Alma Mater Studiorum · Università di Bologna

**FACOLTÀ DI SCIENZE MATEMATICHE, FISICHE E NATURALI**

**Corso di Laurea in Informatica**

**MARKET E TOOLS PER LA PERSONALIZZAZIONE DI APPLICAZIONI ANDROID**

Relatore: Presentata da:

Chiar.mo Prof. ANDREA POLA

VITTORIO GHINI

**Sessione : Seconda**

**Anno Accademico : 2011/2012**

# MaRKET E TOOLs PER LA PERSONALIZZAZIONE DI APPLICAZIONI Android

Analisi DI FAttibilità e studio delle problematiche relative alla personalizzazione di applicazioni in azienda e per terze parti.

Progetto 3cfu - Informatica - Andrea Pola - 0000346690

Sommario

[Personalizzazione di applicazioni in ambiente Android 1](#_Toc326588857)

[Introduzione 2](#_Toc326588858)

* [Dispositivi mobile e Markets 2](#_Toc326588859)
* [Tendenze del mercato 3](#_Toc326588860)
* [Pattern di applicativi android 5](#_Toc326588861)

[Personalizzazione di Applicazioni Android 6](#_Toc326588862)

* [Principi di funzionamento 6](#_Toc326588863)
* [tools e sdk android 8](#_Toc326588864)
* [File di configurazione - Assets 9](#_Toc326588865)
* [Packaging e signing dell'apk 10](#_Toc326588866)
* [Soluzioni e scelte progettuali 12](#_Toc326588867)
* [Stato dell'arte 15](#_Toc326588868)

[Webservice di demo 17](#_Toc326588869)

* [Implementazione 19](#_Toc326588870)

[Possibili Sviluppi 20](#_Toc326588871)

## Introduzione

### Dispositivi mobile e Markets

Questo testo si pone come obbiettivo l'analisi di fattibilità e l'introduzione all'implementazione di sistemi che permettano il riutilizzo di codice sorgente di applicazioni per smartphone.

In questo testo, sarà approfondito l'approccio di soluzioni per la personalizzazione di applicazioni in ambiente Google Android, in quanto al momento attuale questa piattaforma è la miglior candidata per la rappresentazione del più largo settore di utenti e dispositivi. La piattaforma Android infatti conta il maggior numero di dispositivi compatibili sul mercato e lo share di maggioranza assoluta nel mercato statunitense.[[1]](#endnote-2)

Un'attenta valutazione va fatta sulla politica del market della piattaforma scelta. Và analizzato il funzionamento del market e dei tool con cui si vuole operare, per vedere se è possibile generare in maniera automatica o guidata diverse istanze di un'applicazione partendo da un codice sorgente comune.

In questo momento, sul Google Market, che prende il nome commerciale di Google Play è possibile caricare applicazioni in breve tempo e senza controlli manuali da parte del team di Google. Il controllo di qualità su queste applicazioni è lasciato ad un meccanismo di feedback e segnalazione autonomo degli utenti, è possibile infatti votare ogni applicazione e segnalare abusi. Applicazioni meglio valutate avranno migliore visibilità in tutto il market.

In seguito analizzeremo i pregi e i difetti dell'approccio di pubblicazione scelto da Google per il market Android.

**La nascita dei markets**

Con lo sviluppo di dispositivi cellulari con interfacce di rete veloci come wi-fi e 3G, disponibilità di potenza di calcolo e RAM, si è sviluppato velocemente un nuovo modo di rendere disponibile il software verso i consumatori. Store Symbian, App Store , e di seguito Google Market poi rinominato in Google Play.

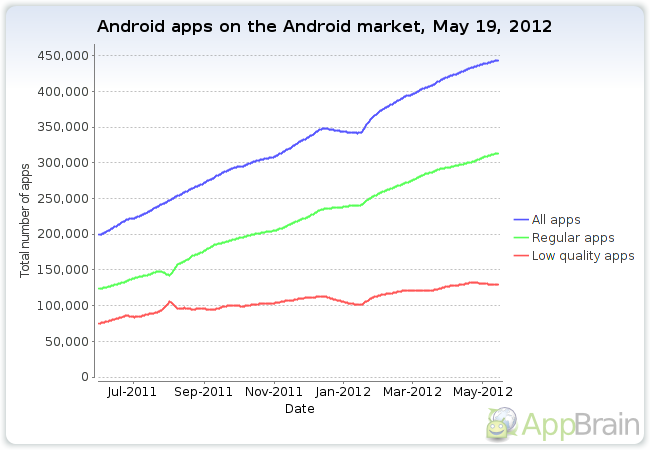
I Markets rendono disponibili direttamente sul cellulare attraverso un portale centralizzato e autorizzato, molte applicazioni, con vantaggi in termini di sicurezza e di visibilità dei prodotti pubblicati. Si tratta sicuramente di una grande opportunità per uno sviluppatore, che una volta sviluppato un buon prodotto deve spendere meno energie e risorse per rendere disponibile il proprio software al pubblico.

Questo approccio è stato un successo, ormai adottato da tutte le aziende del settore.

Come vedremo, i Markets hanno anche dato vita a nuove "legal issue" per la gestione di responsabilità e certificati tra sviluppatori, pubblicatori , utenti e aziende , che in determinate situazioni si scontrano ricoprendo lo stesso ruolo in un mercato molto concorrenziale e meritocratico. Proprio questo tema sarà affrontato in maniera approfondita per quanto riguarda la pubblicazione di applicazioni personalizzate sul market Android in quanto punto critico per lo sviluppo del progetto trattato in questo testo.

### Tendenze del mercato

Grazie allo sviluppo dei markets e grazie al notevole successo di queste soluzioni è sempre più evidente la necessità di avere un'applicazione che rappresenti la propria azienda sul market Android e/o sull'App store. Le statistiche parlano di un rapido tasso di crescita del numero di applicazioni disponibili sul market Android che non mette in dubbio la crescente importanza del settore mobile.



Avere una soluzione mobile dei propri servizi rappresenta un grande valore aggiunto per il proprio marchio oltre che un'estensione della propria soluzione web.

Questa tendenza ha accelerato il mercato delle applicazioni mobile, al punto di aumentare l'aspettativa dell'utente, che si aspetterà di trovare, per una grande azienda, soluzioni mobile, sia su Apple App-store che su Google Play, quindi sia per IOS che per Android. Questo da vita ad un tema molto vicino a quello trattato in questo testo, lo sviluppo cross-platform di applicazioni mobile.

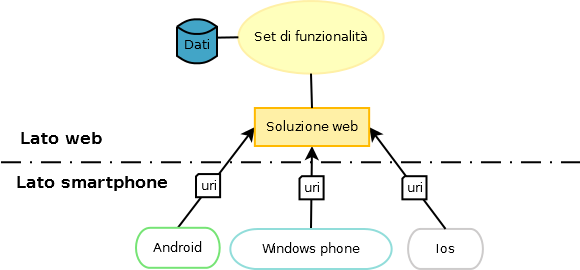
Molte testate giornalistiche ad esempio, sono venute incontro a questa esigenza estendendo i loro quotidiani al settore mobile attraverso queste soluzioni tecniche.

Lo sviluppo Cross-platform risulta come una soluzione unificata e a costo ridotto per risolvere lo stesso problema (rendere disponibili i propri servizi) su diverse piattaforme che non condividono strumenti di sviluppo e linguaggi di programmazione.

Proprio lo sviluppo di applicazioni cross-platform basate su web (web-app, web-based app, webview) può essere un bell'esempio per trattare di come è possibile sviluppare un'applicazione mobile per renderne il codice sorgente riutilizzabile per diversi clienti o diversi utilizzatori, quindi **"personalizzabile"**.

Web-app un ottimo esempio

Di fatto una web app mobile viene sviluppata in due componenti principali, un lato web e sullo smartphone un lato client. Lo sforzo in questa situazione si concentra quasi completamente lato web, sviluppando interfaccia e funzionalità con linguaggi che daranno come output pagine web compatibili con tutti gli smartphones dell'attuale generazione. Esistono frameworks di sviluppo per web app pensati appositamente per questo, che realizzano pagine web per interfacce mobile.



In questo modo la parte client delle applicazioni, che sarà installata sullo smartphone risulta molto semplice. In una tecnologia o in un'altra (IOS, Android etc...), l'essenza del client è una chiamata verso il browser passando come parametro l'url del servizio web, magari con maschere o variazioni sulle funzionalità del browser stesso (Click Handling, Gesture Handling etc...).

Si delinea che la struttura parte client di un applicazione di questo tipo può essere banalmente riutilizzata. Il codice funzionale infatti sarà sempre lo stesso, a cambiare sarà l'unica parte che ha necessità di comunicare con sistemi esterni, in questo caso l'url del servizio web.

E' facile in questo modo capire che il sorgente client di un'applicazione di questo tipo è riusabile ogni volta che c'è la necessità di visualizzare un nuovo servizio web. Una possibile soluzione per il riutilizzo del codice funzionale è l'inserimento di un file di configurazione, che raccoglie i parametri che poi saranno personalizzati.

Durante la personalizzazione dell'applicazione, sarà solamente il file di configurazione ad essere variato, facendo rimanere invariata la parte funzionale.

In questo semplice esempio potremmo realizzare un file di configurazione che contenga l'url del servizio web.

### Pattern di applicativi android

Dall'esempio che è stato appena affrontato e dalle applicazioni pubblicate sul market Android, si può notare in una visione più generale, come gli applicativi mobile possano essere raggruppati in pattern o categorie.

Alcune potrebbero essere:

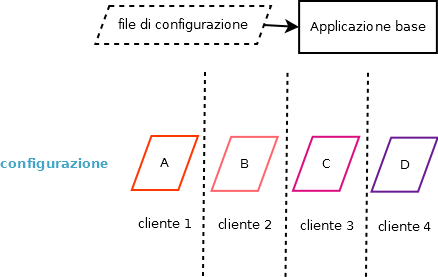
* Gallerie
* Applicazioni che fanno uso di api Web predefinite
* Applicazioni di offerte commerciali
* Applicazioni di news Rss
* Webview / Web App ...

Ogni pattern, visto come classe di applicazioni che realizzano la stessa funzionalità, può essere implementato attraverso un codice funzionale comune e poi personalizzato al volo in maniera guidata con l'uso di form o interfacce utente.

La personalizzazione di questi pattern, come nell'esempio delle webview (web-app,etc..), andrà ancora una volta a riguardare il file di configurazione ed eventualmente le risorse grafiche.

*Un esempio di dati che potrebbero essere inseriti nel file di configurazione e di volta in volta personalizzati potrebbero essere i dati fiscali del cliente, i dati di brand come logo e nome dell'applicazione.*

Questo suggerisce che potrebbe essere utile riutilizzare il sorgente di un' applicazione "BASE" rilasciato con istanze diverse del file di configurazione a seconda del cliente richiedente.

**

**A questo punto risulta chiaro il concetto di personalizzazione di applicazioni Android:** riusare il codice sorgente di un'applicazione per diversi clienti con necessità simili attraverso , ad esempio, un file di configurazione d'istanza.

Questo apre la valutazione e l'analisi di una possibile implementazione in ambiente Android.

## Personalizzazione di Applicazioni Android

### Principi di funzionamento

Ora che abbiamo chiarito quale sia il significato di personalizzazione di applicazioni in ambiente Android, vediamo quali sono i passi per implementare questa tecnica.

Gli argomenti che vanno affrontati e analizzati per valutare la fattibilità sono:

* Sorvolata generale sul Sdk Android
* File di configurazione per applicazioni Android
* Signing Issues in Android
* Fattibilità su piattaforme web

L'idea è di implementare un Web Service che permetta di personalizzare applicazioni Android al volo attraverso un interfaccia grafica.

Il Web Service farà uso di un sorgente di applicazione Android "BASE" preparato in modo da avere un file di configurazione da cui trarre tutte le informazioni per il funzionamento.

L' interfaccia grafica del Web Service permetterà di raccogliere tutte le informazioni necessarie per riempire il file di configurazione.

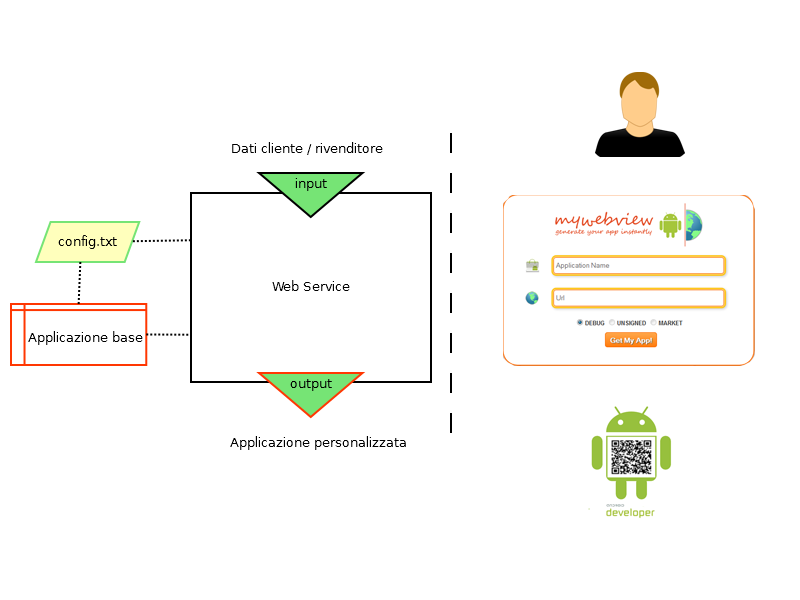
Una volta riempito il file di configurazione dell'applicazione Android "BASE", questa viene compilata e rilasciata al richiedente direttamente attraverso l'interfaccia del Web Service.

L'esempio che verrà trattato è la personalizzazione di una WebView, quindi la personalizzazione di un'applicazione che permetta la visualizzazione di una pagina web, che potrebbe essere una pagina web di un particolare servizio dedicato al mobile (web-app), una pagina privata per la generazione di una sorta di "preferito" o una qualsiasi pagina web.

Proprio le webview /webapp come già detto si addicono molto al tema trattato per il basso numero di parametri e quindi la veloce e semplice implementazione degli script per la gestione del file di configurazione e dell'interfaccia grafica.

Il principio di funzionamento della piattaforma rimane tale anche in caso di applicazioni più complesse, quindi appartenenti ad un'altro pattern.

Nei casi più complessi andranno trattate con particolare attenzione la modularità dei sorgenti del Web Service e l'interfaccia grafica, che dovrà essere adatta a raccogliere molte informazioni senza stressare troppo l'utente.

****Caso d'uso Esempio 1 WebView:**

*Utente A ha un'azienda web e vuole pubblicare un suo servizio web mobile sul market Android a costo ridotto senza riscrivere l'applicazione in modo nativo. Si reca sul Web Service, inserisce l'url del suo servizio web e il nome dell'applicazione. A questo punto il sistema Personalizza l'applicazione Base (Webview in questo caso) e Utente A ottiene senza sforzo un modo per essere nel market Android senza nuovi costi.*

**Caso d'uso Esempio 2 Applicazione Offerte Commerciali:**

*Utente A vuole sviluppare un'applicazione che mostri offerte commerciali ai suoi clienti a basso costo: si reca sul Web Service , utilizza l'interfaccia grafica, inserisce i suoi dati commerciali, il feed rss del suo suo sito e in pochi istanti genera l'applicazione di cui aveva necessità senza conoscere Eclipse, Tool di sviluppo o contattare Aziende Specializzate.*

*--*

*L'esempio potrebbe estendersi per altri tipi di applicazioni, cambierà come detto in precedenza l'interfaccia grafica del Web Service e l'applicazione base.*

### build-simplified.pngtools e sdk android

Risulta necessario analizzare, almeno in una visione generale, come funzionino i Tools e gli strumenti di sviluppo per applicazioni Android.

Le applicazioni Android vengono sviluppate in linguaggio Java e con descrittori xml per i dati di installazione.

Lo sviluppo del sorgente può essere fatto attraverso l'uso di IDE come Eclipse, oppure senza IDE. Per quanto riguarda la parte di compilazione e generazione dei pacchetti invece si fà riferimento agli strumenti rilasciati da Google stesso, l'Android SDK.

L'android SDK ha, tra i vari compiti, l'obbiettivo di generare una variante del pacchetto JAR preparato per sistemi Android, denominato APK. In questo pacchetto, come nel progetto sorgente risiedono le risorse grafiche, alcune risorse testuali, certificati e i sorgenti compilati per la JVM presente su sistema Android, una versione riprogettata di JVM denominata Java Dalvik Virtual Machine. L'apk è il frutto di tutto lo sforzo, andrà installato sui device e conterrà la nostra applicazione pronta all'uso.

Il Web Service dovrà utilizzare questi tool in modo molto simile a quello che viene fatto da Eclipse all'atto della generazione del package.

Un tool che è utilizzato per la generazione dei pacchetti APK è ANT, strumento utilizzato anche per lo sviluppo di pacchetti JAR o WAR in caso di sviluppo in piattaforma JEE e Tomcat. Sarà necessario preparare adeguatamente anche questo strumento sulle macchine che ospiteranno il Web Service.

La necessità di usare questi tool, chiarisce che nel caso di sviluppo di un Web Service come quello di questo testo, non può essere usata una normale piattaforma di hosting, ma la scelta deve ricadere, per numerosi motivi su una macchina virtuale su cui è possibile installare questi servizi e configurarli in modo autonomo.

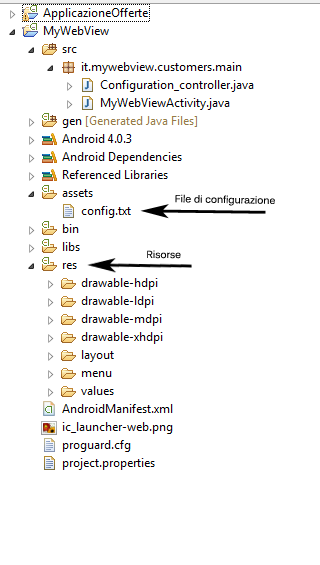
### File di configurazione - Assets

Ora bisogna vedere se è possibile e se è presente una soluzione comoda, possibilmente supportata dalle librerie Android per utilizzare file di configurazione nell' applicazione base.

I dati devono essere presenti e coerenti al momento della compilazione, la soluzione più immediata è quindi costruire un file di testo con un formato definito ad hoc.

Le librerie Android supportano questo modo di operare attraverso il concetto di Assets.

Ecco quanto riportato dalla documentazione Android Developer:

Resources are an integral part of an Android application. In general, these are external elements that you want to include and reference within your application, like images, audio, video, text strings, layouts, themes, etc. Every Android application contains a directory for resources (res/) and a directory for assets (assets/). Assets are used less often, because their applications are far fewer. You only need to save data as an asset when you need to read the raw bytes. The directories for resources and assets both reside at the top of an Android project tree, at the same level as your source code directory (src/).

The difference between "resources" and "assets" isn't much on the surface, but in general, you'll use resources to store your external content much more often than you'll use assets. The real difference is that anything placed in the resources directory will be easily accessible from your application from the R class, which is compiled by Android. Whereas, anything placed in the assets directory will maintain its raw file format and, in order to read it, you must use the [AssetManager](http://developer.android.com/reference/android/content/res/AssetManager.html) to read the file as a stream of bytes. So keeping files and data in resources (res/) makes them easily accessible.

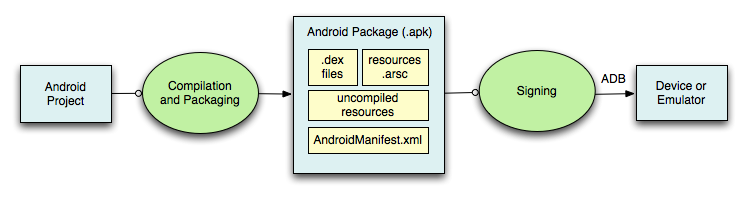
I progetti di applicazioni Android hanno un'organizzazione ben definita delle risorse, che sono organizzate e visibili direttamente durante lo sviluppo dell'applicazione Base.

E' possibile quindi astrarre dal filesystem ed utilizzare direttamente le librerie Android per accedere al file di configurazione come fosse una qualsiasi risorsa del progetto.

L'applicazione Base quindi, oltre a realizzare la propria semantica, dovrà integrare alcune funzionalità per leggere i dati dal file di configurazione e renderli disponibili al resto del pacchetto. Và implementata quindi un'interfaccia per la lettura dei dati dal file di configurazione.

Avere un file di configurazione su cui è definita un'interfaccia di lettura ed essere un'applicazione Android sono i requisiti perchè l'applicazione sia quindi personalizzabile.

### Packaging e signing dell'apk

Nell'introduzione ai Tool Android è stato presentato il formato di packaging che viene utilizzato per le applicazioni Android: il formato APK. Ora Analizziamo più in dettaglio quali sono le caratteristiche dei file Apk da curare per il rilascio di un'applicazione personalizzata utilizzando la soluzione di questo testo.

I file APK sono il risultato della compilazione e del packaging da parte dell'SDK Android attraverso un processo un pò complicato la cui descrizione lascio alla guida ufficiale: http://developer.android.com/guide/developing/building/index.html.

I sorgenti Java del progetto vengono compilati, nell' Apk finale quindi non saranno visibili in chiaro, mentre altre risorse come gli Assets rimangono in chiaro e visibili.

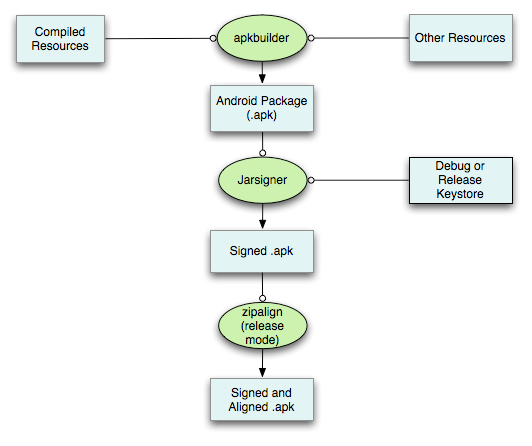
Dopo il packaging dell' Apk, è necessaria una procedura di firma.

Questa procedura di firma è il punto critico di tutto il progetto. Banalmente si potrebbe pensare di poter compilare una volta per tutte il progetto base e di personalizzarlo aprendo il file APK e cambiando i parametri del file di configurazione, visto che questa risorsa è in chiaro. Questa procedura non è possibile ed è sconveniente per due motivi:

1. Il pacchetto deve essere firmato per poter essere installato su dispositivi Android. Quindi la modifica dopo la firma non è possibile.
2. Ogni applicazione, deve avere un package unico in tutto il Market e unico sul dispositivo Android. Questo rende necessaria la generazione in maniera dinamica e automatica del nome del package dell'applicazione personalizzata . Questa procedura è possibile solo prima della compilazione.

Detto questo è necessario personalizzare il package name, il file di configurazione e solo in seguito compilare. Questo intende che il processo di generazione dell'applicazione personalizzata costa di una compilazione per istanza. Costo affrontabile per un'applicazione di piccole dimensione, ma potenzialmente un problema nel caso di applicazioni di grandi dimensioni.

#### Signing Issue

Ora analizziamo i punti critici della procedura di firma. La firma delle applicazioni Android è obbligatoria per poter installare l'applicazione.

La procedura di firma viene portata a termine da un tool integrato nell'sdk che si chiama Jarsigner. Questo tool prende in input una chiave Privata e l'apk non firmato, firma con la chiave privata il pacchetto che poi potrà essere rilasciato al pubblico sul market o attraverso web.

L'SDK usa un meccanismo automatico di generazione di chiavi, le chiavi generate da questo meccanismo vengono chiamate chiavi di Debug (Debug Keystore). Queste chiavi hanno dati fittizi e vengono generate per poter testare la propria applicazione durante la fase di test.

**Con le chiavi di Debug non è possibile rilasciare nessuna applicazione sul market**. A questo punto per poter caricare l'applicazione sul market è necessario generare una chiave privata (Release Keystore) con cui firmare le applicazioni personalizzate.

|  |  |
| --- | --- |
| Tipo di chiave | Modalità di pubblicazione |
| Debug Keystore | Web e Diretta |
| Release Keystore | Market, Web e Diretta |

\* Con web si intende la pubblicazione dell'applicazione personalizzata direttamente dal sito dell'utente, tramite url o qr-code.

Questo è il principale problema da affrontare per questo tipo di sistema, con conseguente scelta progettuale.

##### Gestione di chiavi private

La principale scelta del progetto sta quindi nella soluzione a questo problema. Si tratta di riuscire a trovare una soluzione elegante che permetta all'utente del Web Service di poter personalizzare l'applicazione e poterla pubblicare sul market, in modo semplice ma senza violare le procedure di sicurezza da cui è nata l'idea di firma e crittografia asimmetrica.

### Soluzioni e scelte progettuali

Detto quindi che con le chiavi di Debug, generate automaticamente non è possibile pubblicare nulla sul market, vediamo quali sono le possibili scelte progettuali per poter procedere.

|  |  |
| --- | --- |
| 1. | Generazione al volo di una Release Key |
| 2. | Upload della Release Key |
| 3. | Rilascio dell'Apk non firmato e rilascio di un tool per la firma assistita |
| 4. | Rilascio dell'Apk firmato dal proprietario del Web Service |

###### 1. Generazione al volo di una Release Key \*

Si tratta di raccogliere altri dati oltre a quelli necessari per la compilazione del file di configurazione. I dati minimi in questione sono una password e un nome. Questi dati verranno utilizzati per generare una Release Keystore dell'utente con cui verrà firmato il pacchetto Apk.

|  |  |
| --- | --- |
| Pro | Contro |
| * Semplice da implementare * Non richiede conoscenze avanzate all'utente * Accounting dell'utente * Riuso chiave per altre applicazioni | * Ci si accolla la gestione di chiavi private di terzi. (Legal Issue?) * E' concettualmente sbagliato generare chiavi private di terzi |

\* possibile implementazione con chiave privata usa e getta?

###### 2.Upload della Release Key

Simile alla soluzione precedente, ma sarà l'utente a caricare la propria chiave già generata.

|  |  |
| --- | --- |
| Pro | Contro |
| * Semplice da implementare * Riuso chiave per altre applicazioni | * Condividere una chiave privata è sbagliato * Richiede conoscenza all'utente di gestione chiavi |

###### 3. Rilascio dell'Apk non firmato e rilascio di UN tool per la firma assistita

Il Web Service rilascia l'Apk non firmato e un tool con interfaccia grafica in grado di assistere alla creazione della firma. Senza memorizzare dati personali su sistemi esterni da quello dell'utente.

|  |  |
| --- | --- |
| Pro | Contro |
| * Rispetta tutti i principi della crittografia Asimmetrica * Riuso della chiave privata | * Soluzione non immediata * Dimensione del tool accettabile? * Manutenzione del tool accettabile? |

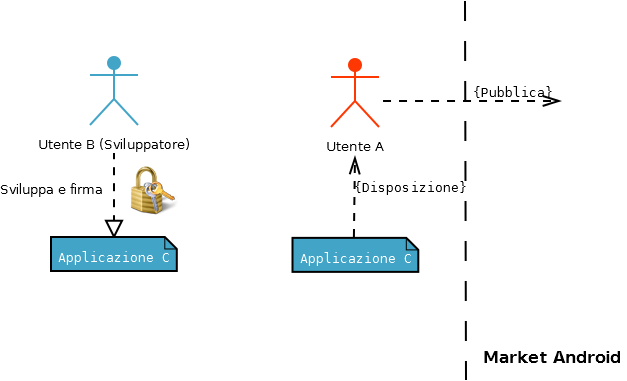
###### 4. Rilascio dell'Apk firmato dal proprietario del Web Service

Il Web Service utilizza una Release Key generata dal proprietario del Web Service, ad esempio Andrea Pola. Firmando le applicazioni personalizzate dei clienti con questa chiave.

|  |  |
| --- | --- |
| Pro | Contro |
| * Semplice da implementare * Semplice da mantenere | * Non rispetta i principi della crittografia asimmetrica |

##### APProfondimento su Android market e firme.

Come detto all'inzio ecco un'approfondimento sui pregi e difetti della pubblicazione su Google Play. Senza scendere in confronti con la piattaforma Apple, vediamo quali sono i punti che ci interessano per l'implementazione di servizi di personalizzazione di applicazioni Android.



Il market Android permette, allo stato attuale delle cose, la pubblicazione da parte di un registrato al Market di applicazioni firmate da terzi. Per essere chiari, l'applicazione C sviluppata da B e firmata da B può essere pubblicata dall' utente A sul market senza problemi. Questa scelta a prima vista non rispetta i principi che stanno alla base delle firme digitali. Probabilmente questa scelta risulta una scelta commerciale, pensata per far accedere al market figure che non sono direttamente gli sviluppatori delle applicazioni.

Questa politica quindi non scarta nessuna delle scelte progettuali presentate prima. Ma approfondiamo la scelta progettuale n.4:

Poniamo il caso che un utente del Web Service voglia generare un'applicazione di un sito web di cui lui non è il proprietario, ad esempio una testata giornalistica. L'utente va sul Web Service che personalizza WebView (il caso studio) e genera un'applicazione di questo sito web pronta per il market, secondo la scelta progettuale 4, firmata dal proprietario del Web Service. A questo punto l'utente pubblica sul market.

**In caso di denuncia, che responsabilità ha il firmatario?**

Il firmatario in questo caso è il proprietario del Web Service, che è una persona diversa dalla persona che pubblica sul market l'applicazione.

E' chiaro come questo processo sia in contraddizione con i principi legali della firma digitale. E' evidente inoltre che questo processo e i servizi di personalizzazione delle applicazioni debbano essere sostenuti da una "licenza d'uso" e un comparto legale accurato.

### Stato dell'arte

Ora vediamo quali sono le soluzioni commerciali presenti sul web che forniscono servizi di personalizzazione di applicazioni. Dal punto di vista commerciale queste soluzioni vengono nominate come "Application Builders" o nomi simili. Soluzioni di questo tipo stanno comparendo e sviluppandosi molto velocemente al momento della scrittura di questo documento.

Valutiamo quali scelte progettuali sono state applicate e come è stato affrontato il problema precedente, limitandoci al contesto Android in caso di soluzioni multipiattaforma.

**App Inventor**

<http://appinventor.mit.edu/>

Ritornato da poco, dopo una storia un pò difficoltosa. Servizio lanciato ufficialmente da Google tramite i Google Labs, poi chiusi,ora è disponibile grazie al suo rilascio Open Source e grazie al Mit che ha preso in mano il progetto e rilanciato da poco come Beta. Si tratta di un progetto per lo sviluppo di applicazioni, è stato il primo nel suo genere. E' un sistema un pò diverso da quello analizzato in questo testo, è di fatto un tool per la programmazione visuale ad alto livello.

|  |  |
| --- | --- |
| Pro | Contro |
| * Potente * Tendenzialmente un Tool per lo sviluppo con interfaccia grafica * Alte potenzialità didattiche | * Non permette il rilascio di applicazioni nel market * Servono conoscenze di programmazione |

**Buzz Touch**

http://www.buzztouch.com/

Progetto per il building di applicazioni, semplice, multipiattaforma, ma che rilascia il codice sorgente e non l'eseguibile. Il sorgente viene lasciato con istruzioni per firma compilazione e pubblicazione. Il nome dell'applicazione deve essere disponibile, in sostanza è presente un meccanismo di mutua esclusione sul nome dell'applicazione (dovuto alla gestione del package name probabilmente).

|  |  |
| --- | --- |
| Pro | Contro |
| * Implementazione semplice * Multipiattaforma | * Non risolve il problema * Mutua esclusione sul nome dell'applicazione |

**The AppBuilder.com**

http://www.theappbuilder.com

La versione per Store/Market è a pagamento quindi non ho verificato quali sono meccanismi di gestione delle chiavi siano presenti. L'applicazione generata viene pubblicata attraverso il loro Account nel market. Probabilmente viene fatto un controllo manuale dell'applicazione finale. La firma è a nome di The AppBuilder. Molto potente è il sistema di gestione degli aggiornamenti dell'applicazione. In sostanza l'applicazione è sempre aggiornata grazie al loro sistema "In the Cloud". Probabilmente le implementazioni sono attraverso Web App.

Di fatto stai acquistando un'applicazione loro tramite abbonamento.

|  |  |
| --- | --- |
| Pro | Contro |
| * Potente * Applicazioni Sempre aggiornate * Multipiattaforma * Soluzione commerciale valida | * Poca autonomia * Aggira il problema * Pubblicazione sul loro Account * Controlli manuali |

**Apps-Builder**

http://www.apps-builder.com/

Apps Builder è sicuramente la soluzione più interessante, inoltre è italiana, sviluppata al politecnico di Torino e poi diventata azienda. E' multipiattaforma e ha un bel set di pattern a disposizione, con interfaccia grafica intuitiva e potente. Il punto critico di questa realizzazione è che nel reparto Android l'applicazione viene firmata a nome dell'intestatario dell'azienda.

Mentre le soluzioni commerciali di prima non aderiscono a nessuna delle scelte progettuali definite in questo testo questa adotta la scelta progettuale n.4 .

E' stata lanciata una soluzione per partner che potrebbe avvicinarsi alla scelta progettuale n.2

|  |  |
| --- | --- |
| Pro | Contro |
| * Multipiattaforma * Ampia scelta * Interfaccia grafica ben realizzata * Primo a lanciare una soluzione B2B | * Firma dell'intestatario dell'azienda |

Concludendo questa parte sullo Stato dell'arte, possiamo vedere che le soluzioni in commercio utilizzano diverse modalità per aggirare o risolvere il problema di firma trattato. Sarebbe interessante realizzare una soluzione che risolva il problema delle firme dal punto di vista tecnico e nel rispetto dei principi della firma digitale. Anche in vista di possibili nuove politiche del market Android.

## Webservice di demo

Analizzati, requisiti e problematiche, vediamo se ci sono problematiche di realizzazione da affrontare.

Requisiti in ambiente web:

|  |  |
| --- | --- |
| Ambiente JDK | OK |
| ANT | OK |
| Android SDK | OK |
| LAMP | OK |

Problematiche tecniche

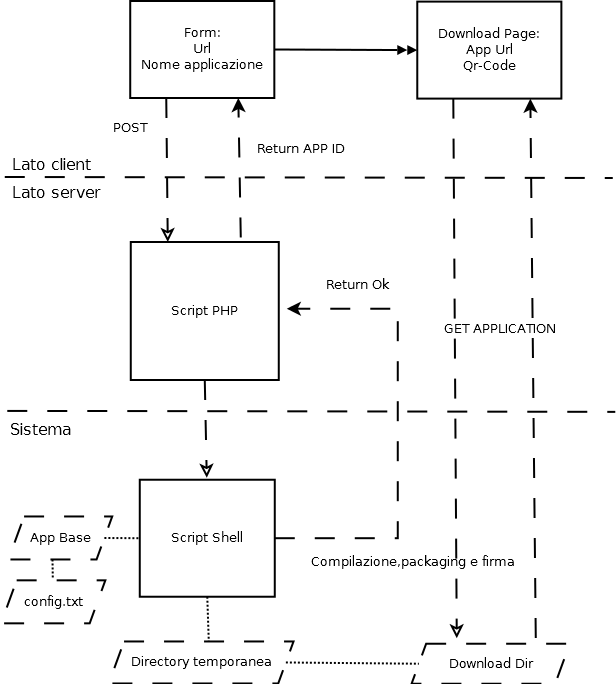
|  |  |
| --- | --- |
| Generazione dinamica di package name | OK |
| Compilazioni concorrenti della stessa applicazione base | OK |
| Modalità di firma | DEBUG |

Generazione dinamica di package name: questa problematica è stata risolta facendo "search and replace" tra i sorgenti dell'applicazione base. Parte del package name base viene sostituito con una stringa generata dai dati forniti dall'utente.

Compilazioni concorrenti della stessa applicazione base: questa problematica è stata risolta con l'uso di directory temporanee generate in funzione dei dati forniti dall'utente.

La versione di demo del progetto, personalizza un'applicazione fermandosi alla firma in modalità Debug.

Schema di funzionamento:



### Implementazione

Il Web Service si mostra con un'interfaccia grafica molto semplice, si tratta di un form con 2 input, uno per il nome dell'applicazione e l'altro per l'url. Questi due dati vengono inviati tramite POST ad uno script PHP che genererà un nome temporaneo dai due dati in ingresso, questo sarà l'identificativo dell'applicazione e della directory temporanea ad essa assegnata. L' id è generato con funzione md5 a partire da dagli input , per evitarne la previsione è stata inserita anche una stringa in modo da realizzare una semplice funzionalità di salting.

Nell'attesa di una risposta da parte del server, il form viene sostituito da una pagina di attesa.

Questi 3 dati: url, nome e tmp\_dir verranno passati ad uno script shell, che gestirà le directory per la mutua esclusione e riempirà il file di configurazione dell'applicazione base. Fatto questo lo script chiama ANT per la generazione dell'apk. ANT e l'sdk android ,chiamato a sua volta, utilizzano per la firma dell'Apk una chiave di Debug precostruita e depositata in una directory predefinita.

Una volta concluso il processo di compilazione e packaging, l'Apk generato viene depositato in una directory di download.

A questo punto lo script Shell ha concluso, l'esecuzione ritorna allo script PHP che restituirà risposta alla chiamata POST avviata dal form. La risposta conterrà il codice identificativo dell'applicazione appena generata. Nella pagina di attesa sarà inserito un link ad uno script che si occuperà di permettere il download l'APK. Infine viene visualizzato un qr-code che punta all'applicazione appena creata per il download diretto su cellulare.

Per la comunicazione script PHP -> Shell è stata utilizzata una chiamata exec. Questo deve essere fatto con cura, poichè i dati passati alla shell sono variabili inserite dall'utente, questo significa che potrebbe essere inserito codice malevolo o comandi shell non previsti. Per questi motivi sono stati usati filtri forniti dalle librerie PHP per l'Excaping di caratteri.

|  |  |
| --- | --- |
| Input utente | Url, Nome Applicazione |
| Input shell script | Url,Nome Applicazione, Directory Temporanea |
| Output Shell Script | Stringa di conferma |
| Output PHP Script | APP ID |
| Output Utente | APK URL, QR-CODE |

Nel caso di file di configurazione più complessi, dovrà essere messo appunto uno script shell, in grado di prendere in input altri parametri di personalizzazione oltre quello della directory temporanea che comunque va mantenuto per risolvere i problemi di mutua esclusione.

La parte web è stata realizzata con chiamate asincrone post, con l'uso della libreria Jquery.

La demo è visibile all'indirizzo: www.mywebview.com

## Possibili Sviluppi

1. Analisi del diritto d'autore nei Markets
2. Approfondimento sui Contratti dei Markets
3. Analisi su contratti d'uso di soluzioni di tipo "Application Builder"
4. Personalizzazione di Applicazioni in ambienti Multipiattaforma
5. Implementazione di una delle scelte progettuali elencate
6. Implementazione di una soluzione senza compilazione live
7. Implementazione di un servizio di Personalizzazione Applicazioni multipiattaforma Open Source
8. Analisi approfondita di App Inventor
9. Plugin per la gestione di personalizzazione di App in ECLIPSE

1. Fonti statistiche su applicazioni e market android : http://www.appbrain.com [↑](#endnote-ref-2)