallo di esecuzione del codice maligno, portandolo all'esecuzione con frequenza ad ogni secondo ha provocato un aumento considerevole di scansioni automatiche seguite da una revisione manuale e dalla disattivazione dell'account da sviluppatore dei due ricercatori.

Questi studi hanno portato alla luce alcuni aspetti non del tutto chiari e alcune importanti limitazioni nello strumento a disposizione di Google per contrastare la diffusione di malware nei dispostivi Android.

Sicuramente le armi piú potenti a disposizione degli utenti contro attacchi maligni restano, indipendentemente dalla tecnologia utilizzata, le norme di buonsenso e di buon utilizzo di sistemi e oggetti in grado di connetterci, nel bene e nel male, a tutto il mondo.

Security Enhanced (SE) Android: Bringing Flexible MAC to Android

Stephen Smalley e Robert Craig sono gli autori della ricerca denominata *'Security Enhanced (SE) Android: Bringing Flexible MAC to Android'*.[27] Nel loro trattato documentano come utilizzare il modulo di sicurezza Security-Enhanced Linux (SELinux) nei dispositivi Android dimostrando i benefici che tale approccio di sicurezza porta nel sistema. Il SELinux é un insieme di strumenti del kernel Linux utilizzati per la gestione ed i controlli di sicurezza,tra i quali il sistema MAC (Mandatory Access Control).
Quest'ultimo permette al sistema operativo di limitare la capacitá di un soggetto(processo or thread) di accedere o generalmente eseguire qualche tipo di operazione su un oggetto (files, directories, porte TCP/UDP, periferiche

IO, ecc...).
Nella loro documento i ricercatori illustrano il principali ostacoli trovati e le soluzioni adottate per implementare il modulo SELinux nella piattaforma Android.
Il primo ostacolo consiste nell'abilitare il modulo di sicurezza nel kernel Linux del dispositivo: il problema principale risiede nel fatto che SELinux richiede che il filesystem fornisca un supporto per i marcatori di sicurezza,ma il tipo di filesystem originariamente scelto per Android (yaffs2) non prevede tale supporto.
I ricercatori sono riusciti ad implementare i requisiti per associare gli attributi necessari alla sicurezza sfruttando il supporto per attributi estesi aggiunto nelle ultime versioni di yaffs2.
Nei dispositivi piú recenti invece il problema non viene piú riscontrato in quanto usano il filesystem ext4 che supporta tutti gli attributi necessari ed il supporto per i marcatori di sicurezza.
Il secondo problema riscontrato dai ricercatori é la sostanziale di tutti i livelli soprastanti al kernel rispetto aduna qualsiasi distribuzione Linux. Di conseguenza nessuno dei lavori di integrazione giá sviluppati potevano essere riutilizzati.
Il SELinux prevede inoltre un uso esteso di system call per poter ottenere e settare le informazioni di sicurezza,non presenti nelle librerie del kernel di Android.
Gli autori di questa ricerca hanno perció esteso la libreria C Android, chiamata bionic, per coprire tutte le chiamate di sistema necessarie al modulo di sicurezza.
Il trattato procede mostrando una serie di casi di studio atti a dimostrare l'impatto dell'implementazione del modulo SELinux su le vulnerabilitá delle applicazioni Android.
In conclusione viene quindi dimostrato l'impatto benefico dell'utilizzo di questo modulo di sicurezza sui dispositivi Android, senza che questa implementazioni porti un effetto tangibile sulle prestazioni del dispositivo.

Capitolo 3

Tecnologie integrate: PayPal e GCM

Lo sviluppo dell'applicazione oggetto di questa tesi ha comportato l'utilizzo di alcuni servizi esterni come PayPal e GCM (GoogleCloudMessaging) per la gestione di un protocollo di pagamento sicuro e per il controllo di un sistema di notifiche push rispettivamente.
La scelta di PayPal per l'integrazione di un meccanismo di pagamento é dettata dai grandi numeri e dalle forti certezze in ambito di sicurezza che il sistema offre.
La scelta di GoogleCloudMessaging (GCM) come sistema di gestione dell'invio di notifiche lato server, nasce dalla personale fiducia nella soliditá del gruppo Google oltre che da un'estesa documentazione presente nella rete.

Di seguito vengono presentati ed esposti i servizi appena citati.

3.1 PayPal

3.1.1 Introduzione a PayPal

Paypal é una societá americana, nata nel 1998 sotto il nome di *Confinity*, per offrire servizi di trasferimento di denaro tramite internet.

É possibile aprire il proprio conto PayPal semplicemente registrandosi gratuitamente associando una o piú carte di credito (per un massimo di otto), o carte prepagate.
L'idea innovativa di questo sistema é la possibilitá di effettuare transazioni di denaro senza dover condividere i dati delle carte associate al conto.
Oltre a ció la societá ha sviluppato un sistema proprietario di prevenzione frodi considerato uno dei piú sicuri al mondo.

3.1.2 Un po' di storia

La societá nasce dall'incontro da tra Peter Thiel(imprenditore e venture capitalist) e Max Levchin(esperto programmatore) ed Elon Musk (cofondatore di X.com, una compagnia di servizi finanziari online e di pagamenti via e-mail) nell'universitá di Standford.
Il successo é quasi immediato, grazie anche ai fondi stanziati da importanti societá come Nokia Ventures e Deutsche Bank che credono da subito nel progetto.
La rapida ascesa continua nel 2002, quando Paypal viene quotata in borsa e acquisita, pochi mesi dopo, da *eBay Inc.* uno dei siti di compra-vendita online piú grandi al mondo.
Successivamente Paypal si espande nel mondo, offrendo:
-servizi in 25 valute (dal dollaro americano, al baht tailandese);
-una serie di servizi accessori
-la possibilitá di inviare pagamenti verso altri conti Paypal semplicemente con l'invio di un'email.

Paypal é leader nel settore dei pagamenti online tanto che nel 2013 ha fatturato 6, 6 miliardi di dollari, coprendo il 41% del fatturato di eBay Inc. nello stesso anno.
Ad oggi (secondo trimestre 2014) PayPal ha 152 milioni di account attivi

e permette pagamenti in piú di 100 valute differenti e gestisce 7,001 $ ogni secondo, con un ricavo di 1,95 miliardi dollari, in crescita del 20% .[1]

3.1.3 PayPal e sicurezza

Il successo e la diffusione di PayPal sono dovuti anche, e sopratutto, al livello di sicurezza che il sistema é in grado di offrire all'utente.
Il primo grado di sicurezza deriva direttamente dalla struttura e dal funzionamento insito di PayPal.Il sistema, infatti, funge da intermediario tra il mittente e il destinatario.
Il controllo della titolaritá tra l'intestatario del conto PayPal e quello sulla carta di credito fornita in fase di registrazione fornisce un altro importante tassello da aggiungere alla fiducia degli utenti verso questo servizio.
Gli utenti sono, infatti, tenuti a confermare tale verifica e in tal caso vengono evidenziati agli altri in modo opportuno.

Da un punto di vista prettamente tecnico l'infrastruttura PayPal utilizza connessioni criptate attraverso SSL 3.0 con chiave a 168bit, sia nelle fasi di gestione del proprio profilo utente che nelle fasi di comunicazioni vere e proprie.

PayPal inoltre mette a disposizione dell'utente e del commerciante un gruppo antifrode, attivo 24 ore su 24, composto da professionisti ed esperti di sicurezza telematica, tra cui anche ex agenti delle forze dell'ordine che vantano una notevole esperienza nella prevenzione delle frodi online.

3.1.4 PayPal nei sistemi mobili

Il gruppo PayPal non é rimasto di certo a guardare mentre l'evoluzione dei dispositivi con accesso alla rete cresceva e mutava, orientandosi sempre piú verso smartphone e tablet, rilasciando, nel 2013, i pacchetti di sviluppo

per Android e iOS. Il pacchetto PayPal mobile consente di creare applicazioni in grado di accettare pagamenti PayPal e carte di credito.
Come precedentemente illustrato esiste una versione per piattaforma Android e una per iOS.
PayPal mobile SDK fornisce una serie di API utili per integrare un servizio di pagamento built-in nelle applicazioni per smartphone, tra le quali: -identificazione client PayPal -Sistema di pagamento PayPal classico -Sistema di pagamento con carte di credito e credit card scanning -Riepilogo dell'ordine
Durante lo sviluppo dell'applicazione é possibile settare l'ambiente di sviluppo in modo da evitare movimenti di denaro reale (SANDBOX)

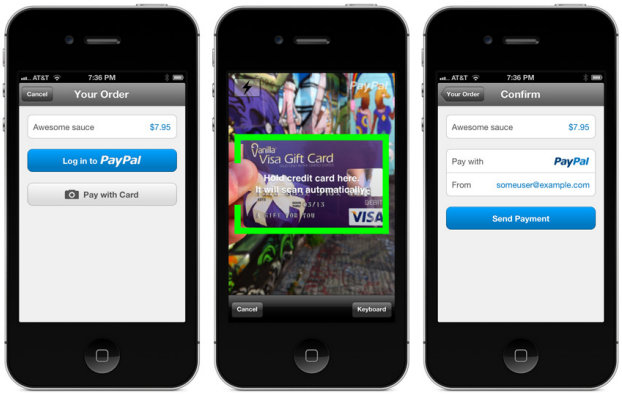

Figura 3.1: Implementazione del Pacchetto PayPal

3.2 GoogleCloudMessaging

Nello sviluppo di applicazioni mobili risulta utile, se non addirittura necessario, implementare un servizio di notifiche push.
Le notifiche push sono messaggi inviati in modo proattivo dai server alle applicazioni. Questi sono usate generalmente per segnalare, ad esempio, la visualizzazione delle ultime notizie, aggiornamenti dei social network, nuovi messaggi in entrata, richiesta di aggiornamenti.
Android supporta le notifiche push tramite specifici servizi cloud.

Google Cloud Messaging per Android (GCM) é il servizio che permette di inviare dati da un server ad un dispositivo Android
Il servizio GCM gestisce tutti gli aspetti di accodamento dei messaggi e della consegna all'applicazione di destinazione in esecuzione sul dispositivo Android .
Per inviare o ricevere messaggi, l'applicazione deve prima ottenere un ID di registrazione. L'ID di registrazione identifica il dispositivo e l'applicazione, e determina anche quali applicativi server possono inviare messaggi a questa applicazione.
Il processo di Google Cloud Messaging (GCM) segue alcuni semplici passaggi:

- L'app si registra per le notifiche push lanciando un Intent apposito insieme al numero di progetto
- GCM trasmette un REGISTRATION, con un ID registrazione, all'app che ha eseguito la sottoscrizione.
- Il server crea la notifica push contattando GCM e fornendo l'ID registrazione, la chiave API e un messaggio
- il servizio GCM archivia e inoltra il messaggio di notifica push all'app

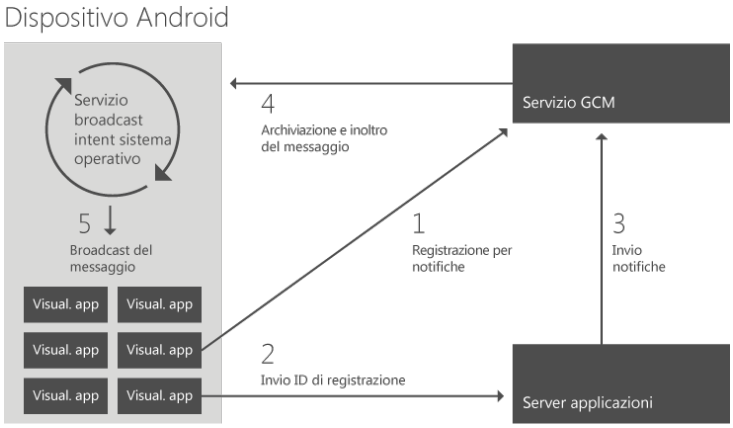

Figura 3.2: Flusso di esecuzione GCM

Capitolo 4

Dall'idea allo sviluppo: VERiSMARTpay

4.1 Introduzione

VERiSMARTpay é un'applicazione per dispositivi basati su piattaforma Android nata con l'idea di sviluppare uno strumento che rendesse piú facile gestire situazioni quotidiane di scambi di denaro tra amici e conoscenti.

L'applicazione si presenta come una novitá nel panorama degli applicativi per smartphone in quanto allo stato attuale i pagamenti su piattaforma si limitano, in prevalenza, ad una relazione

utente - azienda

utente - esercizio commerciale
utente - e-commerce
utente - professionista (P.IVA)

VERiSMARTpay si pone l'obiettivo di rendere facile e immediato lo scambio di denaro direttamente tra

utente $\Leftrightarrow$ *utente*

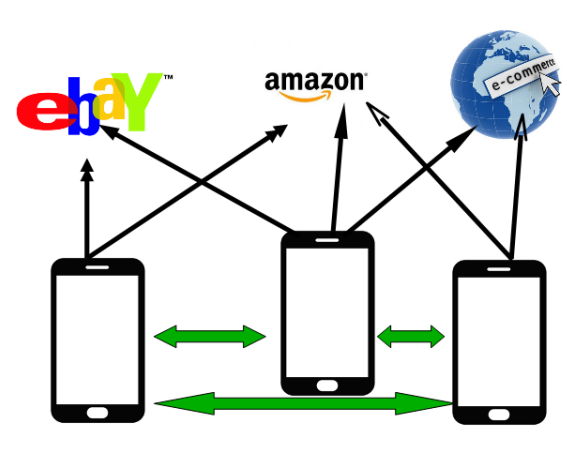

Figura 4.1: Schema di relazione pagamenti elettronici

Dall'idea iniziale di creare un vero e proprio portafoglio elettronico, il progetto si é successivamente esteso coprendo funzionalitá sempre piú evolute fino ad ottenere un'applicazione di gestione dei pagamenti ricca e completa .

L'applicazione, infatti, permette anche di :

- gestire le richieste ricevute e quelle inviate in un'interfaccia intuitiva e user-friendly
- creare una richiesta collettiva (collection) ed inviarla a piú contatti per richiedere denaro a piú persone per uno scopo comune (es: raccogliere una cifra per comprare un regalo ad un amico comune, dividere il conto di una cena, ...)
- partecipare a richieste pubbliche, a scopo benefico.
- gestire i crediti (e i debiti) tra i contatti, in modo da permettere di utilizzarli per saldare eventuali nuove richieste.
- concedere la possibilitá di pagamenti rateizzati di una richiesta creata

- comunicare con gli altri utenti tramite uno strumento di chat asscociato ad ogni richiesta ricevuta e/o inviata.
- Ricevere notifiche Push relative a informazioni riguardanti richieste ricevute o inviate
- Mandare promemoria di pagamento ai destinari di richieste di cui si é proprietari
- Creare una blacklist di utenti da cui non si vogliono ricevere richieste
- Creare e gestire gruppi di contatti per rendere la compilazione di nuove richieste piú rapida (es: coinquilini, compagni di calcetto, ...)

.. e molto altro ancora.

Le componenti principali per comprendere il funzionamento dell'applicazione sono principalmente tre:

- Le richieste
- Le richieste collettive
- Le offerte

Nel seguito verrano trattate e esposte singolarmente.

4.2 Architettura

4.2.1 Le richieste

Una richiesta di pagamento é lo strumento basilare mediante il quale gli utenti possono richiedere ed inoltrare pagamenti.
Ogni utente puó richiedere un pagamento ad una o piú persone.
Ogni utente destinatario diventa, se decide di accettare la richiesta, immediatamente debitore nei confronti di chi ha inviato la richiesta.
Un utente che riceve una Request potrá infatti decidere se:
-**Accettare** la richiesta e quindi, in seguito, pagare
-**Rifiutare** la richiesta

L'utente che decide di accettare una richiesta potrá comunque scegliere di rifiutarla in un secondo momento;
allo stesso modo un utente che inizialmente ha deciso di rifiutarla potrá decidere di accettarla successivamente.

Se la richiesta é stata accettata l'utente destinatario potrá in qualsiasi momento effettuare il pagamento,
altrimenti, se la richiesta é stata rifiutata, l'utente non potrá effettuare il pagamento, ma continuerá a ricevere le notifiche relative.

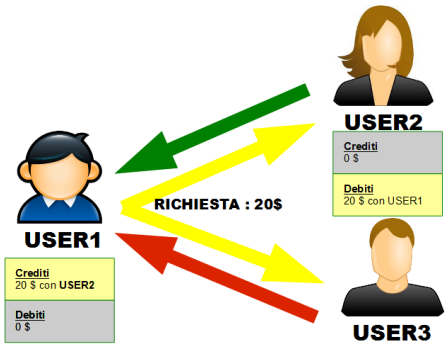

Figura 4.2: Funzionamento di una richiesta di pagamento

4.2.2 Le richieste collettive

Come detto in precedenza é possibile creare delle raccolte di pagamenti (dette Collection) tramite le quali é l'utente puó richiedere denaro a piú persone per uno scopo unico

L'idea alla base di questa funzionalitá é quella di permettere la gestione di tutti quei casi in cui lo scopo della richiesta di pagamento é lo stesso per molteplici destinatari.

Esempi di vita quotidiana in cui si verifica questa necessitá sono numerosi:

- raccogliere la partecipazione di vari amici per un regalo comune
- dividere il conto di una pizzeria
- gestire le spese tra condomini - condividere i costi di una vacanza -ecc...

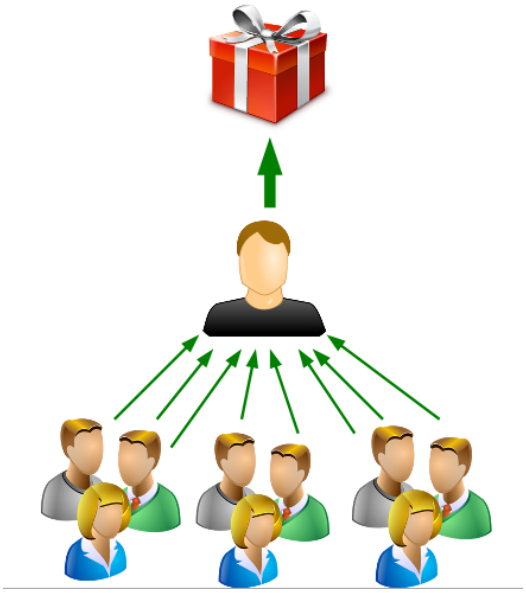

Figura 4.3: Richieste collettive

4.2.3 Le offerte

L' offerta é lo strumento messo a disposizione dell'utente destinatario per utilizzare eventuali crediti come moneta di scambio per il pagamento di nuove richieste.
Al momento del pagamento si potrá decidere, infatti, se pagare direttamente o patteggiare la richiesta annullando o decrementando eventuali crediti che si hanno con il proprietario della richiesta ricevuta.
Il proprietario riceverá, quindi, l'offerta e potrá decidere se accettare lo scambio o meno.

Uno schema puó facilitare la comprensione del funzionamento:

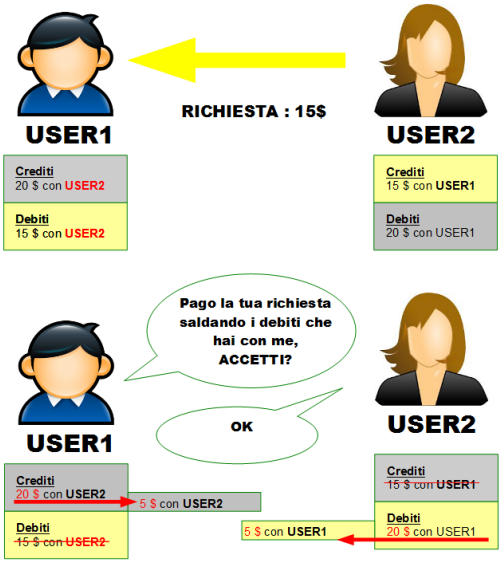

Figura 4.4: Schema di funzionamento delle offerte

4.2.4 Le richieste pubbliche

Le richieste pubbliche sono un particolare tipo di richiesta.
Ció che le contraddistingue dalle richieste semplici é in primo luogo lo scopo per cui vengono create:le richieste pubbliche, infatti, nascono per raccogliere somme di denaro per scopi benefici.
Un altro carattere distintivo é che queste richieste saranno visibili a chiunque sia in possesso dell'applicazione.
Esistono essenzialmente due tipi di richieste pubbliche:

- Richieste pubbliche ad offerta libera: ogni utente puó decidere l'importo da donare
- Richieste pubbliche ad offerta fissa: l'importo richiesto é fissato a priori

Entrambe le tipologie, comunque, seguono la filosofia 'reach a goal': ovvero si pone un'importo da raggiungere tra tutti gli utenti. L'idea é tutt'ora in fase di sviluppo, ed al momento non é possibile creare direttamente una richiesta pubblica a livello di applicazione, in quanto é necessario sviluppare un sistema di controllo per eventuali usi fraudolenti del servizio.

4.3 Interfaccia utente

4.3.1 MainActivity

L'applicazione si presenta all'utente con una grafica ricca ed user-friendly Si compone di cinque livelli:

1. Livello per la creazione di una nuova richiesta
2. Livello per la visualizzazione delle richieste pubbliche
3. Livello per la gestione delle richieste ricevute
4. Livello per la gestione delle richieste inviate
5. Livello per la gestione dei settaggi

Successivamente verrano esposte le componenti grafiche e le funzioni di ognuno.

Figura 4.5: Main Activity

Al momento del lancio dell'applicazione viene richiesto l'inserimento delle credenziali d'accesso, impostate in fase di registrazione.
Sará comunque possibile cambiare tali informazione accedendo al pannello dei settaggi. Se il dispositivo non risulta essere registrato nel database, viene lanciata l'activity per la registrazione di un nuovo utente.

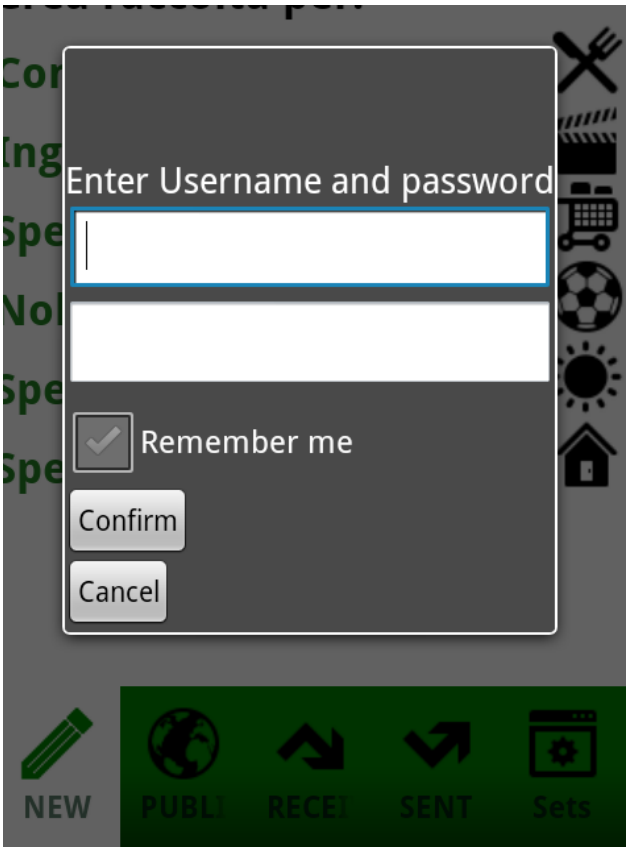

Figura 4.6: Inserimento credenziali

4.3.2 Nuova richiesta

Questo tab viene utilizzato per creare nuove richieste o modificarne di giá esistenti (ovviamente solo quelle di cui si é proprietari).
Come illustrato in precedenza una richiesta é lo strumento a disposizione dell'utente per domandare un impegno economico ad un altro utente.
La richiesta viene creata seguendo quattro semplici passaggi:

1.Scelta della categoria L'applicazione mette a disposizione una serie di tipologie di scopi per le richieste .
La scelta dello scopo della richiesta serve a rendere piú user-friendly l'interfaccia all'utente, difatti, non influisce in alcun modo nei passaggi successivi.

Figura 4.7: Creazione nuova richiesta - Step 1

2.Titolo e descrizione I campi titolo e descrizione sono obbligatori.
Queste informazioni serviranno ai destinatari per riconoscere la richiesta ricevuta e comprenderne lo scopo.

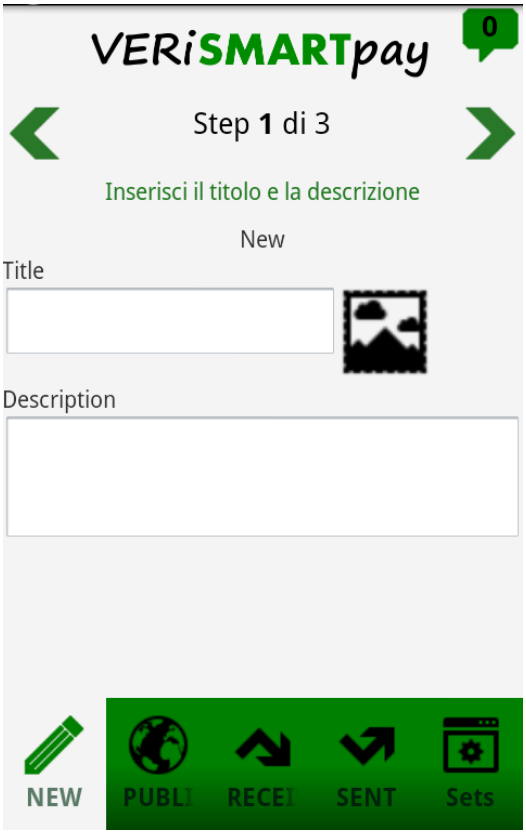

Figura 4.8: Creazione nuova richiesta - Step 2

3.Importo In questa terza fase é possibile inserire l'importo della richiesta e la valuta.
L'importo inserito viene inteso in modo differente a seconda della choose selezionata:importo totale da dividere tra i destinatari oppure importo medesimo da moltiplicare per ogni destinatario ;
una terza opzione permette di specificare un importo specifico per ogni destinatario.
In questa schermata é inoltre possibile decidere se gli utenti destinatari dovranno pagare l'importo in un'unica soluzione o meno.

Nel secondo caso si puó impostare anche l'importo minimo accettato per ogni singolo pagamento parziale.

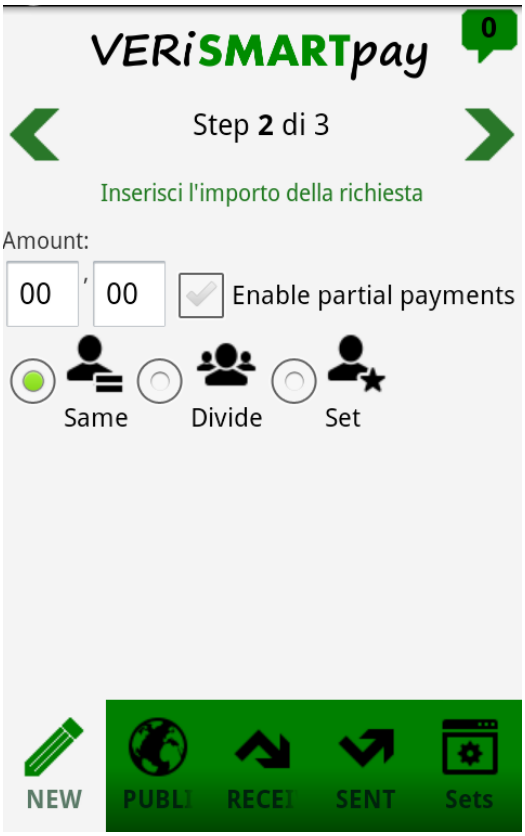

Figura 4.9: Creazione nuova richiesta - Step 3

4.Inserimento destinatari Questo step é adibito all'inserimento dei destinatari della richiesta (intesi come contatti telefonici)
L'inserimento dei contatto puó avvenire:

- manualmente: digitando il numero telefonico direttamente nella tex-View adibita a tale scopo

- dalla rubrica: é possibile selezionare i contatti direttamente dall'elenco

- dai gruppi: selezionando un gruppo tutti i contatti di cui é composto verranno aggiunti automaticamente alla lista dei destinatari.

Al termine dell'inserimento dei destinatari, se tutti i parametri necessari sono stati inseriti e sono corretti(ad esempio l'importo deve essere maggiore di zero), la richiesta potrá essere inviata;
una progressbar visualizza il progresso dell'esecuzione.
Al termine dell'operazione, se andata a buon fine, la richiesta sará stata aggiunta nel database e i destinatari verranno avvisati con una notifica del tipo:'Utente1 ha creato una nuova richiesta' (dove 'Utente1' é lo username del mittente).
I destinatari che non possiedono l'applicazione, invece, verrano invitati a scaricarla dallo Store.

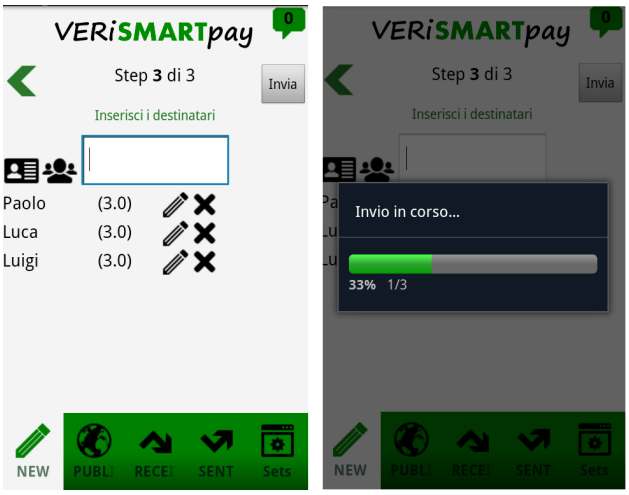

Figura 4.10: Creazione nuova richiesta - Step 4

4.3.3 Richieste pubbliche

Questa sezione dell'applicazione nasce con un'idea di creare richieste accessibili da chiunque con un fine benefico.
Le informazioni relative ad ogni richiesta sono visualizzate in una lista, dove per ogni richiesta sono presenti il titolo, l'importo, il nome (username) del mittente e il numero di utenti che hanno giá aderito alla richiesta.
Le operazioni a disposizione dell'utente sono, in questo particolare caso, limitate: un utente non puó, ad esempio, rifiutare o cancellare una richiesta pubblica, in quanto non é stato inserito come uno specifico destinatario;per questa specifica categoria di richieste é, inoltre, disabilitato l'utilizzo del meccanismo delle offerte come metodo di pagamento.
L'utilizzatore finale, potrá quindi decidere di pagare direttamente l'importo suggerito per un numero illimitato di volte;
proprio perché lo scopo di queste richieste é filantropico ogni utente puó partecipare piú volte alla medesima raccolta.

Nella schermata in oggetto é presente anche un form per la ricerca di una determinata raccolta in base al nome.
La query restituirá le richieste pubbliche che hanno come titolo o autore quello specificato oppure che iniziano, finiscono, contengono la stringa specificata.
Selezionando una delle richieste della lista si accede ad una finestra informativa, contente il titolo, l'importo, l'ammontare attuale della raccolta in relazione all'obiettivo prefissato oltre agli strumenti per condividere la raccolta sui piú diffusi social network e per effettuare la donazione.

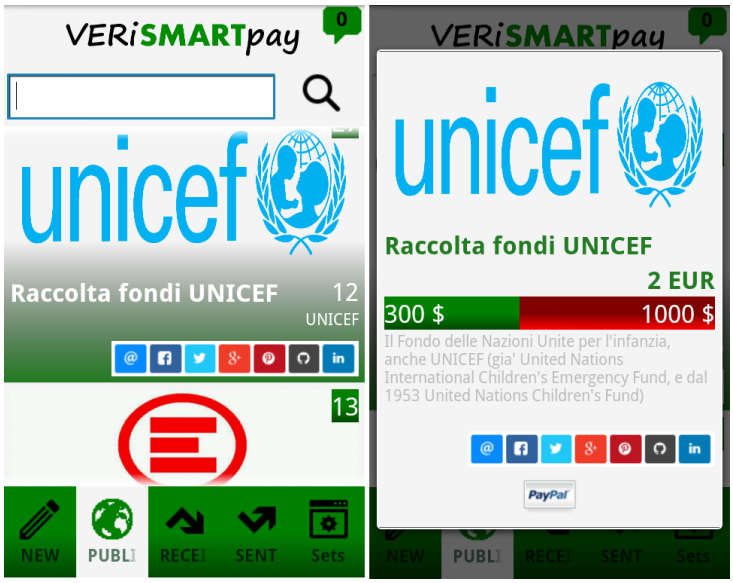

Figura 4.11: Richieste pubbliche

4.3.4 Richieste ricevute

Questo pannello viene utilizzato per visualizzare e gestire tutte le richieste ricevute, ovvero tutte le richieste di cui l'utente é destinatario.
In questa sezione sono presenti sia le richieste accettate che quelle rifiutate, ovviamente opportunamente evidenziate. Questa sezione presenta, infatti, una lista contente, per ogni riga, le informazioni della singola richiesta.
Nello specifico sono presenti:

- l'immagine della richiesta (se il mittente di tale richiesta non ha impostato alcuna immagine, viene impostato il logo dell'applicazione come immagine di default).
  L'immagine originale viene dinamicamente modificata, per renderla esteticamente piú gradevole, dandole una forma circolare

- il titolo, il mittente, l'importo, la data di creazione della richiesta.

- lo stato della richiesta (rifiutata o accettata)

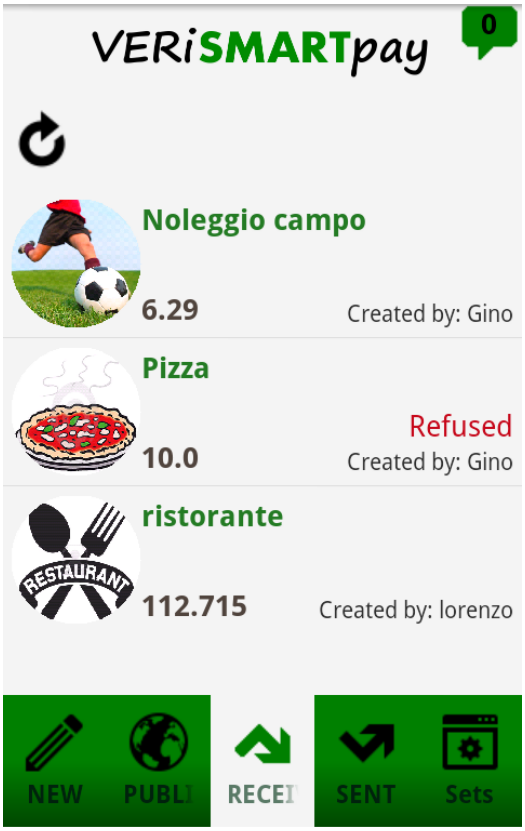

Figura 4.12: Tab richieste ricevute

Selezionando un elemento della lista viene aperta una finestra contenente tutte le informazioni relative alla richiesta e gli strumenti a disposizione dell'utente per effettuare operazioni su di essa (pagamento, accettare/rifiutare, cancellare)
Alcune di queste operazioni(accettare o rifiutare, cancellare) sono accessibili direttamente in quanto associati all'evento OnLongclick.
In particolare ecco come si presenta la finestra:

- Tutte le informazioni sulla richiesta quali titolo, l'importo, la descrizione, il mittente.

- Button Refuse/Accept : Permette all'utente di decidere di accettare o meno la richiesta
  (L'informazione viene inviata e salvata sul server e viene inviata una notifica al proprietario della richiesta)
  L'utente viene comunque informato di ogni eventuale modifica alla richiesta

- Button Delete:Permette all'utente di decidere di eliminare definitivamente la richiesta
  (Viene inviata una notifica al proprietario della richiesta)
  L'utente verrá escluso da ogni notifica relativa alla richiesta(viene eliminata anche dal server)

- Button Pay: Permette all'utente di inviare il denaro al proprietario della richiesta seguendo alcuni semplici passaggi illustrati nel dettaglio in seguito

- Button Chat:Permette all'utente di accedere all'Activity relativa al servizio di chat relativo a quella specifica richiesta

Figura 4.13: Finestra per la gestione delle richieste ricevute

4.3.5 Richieste inviate

In questa sezione é possibile accedere alla gestione delle richieste di cui si é proprietario, ovvero quelle di cui si é il mittente.
Le richieste sono visualizzate in un lista che contiene:

- l'immagine della richiesta (se non é stata impostata alcuna immagine, viene impostato il logo dell'applicazione come immagine di default). L'immagine originale viene dinamicamente modificata, per renderla esteticamente piú gradevole, dandole una forma circolare
- il titolo, l'importo (la somma degli importi per ogni destinatario se é una richiesta collettiva), la data di creazione della richiesta.
- lo stato della richiesta: numero di destinatari che hanno giá pagato e numero di quelli che ancora devono saldarla

Selezionando un elemento della lista si accede alla gestione della richiesta medesima.
In particolare la lista permette di visualizzare:

- il titolo e la descrizione della richiesta
- l'ammontare raccolto e quello ancora da ricevere.
- lo stato del pagamento di ogni destinatario singolarmente

I destinatari sono suddivisi in base allo stato del pagamento:

- Confermato: il pagamento é stato effettuato e confermato
- Da confermare: il pagamento é stato effettuato, ma non é stato completato (es: se in fase di pagamento é mancata la connesione).Il proprietario della richiesta puó confermare, o meno, il pagamento.
- Non pagato: il pagamento non é stato ancora effettuato

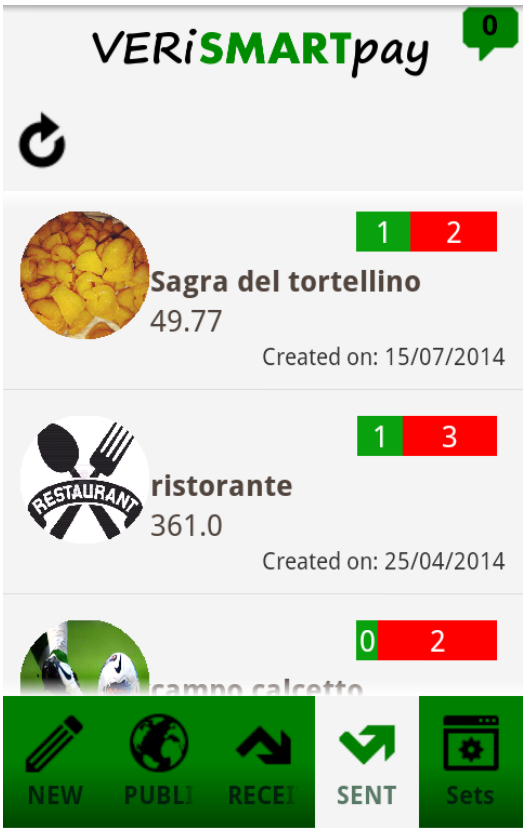

Figura 4.14: Tab richieste inviate

- Rifiuato: il destinatario ha deciso di rifiutare il pagamento
- Offerta: il destinatario ha inviato un'offerta per saldare il pagamento
- In attesa: la richiesta in oggetto é stata inserita in un'offerta della quale si attende una risposta di accettazione o rifiuto;nel primo caso verrá scalato l'importo proposto altrimenti la richiesta tornerá allo stato 'non pagato'

Per ogni destinatario é possibile eseguire diverse operazioni a seconda dello stato del pagamento come, ad esempio, modificare l'importo, cancellare la richiesta, confermare manualmente il pagamento, mandare un promemoria . Ognuna di queste azioni viene notificata al destinatario con un differente messaggio.

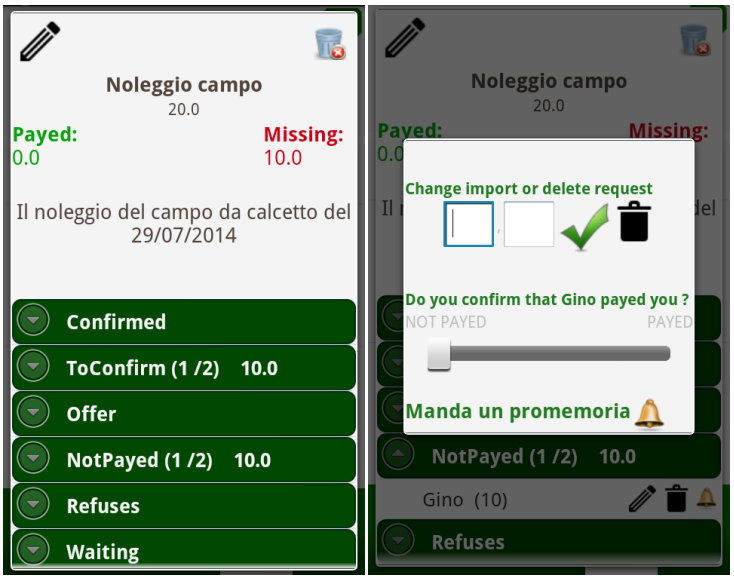

Figura 4.15: Finestra per la gestione delle richieste inviate

Se la richiesta é di tipo collettiva, é possibile eseguire operazioni generali e condivise a tutti i destinatari come la modifica del titolo e/o destinatario oppure l'eliminazione definitiva.
Queste operazioni verranno segnalate a tutti i destinatari con un apposito messaggio.

4.3.6 Settings

Questa Activity permette all'utente di impostare i settaggi relativi a:

- Dati utente: possibile modificare username, password, email, foto utente
- Impostazioni notifiche: é possibile impostare se ricevere o meno le notifiche
- Impostazioni contatti: é possibile accedere alla gestione dei gruppi, dei contatti bloccati o della condivisione

Le impostazioni relative ai dati utente (username, password, email, foto) vengono aggiornate anche nel database, cosí che siano visibili anche agli altri utenti, ad esclusione della password .
Le impostazioni di carattere specifico all'utilizzo dell'applicazione (impostazioni notifiche, contatti bloccati, gestione gruppi) verrano memorizzate come preferenze dell'applicazione stessa.

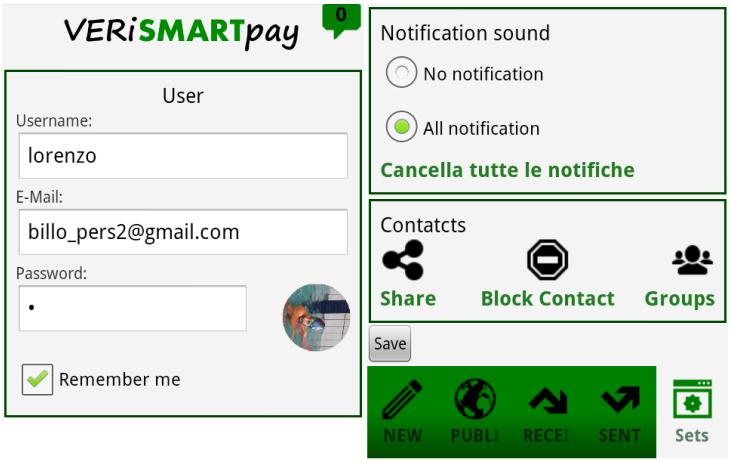

Figura 4.16: Tab settings

4.3.7 Contatti bloccati

In questa activity é possibile impostare gli utenti dai quali non si desidera ricevere richieste.

Le richieste ricevute dagli utenti presenti in questa lista vengono immediatamente eliminate dal databse, l'utente non riceverá alcuna informazione di tale operazione.
É possibile inserire un contatto da bloccare inserendo direttamente il suo contatto telefonico, o selezionandolo dalla rubrica.
In qualsiasi momento l'utente puó decidere di rimuovere un contatto dalla lista semplicemente selezionandolo e confermando l'intenzione di riabilitarlo ad inviargli richieste.

Figura 4.17: Contatti bloccati

4.3.8 Notifiche

L'applicazione implementa un accurato e preciso sistema di notifiche push verso l'utente.
Le notifiche di tipo push sono una feature ormai quasi indispensabile per qualsiasi sviluppatore che voglia fornire un ambiente di utilizzo completo, pratico e funzionale all'utente.
L'affermarsi di questo strumento é stato reso possibile sopratutto grazie alla crescita sempre maggiore di sistemi operativi in grado si supportare multita-

sking : l'applicazione che implementa questo servizio, infatti, resta in ascolto di eventuali notifiche in background.

L'applicazione che ho sviluppato, sfrutta il servizio GoogleCloudMessage per l'invio e la gestione delle notifiche lato server.
Nello specifico del mio progetto l'utente riceve notifiche di informazioni relative alle richieste ricevute e inviate.

Un apposito contatore posto in alto e visibile in ogni activity tiene traccia delle notifiche ancora da visualizzare.
Un'apposita schermata mostra tutte le notifiche in ordine di data di ricezione con un testo significativo(evidenziando quelle ancora da leggere con uno stile italico). La tabella mostra nel dettaglio i tipi di notifica e ció che

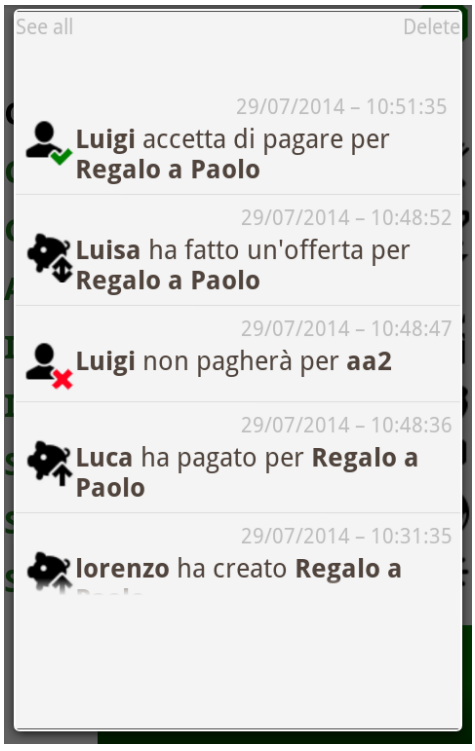

Figura 4.18: Visualizzazione notifiche

rappresentano.

Icona	Tipo	Descrizione
	Nuova richiesta ricevuta	Hai ricevuto una richiesta da un altro utente
	Pagamento ricevuto	Un utente ha pagato per una tua richiesta
	Richiesta modificata	Il proprietario di una richiesta ha modificato alcune informazioni di essa.
	Richiesta rifiutata	Un utente a cui avevi mandato una richiesta non ha aderito
	Richiesta accettata	Un utente a cui avevi mandato una richiesta ha aderito
	Pagamento confermato	Il tuo pagamento per una richiesta é stato confermato
	Nuova offerta ricevuta	Un utente richiede di pagare per una tua richiesta con una offerta
	Offerta accettata	Un utente ha accettato una offerta
	Offerta rifiutata	Un utente non ha accettato di offerta
	Commento	Un utente ha commentato una sua richiesta

Tabella 4.1: Tipologie di notifiche

4.3.9 Primo accesso

Questa activity viene caricata nel caso in cui l'utente non sia ancora registrato al servizio.
Ad ogni accesso all'applicazione, infatti, il sistema controlla se l'utente é presente nel database, utilizzando il numero di telefono associato al dispositivo come riferimento.
Nel caso in cui l'utente dovesse risultare non registrato, l'activity lanciata permette all'utente di inserire i dati indispensabili al funzionamento dell'applicazione:username, password, email (per ricevere pagamenti), immagine del profilo.
Prima di confermare la registrazione viene effettuato un controllo per garantire l'unicitá del nome utente.
Ovviamente tutti i campi sono obbligatori.
Una volta completata la registrazione si potrá accedere all'applicazione nella sua completezza.

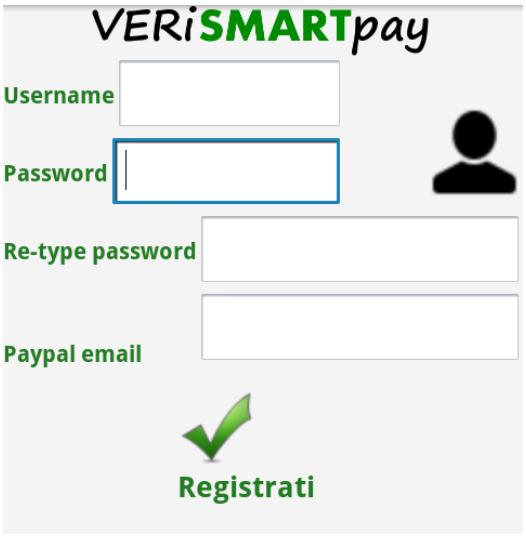

Figura 4.19: Primo accesso

4.3.10 Chat

Una delle ultime componenti aggiunte a questo progetto é un servizio di chat.
Questo servizio nasce dall'idea di mettere a disposizione degli utenti, che condividono una richiesta, un metodo per comunicare.
Il servizio permette di inviare una stringa di testo o un'immagine agli altri utenti.
Le conversazioni sono memorizzate nel database in modo da non saturare la memoria interna del dispositivo.
Seguendo la politica di sicurezza intrapresa in tutto lo sviluppo di questo progetto, anche le conversazioni sono cifrate utilizzando la chiave pubblica e privata dell'utente.
Ad ogni nuovo commento (inteso sia come testo che come immagine) inserito viene inviata una notifica ad ogni utente coinvolto nella richiesta.

Figura 4.20: Servizio di chat

4.3.11 Gestione dei gruppi

Con lo scopo di permettere un utilizzo piú veloce e fruibile all'utente é stato anche aggiunto uno strumento per la gestione in gruppi dei contatti. Questa utility permette, infatti, di suddividere i propri contatti in gruppi per velocizzare l'inserimento di essi come destinatari di una richiesta.
Le situazioni in cui questo strumento risulta veramente utile sono tutti quei casi in cui un insieme di utenti condivide spesso delle spese come ad esempio i coinquilini di un appartamento.
É possibile accedere al panello di gestione dei gruppi dalla sezione settings.
La schermata di gestione permette di creare o eliminare un gruppo e di aggiungere o togliere un elemento da un determinato gruppo.
L'inserimento dei contatti puó avvenire in modo manuale o selezionando il numero di telefono dalla propria rubrica.
Da un punto di vista implementativo e strutturale é importante sottolineare che i gruppi vengono memorizzati nel device dell'utente in quanto il loro utilizzo rimane confinato ad un utilizzo locale.

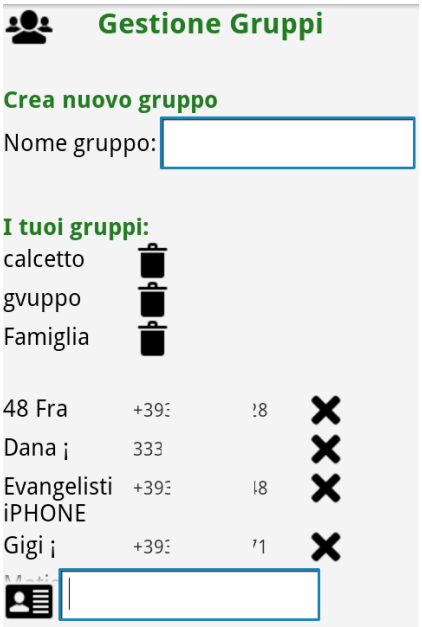

Figura 4.21: Schermata per la gestione dei gruppi

4.3.12 Pagamento

La fase di pagamento della richiesta segue alcuni semplici passaggi:

1. Se la richiesta prevede la possibilitá di effettuare un pagamento parziale, allora viene richiesto direttamente al destinatario di inserire l'importo che si vuole versare(entro i limiti imposti dal mittente)

2. Il sistema verifica la presenza di eventuali crediti che si hanno con il mittente della richiesta, e fornisce la migliore proposta di offerta (intesa come l'offerta che riesce ad annullare il maggior numero di richieste)

3. - Se l'utente pagante decide di sfruttare l'offerta propostagli dal sistema, allora l'offerta viene inviata al mittente e la richiesta rimane in sospeso
   - Se l'utente pagante decide di non inviare l'offerta (oppure non vi erano i requisti per crearne una) viene lanciata l'activity per il pagamento vero e proprio.

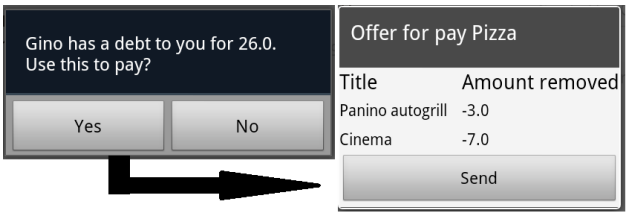

Figura 4.22: Pagamento tramite offerta

La fase di pagamento vera e propria viene implementata, come giá illustrato, con le API rese disponibili da PayPal per lo sviluppo su piattaforme Android.
PayPal mobile SDK fornisce una serie di API utili per integrare un servizio di pagamento built-in nelle applicazioni per smartphone, tra le quali:
identificazione client PayPal
Sistema di pagamento PayPal classico

Sistema di pagamento con carte di credito e credit card scanning
Riepilogo dell'ordine

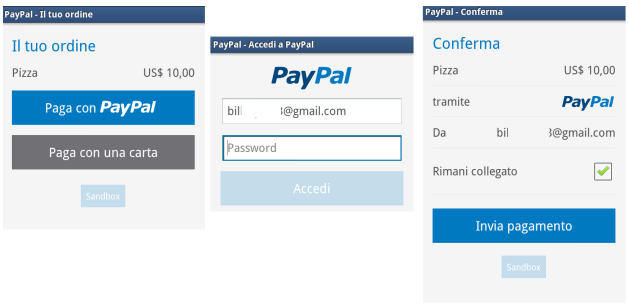

Figura 4.23: Fasi del pagamento con PayPal SDK

4.4 Implementazione

In questa sezione intendo presentare gli aspetti implementativi piú interessanti e utili alla comprensione delle funzionalitá dell'applicazione.
Intendo, quindi, descrivere le classi e gli oggetti fondamentali su cui si basa l'intera architettura del software, oltre che alcuni cenni sulle funzioni piú rilevanti.

4.4.1 Le classi principali

Le classi principali utilizzate dal sistema sono:

- Request : classe che descrive la struttura e i metodi per le richieste
- Offer : classe che descrive la struttura e i metodi per le offerte
- Notification : classe che descrive la struttura e i metodi per notifiche

La classe Request Questa classe permette di definire nuove richieste di pagamento.

```
class Request  {
   String req_id;
   float amount;
   float min_amount
   String curr;
   String title;
   String descr;
   String creator_id;
   String creator_name;
   int payed;
   int type;
   int wanna_pay;
   String collection_id;
   String friend_id;
   String friend_name;
   String created_at;
}
```

Dove, in particolare:

- req_id :identificatore alfanumerico univoco di ogni richiesta creato in modo casuale
- collection_id :identificatore alfanumerico univoco di un insieme di richieste (se la richiesta é stata inviata a piú persone queste avranno lo stesso collection_id)
- amount :importo della richiesta
- curr :valuta dell'importo richiesto ('EUR', 'USD', 'YEN', ...)
- min_amount: importo minimo che il mittente richiede se la richiesta prevede pagamenti parziali
- title : titolo della richiesta
- descr: breve descrizione della richiesta
- creator_id :identificatore univoco di chi ha creato la richiesta
- creator_name: nome associato a creator_id
- payed : valore intero che indica lo stato del pagamento
  - Pagamento effettuato e confermato
  - Pagamento non effettuato

  - Pagamento effettuato, ma non confermato
  - É stata inviata un'offerta come merce di cambio, ma é da confermare
  - La richiesta é stata inserita in una offerta inviata
- wanna_pay: indica se l'offerta é stata accettata o rifiutata
- friend_id: identificatore univoco dell'utente destinatario
- type: indica il tipo di richiesta
  - privata con pagamento fisso
  - privata con pagamento parziale abilitato
  - pubblica con offerta libera
  - pubblica con importo fisso
- created_at : data e ora creazione della richiesta (nel formato hh:mm:ss -GG-MM-YYYY)

Questa classe contiene i getters ed i setters per ogni componente.
In piú sono presenti metodi per la conversioni di una richiesta in parametri utilizzati nella richiesta HTTP.

La classe Offer La classe offer é costituita dalla seguente struttura:

```
1 class Offer {
2    int id;
3    String req_id;
4    String req_title;
5    float amountToremove;}
```

Questa classe permette di definire un'offerta di pagamento.
Ricordiamo che un'offerta di pagamento é uno strumento a disposizione degli utenti per proporre un accordo di pagamento utilizzando i propri crediti/debiti come moneta di scambio.
In quest'ottica risulta di facie comprensione la struttura implementata per la classe Offer.

- id: identificatore univoco dell'offerta
- req_id: identificatore univoco della richiesta che si vuole estinguere

- req_title: nome della request associato a req_id
- amountToRemove:importo da scalare nella richiesta identificata da req_id

Una proposta di pagamento tramite offerta é composta da un'array di oggetti Offer. La classe é composta dai metodi per definire o recuperare determinati valori di un'offerta oltre che un metodo per la conversione di un oggetto Offer in una serie di parametri per eseguire richieste HTTP.

La classe Notification Questa classe permette di definire un formato per la gestione delle notifiche ricevute dal server.
Ogni notifica creata lato serveré infatti composta dai campi :

- mittente
- destinatario
- tipo
- titlo richiesta

```
1 Class Notification {
2     String message;
3     int type;
4     int icon;
5     String date;}
```

Utilizzando questi dati viene creata un'istanza della classe Notification che é formata da

- message : messagio creato dinamicamente in base al tipo (es: USER1 ha creato la richiesta TITOLO1)
- type : intero che identifca il tipo della notifica
- icon : icona associata alla notifica in relazione al tipo

- id: identificatore univoco dell'offerta
- req_id: identificatore univoco della richiesta che si vuole estinguere
- req_title: nome della request associato a req_id
- amountToRemove:importo da scalare nella richiesta identificata da req_id

Una proposta di pagamento tramite offerta é composta da un'array di oggetti Offer. La classe é composta dai metodi per definire o recuperare determinati valori di un'offerta oltre che un metodo per la conversione di un oggetto Offer in una serie di parametri per eseguire richieste HTTP.

4.4.2 Comunicazione client-server

Una componente importante nell'architettura del progetto é sicuramente la comunicazione tra client e server.
Nello specifico il sistema implementato nel dispositivo mobile effettua richieste POST HTTP al server per ottenere le informazioni memorizzate nel database come, ad esempio, informazioni sulle richieste o sui contatti, oppure per chiedere inserire nuovi elementi (nel caso della creazione di una nuova richiesta, registrazione di un utente o invio di un messaggio in chat, ecc..), oppure effettuare modifiche di alcuni campi del database (es: modifica del titolo di una richiesta o conferma manuale di un pagamento, ecc...)
Il client ha a disposizione una numero limitato di funzioni, identificate da un numero intero al quale il servwer associa una specifica query da eseguire nel database.
In questo modo una connessione al server con la richiesta di eseguire la funzione numero 1 corrisponde alla query di selezione di tutte le richieste pubbliche nel database.
Il protocollo di comunicazione tra un dispositivo Android e un server segue alcuni passaggi standard: 1. Creazione di una lista di parametri attributo-valore da passare al server.
Per questa operazione risulta utile utilizzare la classe NameValuePair:una semplice classe per incapsulare una coppia attributo / valore.

```
1 ArrayList<NameValuePair> PostParams = new ArrayList<NameValuePair
    >();
```

```
PostParams.add(new BasicNameValuePair("PublicRequestTitle",
    PubTitle));
```

(nell'esempio l'unico parametro passato é il titolo di una richiesta pubblica della quale si vogliono ottenere le informazioni) 2.Creazione richiesta HTTP Dopo aver settato tutti i parametri necessari, si procede a creare la richiesta HTTP vera e propria; in questa fase é bene settare i timeout di connessione, oltre che ovviamente il parametro contenente l'url al quale la connessione deve puntare

```
//Creazione richiesta HTTP
HttpParams httpParameters = new BasicHttpParams();

//Impostazione parametri timeout della connessione (in ms)
HttpConnectionParams.setConnectionTimeout(httpParameters,  6500);
HttpConnectionParams.setSoTimeout(httpParameters,  6500);

HttpClient httpclient = new DefaultHttpClient(httpParameters);
HttpPost httppost = new HttpPost("URL DEL SERVER");
httppost.setEntity(new UrlEncodedFormEntity(PostParams));
```

3.Lanciare la richiesta e gestire la risposta Il seguente codice mostra come lanciare il metodo, il risultato verrá fornito sottoforma di enitá HTTP, ma con l'aiuto delle librerie native JAVA risulta facile convertire il risultao nel formato piú idoneo

```
HttpResponse response = httpclient.execute(httppost);
//Ottieni la risposta
HttpEntity entity = response.getEntity();

//Converti il risultatoin stringa e poi in formato JSON
String result = EntityUtils.toString(entity);
JSONObject jsonResult = new JSONObject(result);

//Recupera i dati
String descrizione = jsonObject.getString("Titolo");
float import = jsonObject.getInt("Importo");
```

4.4.3 Integrazione con le API PayPal

Il pacchetto di sviluppo rilasciato da PayPal per sistemi Android permette di creare un'interfaccia di pagamento Durante lo sviluppo dell'applicazione é possibile settare l'ambiente di sviluppo in modo per esguire dei test su conti virtuali, evitando movimenti di denaro reale.
Nel progetto viene chiamata un'activity secondaria gestita dalla classe PayPalService con l'aggiunta di alcuni parametri per gestire l'autenticazione dell'utente

```
1 Intent intent = new Intent(this, PayPalService.class);
2 intent.putExtra(PaymentActivity.EXTRA_PAYPAL_ENVIRONMENT, ENV);
3 intent.putExtra(PaymentActivity.EXTRA_CLIENT_ID, CLIENT_ID);
```

Dove CLIENT_ID si riferisce ad una variabile che identifica l'applicazione, mentre la variabile ENV corrisponde all'ambiente di sviluppo (reale o sandbox) Successivamente vengono effettuate le operazione di pagamento vere e proprie questa volta tramite la classe PaymentActivity alla quale vengono passate come parametri le informazioni sul pagamento.
In particolare in questa fase é possibile impostare il destinatario del pagamento, l'importo e la valuta, la descrizione del prodotto acquistato

```
1 Intent pay_intent = new Intent(this, PaymentActivity.class);
2 pay_intent.putExtra(PaymentActivity.EXTRA_RECEIVER_EMAIL, "
    Destinatario del pagamento");
3 pay_intent.putExtra(PaymentActivity.EXTRA_PAYMENT, "Oggetto da
    comprare");
4 pay_intent.putExtra(PaymentActivity.EXTRA_PAYER_ID, "ID del
    compratore");
```

Il risultato ottenuto é uun insieme di dati in formato JSON che specificano i dettagli del pagamento.

```
1 {
2   "client": {
3     "environment": "sandbox",
4     "paypal_sdk_version": "2.0.0",
```

```
    "platform": "Android",
    "product_name": "PayPal Android SDK;"
  },
  "response": {
    "create_time": "2014-02-12T22:29:49Z",
    "id": "PAY-5000000234KL57LXI",
    "intent": "sale",
    "state": "approved"
  },
  "response_type": "payment"
}
```

4.5 Server-side

Il progetto prevede e necessita dello sviluppo di parte del codice lato server.
In particolare sono implementate tutte le funzioni per l'invio di notifiche push (utilizzando le api GCM) e lo storage e la gestione di dati un database.

Il database Le comunicazioni con il server permettono all'applicazione di inserire dati nel database, eseguire query predefinite, cancellare e modificare items delle tabelle presenti.
Il database é composto da quattro tabelle:

1. REQUEST
2. CONTACT
3. OFFER
4. TEXT

La tabella REQUEST La tabella contiene tutte le richieste create.

Campo	Tipo	Descrizione
Request Id	Text	Id univoco della richiesta
Type	Int	Intero che identifica il tipo di richiesta: - Privata con possibilità di pagamenti parziali - Privata senza possibilità di pagamenti parziali - Pubblica ad importo fisso - Pubblica ad offerta libera
Amount	Float	Importo della singola richiesta
Currency	Text	Valuta relativa ll'importo
Title	Text	Titolo
Descr	Text	Descrizione
CreatorId	Text	Id univoco dell'utente che ha creato la richiesta
Payed	Int	Intero che indica lo stato del pagamento () - Pagamento effettuato e confermato - Pagamento non effettuato - Pagamento effettuato,ma non confermato - Esiste un'offer per pagarla ed è da confermare - La richiesta è stata inserita in una offer
WannaPay	Int	Intero che indica se la richiesta è stata accettata o rifiutata
CollectionId	Text	Id univoco della collection (insieme di richieste)
FriendId	Text	Id univoco del destinatario della richiesta
CreatedAt	Text	Data e ora della creazione della richiesta
AccessToken	Char	Valore random

Figura 4.24: Tabella REQUEST del database

Di particolare rilevanza sono i campi:
Payed : identifica con un valore compreso tra -3 e 1 lo stato del pagamento per quella specifica richiesta
Type : identifica il tipo di richiesta (Pubblica a offerta libera, pubblica ad offerta fissa, richiesta privata con pagamento parziale, richiesta privata con importo fissato)

La tabella CONTACT La tabella contiene tutti i dati relativi ad un utente registrato, tra i quali risultano di particolare importanza:
Public key: la chiave pubblica associata all'utente ed utilizzata per cifrare i dati nelle comunicazioni verso di esso
Login_reg_id: identificatore univoco creato in fase di registrazione al servizio GCM

Campo	Tipo	Descrizione
Login_phone	Char	Numero di telefono del contatto
Login_name	Char	Username del contatto
Login_mail	Char	E-mail del contatto
Login_regid	Char	Id GCM associato al numero di telefono Login_phone
Login_pubkey	Text	Chiave pubblica associata al contatto
Login_psw	Char	Password del contatto per l'accesso

Figura 4.25: Tabella CONTACT del database

La tabella OFFER Questa tabella del database viene utilizzata per memorizzare ogni proposta di offerta che un utente invia con lo scopo per saldare un debito con un altro utilizzatore.
Le offerte memorizzate sono solo quelle in attesa di una conferma (o rifiuto). Ogni elemento di questa tabella deve quindi tenere traccia dell'importo da decrementare e dell'id relativi alle richieste oggetto dell'offerta(cioé quelle che si vogliono utilizzare come 'merce di scambio'), nonché dell'id della richiesta soggetto dell'offerta (ovvero quella che si vuole saldare utilizzando i crediti).

Campo	Tipo	Descrizione
Offer Id	Int	Id univoco dell'offerta
Request Id	Text	Id univoco della richiesta che si vuole di scontare
Request Title	Text	Titolo della richiesta associato alla richiesta da scontare
Amount	Float	Importo che si vuole scontare dalla richiesta
Request Id To Pay	Text	Id univoco della richiesta per la quale si è fatta l'offerta

Figura 4.26: Tabella OFFER del database

La tabella TEXT La scelta progettuale di salvare tutte le conversazioni nel server é data dalla necessitá di rendere il sistema il piú possibile autonomo dalla piattaforma client (futuri sviluppi potrebbero dirigersi verso lo sviluppo dell'applicazione per smartphone con sistemi operativi diversi da Android).
La tabella contiene semplicemente l'id del client che é l'autore del commento, l'oggetto stesso del commento, la tipologia (testo o immagine) ed un riferimento alla richiesta per la quale il testo é stato creato.

Campo	Tipo	Descrizione
Collection Id	Text	Id univoco della richiesta alla quale il commento si riferisce
Author Id	Text	Id univoco dell'autore del commento
Text	Text	Testo del commento
Type	Int	Tipo (testo o immagine)
Date	Date	Data creazione del commento

Figura 4.27: Tabella TEXT del database

4.6 Sicurezza

4.6.1 Crittografia asimmetrica

L'utilizzo e la memorizzazione di dati sensibili necessari all'applicazione ha richiesto l'implementazione di un livello di sicurezza adeguato.
Ho scelto di sfruttare l'algoritmo RSA (o crittografia a chiave pubblica/privata) a livello applicativo, non avendo a disposizione un server che implementasse un trasporto sicuro di dati (https).
Per comprendere piú facilmente l'idea alla base di questo algoritmo basta associare l'idea di chiave pubblica a quella di lucchetto e quella privata a quella di chiave.
Nella comunicazione di un messaggio cifrato fra due utenti A e B, A dovrá innanzitutto chiedere a B il suo lucchetto, giá aperto(B dovrá prestare attenzione a conservare segretamente la chiave del lucchetto medesimo).
Ora A chiude il messaggio con il lucchetto ricevuto e lo spedisce a B. Da questo punto in poi solo B potrá aprire e quindi decifrare il messaggio inviato da A.

4.6.2 Sicurezza dei dati

Nello sviluppo dell'applicazione ad ogni nuovo accesso vengono create le due chiavi (pubblica e privata) personali a 2048 bit.
La chiave pubblica verrá memorizzata dal server, in modo che possa cifrare

tutti i dati verso ciascun utente.
La chiave privata, invece, viene memorizzata nel device dell'utente in modo che possa decifrare i dati ricevuti dal server.

Ogni utente ha ovviamente, a disposizione la chiave pubblica del server, in modo che ogni dato inviato venga preventivamente cifrato.
Sará poi compito del server decifrare le informazioni ricevute prima di poterle utilizzare.

L'implementazione di un algoritmo di cifratura é reso possibile grazie all'utilizzo di librerie come: javax.crypto, che fornisce le classi e le interfacce per le operazioni di crittografia, o pacchetti come java.security che include un ampio set di API, strumenti e implementazioni degli algoritmi di sicurezza comunemente utilizzati, meccanismi e protocolli.

La scelta di seguire un forte orientamento 'security-oriented', porta con se alcuni svantaggi soprattutto a livello di performance e efficienza dell'applicazione.
Si é notato infatti un leggero ritardo nelle operazioni di ricezioni dei dati dal server, dovuti per l'appunto alle operazioni intermedie di cifratura e decifrazione dei dati.
Nel complesso, peró, nell'applicazione non si osservano gravi ritardi e tempi d'attesa sproporzionati tali da non giustificare un'importante interfaccia di sicurezza, anche perché la mole dei dati trasmessi non é particolarmente rilevante.

Capitolo 5

Conclusioni e sviluppi futuri

In questo documento di tesi ho avuto modo di spiegare nei minimi particolari tutto il lavoro svolto nella progettazione, nello sviluppo e nell'implementazione di un impegno che mi ha trascinato con entusiasmo nel corso degli ultimi mesi.
Il risultato finale si presenta come un valido documento di approfondimento sulle interessanti argomentazioni e specifiche tecniche inerenti l'architettura del sistema operativo Android, oltre che fornire una panoramica sui maggiori competitors nel settore dei sistemi operativi per dispositivi mobili.
Vengono, inoltre, studiate tematiche inerenti il tema trattato approfondendo scientificamente argomentazioni attinenti la sicurezza nei dispositivi mobili con sistema operativo Android, presentando le sue vulnerabilitá, gli strumenti architetturali e quelli specifici, messi a disposizione per contrastare i pericolosi temi di malware o piú in generale dei software dannosi.
In piú ho presentato una panoramica approfondita sulla pervasivitá del sistema operativo di casa Google per dimostrare la sua estrema duttilitá e capillaritá, oltre che dare uno spunto sulle piú moderne tecnologie presenti nel mercato delle comunicazioni mobili.

Alla conclusione del lavoro svolto penso di potermi tranquillamente esprimere affermando che gli obiettivi preposti nella fase preliminare di progettazione

siano stati tutti ampiamente raggiunti.
Ad oggi VERiSMARTpay presenta un'interfaccia client completa, funzionale e graficamente gradevole.Ai requisiti preposti inizialmente sono state aggiunte una serie di feauture che rendono l'applicazione un vero e proprio sistema per l'organizzazione del proprio portafoglio elettronico.

Il lavoro svolto mi ha permesso di integrare numerose conoscenze, strumenti e tecnologie, potendo cosí approfondire e confrontarmi con molti dei concetti appresi durante gli studi universitari.
Seguire il progetto dall'idea iniziale fino all'implementazione vera e propria, mi ha visto coinvolto in prima persona in tematiche decisionali di progettazione e project planning, trasformando un'intuizione in un mesh-up di tecnologie e strumenti differenti.
L'idea iniziale si é evoluta portandomi al confronto diretto con svariate tematiche in ambito informatico come ad esempio: aspetti della comunicazione client-server attraverso il protocollo HTTP, tematiche di sicurezza e implementazione di algoritmi idonei alla conservazione di dati sensibili, disegno e sviluppo di database, integrazione di API esterne, oltre che ad uno utilizzo esteso e esperto delle tecniche di programmazione lato client e lato server.

Sono stati eseguiti numerosi test sia su device virtuali che su una serie di dispositivi differenti sia per caratteristiche tecniche che per versione del sistema operativo montato.
Al momento della stesura di questa tesi i test eseguiti non hanno evidenziato errori progettuali o implementativi, anche se certamente solo l'uso su larga scala permetterá di sottolineare alcuni bug minori altresí non riscontrabili.
Con questi risultati intendo affermare che il software sia sufficientemente stabile per una prossima pubblicazione negli store.

Le possibilitá di estensione sono numerose e vanno di pari passo con la crea-

tivitá che mi ha portato fino a qui oggi.
Sicuramente sará interessante estendere l'applicazione per la fruizione su altri sistemi operativi in modo da permettere un ampliamento dello spettro di utilizzo del prodotto.
Da un punto di vista tecnico puó risultare utile integrare un sistema piú versatile nella fase di pagamento, che possa fornire all'utente altre possibilitá per concludere una richiesta; tra le possibilitá al momento piú interessanti risulta degno di nota il servizio GoogleWallet dell'omonima Google Inc.che ovviamente si integra facilmente con il sistema Android.
Non escludo, inoltre, l'impiego di VERiSMARTpay per scopi commerciali, come ad esempio un market di beni o servizi acquistabili singolarmente oppure in gruppo, per il quale occorre un'accurata analisi di mercato oltre ad un importante approfondimento su alcune tematiche burocratiche-legali da non sottovalutare per un impiego corretto del prodotto.

Questo lavoro mi ha coinvolto completamente e con ardore per buona parte degli ultimi mesi del mio percorso universitario e spero di poter ottenere un riscontro diretto con gli utenti per poter migliorare ulteriormente il prodotto.
Nel prossimo futuro intendo continuare lo sviluppo di questo progetto, in quanto sono fermamente convinto possa essere uno strumento molto utile oltre che un lavoro maturo, completo e di qualitá.

Elenco delle figure

Bibliografia

[1] *https://www.paypal-media.com/about*, Dati utilizzo paypal nel mondo.

[2] Y. Fledel A. Shabtai, *Google android: A state-of-the-art review of security mechanisms*, Department of Information Systems Engineering Ben-Gurion University Israel,Department of Computer Science Ben-Gurion University Israel, Deutsche Telekom Laboratories at Ben-Gurion University Israel (2009).

[3] Texas AM, *Texas am's annual mobility study*, 2014.

[4] Apache, *http://www.apache.org/licenses/license-2.0*, Licenza Apache.

[5] Mobile Marketing Association, *Mobile banking overview*, 2009.

[6] Massimiliano Bigatti, *E-commerce con paypal. guida completa per lo sviluppatore*, Apogeo, 2008.

[7] Han Bing, *Analysis and research of system security based on android*, North China University of Technologies.

[8] Pietro Brunetti, *Cocoatouch layer memory management guidelines.*, (2011).

[9] International Data Corporation, *Worldwide smartphone os market share*, 2014.

[10] Ericsson, *http://www.ericsson.com/ericsson-mobility-report*, Rapporto di Ericsson Mobility Report sulla Networked Society.

[11] F-Secure, *Q1 2013: Game changer for android malware?*, (2013).

[12] Wong Feng, Caire, *Immunochromatographic diagnostic test analysis using google*, American Chemical Society (2014).

[13] Marko Gargenta, *Learning android*, O'Reilly, 2011.

[14] Gartner, *Market share analysis: Mobile phones, worldwide, 4q13 and 2013*, Gartner:analisi di maercato vendita smartphone.

[15] Google Inc., *https://www.google.com/intl/it_it/about/company/philosophy/*, Filosofia Google.

[16] G.Ghinea J.Hansen, T.Gronli, *Cloud to device push messaging on android: A case study*, Sch. of Inf. Syst., Comput.and Math., Brunel Univ.

[17] Charlie Miller Jon Oberheide, *Dissecting the android bouncer*, 2012.

[18] Neal Leavitt, *Payment applications make e-commerce mobile*, IEEE Computer Society (2010).

[19] Andrea Leganza, *iphone programming*, PuntoInformatico Libri, Edizioni Master.

[20] Ding Li-pin, *Analysis the security of android*, Chinese Academy of Sciences.

[21] Angelico Massimo, *E-commerce and security*, (2011).

[22] Rob Miller Min Wu, Simson Garfinkel, *Secure web authentication with mobile phones*, MIT Computer Science and Artificial Intelligence Laboratory.

[23] Sean Schultenel Nicholas J.Percoco, *Adventures in bouncerland - failures of automated malware detection within mobile application markets*, BlackHat (2012).

[24] Carlo Pelliccia, *Android programming*, PuntoInformatico Libri, Edizioni Master, 2012.

[25] Anwar Usman Shaheed Zulfiquar Saleem Kadhiwal, *Analysis of mobile payment security measures and different standards.*

[26] Dave MacLean Satya Komatineni, *Pro android 4*, Apress, 2012.

[27] Craig Smalley, *Security enhanced (se) android: Bringing flexible mac to android*, Trusted Systems Research National Security Agency.

[28] Stephen Paine Steve Burnett, *The rsa security's official guide to cryptography*, McGraw-Hill, 2001.

[29] Pan Wang, Qi, *Design of remote monitoring system for aquaculture cages based on 3g networks and arm-android embedded system*, SciVerse ScienceDirect.

[30] D.Octeau W.Enck, *A study of android application security*, Tech. report, 2011.

[31] Patrick McDaniel William Enck, Machigar Ongtang, *Understanding android security*, Pennsylvania State University.

Ringraziamenti

Ringrazio il Chiar.mo prof. **Luciano Bononi** per il sostegno professionale e competente che ha saputo darmi nello sviluppo di questo mio lavoro di tesi mettendo sempre a disposizione tutto il suo vastissimo bagaglio culturale e professionale nel comprendere e migliorare le mie intuizioni.

Ringrazio il dott.**Luca Bedogni** nelle vesti del mio correlatore perché ha saputo guidarmi motivandomi sempre con grande entusiasmo da quando tutto questo era solo un'idea nella mia mente; i suoi consigli e suggerimenti sono il vero valore aggiunto di questo lavoro.

Voglio ringraziare mia madre **Marina**, la quale ha saputo trasmettermi la sua grandiosa forza d'animo anche nei momenti piú difficili;penso che raramente potró mai incontrare una persona tanto forte e coraggiosa quanto lei e per questo lei sará sempre un modello da imitare nella mia vita.
A questi ringraziamenti segue un sincero grazie a mio padre **Marco**;in questi anni non ha mai dubitato delle mie capcitá anche quando io stesso vacillavo.Il mio primo maestro, il mio modello, la mia piú grande aspirazione, la mente piú brillante che ho avuto l'onore di avere al mio fianco a sostenermi sempre senza smettere mai di incoraggiarmi ed al quale sono veramente grato.I suoi preziosi insegnamenti e i suoi ideali sono la mia guida nella vita.
Ringrazio mio fratello **Walter**, che con le sue continue battute e prese in giro alla fine é riuscito davvero a farmi studiare.Del suo sostegno non dubiteró mai.
Ringrazio i tutti i miei parenti: **'nonni di Pisa'** che oggi sono qui con me:li ringrazio davvero di cuore...e loro sanno il perché; mia zia **Laura**, che nonostante abiti lontano é sempre molto presente e vicina a me e alla mia famiglia; mia nonna **Giovanna** e mio zio **Paolo**, che da Roma sono arrivati fino a qui per condividere con me questa giornata, confermando ancora una volta il loro affetto e la loro vicinanza.

Ora voglio ringraziare una persona veramente speciale:
la mia ragazza **Veronica**.Non so come presentare Veronica, se non come la mia metá.
Lei é la mia certezza, la mia ispirazione per tutto quello che faccio e che penso, la mia confidente per tutto quello che provo, il mio sentimento piú grande.
Sono stato veramente fortunato ad incontrala nella mia strada, e credo che tutti i miei sforzi per conquistarla siano stati del tutto ripagati nell'avere al mio fianco una ragazza tanto meravigliosa quanto unica.
La ringrazio per tutto quello che ha fatto per me, e per essermi stata sempre accanto preoccupandosi di non farmi mai mancare un suo sorriso, la sua allegria ed una parola di conforto.
Sono sicuro che la sua eccezionale forza d'animo, la sua grande determinazione e la sua brillante intelligenza le permetteranno di raggiungere qualsiasi obiettivo deciderá di porsi nella vita.

Ringrazio la mia migliore amica **Sara Zazzaroni**, con la quale in questi anni ho condiviso tutto, le voglio bene come se ne vuole ad una sorella e lei lo sa.

Nel corso di questi anni universitari ho avuto modo di conoscere molti ragazzi e molti amici.
Tra questi voglio ringraziare **Flavio Picinelli**:grande compagno di studi e di progetti;**Mauro Midolo**: un ragazzo brillante che rimpiango di non aver incontrato prima;**Andrea Aquino** :geniale ragazzo senza il quale chissá se sarei qui oggi.
Ringrazio anche tutti i miei **compagni del liceo**, con i quali ho passato sicuramente gli anni piú divertenti e con i quali ho vissuto esperienze che non sarebbe saggio mettere qui nero su bianco.

Ultimi, ma solo per costringerli a leggere tutto il resto, sono i miei amici di sempre;un 'gruppo di matti' che considero la mia seconda famiglia:**Luca** (un ragazzo molto intelligente e dai forti ideali.Un amico vero e sincero), **Luigi**(un vulcano di energia e simpatia che riesce a strapparti un sorriso anche nei momenti piú difficili.Peccato solo per le sue origini lucchesi), **Matia**(il piú sfrenato di tutti noi, una bomba pronta ad esplodere.Ma serio e affidabile quando un amico ha bisogno), **Federico**(un sognatore ed un visionario.Con lui ho condiviso tantissimo e che non sento da troppo tempo e mi dispiace tanto.Torna presto!), **Elena**(la nostra sorellona acquisita, resiste e sopporta tutti noi da tempo immemorabile, ma senza mai trascurare un sorriso), **Francesca**(con la testa un po' sulle nuvole, ma sempre disponibile e con i piedi ben fissati per dare un aiuto), **Giulia**(che prestissimo sará una splendida mamma ed alla quale faccio tanti auguri), e poi, non me ne vogliano per il poco spazio, **Elisa**, **Alex**, **Ivan**, **Fabio**, **Ilaria**, **Valentina**, **Pigi**, **Lorenzo**, **Sandra**, **Carlotta**, **Selene**, **Matteo**, **Giulia** e tutti gli altri che in un modo o nell'altro mi hanno accompagnato in questi anni meravigliosi.