Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea Magistrale in Informatica

Analisi e progettazione di architetture private cloud per servizi di push notification verso dispositivi mobile

Tesi di Laurea in Laboratorio di Applicazioni Mobili

Relatore:
Prof. Luciano Bononi

Presentata da:
Fabrizio Nuzzo

II Sessione
Anno Accademico 2011-2012

" Anything Worth Doing Is Worth Overdoing"

(Mick Jagger)

Introduzione

Negli ultimi anni i sistemi operativi per dispositivi mobile hanno avuto un evoluzione esponenziale, suffragata da un forte interesse delle aziende presenti con le loro applicazioni. Analizzando l'attuale contesto tecnologico è facile notare come la maggior fetta di mercato è occupata da sistemi evoluti che hanno saputo rinnovarsi ed estendere le possibilità implementative da parte di sviluppatori terzi. Oltre ad un incremento delle funzionalità offerte tramite le API ufficiali ed una documentazione sempre più complete rilasciata dalle case produttrici, è nato un mercato parallelo di aziende che sviluppano framework esterni che aiutano lo sviluppatore a realizzare le proprie applicazione o offrono servizi per estendere le funzionalità delle stesse. In questa tesi verrà preso in considerazione un aspetto che nell'ultimo periodo ha riscosso un grande interesse sia da parte delle case produttrici, che da parte degli sviluppatori: le push notification. Con questo termine si fa riferimento a un messaggio asincrono inviato da un server di terze parti verso il dispositivo. Tramite questa tecnologia le applicazioni possono continuare a interagire con l'utente anche quando non sono in primo piano (foreground) a diretto contatto con esso . Ricordiamo che inizialmente le applicazioni per smartphone sviluppate da terzi avevano grosse limitazioni, potevano interagire con l'utente solo quando erano in foreground e l'unico modo per aggiornare i contenuti o fruire di servizi da un web service era sottomessa alla volontà dell'utente che richiedeva l'aggiornamento della risorsa (tecnologia pull). Sia in un contesto business che consumer è facile percepire l'utilità che un servizio del genere offre. Per esempio i maggiori social network usano le push notification

per notificare l'utente di un cambiamento di stato o di un messaggio ricevuto nel proprio account. Le aziende possono inviare tramite questa tecnologia delle offerte o informare l'utente di un evento.
Il Reparto Android di Google ha rinnovato completamente la sua tecnologia di notifiche push presentando nel Google I/O session di luglio 2012 [1] il nuovo servizio Google Cloud Messanging for Android (GDM) rimpiazzando il vecchio servizio Android Cloud to Device Messaging Framework (C2DM) [2]
BlackBerry da circa un anno ha esteso il suo noto servizio di notifiche push: BlackBerry Push Service (BBPS) [3], riservato agli sviluppatori enterprise verso tutti gli sviluppatori. Apple dal canto suo è forte del suo solido servizio Apple Push Notification Service (APNs) [4] e negli ultimi due anni ha revisionato e semplificato la gestione delle notifiche sia da parte dello sviluppatore che da parte dell'utente finale. La volontà di apertura, da parte di queste grandi aziende, della loro tecnologia che possiamo definire cloud to device una volta riservata ad un uso esclusivo da parte di esse (notifiche email, aggiornamenti di sistema ecc.) apre nuove orizzonti per l'implementazione di servizi cloud per dispositivi mobile.
Questa tesi ha come scopo l'analisi approfondita dei diversi servizi di push notification per dispositivi mobile e la progettazione di una componente integrabile in una private cloud per l'inoltro e la gestione delle push notification. L'idea di un private cloud è giustificata dal fatto che il servizio che si vuole implementare utilizza gli identificativi univoci dei dispositivi,che se abbinata all'identità do ogni singolo possessore del dispositivo rappresenta un dato riservato che molte aziende sono ostili a consegnare ad altri fornitori di servizi public cloud per sistemi mobile. Un uso fraudolento di questo servizio potrebbe ledere all'immagine dell'azienda o far perdere credibilità all'appli-

[1] `https://developers.google.com/events/io/sessions#day-1`
[2] `https://developers.google.com/android/c2dm/`
[3] `https://developer.blackberry.com/services/push/?CPID=PUSHAPI00`
[4] `http://developer.apple.com/library/mac/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html`

cazione. Un altro obbiettivo di questa tesi è di trovare soluzioni architetturali e implementative che non siano adatte solo alle quattro piattaforme prese in esame, ma il sistema deve essere progettato in un ottica di espansione dei suoi confini e funzionalità, poiché nell'attuale contesto mobile l'interoperabilità è più importante della specializzazione. L'esposizione di questa tesi è organizzata come segue: Nel capitolo 1 del presente documento, descrivo in dettaglio la tecnologia push, ripercorre l'evoluzione di questo tipo di servizio nelle varie piattaforme ed elenca i benefici e i rischi associati. Procedendo con la stesura del capitolo 2 viene presentata la definizione di Cloud Computing con la descrizione dei modelli e visti alcuni dei problemi legati a questo trend. Nel capitolo 3 verrano visti in dettaglio casi di studio di private cloud che integrano la tecnologia push e servizi public cloud che basano la loro offerta su questa tecnologia. Nel capitolo 4 il focus verrà spostato verso le piattaforme mobile; per ogni piattaforma verrano descritte le caratteristiche generali del servizio push offerto dalla casa produttrice e l'architettura di riferimento. Verrano presi in considerazione i componenti principali che fanno parte dell'architettura, le credenziali necessarie per accedere al servizio e il percorso di una notifica dal server di terze parti fino al dispositivo. Alla fine del capitolo verrano comparate le quattro piattaforme prese in esame. Il capitolo 5 è dedicato alla progettazione del sistema che si vuole realizzare e prevede la formulazione dell'architettura generale, requisiti e casi d'uso; verranno discussi i pattern architetturali e le tecnologie per realizzare un sistema che possa interagire con le piattaforme viste nel capitolo precedente. Il capitolo 6 è la presentazione, con relativa spiegazione, dei punti critici del sistema switchboard; descrive i punti focali del programma, e i dettagli implementativi più importanti. Tale progetto nasce con l'intento di concretizzare questo studio tramite l'applicazione della conoscenza sugli argomenti trattati.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Push Tecnology

In questo capitolo introdurrà il concetto di push notification, applicato nell'ambito mobile con tutti i benefici che questa tecnologia può portare e i rischi che ne possono derivare da un suo uso non corretto.

1.1 Qualche definizione

Una notifica push è un breve messaggio asincrono inviato da un server ad una specifica applicazione installata nel dispositivo finale. [14] Questi messaggi sono ampiamente usati dalle case produttrici per informare l'utente che è disponibile una nuova versione del sistema operativo o che è disponibile un aggiornamento di un'applicazione. Sviluppatori di terze parti potrebbero utilizzare queste notifiche per informare l'utente che è disponibile un aggiornamento nei contenuti dell'applicazione o un evento relativo all'applicazione. Per esempio, le push notification possono essere utilizzate in un applicazione per informare l'utente dell'andamento delle proprie azioni nel mercato azionario o il cambiamento del risultato in un match sportivo. iOS [1] supporta questa tecnologia dalla versione 3.0 del suo noto sistema operativo. Le API sono state rilasciate agli sviluppatori nel 2009 iOS presenta le push notification come text, alerts, badges o una combinazione di questi.

[1] `http://www.apple.com/ios/`

Android [2] supporta questa tecnologia dalla versione 2.2 del suo sistema e le API sono state rilasciate agli sviluppatori nel 2010 Android OS presenta le push notification come toast notification (brevi messaggi che appaiano per pochi secondi sullo schermo e in maniera permanete nella barra di stato del sistema). BlackBerry [3] supporta questa tecnologia dalla versione 4.0 del suo sistema operativo e le relative API sono state rilasciate inizialmente solo per gli sviluppatori BlackBerry Enterprise Server [4] (BES) e nel marzo 2010 sono state aperte anche all'utenza BlackBerry Internet Service [5] (BIS). BlackBerry OS presenta le push notification come text, alerts, badges o una combinazione di questi.

1.1.1 Benefici

Di seguito verranno elencati e descritti i benefici indotti dall'utilizzo di questa tecnologia:

1. Fornire informazioni in maniera efficiente

   Inviare dati tramite la tecnologia push è il modo migliore per informare l'utente finale in tempo reale di possibili aggiornamenti. Inoltre la tecnologia push riduce notevolmente l'impatto della latenza di rete. Dato che le reti wireless hanno larghezza di banda limitata rispetto alle reti cablate, la velocità di trasferimento dati è più lenta. Tuttavia poiché la tecnologia push lavora in background senza l'intervento dell'utente, da punto di vista di quest'ultimo non c'è tempo di attesa. I nuovi dati possono essere sincronizzati e l'utente viene informato solo nel momento in cui il dato è pronto per essere visualizzato.

2. Ridurre il consumo di batteria e di banda

[2] `http://www.android.com/`

[3] `http://www.blackberry.com`

[4] `http://us.blackberry.com/business/software/bes.html`

[5] `http://it.blackberry.com/business/software-and-services/blackberry-internet-service/overview.jsp`

Preservare l'utilizzo di batteria è un aspetto importante nella programmazione di applicazioni per dispositivi mobile; al fine di recuperare dati sempre aggiornati da un server, alcune applicazioni interrogano con frequenza i loro server per determinare se sono disponibili nuovi dati. Questa pratica influisce notevolmente sulla vita della batteria ed inoltre consuma inutilmente banda c che si riflette in costi maggiori per gli utenti con un piano dati a consumo. Tramite la tecnologia Push invece le applicazioni semplicemente rimangono in ascolto fino a quando il server invia loro una nuova notifica.

3. Estende le possibilità di sviluppo

   Tramite la tecnologia Push le applicazioni posso arricchirsi di nuove funzionalità, interagire con più efficienza con i servizi di terze parti e soddisfare un maggior numero di casi d'uso. Ricerche di mercato hanno dimostrato che le applicazioni abilitate alla ricezione di notifiche push vengono utilizzate con più frequenza, questo si trasforma per le aziende in un ritorno di investimento dovuto ad un maggiore coinvolgimento da parte dei propri utenti.

1.1.2 Rischi

Un uso scorretto di questo servizio può portare l'utente a catalogare in maniera unilaterale le push notification come un nuovo canale per far fluire pubblicità indesiderata. [17] Per tutelare l'utente da questa nuova forma di spam le case produttrici con il susseguirsi delle release hanno aggiunto ai loro sistemi la possibilità di disattivare la ricezione delle push notification da parte di ogni singola applicazione. La piattaforma iOS tutela i propri utenti in due modi: avvertendo al momento dell'installazione che l'applicazione necessita del permesso di ricevere notifiche e dalla versione 5.0 del suo sistema operativo tramite il notification center; funzionalità di iOS dove vengono visualizzate tutte le applicazioni abilitate alla ricezione delle notifiche; da questa semplice schermata è possibile per ogni applicazione scegliere diverse

opzioni. Gli utenti Android invece possono accorgersi di questa evenienza durante l'installazione, quando vengono visualizzati i permessi richiesti dall'applicazione, e decidere di non installarla. Dalla versione 4.1 (Jelly Bean) tramite app's info screen (schermata che raccoglie le informazioni di ogni singola applicazione) nel gestore delle applicazione è possibile negare il permesso di ricevere notifiche alle applicazione che ne abusano. BlackBerry invece integra fin dalla prima versione la possibilità di disattivare le push notification delle applicazioni. I permessi dell'applicazione vengono visualizzati durante il primo avvio e successivamente è possibile cambiare la configurazione dal menu avanzato della stessa.

Per quanto lo Spam influisca negativamente sull'user experience, non è paragonabile ai danni prodotti da un attacco di tipo phishing utilizzando la tecnologia push. E' giusto precisare che non tutte le piattaforme sono predisposte ad attacchi del genere. Il problema riguarda solo due delle piattaforme prese in considerazione (Android, BlackBerry). Non a caso si tratta di piattaforme che permettono personalizzazioni nell'aspetto grafico delle view (veste grafica di una schermata). Per capire meglio il problema bisogna prima di tutto fare una distinzione tra OS-Controlled e App-Controlled notification views. iOS rientra nella prima categoria, poiché nessuna personalizzazione è permessa ai messaggi di notifica, le notifiche sono gestite interamente dal sistema quindi con grafica standard. BlackBerry OS e Android invece appartengono alla seconda categoria. Il sistema fa solo da tramite nel momento della ricezione della notifica poi lascia il controllo all'applicazione proprietaria. In tutte due le piattaforme è possibile personalizzare l'aspetto grafico della view e proprio per questa particolarità è possibile generare delle view che rispecchiano le schermate di login dei social network o di altri servizi e trarre in inganno l'utente. Su questo argomento rilevante è l'articolo [20].

Capitolo 2

Introduzione al Cloud Computing

Dopo aver introdotto nel Capitolo 1 la Push Tecnology, in questo capitolo introdurrò il concetto di Cloud computing. Inizieremo con il definire questo concetto poi in 2.2 verranno elencate le caratteristiche e in confrontati diversi tipi di modelli di servizio.

2.1 Definizione

In accordo con NIST [1] possiamo definire il cloud in questo modo:

" Il cloud computing è un modello per abilitare un accesso conveniente e su richiesta a un insieme condiviso di risorse computazionali configurabili (ad esempio reti, server, memoria di massa, applicazioni e servizi) e possono essere rapidamente procurate e rilasciate con un minimo sforzo di gestione o di interazione con il fornitore del servizio." [19] Questo modello promuove la disponibilità del servizio ed è caratterizzato da cinque fattori essenziali, tre modelli di servizio e quattro modelli di rilascio. I progressi nelle prestazioni dei componenti digitali hanno provocato un enorme aumento della porta-

[1] `http://www.nist.gov/index.html`

ta degli ambienti IT, e di conseguenza, è nata l'esigenza di poterli gestire uniformemente in un'unica "nuvola" (cloud).

2.2 Caratteristiche essenziali:

Di seguito un elenco delle caratteristiche che contraddistinguono questo tipo di modello

1. On-demand self-service: Il consumatore può unilateralmente disporre di capacità di calcolo, come server time e network storage, se necessario, senza richiedere l'interazione umana con il fornitore del servizio.

2. Broad network access: accesso ai servizi del cloud da qualsiasi dispositivo tramite l'uso di standard.

3. Resource pooling: in cui sono raggruppate le risorse di calcolo del provider per servire più consumatori utilizzando un modello multi-tenant, con diverse risorse fisiche e virtuali assegnate in modo dinamico e riassegnate secondo le esigenze dei consumatori;

4. Rapid elasticity: capacità di scalare verso l'alto ma anche verso il basso a seconda di diversi fattori esterni e per periodi di tempo definibili dall'utente garantendo lo stesso modello di business.

5. Measured service: tutte le risorse, dallo IaaS al SaaS sono controllate e monitorate ad uso sia del provider che del consumer.

2.3 Classificazione dei modelli di servizio

Nel corso del tempo si è dibbattuto molto su come catalogare questi nuovi sistemi distribuiti fino ad arrivare ad una classificazione a livelli come descritto di seguito:

1. **IaaS** (Infrastructure as a Service): rappresenta lo strato del cloud di più basso livello e permette di avere come servizi il networking, Server, storage ecc..

2. **PaaS** (Platform as a Service): rappresenta una vera e propria piattaforma applicativa (o infrastruttura applicativa) su cui sviluppare applicazioni "tradizionali" o in modalità SaaS. Tale piattaforma è disponibile e gestibile tramite servizi.

3. **SaaS** (Software as a Service) è l'erogazione di applicazioni stanziate ad end-users esterni capaci di self-provisioning e adattamento alle richieste degli utenti.

E' importante per chiarificare l'argomento dopo aver elencato i 3 modelli di servizio capire chi sono gli attori principali:

- **IaaS:** si rivolge prevalentemente a sistemisti e architetti di infrastruttura.

- **PaaS:** si rivolge prevalentemente a sviluppatori ed architetti di soluzioni.

- **SaaS:** si rivolge ad utenti finali.

2.4 Modello di Rilascio

1. **Private cloud:** Il Private Cloud introduce all'interno di un'azienda i principi del cloud computing. Inizialmente il private cloud era composto interamente da da risorse interne, ma con il rapido evolversi dei bisogni e per soddisfare esigenze sempre più legata e alla mobilità viene combinato con risorse cloud esterne lasciando comunque la totalità del controllo all'azienda. I motivi per cui un'azienda possa ricorrere a questa modalità, in alternativa a quella pubblica, sono diversi [8]:

- L'infrastruttura IT esistente è ben gestita e ben configurata e passare ad un Public Cloud significherebbe perdere gli investimenti già fatti nel costruirla e operarla.
- Determinate applicazioni aziendali richiedono servizi o tecnologie presenti sono all'interno del perimetro definito dal firewall.
- La necessità di aderire a specifiche prescrizioni normative o legislative sul controllo dei dati renderebbe particolarmente complessa l'adozione di una tecnologia di Public Cloud.

2. **Public cloud:** sono servizi di cloud computing erogati attraverso la rete Internet da un service provider a diversi clienti. L'infrastruttura, la piattaforma, le applicazioni sono di proprietà del service provider, sono gestite dal service provider e sono condivise con più clienti.

3. **Hybrid cloud:** : i servizi sono costruiti su infrastrutture ibride che utilizzano la modalità privata per alcuni aspetti (ad esempio la conservazione dei dati) e la modalità pubblica per altri (ad esempio le interfacce di accesso).

4. **Community cloud:** è uno sforzo di collaborazione in cui l'infrastruttura è condivisa tra varie organizzazioni da una specifica comunità con interessi comuni (sicurezza, la conformità, la giurisdizione, ecc.), sia se gestiti internamente o da una terza parte ed ospitati internamente o esternamente.

2.5 Problemi del Cloud Computing

I sistemi di cloud computing vengono criticati principalmente per l'esposizione degli utenti a rischi legati a:

1. **Sicurezza informatica e privacy degli utenti:** Utilizzare un servizio di cloud computing per memorizzare dati personali o sensibili, espone l'utente a potenziali problemi di violazione della privacy. I dati

personali vengono memorizzati nelle Server Farms di aziende che spesso risiedono in uno stato diverso da quello dell'utente. Il cloud provider, in caso di comportamento scorretto o malevolo, potrebbe accedere ai dati personali per eseguire ricerche di mercato e profilazione degli utenti. In presenza di atti illegali, come appropriazione indebita o illegale di dati personali, il danno potrebbe essere molto grave per l'utente, con difficoltà di raggiungere soluzioni giuridiche e/o rimborsi se il fornitore risiede in uno stato diverso da paese dell'utente. Nel caso di industrie o aziende, tutti i dati memorizzati nelle memorie esterne sono seriamente esposti a eventuali casi di spionaggio industriale. [1]

2. **Problemi internazionali di tipo economico e politico:** Possono verificarsi quando dati pubblici sono raccolti e conservati in archivi privati, situati in un paese diverso da quelli degli utenti della nuvola. Produzioni cruciali e di carattere intellettuale insieme a una grande quantità di informazioni personali sono memorizzate in forma di dati digitali in archivi privati centralizzati e parzialmente accessibili. Nessuna garanzia viene data agli utenti per un libero accesso futuro. Maggiore sicurezza e garanzie vi sono nel caso in cui il fornitore del servizio appartiene alla stessa nazione,area applicando le medesime leggi,normative sulla Privacy e Sicurezza del cliente (la legislazione USA o di altre nazioni e' molto diversa dall'italiana e diventa impossibile pensare di soddisfare normative nazionali con servizi in cloud di altre nazioni).[1]

3. **Continuità del servizio offerto:** Delegando a un servizio esterno la gestione dei dati e la loro elaborazione l'utente si trova fortemente limitato nel caso in cui i suddetti servizi non siano operativi (out of service). Un eventuale malfunzionamento inoltre colpirebbe un numero molto elevato di persone contemporaneamente dato che questi sono servizi condivisi. Anche se i migliori servizi di cloud computing utilizzano architetture ridondate e personale qualificato al fine di evitare malfunzionamenti dei sistema e ridurre la probabilità di guasti visibili

dall'utente finale, non eliminano del tutto il problema. Bisogna anche considerare che tutto si basa sulla possibilità di avere una connessione Internet ad alta velocità sia in download che in upload e che anche nel caso di una interruzione della connessione dovuta al proprio Internet Service Provider/ISP si ha la completa paralisi delle attività. [1]

4. **Interoperabilità:** Con interoperabilità si intende la possibilità di migrare un'applicazione da una Cloud ad un'altra senza dover sostenere spese di migrazione intese come riscrittura di runtime per adattarla alla nuova interfaccia. Questo fenomeno è definito con il termine - data lock-in". Una volta infatti scelto il fornitore più adeguato alle proprie esigenze, sarà molto difficile cambiarlo in futuro per via delle differenti API, anche se i sistemi siano catalogati secondo lo stesso modello di servizio. Questo problema al momento è molto discusso poiché rappresenta un punto cruciale per lo sviluppo del Cloud Computing. Un'iniziativa che va in tale direzione è quella supportata da opencloudmanifesto.org, in cui è pubblicato un manifesto [2] contenente una serie di principi a favore di standard aperti per il cloud. Al momento, questo manifesto è supportato da oltre 400 organizzazioni.

[2]http://www.opencloudmanifesto.org

Capitolo 3

Stato dell'arte

Con la crescente diffusione del trend BYOD [1] (Bring Your Own Device) [18], i reparti IT delle aziende devono trovare un modo efficiente e sicuro per consentire ai dipendenti di usare i propri dispositivi mobili nel luogo di lavoro.[12] Questa sessione prende in esame alcuni esempi di servizi cloud per dispositivi mobile che utilizzano la tecnologia delle push notification come parte integrante dell'architettura. Verranno presi in considerazione solo i sistemi che garantiscano un certo grado di interoperabilità con le piattaforme di maggior utilizzo. Verranno descritti servizi che integrano la tecnologia push per creare delle private cloud e servizi che offrono accesso a questa tecnologia lasciando libertà allo sviluppatore di integrarli con le proprie applicazioni.

[1] `http://en.wikipedia.org/wiki/Bring_your_own_device`, Ottobre 2012

3.1 BlackBerry Mobile Fusion

BlackBerry Mobile Fusion [2] viene presentato dalla nota azienda Research in Motion (RIM) come: " lo strumento ideale per gestire i dispositivi mobile nel modo più veloce, facile e strutturato che mai "; BlackBerry Mobile Fusion é uno strumento riservato ad un uso aziendale, che permette di connettere dispositivi BlackBerry OS così come dispositivi Apple iOS e Google Android da un'unica interfaccia. Questo strumento aiuta a proteggere le informazioni aziendali importanti, sincronizzare i calendari, applicazioni e rendere il tutto disponibile per i dipendenti in movimento. E' importate notare che un'azienda come RIM da sempre legata al suo prodotto di punta BlackBerry abbia deciso di aprire i propri sistemi verso dispositivi della concorrenza, questa strategia è giustificata dal trend BYOD sopraccitato e dal fatto che molte aziende utilizzano svariati applicativi a volte presenti solo su alcune piattaforme.

3.1.1 Architettura

In Figura 3.1 si può osservare l'architettura di questo sistema divisa in moduli. Interessante per quest'analisi è il core module che ha il compito di comunicare con i server Push delle case produttrici. In questo contesto la tecnologia delle Push notification svolge un ruolo fondamentale, informa l'utente in tempo reale che deve instaurare una connessione con il private cloud aziendale per ricevere aggiornamenti o effettuare un operazione. Senza il supporto di questa tecnologia non sarebbe possibile usufruire degli stessi servizi riservati ad esempio ai possessori dei dispositivi BlackBerry piuttosto che ai possessore di dispositivi Apple.

Di seguito la descrizione dei vari componenti in visibili in Figura 3.1 reperibile dalla documentazione ufficiale [10] :

[2] `http://us.blackberry.com/business/software/blackberry-mobile-fusion.html`

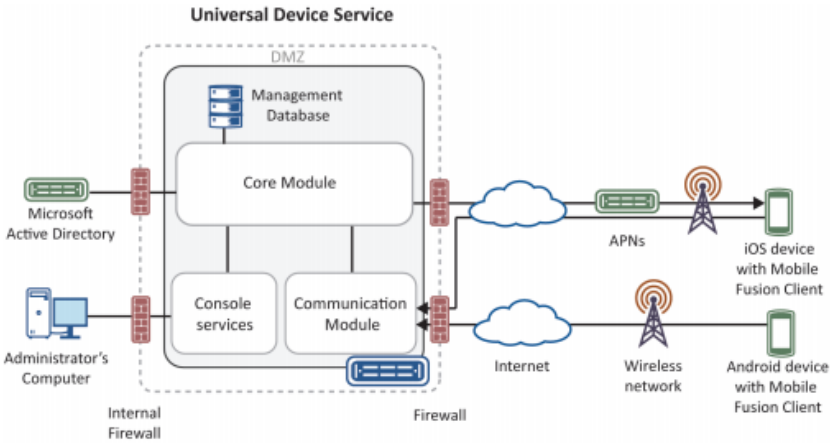

Figura 3.1: Black Berry Universal Device Architecture

- **Core Module:** Il Core Module gestisce l'accesso al database e comunica con l'ambiente aziendale. Il core module si connette a i seguenti servizi:

  - The Universal Device Service: recupera le informazioni sull'account utente da Microsoft Active Directory per creare account utente.
  - Il profilo di Microsoft ActiveSync sui dispositivi è configurato per consentire ai dispositivi di accedere al servizio di posta elettronica aziendale.
  - Il core module invia le notifiche tramite i servizi di push delle case produttrici (ex. APNs per i dispositivi iOS). Le notifiche informano l'utente che sono disponibili aggiornamenti e l'applicazione deve contattare il *Comunication Module* per sincronizzarsi

- **Communication Module:** Questo modulo gestisce il servizio Universal Device Service communication. Esso deve essere accessibile a Internet in modo che possa comunicare con i dispositivi Apple iOS, Google Android o Black Berry.

- **Console services:** È possibile utilizzare i servizi della console per gestire gli account utente, i criteri IT, profili, applicazioni e dispositivi.

- **Mobile Fusion Client:** entità installata su dispositivi che gestisce la comunicazione con i server Push proprietari e con l'interfaccia del *Communication Module*.

3.2 IBM Worklight

IBM negli ultimi anni ha investito molte risorse nel reparto mobile , estendendo molte delle proprie soluzioni a questo nuovo contesto e venendo incontro a esigenze di interoperabilità di applicazioni e sistemi. [6] Tra le tante soluzioni offerte da IBM è stata presa in considerazione Worklight [3] perché si distingue dalle altre per la completezza e le possibilità offerte. Essa fornisce una piattaforma di sviluppo per applicazioni mobile avanzata per smartphone e tablet. Questa soluzione consente di creare applicazioni cross-platform complete senza conversione del codice, interpreter proprietari o linguaggi di scripting poco diffusi.
Inoltre tramite la componente IBM Worklight Server e la tecnologia push è possibile offrire servizi avanzati ai propri utenti. Worklight come è possibile vedere in Figura 3.2 reperibile dalla documentazione ufficiale [7] si divide in 4 componenti principali :

- **IBM Worklight Studio** è un IDE (Integrated Development Environment) basato su Eclipse che consente agli sviluppatori di gestire tutte le attività di programmazione e integrazione per un'applicazione operativa completa.

- **IBM Worklight Server** è un middleware mobile ottimizzato. Questo server basato su Java opera da gateway tra le applicazioni, sistemi di back-end e servizi basati sul cloud.

[3] `http://www.ibm.com/software/mobile-solutions/worklight/`

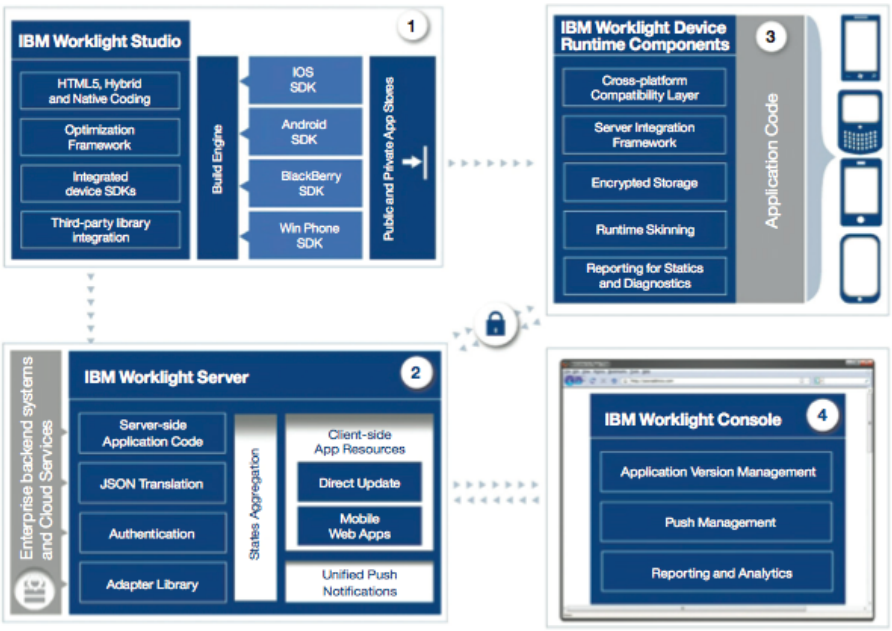

Figura 3.2: WorkLight Components

- Gli **IBM Worklight Device Runtime Component** sono API (Application Programming Interface) di client runtime, progettato per migliorare le funzioni dei server esistenti.

- **IBM Worklight Console** è un'interfaccia utente amministrativa, basata sul Web, per server, adattatori, applicazioni e servizi push.

3.2.1 Architettura

La componente interessante per la mia analisi è IBM Worklight Server questo sistema integrabile nell'ecosistema aziendale permette grazie agli adattatori, di connettere il back-end system con gli applicativi installati nei dispositivi mobile. Gli adattatori sono componenti software che svolgono funzioni di message broker per mettere in comunicazione i back-end lato server agli applicativi dei dispositivi mobile. Tramite essi è possibile configurare, testare e distribuire file XML descrittivi per collegarsi a una serie di interfacce

back-end utilizzando gli strumenti di IBM Worklight Studio e Integrare meccanismi di autenticazione e di sicurezza esistenti. Anche in questo caso la tecnologia push svolge un ruolo fondamentale nel processo di comunicazione poiché viene utilizzata per instaurare una connessione con i dispositivi. Come mostrato in Figura 3.3 Il back-end system comunica con questa componente mediante il *message-based adapters* poi tramite le Unified push API dove il messaggio viene adattato per le varie piattaforme e inviato ai Server proprietari. In base al servizio richiesto il dispositivo potrebbe instaurare una connessione persistente con il sistema o soltanto elaborare l'informazione ricevuta nella notifica.

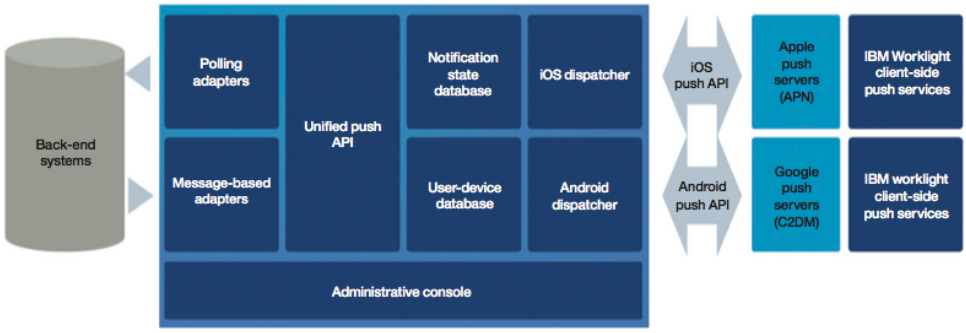

Figura 3.3: Worklight architecture

3.3 Urban Airship

Urban airship azienda leader nel settore dei servizi offerti tramite tecnologia Push per dispositivi mobile offre agli sviluppatore tramite le sue API un modo semplice per aggiungere funzionalità avanzate alle proprie applicazioni disponibile per le tre piattaforme di maggior successo (iOS, Blackberry e Android). A differenza degli altri due casi analizzati Urban Airshp offre un servizio basato interamente su una public cloud.
La loro offerta comprende:

- *Push Composer:* invio push notification tramite una semplice interfaccia web verso le piattaforme supportate.

- *Rich Push:* invio di messaggi push con l'aggiunta di codice HTML, video, audio, mappe e altri componenti multimediali. Possibilità da parte del mittente di tracciare il messaggio e consentire all'utente conservarli in maniera persistenti.

- *Segments:* servizio di notifiche push basato sulla geolocation; questo tipo di servizio viene usata per offrire nuove forme di marketing.

- *In-App Purchase:* questo servizio permette agli utenti di acquistare, nuovi contenuti e funzioni complementari direttamente dall'interno dell'applicazione. Permette anche di semplificare e abbreviare il processo di acquisto e permettere la semplice implementazione di un modello - preemium quindi la realizzazione di nuove opportunità di guadagno per gli sviluppatori.

- *Newsstand Publisher:* servizio basato sul pattern publisher subscriber implementato tramite la tecnologia push, permette agli utenti l'iscrizione a liste e la ricezione di aggiornamenti in base alle preferenze della lista d'iscrizione.

3.3.1 Architettura

L'azienda per il momento non ha reso disponibile una descrizione accurata della propria architettura, ma possiamo dire con certezza che i servizi offerti da Urban Airship sono un esempio di Hybrid cloud [2] dove parte dei loro server e servizi sono fruibili dai propri data center e in caso di picchi di lavoro l'azienda si appoggia ai Web Service di Amazon [4]. Come accennato nel paragrafo precedente L'azienda offre diversi servizi per piattaforme mobile dalla documentazione ufficiale [11] si evince che per ogni piattaforma sono disponibili delle API integrabili nella propria applicazione che estendono il servizio di base offerto dalle case produttrici delle piattaforme.

[4] `http://aws.amazon.com`

Capitolo 4

Piattaforme

In questo capitolo prendo in esame le piattaforme mobile di maggior successo che offrono un servizio di push notification tramite le infrastrutture delle case produttrici. Elencherò per ogni piattaforma le caratteristiche più importanti. In seguito analizzerò l'architettura di riferimento descrivendo le componenti e le credenziali che prendono parte nel processo di invio di una push notification. In seguito l'attenzione ricadrà sul lato dispositivo per vedere come il sistema operativo gestisce una push notification in entrata e come è strutturato il formato di un messaggio; alla fine di questo capitolo verranno comparate le piattaforme e le loro caratteristiche.

4.1 Google Cloud Messaging (GCM)

Google Cloud Messaging (GCM) è un servizio offerto da Google che aiuta gli sviluppatori a creare applicazioni che possono ricevere notifiche da server di terze parti o da altri dispositivi Android. Queste notifiche possono essere usate come avviso per rendere disponibili nuovi dati da prelevare da un server o possono essere messaggi contenenti fino a 4kb di payload. Il servizio GCM gestisce tutti gli aspetti riguardanti l'inoltro e la consegna del messaggio direttamente all'applicazione destinataria, a prescindere dallo stato in cui si trova (background, foreground, shutdown down) che deve essere abilitata alla ricezioni di questi messaggi.
Le informazioni e le figure esplicative contenute nei paragrafi seguenti relative al servizio di Push Notification di Google Android sono contenute nella documentazione ufficiale [5] ultima revisione 14/09/2012

4.1.1 Caratteristiche

- Consente di inviare messaggi verso i dispositivi Android da server di terze parti.
- Non fornisce alcuna garanzia circa la consegna o l'ordine dei messaggi.
- Un'applicazione Android installata su un dispositivo non ha bisogno di essere in esecuzione per ricevere i messaggi Il sistema sveglierà l'applicazione tramite un componente del sistema Android chiamato broadcast receiver che permette di ricevere avvisi propagati a tutto il sistema.
- Non effettua nessun controllo sui messaggi, semplicemente gli inoltra verso il dispositivo.
- E' richiesto Android 2.3 o superiore e Google Play store [1] installato o un simulatore che supporti la versione 2.2 di Android e con le Google APIs installate.

[1] https://play.google.com/store

- Per le versioni di Android inferiori alla 4.0.4 è richiesto un account Google; le versioni successive necessitano del solo play store installato.

4.1.2 Architectural Overview

In questo paragrafo sono descritte le componenti e le credenziali che prendono parte nel processo di inoltro del messaggio essi si dividono in due categorie:
Componenti: Entità fisiche che svolgono un ruolo in GCM.

- *Mobile Device :* Il Dispositivo con un applicazione Android che utilizza GCM

- *Server Provider:* Un application server che invia i dati al dispositivo tramite GCM

- *GCM Servers:* I server di Google che inoltrano i messaggi ricevuti dal Server Provider verso i dispositivi.

Credentials: Identificativi e Token coinvolti nelle diverse fasi per garantire che tutte le parti sono state autenticate e che il messaggio è nella posizione corretta.

- *Project ID:* Identificativo alfanumerico rilasciato dalla APIs console di Google che abilita il servizio GCM per una determinata applicazione.

- *Application ID:* Un identificativo univoco dell'applicazione, esso viene creato in base al nome dell'applicazione stessa dal sistema operativo in modo da assicurare che i messaggi siano inoltrati alla corretta applicazione.

- *Registration ID:* Una identificativo rilasciato dai server GCM all'applicazione Android che permette di ricevere messaggi. Una volta che l'applicazione Android ha ricevuto questo identificativo, lo invia al Server Provider, che lo utilizza per identificare ogni dispositivo. In altre

parole, un Registration ID è l'indirizzo di una particolare applicazione Android in esecuzione su un particolare dispositivo.

- *Google User Account:* Un account rilasciato da Google, il dispositivo mobile deve includere almeno un account Google se nel dispositivo è in esecuzione una versione precedente di Android 4.0.4.
- *Sender Auth Token* Un token univoco che viene salvato dal server provider che fornisce ad esso l'autorizzate e l'accesso ai servizi di Google. Il *Sender Auth Token* viene incluso nel header delle richieste POST.

4.1.3 Life Cycle Flow

In questo paragrafo sono descritti i principali processi e componenti relativi al servizio fornito da Google visibili in Figura 4.1

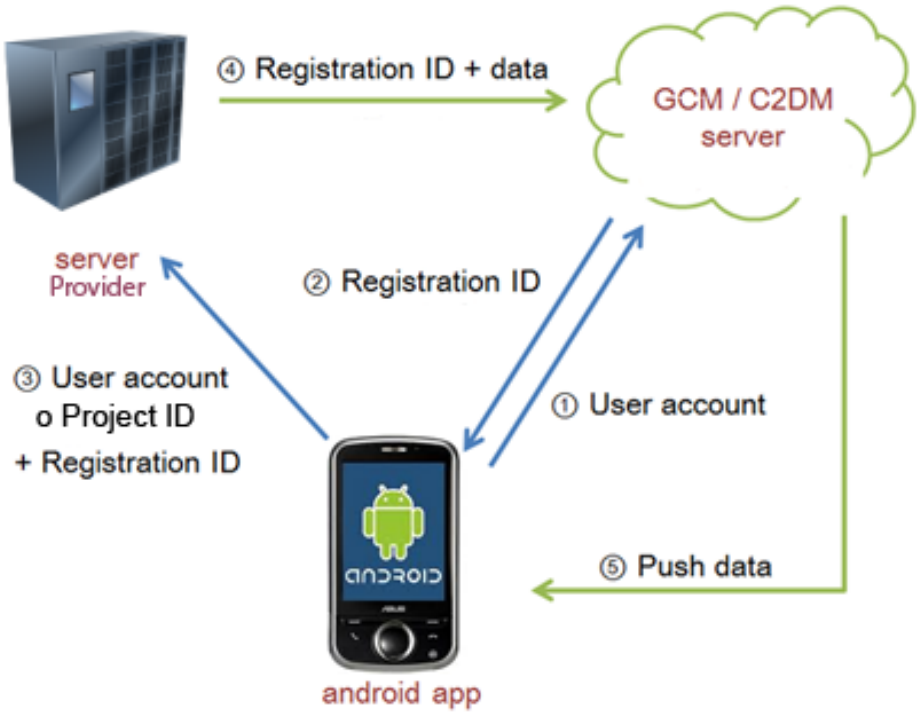

Figura 4.1: Architettura del servizio fornito da Google

Attivazione GCM

Questa è la sequenza di eventi che si verifica quando un'applicazione Android in esecuzione su un dispositivo mobile si registra al servizio GCM per ricevere messaggi:

1. La prima volta che l'applicazione Android deve utilizzare il servizio di messaggistica, avvia la registrarzione al server GCM tramite un *register intent:*

   ```
   com.google.android.c2dm.intent.REGISTER
   ```

   includendo il *Sender ID* e l' *Application ID* nella richiesta.

2. Se la registrazione va a buon fine, il server GCM comunica al dispositivo l'avvenuta registrazione tramite un *registration intent:*

   ```
   com.google.android.c2dm.intent.REGISTRATION
   ```

   che restituisce all'applicazione Android un *Registration ID*. L'applicazione Android deve salvare in persistenza il *Registration ID* ricevuto per trasmetterlo al server provider che vuole comunicare con il dispositivo. E' importante tenere presente che Google può periodicamente aggiornare *Registration ID*, quindi è necessario progettare l'applicazione Android con la consapevolezza che il *Registration intent* può essere richiamato più volte. L'applicazione Android deve essere in grado di gestire quest'evenienza

3. Per completare il ciclo di attivazione, l'applicazione Android invia il *Registration ID* al Server Provider. Esso utilizzerà questo identificativo per comunicare con il dispositivo. Il *Registration ID* ha validità fino a quando l'applicazione è installata sul dispositivo o nel caso Google per qualche ragione decide di cambiare l'identificativo tramite l'*intent register* visto nel secondo passo. Nota: Quando un applicazione viene disinstallata dal dispositivo il *Registration ID* non viene rimosso

automaticamente dai server GCM. Il servizio di Google si accorge che l'identificativo non è più valido, poiché l'applicazione non è più disponibile o per volontà dell'utilizzatore, solo quando il Server Provider tenta di inviare una nuova notifica.

Invio di una notifica

Per il corretto inoltro di una push notification da parte del Server Provider è importante che lo sviluppatore abbia acquisito tramite la console di Google l'API key per comunicare con il server GCM. Inoltre l'applicazione Android deve aver completato con successo la procedura di registrazione vista nel paragrafo precedente e inviato al Server Provider Il *Registration ID*.
Di seguito la sequenza di eventi che si verificano quando un Server Provider tenta di inviare un messaggio:

1. Il Server Provider invia un messaggio al server GCM.
2. Google salva la notifica se il dispositivo non è raggiunto da una connessione dati.
3. Quando il dispositivo è raggiunto da una connessione dati il servizio GCM recapita la notifica.
4. Il sistema operativo Android recapita la notifica ricevuta dal server verso l'applicazione corretta identificata in maniera univoca dal *Registration ID*. Questo passaggio avviene tramite un *intent broadcast* che attiva l'applicazione nota: L'applicazione Android non ha bisogno di essere in esecuzione per ricevere il messaggio.
5. L'applicazione Android elabora il messaggio ricevuto.

Ricezione di una notifica

Questa è la sequenza di eventi che si verifica quando un'applicazione Android installata su un dispositivo riceve un messaggio:

1. Il sistema riceve il messaggio ed estrae i dati (coppie chiave : valore) dal payload del messaggio, se presenti.

2. Il sistema passa i dati grezzi, (coppie chiave : valore) all'applicazione corretta tramite l'Intent:

   `com.google.android.c2dm.intent.RECEIVE`

   Nota: Nessun controllo viene effettuato da parte del sistema sui dati.

3. L'applicazione Android estrae i dati grezzi dall' *Intent reciver* ed elabora i dati.

4.1.4 La comunicazione con il server GCM

In questo paragrafo sono descritti quali sono i requisiti per comunicare con il server GCM, come avviene la comunicazione, il formato del messaggio, i parametri e gli errori da gestire. [16]

Requisiti

1. Attivare il servizio Google Cloud Messaging nella Google API console raggiungibile da `https://code.google.com/apis/console/` e salvare l'API Key da inserire nella richiesta http diretta al server GCM.

2. Il Server Provider deve essere in grado di instaurare una comunicazione tramite il protocollo HTTPS verso il server GCM.

3. Il Server Provider deve essere in grado di salvare in persistenza la coppia *Registration ID* e *API key*.

Per inviare un messaggio il Server Provider invia una richiesta tramite il metodo POST del protocollo HTTP verso l'interfaccia:

`https://android.googleapis.com/gcm/send`

Formato del messaggio

Un messaggio di richiesta è composto da 2 parti: HTTP header and HTTP body. il server GCM accetta nel body del messaggio solo due formati: JSON [2] e plain text; il secondo formato fornisce meno possibilità di personalizzazione. L'header del messaggio deve contenere l'API key e il formato scelto.
Esempio di header:

```
Authorization: key=API KEY
Content-Type:    application/json (per dizionario JSON)
 application/x-www-form-urlencoded;charset=UTF-8 (per testo semplice)
```

Il body del messaggio deve contenere la lista dei *registration ids* e il dizionario JSON contente l'informazione che si vuole inviare; in maniera opzionale è possibile specificare ulteriori paramenti elencati e descritti nella tabella 4.1
Esempio di body:

```
{ "collapse_key": "score_update",
  "time_to_live": 108,
  "delay_while_idle": true,
  "data": {
    "score": "4x8",
    "time": "15:16.2342"
  },
  "registration_ids":["4", "8", "15", "16", "23", "42"]
}
```

[2] http://json.org/

Parametro	**Descrizione**
registration_ids	Una matrice di stringhe che rappresenta l'elenco dei dispositivi registrati al servizio GCM; deve contenere almeno 1 e al massimo 1000 Registration ID. Per inviare un messaggio multicast, è necessario utilizzare JSON.
collapse_key	Una stringa arbitraria (ad esempio "aggiornamenti disponibili"), che viene utilizzato per riassumere il significato di un gruppo di messaggi inviati dal Server Provider mentre il dispositivo non è raggiunto da una connessione dati, in modo che solo l'ultimo messaggio viene recapitato al client. Poiché non vi è alcuna garanzia di ordine in cui vengono inviati i messaggi, il messaggio ultimo potrebbe in realtà non essere l'ultimo messaggio inviato dal Server Provider.
data	Un dizionario JSON i cui campi rappresenta le coppie chiave : valore. Non vi è alcun limite al numero di coppie chiave : valore, l'unico limite riguarda la dimensione totale del messaggio (4kb). I valori possono essere di qualsiasi tipo, ma è consigliato l'utilizzo di stringhe, in quanto i valori verranno convertiti comunque in questo formato dal server GCM.
delay_while_idle	Se incluso, indica che il messaggio non deve essere inviato immediatamente se il dispositivo è inattivo. Il server attende che il dispositivo si attiva, e solo l'ultimo messaggio per ogni valore collapse_key verrà inviato
time_to_live	Per quanto tempo (in secondi) il messaggio deve essere conservata a memoria nel server GCM se il dispositivo non è in linea. Opzionale (default timetolive è di 4 settimane, e deve essere impostato come numero).

Tabella 4.1: Parametri disponibili per il servizio GCM

Formato della risposta

Quando il messaggio viene elaborato con successo, il sistema ritorna con il codice 200 del protocollo HTTP, nel body sono contenute ulteriori informazioni sullo stato del messaggio (inclusi eventuali errori). Quando la richiesta è respinta, la risposta contiene un codice diverso dal 200 (ad esempio 400, 401 o 503) è importante gestire in maniera opportuna questi errori per capire se il server non è temporaneamente disponibile (codice 503) quindi è possibile procedere con un rinvio o se ci sono degli errori nel body della richiesta(codice 400).
Esempio di body di una risposta ricevuta dal server GCM

```
{ "multicast_id": 216,
  "success": 3,
  "failure": 3,
  "canonical_ids": 1,
  "results": [
    { "message_id": "1:0408" },
    { "error": "Unavailable" },
    { "error": "InvalidRegistration" },
    { "message_id": "1:1516" },
    { "message_id": "1:2342", "registration_id": "32" },
    { "error": "NotRegistered"}
  ]
}
```

Nella tabella 4.1.4 sono descritti i possibili parametri contenuti nel messaggio di risposta ritornato dal server GCM

Parametro	**Descrizione**
multicast_id	ID univoco che identifica il messaggio multicast.
success	Numero di messaggi che sono stati elaborati senza errori.
failure	Numero di messaggi che non possono essere elaborati.
canonical_ids	....
results	Matrice di oggetti che rappresenta lo stato dei messaggi elaborati. Gli oggetti sono elencati nello stesso ordine della richiesta e possono avere questi campi: - *message_id:* stringa che rappresenta il messaggio quando è stato elaborato correttamente. - *registration_id*: se presente, significa che il server GCM ha elaborato il messaggio, ma ha un altro canonical ID registrato per tale dispositivo, il mittente deve sostituire gli ID per le richieste future (altrimenti potrebbe essere respinto). - *error*: stringa che descrive un errore che si è verificato durante l'elaborazione del messaggio per il destinatario.

Tabella 4.2: Parametri di risposta del server GCM

4.2 Apple Push Notification Service

Il meccanismo di push notification fornito da Apple si basa sull'infrastruttura del Apple Push Notification service, o più brevemente APNs; il servizio consente al dispositivo iPhone/iPad di instaurare una connessione con un Server Provider. Per inoltrare una notifica push, bisogna contattare il servizio APNs affinchè consegni il messaggio alla specifica applicazione installata sul dispositivo desiderato. Quando una applicazione utilizza le push notification, il dispositivo resta connesso al server APNs utilizzando una connessione TCP/IP. É opportuno sottolineare che il servizio di notifica non è affidabile e che non c'è garanzia sull'effettiva ricezione del messaggio e il tempo di spedizione può variare da un secondo a un'ora. Il contenuto di una Push Notification é chiamato Payload, e la sua dimensione massima consentita é di 256 bytes. Ogni notifica che eccede tale dimensione verrá rifiutata dai server Apple. Le informazioni e le figure esplicative contenute nei paragrafi seguenti relative al servizio di Push Notification di Apple sono contenute nella documentazione ufficiale [3] ultima revisione del 2011-08-09

4.2.1 Caratteristiche

- Consente di inviare messaggi verso i dispositivi Apple da server di terze parti
- Non fornisce alcuna garanzia circa la consegna e l'ordine dei messaggi
- Massima dimensione della notifica 256 bytes
- Non è richiesto che l'applicazione installata su dispositivi Apple sia in esecuzione nel momento della ricezione della notifica
- E' richiesto una versione pari o superiore a iOS 3.0 per i dispositivi mobile e OS 10.4 per gli altri dispositivi Apple
- Bisogna essere registrati come sviluppatori presso il iOS Developer Program membership.

- Bisogna essere in possesso di tutti i certificati e del profilo provisioning [3] necessario al testing e alla distribuzione.

- iOS prevede tre tipi di notifiche, all'interno di una Push Notification:

  - suoni: viene riprodotto un avviso sonoro
  - testi: viene mostrato un aller/banner sullo schermo
  - badge: viene mostrato un numero sull'icona dell'applicazione

4.2.2 Architectural Overview

In questo paragrafo sono descritte le componenti e le credenziali che prendono parte nel processo di inoltro del messaggio essi si dividono in due categorie:
Componenti: Entità fisiche che svolgono un ruolo in APNS.

- *Mobile Device:* Il Dispositivo con un applicazione Apple installata che utilizza APNs.

- *Provider:* Un application server che invia i dati al dispositivo tramite APNS

- *APNS Server:* I server di Apple che inoltrano i messaggi ricevuti dal Provider e gli inoltrano verso i dispositivi registrati.

Credenziali: Identificativi e credenziali coinvolti nelle diverse fasi per garantire che tutte le parti comunicano in maniera corretta.

- *Application ID:* Identificativo alfanumerico rilasciato durante la procedura di registrazione presso il provisioning portal che identifica un'applicazione; ad esso verranno associati i certificati SSL.

[3] `http://developer.apple.com/library/ios/#documentation/ToolsLanguages/Conceptual/YourFirstAppStoreSubmission/ProvisionYourDevicesforDevelopment/ProvisionYourDevicesforDevelopment.html`

- *SSL certificates:* ll funzionamento di APNS è legato al due certificati SSL il primo certificato riguarda il Provider consente la comunicazione con i server APNs Il secondo certificato riguarda il Dispositivo serve per abilitare il dispositivo ad instaurare una connessione con il server APNs.

- *iOS Developer Program membership :* Account rilasciato da Apple, per la gestione dell'applicazione sia in fase di test e produzione .

- *device token:* Token univoco generato dal dispositivo e registrato dal server APNs che consente al server Provider di identificare un dispositivo.

- *password:* Password abbinata al certificato SSL per l'autenticazione con il server APNs.

4.2.3 Life Cycle Flow

In questa sessione sono descritti i principali processi relativi al servizio fornito da Apple visibili in Figura 4.2 .

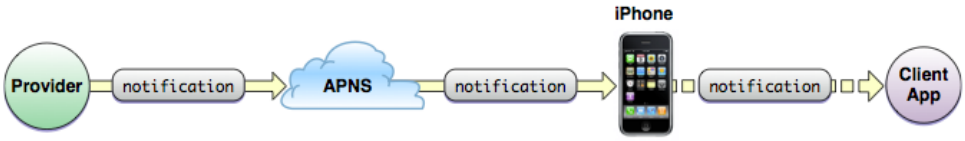

Figura 4.2: Vista delle entità coinvolte nel servizio APNs

La comunicazione con il server APNS

La comunicazione con il server APNs sia da parte del server Provider che da parte del dispositivo avviene tramite il protocollo TLS che permettono di instaurare un canale sicuro dal sorgente al destinatario (end-to-end) su reti TCP/IP. L'autenticazione con il server APNs è di tipo bilaterale, poiché entrambe le parti si scambiano i relativi certificati. Questa autenticazione (definita Mutual authentication) richiede che entrambi le parti (Server provider e Dispositivo) possiedano il certificato precedentemente ottenuto dal provisioning portal. Una volta completata la procedura, viene instaurata una connessione persistente con il server APNs tra da entrambe le parti

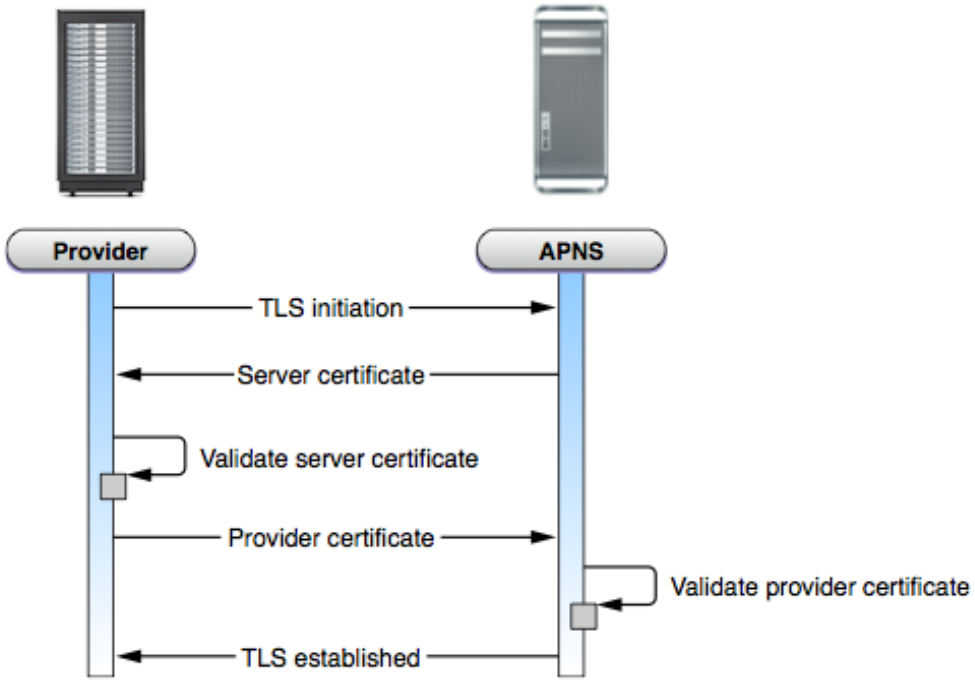

Figura 4.3: Comunicazione Provider APNs

Generazione e utilizzo del Device Token

Dopo il primo avvio dell'applicazione, il dispositivo stabilisce una connessione con il server APNs registrandosi al servizio. Un' applicazione che vuole ricevere le push notification deve registrarsi al server invocando il metodo

```
registerForRemoteNotificationTypes: (UIRemoteNotificationTypeAlert)];
```

dove viene specificato anche il tipo di notifica che l'applicazione può riceve. Se la registrazione ha successo, verrà invocato il metodo:

```
didRegisterForRemoteNotificationsWithDeviceToken:
```

questo metodo restituisce il device token, che dovrà essere inviato al Server Provider per permettere l'invio delle notifiche.

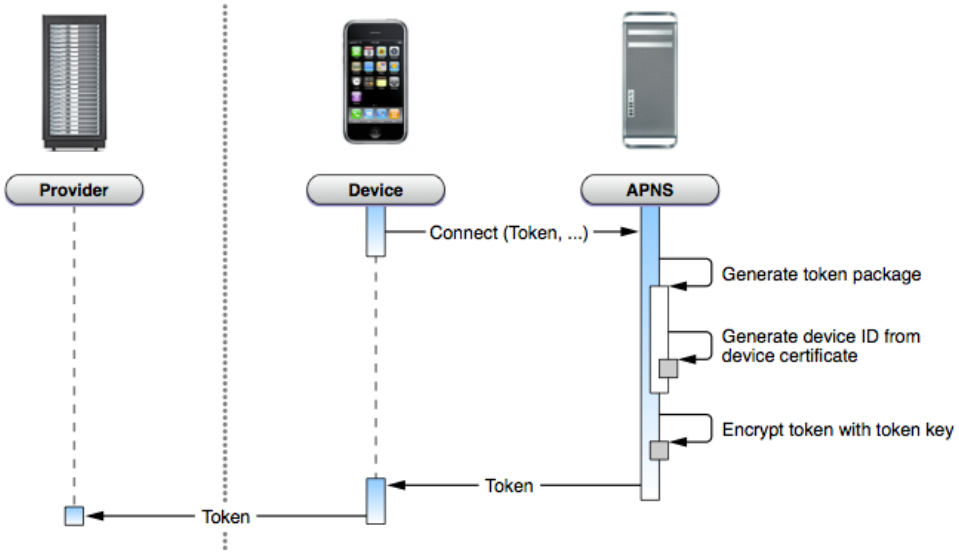

Figura 4.4: Diagramma di sequenza: generazione del device token

Comunicazione con il server APNs

Come già accennato nei paragrafi precedenti per inviare una notifica verso il server APNs di Apple bisogna essere in grado di instaurare una connessione tramite socket TCP. Le interfacce binarie per instaurare una connessione sono:

`ssl://gateway.sandbox.push.apple.com:2195` (in fase di sviluppo)

`ssl://gateway.push.apple.com:2195` (in fase di produzione)

Il server APNs accetta messaggi solo nel formato JSON, in esso sono contenuti una serie di parametri che servono per personalizzare la notifica che riceverà il dispostivo. Nella tabella 4.3 sono descritti tutti i parametri permessi dal servizio
. Esempio di un messaggio:

```
  {
"aps" : {
          "alert" : "You got your emails.",
  "badge" : 9,
          "sound" : "bingbong.aiff"
      },
      "acme1" : "bar",
"acme2" : 42 }
```

parametro	Tipo	Descrizone
alert	string or dictionary	Se questo parametro è incluso, iOS visualizza un avviso standard. È possibile specificare una stringa come valore o un ulteriore dizionario. Se si specifica una stringa, questa diventa il testo del messaggio di una finestra d'avviso con due pulsanti: Chiudi e Vedi.
badge	number	Il numero da visualizzare come badge nell'icona dell'applicazione. Se questo parametro è assente, non verrà visualizzato nessun numero
sound	String	Il nome di un file audio precedentemente inserito nel bundle dell'applicazione. Il suono in questione viene riprodotto come avviso. Se il file audio non esiste viene riprodotto il suono di default. L'audio deve essere in uno dei formati di dati audio che sono compatibili con i suoni di sistema.
body	String	testo del messaggio di allerta
action-loc-key	string or null	Questo parametro serve per specificare l'etichetta del bottone di default viene utilizzato"Visualizza"
loc-key	string	Questo parametro viene utilizzato per richiamare stringhe presenti nel file Localizable.strings dell'applicazione
loc-args	array of strings	identico al precedente parametro ma utilizza un array di stringhe
launch-image	string	Con questo parametro è possibile specificare il nome di un file immagine precedentemente inserito nell'applicazione da visualizzare nella finestra di notifica.

Tabella 4.3: Parametri disponibili per il servizio APNs

Codici di risposta

Il server APNs risponde al Server Provider con dei codici di risposta descritti in Tabella 4.4; è importante che il Server Provider gestisca in maniera corretta questi codici.

Codice	**Descrizione**
0	Nessun errore riscontrato
1	errore nel processo
2	devise token non presente
3	Missing topic
4	payload mancante
5	dimensione token errata
6	Idimensione topic errata
7	dimensione payload errata
8	token non valido
255	None (unknown)

Tabella 4.4: Codici di risposta del server APNs

Feedback Service

Il feedback service ha il compito di informare il Provider se l'applicazione non è più installata nel dispositivo o l'utente ha espresso la volontà di non ricevere più le notifiche. In tutti e due i casi il Provider deve cancellare il device token dalla lista dei dispositivi raggiungibili per ridurre l'overhead inutile del messaggio e migliorare le prestazioni generali del sistema. A questo scopo Apple Service Push Notification include un servizio di feedback utilizzabile per richiedere informazione sulla richiesta precedentemente inoltrata. Il sistema APNs aggiorna l'elenco dei dispositivi associati ad ogni singola applicazione ogni volta che si tenta di inviare una notifica; quindi è compito del Provider interrogare il servizio per sincronizzare l'elenco dei dispositivi, Il servizio di feedback è raggiungibile tramite l'interfaccia:

`ssl://feedback.sandbox.push.apple.com:2196` (in fase di sviluppo)

`ssl://feedback.push.apple.com:2196` (in fase di produzione)

Per accedere al servizio si utilizza lo stesso certificato SSL per l'invio delle notifiche, una volta ottenuta la connessione il servizio restituisce immediatamente la risposta contenete i device Token da cancellare. I parametri di risposta sono descritti nella tabella 4.5

Parametro	**Descrizione**
Timestamp	Un timestamp (come un valore a quattro byte) che indica quando il server APNs ha stabilito che l'applicazione non è più presente sul dispositivo.
Token length	Lunghezza del token
Device token	Il token del device da eliminare

Tabella 4.5: Parametri di riposte del feedback service

4.3 Black Berry Push Service

Push technology of the BlackBerry(BBP) è il servizio offerto da Research In Motion (RIM) per la sua piattaforma Black Berry che aggiunge la possibilità di ricevere le notifiche push inviate da server di terze parti. A differenza delle altre aziende RIM non è nuova ad un servizio del genere infatti una prima versione dello stesso servizio è stata introdotta nel 2008 ma solo per un utenza business tramite l'ecosistema BlackBerry Enterprise Server (BES). Nel mese di marzo 2010 RIM ha esteso il servizio a tutti gli utenti, anche se, con un considerevole taglio delle funzionalità; offrendo una variante del suddetto servizio chiamata BlackBerry Push Essentials (BIS). Una delle funzionalità eliminate che marca maggiormente le due versioni è che nella versione BIS non c'è garanzia di ricezione del messaggio. Fattore molto importate in fase di progettazione di un sistema che necessita di scambiare messaggi in modo affidabile. Un'altra differenza con le altre piattaforme viste riguarda lo standard utilizzato per l'invio del messaggio infatti RIM utilizza lo standard PAP [9] definito dal Open Mobile Alliance [4] come standard per l'invio dei messaggi push. La dimensione massima del messaggio è di 8 KB, i messaggi che eccedono il limite o che non rispettano lo standard verrano rifiutati dal server.
Le informazioni e le figure esplicative contenute nei paragrafi seguenti relative al servizio di Push Notification di BlackBerry sono contenute nella documentazione ufficiale [4] versione 1.1.0.16

4.3.1 Caratteristiche

- Consente di inviare messaggi verso i dispositivi BlackBerry da server di terze parti.
- Non fornisce alcuna garanzia circa la consegna e l'ordine dei messaggi
- Massima dimensione del messaggio 8 KB

[4] http://www.openmobilealliance.org/

- Non è richiesto che l'applicazione installata sul dispositivio BlackBerry sia in esecuzione nel momento della ricezione della notifica
- Bisogna essere registrare la propria applicazione presso BlackBerry Developer Program
- Bisogna essere in possesso di un identificativo univoco e delle credenziali per comunicare con i server RIM
- E' possibile includere nel corpo del messaggio: testo, suoni o immagini
- Sia in fase di valutazione che in fase di distribuzione RIM limita il numero di notifiche giornaliere
- Entrambi i servizi sono disponibili solo se l'utente ha un abbonamento di tipo BIS o BES

4.3.2 Architectural Overview

In questo paragrafo sono descritte le componenti e le credenziali che prendono parte nel processo di inoltro del messaggio essi si dividono in due categorie:
Componenti: Entità fisiche che svolgono un ruolo in BBPS.

- *Mobile Device:* Il dispositivo con un'applicazione BlackBerry installata che include le librerie ufficiali per l'utilizzo delle push notification
- *Server Provider* Un application Server che consente di creare messaggi di richiesta (push request, cancel request, stato-query request) e messaggi di risposta (result-notification-response) utilizzando le librerie ufficiali fornite da RIM e sottopone le richieste al Gateway Proxy Push (PPG).
- *Push Proxy Gateway:* Il Push Proxy Gateway (PPG) processa le richieste inviate dai vari Servers Provider. Dopo l'elaborazione di una

richiesta da parte del Push Initiator, il PPG invia un messaggio di risposta che comunica se la richiesta è stata accettata e verrà elaborata. Il messaggio di risposta contiene un codice di avvenuta accettazione o di errore nello standard PAP.

Credenziali: Le credenziali coinvolti nelle diverse fasi per garantire che tutte le parti sono state autenticate e che il messaggio è nella posizione corretta.

- *App ID:* Identificativo univoco dell'applicazione rilasciato da RIM
- *password* una password abbinata al App ID
- *CPID (Content Provider ID):* identificativo univoco che fa parte del PPG address
- *PPG Base URL:* URL del Push Proxy Gataway (https://[CPID]+pushapi.eval.blackberry.com)
- *Push Port:* numero di porta per instaurare la comunicazione con il PPG lato client

4.3.3 Life Cycle Flow

In questa sessione sono descritti i principali processi relativi al servizio fornito da RIM

1. Il Server Provider tramite le librerie lato server invia messaggio verso il Push Proxy Gateway sotto forma di una richiesta POST HTTP. Il messaggio push è un messaggio MIME multipart [5], che contiene i seguenti elementi:

    - Un PAP WAP 2.1 XML, che descrive i parametri di consegna e specifica uno o più Dispositivi BlackBerry in cui il contenuto sarà consegnato.

[5] http://tools.ietf.org/html/rfc2387

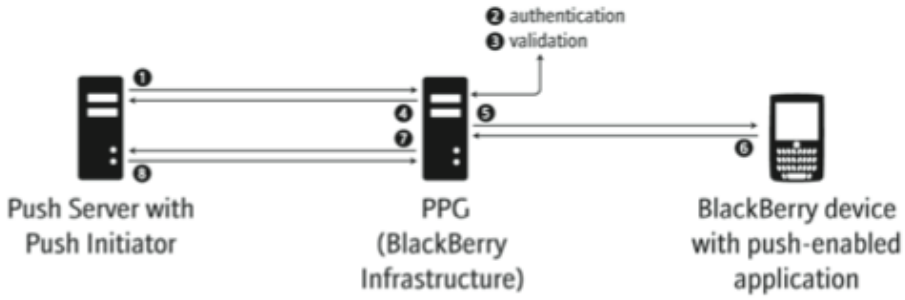

Figura 4.5: Architettura del servizio fornito da RIM

- i contenuti push da consegnare alla lista dei dispositivi BlackBerry presenti nel corpo del PAP

2. Il Push Proxy Gateway riceve la richiesta di comunicazione da parte del Server Provider.

3. Se il push Initiator viene autenticato, il Push Proxy Gateway convalida il messaggio push:

   - Se l'account del Server Provider rientra nella quota giornaliera di messaggi push riservati a lui e la richiesta contiene tutte le necessarie informazioni, Push Proxy Gateway accetta il messaggio push.
   - Se ha ha superato i limiti di push, Push Proxy Gateway rifiuta il messaggio.

4. Il Push Proxy Gateway restituisce una risposta all'initiator e indica se il messaggio è stato accettato per l'elaborazione o rifiutato. Se il messaggio è stata respinto, Push Proxy Gateway restituisce un codice di errore all'initiator che fornisce il motivo del rifiuto.

5. Il Push Proxy Gateway si occupa di recapitare i contenuti push ai dispositivi BlackBerry specificati.

6. Ogni dispositivo BlackBerry che riceve il messaggio notifica al Push Proxy Gateway se il messaggio è stato ricevuto. Un messaggio push è considerato di successo se il contenuto viene consegnato prima della data di scadenza specificata e soddisfa i criteri specificati dall'elemento quality-of-service.

7. Se l'initiator ha accesso al BES può richiedere un feedback al Push Proxy Gateway, nel messaggio di feedback è compresa la garanzia dell'avvenuta ricezione da parte di ogni singolo dispositivo.

Comunicazione con il server Push Proxy Gateway

In questo paragrafo sono descritti quali sono i requisiti per comunicare con il server PPG come avviene la comunicazione, il formato del messaggio, i parametri e gli errori da gestire.

Requisiti

1. Richiedere l'attivazione del servizio di notifiche push presso `https://www.blackberry.com/profile/?eventId=8121` il nome dell'applicazione inserito deve combaciare con il nome del progetto che andrà in distribuzione

2. Aver ricevuto per email da RIM tutte le credenziali viste in 4.3.2

3. Il Server Provider deve essere in grado di instaurare una comunicazione tramite il protocollo HTTPS con il Push Proxy Gateway

4. Il Server Provider ha in persistenza almeno un *App ID* dei dispositivi a cui si vuole inviare la push notification ;

Per inviare un messaggio il Server Provider invia una richiesta tramite il metodo POST del protocollo HTTP all'indirizzo *PPG Base URL:*

Formato del messaggio

Come anticipato in 4.3 il servizio offerto da RIM utilizza lo standard MIME [6]; ricordiamo che MIME, conosciuto come standard per la comunicazione email, è anche un componente fondamentale dei protocolli di comunicazione come l'HTTP, il quale richiede che i dati siano trasmessi come messaggi complessi simil-email come potrebbero essere le push notification. In accordo con lo standard PAP un messaggio è composto da tre parti:

1. *RFC 2822 email message:* questa componente contiene l'intero messaggio. La specifica RFC si riferisce al formato di un messaggio di posta elettronica, l'indirizzo è l'URL del Push Proxy Gateway ricevuto al momento della registrazione

2. *RFC 2045 MIME section:* questa componente è un allegato MIME di tipo application/xml.

3. *XML command:* Comandi XML riferiti allo standard PAP ho proprietari di RIM.

[6] http://tools.ietf.org/html/rfc2387

Esempio di messaggio:

```
This sample includes the boundary line for a multipart MIME message.
 --PMasdfglkjhqwert
Content-Type: application/xml
<?xml version="1.0"?>
<!DOCTYPE pap PUBLIC "-//WAPFORUM//DTD PAP 2.1//EN"
"http://www.openmobilealliance.org/tech/DTD/pap_2.1.dtd">
<pap>
  <push-message push-id="999999999"
                source-reference="AAAAAAAAAAAA"
                deliver-before-timestamp="2008-09-31T13:30:00Z"
                ppg-notify-requested-to="notify_url_path">
    <address address-value="PIN00001"/>
    <address address-value="PIN00002"/>
    <address address-value="PIN00003"/>
    <quality-of-service  delivery-method="confirmed"/>
  </push-message>
</pap>
--PMasdfglkjhqwert
Content-Encoding: binary
Content-Type: text/html
testo della notifica
 --PMasdfglkjhqwert--
```

4.4 Push Notifications per Windows Phone

Dal versione di Windows Phone 7, sono stati introdotti una serie di nuovi concetti relativi allo sviluppo su dispositivi mobili, tra questi di vitale importanza per rimanere al passo con la concorrenza Microsoft ha aperto il suo servizio di push notification verso le applicazioni di terze parti. Il sistema operativo Winows Phone come iOS non permette il multitasking per le applicazioni di terze parti, se una applicazione passa in background, di fatto viene "congelata" per poi essere "scongelata" quando le viene ridato il focus (operazione che viene definita tombstoning). Il servizio di notifica push di Microsoft in Windows Phone (MPN) offre agli sviluppatori di terze parti un canale elastico, dedicato, e persistente per inviare i dati ad un'applicazione Windows Phone da un servizio Web in maniera semplice ed efficiente.

4.4.1 Caratteristiche

- Consente di inviare messaggi verso i dispositivi Windows Phone da server di terze parti.
- E' possibile avere un solo canale per applicazione per ricevere notifiche push.
- Ogni dispositivo può avere al massimo 30 canali per ricevere notifiche push; in pratica solo le prime 30 applicazioni possono accedere a questo servizio.
- Le notifiche push non dovrebbero includere dati confidenziali.
- Il servizio offerto da Microsoft non fornisce alcuna garanzia circa la consegna e l'ordine dei messaggi.
- La massima dimensione del corpo di una notifica push è di 3 KB.
- Le notifiche push non verranno ricevute se il dispositivo è connesso ad un SOCKS proxy.
- L'architettura di Windows Phone 7 mette a disposizione 3 diversi tipi di notifica:
  - Toast Notification
  - Raw Notification
  - Tile Notification
- Un applicazione installata su windows Phone non ha bisogno di essere in esecuzione per ricevere le notifiche.
- E' richiesta la versione 7.0 di windows Phone.

4.4.2 Architectural Overview

In questo paragrafo sono descritti i termini e i concetti chiave inclusi in MPN. Essi si dividono in due categorie:
Componenti: Entità fisiche che svolgono un ruolo in MPNS.

- *Mobile Device :* Il Dispositivo con un applicazioneWindows Phone installata abilitata alla ricezione delle push notification.
- *Server Provider:* Un application server che invia i dati al dispositivo comunicando con il server di push notification di Microsoft.
- *MPN Server:* I server di Microsoft che inoltrano i messaggi ricevuti di Server Provider verso i dispositivi registrati.

Credentials: Identificativi e Token coinvolti nelle diverse fasi per garantire che tutte le parti sono state autenticate e che il messaggio è nella posizione corretta.

- *Un account Windows Phone Marketplace* Account rilasciato da Microsoft per la gestine delle applicazioni.
- *SSL Root Certificates* certificato per la comunicazione con il server MPN rilasciato da Microsoft.
- *Key-Usage:* identificativo univoco per accedere al servizio dal lato Provider.

4.4.3 Life Cycle Flow

In questa sessione sono descritti i principali processi relativi al servizio fornito da MPN Il diagramma in Figura 4.6 mostra come l'applicazione client in esecuzione sul telefono può richiedere un push notification URI dal servizio Push client del sistema operativo(1). Il servizio Push client negozia poi con il servizio Push di Microsoft Notification (MPN) e restituisce un URI per l'applicazione client (2 e 3). L'applicazione client può quindi inviare l'URI

al Server Provider (4). Quando il Provider ha informazioni da inviare all'applicazione client, utilizza l'URI per l'invio della push notification al servizio di notifica push di Microsoft (5), che a sua volta provvede a consegnare la Push notification all'applicazione installata su un dispositivo Windows Phone (6). Il servizio MPN invia un codice di risposta al Server Provider, dopo che una notifica push viene inviata, per indicare che la notifica è stata ricevuta e verrà consegnato al dispositivo il prima possibile, tuttavia il servizio MPN di Microsoft non fornisce garanzie di consegna delle push notification al dispositivo.

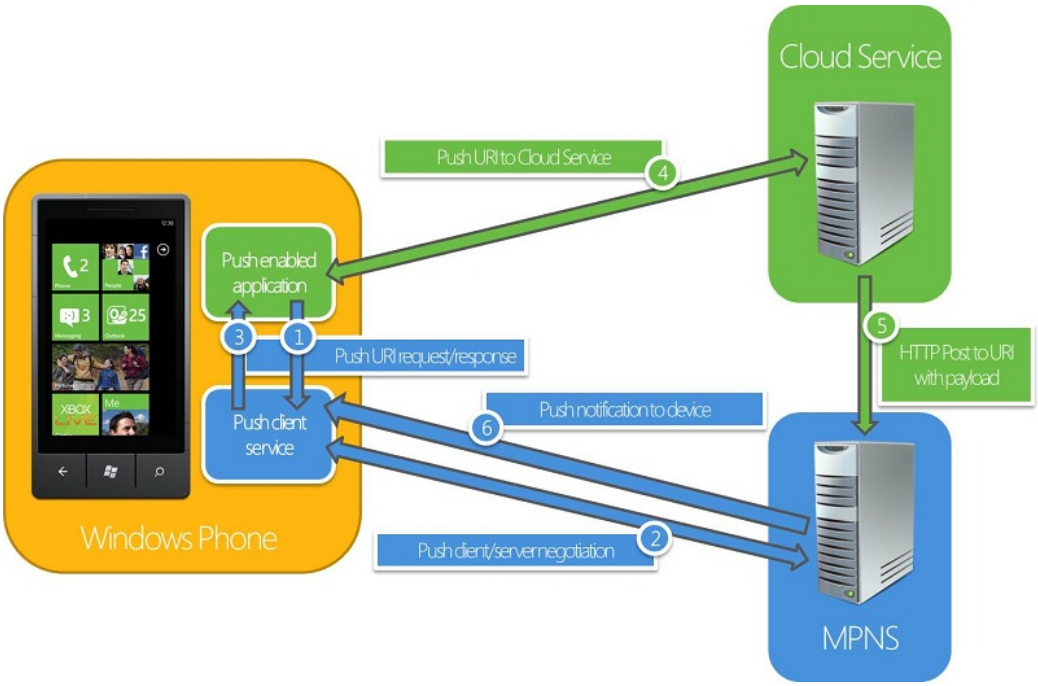

Figura 4.6: Vista architetturale di alto livello del servizio fornito da Microsoft

4.4.4 Tipi di Notifiche

A differenza delle altre piattaforme Microsoft associa a ogni tipo di notifica differenti usi, ad esempio la *Raw Notification* viene visualizzata dall'utente solo se nel momento in cui viene ricevuta l'applicazione destinatario è in primo piano. Le *toast notification* invece possono essere usate in maniera

trasparente all'utente per trasmettere dati all'applicazione. Di seguito sono elencate e descritte le varie tipologie di notifiche disponibili:
Una **Toast Notification** viene visualizzata nella parte superiore dello schermo per avvisare gli utenti di un evento, come ad esempio una notizia o un avviso meteo. Essa sarà impressa nello schermo per circa 10 secondi, a meno che l'utente non la respinga. Se l'utente tocca il riquadro della notifica, l'applicazione a cui è legata la notifica verrà avviata.
Gli elementi che compongono una toast notification sono:

- *Title:* Una stringa in grassetto che viene visualizza immediatamente dopo l'icona dell'applicazione. Questa è definita come proprietà *Text1* nello schema XML.

- *Content:* Il contenuto della notifica proprietà *Text2* nello schema XML.

- *Parameter:* dati non visibili nella notifica ma che vengono passati all'applicazione per specifici compiti proprietà *Param* nello schema XML.

Una **Una Tile Notifications** è un link ad una applicazione visualizzata in *Start*. Ci sono due tipi di Tiles Application Tiles e secondary Tiles. questo tipo di notifica viene utilizzata anche per la funzione badge.

Una **Raw Notifications** è una notifica che viene visualizzata solo se l'applicazione è in esecuzione, se l'applicazione non è in esecuzione la notifica raw viene scartata.

4.5 Confronto tra le piattaforme

Nella Tabella 4.5 sono riassunti le caratteristiche salienti di ogni piattaforma visti nei paragrafi precedenti. Questa tabella sarà un utile riferimento in fase di progettazione del componente che si vuole realizzare.

	Android	**iOS**	**BlackBerry OS**	**Windows Phone**
Message reliability	NO	NO	NO	NO
Formato del messaggio	JSON	JSON	MIME multipart message	XML
Dimensione max payload	4KB	256Byte	8KB	3KB
Formati accettati	text	text,immagini, audio	text	text
Time to live di una notifica in attesa di essere conseganta	30g	N.S	30gg	N.S.
Quota messaggi giornaliera	no quota	no quota	5000 al giorno	500 al giorno
Report sullo stato della richiesta	SI	SI	SI	SI

Tabella 4.6: Tabella comparativa delle piattaforme analizzate

N.S. = Non Specificato

Capitolo 5

Progettazione di Switchboard

In questo capitolo verrà discussa la fase di progettazione del sistema Switchboard al fine di costruire un prototipo con funzionalità base. Nello studio descritto nei capitoli precedenti, in particolare nel capitolo 2 per quanto riguarda le soluzioni già in commercio che sfruttano questa tecnologia, nel capitolo 3 per quanto riguarda le varie caratteristiche, in comune e non, di ogni piattaforme; ho acquisito un background sufficiente per iniziare a progettare una componente che nel panorama dei sistemi private cloud open source non è stata ancora approfondita. L'idea è di realizzare una componente, facilmente integrabile in un sistema presistente in grado di fornire una semplice interfaccia per permettere l'interoperabilità con i vari servizi di push presi in esame. In 5.1 descriverò l'architettura di alto livello del sistema che si vuole implementare, poi in 5.2 l'analisi dei requisiti e in 5.3 la stesura dei casi d'uso utili per capire gli attori che interagiscono con il sistema.

5.1 Architettura

L'immagine 5.1 mostra le componenti fondamentali coinvolte nel progetto. Grazie a questa vista di alto livello è facile percepire l'utilità di questa applicazione. L'applicazione switchboard si occupa interamente della comunicazione con i vari servizi di push; in questo modo l'invio di una notifica verso i dispositivi di differenti piattaforme viene trasformata in una semplice richiesta http facilmente integrabile in un sistema informativo già esistente o in una semplice pagina web. Il sistema deve anche fornire delle funzionalità di gestione dei dispositivi, come ad esempio la registrazione e disiscrizione dei dispositivi e deve poter fornire all'utilizzatore le stesse opzioni disponibili per ogni piattaforma se bypassasse il sistema switchboard e comunicasse direttamente con il server del produttore. In questo capitolo utilizzerò la parola Provider con riferimento a un server di terze parti che si interfaccia con i sevizi di push delle case produtrici, Vendor Server con riferimento ad un server di gestione delle notifiche proprietario (GCM, APNs, BGP, MPN) e Device per identificare in maniera generale un dispositivo mobile che appartiene a una specifica piattaforma.

5.2 Raccolta dei requisiti

Per la raccolta dei requisiti mi sono servito dell'analisi dei casi di studio visti nel capitolo 3 di aziende che offrono servizi simili. Il progetto Switchboard si differenzia da questi servizi perché vuole essere solo un adapter, lasciando all'utilizzatore piena libertà di integrazione con un sistema esistente e di sviluppo di una propria libreria dal lato device con funzionalità che estendono questo servizio.

Requisiti funzionali:

1. Permettere l'invio di una notifica verso differenti piattaforme.

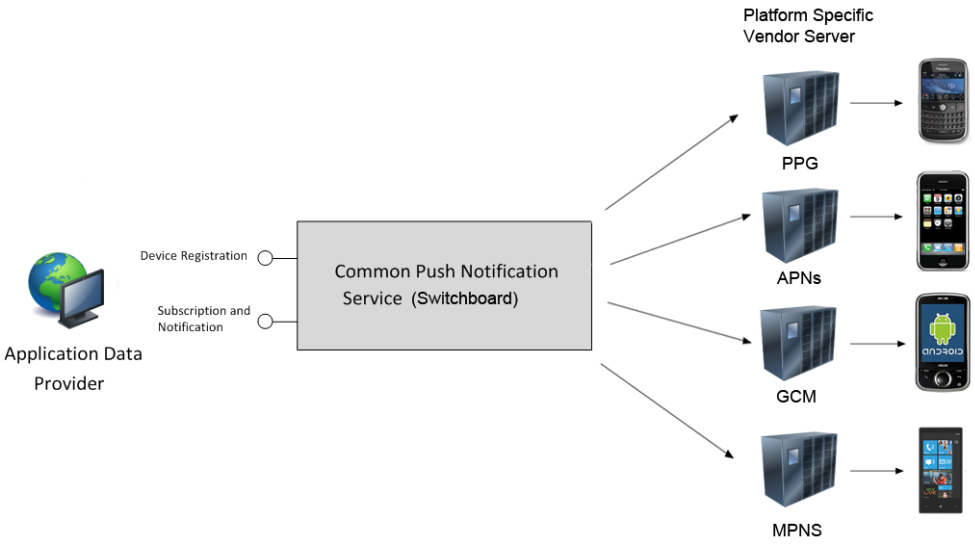

Figura 5.1: Vista architetturale di alto livello del sistema

2. Registrare i device e visualizzare informazioni utili per l'aggiornamento delle applicazioni.

3. Fornire un semplice interfaccia per permettere a sistemi informativi terzi di interagire con il sistema.

4. Fornire all'utente le opzioni di invio che avrebbe se si interfacciasse direttamente con i Vendor Server.

5. Permettere l'invio di notifiche selettive (in base alla piattaforma o ad un sottoinsieme di dispositivi registrati).

Requisiti non funzionali:

1. Il sistema deve avere un architettura modulare che faciliti l'aggiunta di nuove piattaforme.

2. Il sistema deve utilizzare protocolli comuni e tecnologie supportate da tutte le piattaforme.

3. Il sistema deve gestire gli errori di comunicazioni con i vari Server Push

4. Fornire report sullo stato delle richieste (per quanto permesso dalle piattaforme)

5.3 Casi d'uso

Dopo aver analizzato i requisiti di sistema, approfondiamo ora il progetto che si vuole realizzare attraverso i casi d'uso, suddivisi in tre scenari. Il primo scenario rappresentato dalla Figura 5.2 mostra l'interazione tra il sistema e il manutentore del sistema. Il compito del manutentore è quello di Figura configurare il sistema Figura 5.3 . Quindi il manutentore deve aver precedentemente concluso la fase di registrazione presso i Vendor Server vista nel capitolo 4 e ricevuto le credenziali per accedere al servizio. Il passo successivo è creare un nuovo profilo Figura 5.3 e inserire le credenziali nel sistema per testare la corretta comunicazione con il/i Vendor Server, Questa operazione avviene tramite l'interfaccia web del sistema e restituisce le chiavi di accesso che identificano la risorsa. Il secondo scenario in Figura 5.3 rappresenta l'interazione che gli altri attori del sistema hanno con Switchboard, per facilità di comprensione ho rappresentato la situazione in cui un attore umano interagisca con le API del sistema, ma queste stesse API potrebbero essere utilizzate da altri componenti o tramite meccanismi automatizzati. L'attore che chiameremo fruitore, per interagire con il sistema deve aver ricevuto dal manutentore le credenziali di accesso associate ad un determinato profilo. Tramite queste credenziali può utilizzare le Client API per inviare notifiche verso i dispositivi registrati. Il Diagramma in Figura 5.4 invece delinea i compiti dell'utente finale. L'utente ha il compito di registrarsi e disiscriversi dal sistema tramite le funzioni incluse nell'API lato device e se il profilo lo offre iscriversi a determinate liste. Come spiegato nel libro [15], i diagrammi presentati derivano dalla lista dei compiti degli attori, nella quale vengono elencati per ciascuno le attività di più alto livello e fissati i punti principali per la loro esecuzione.

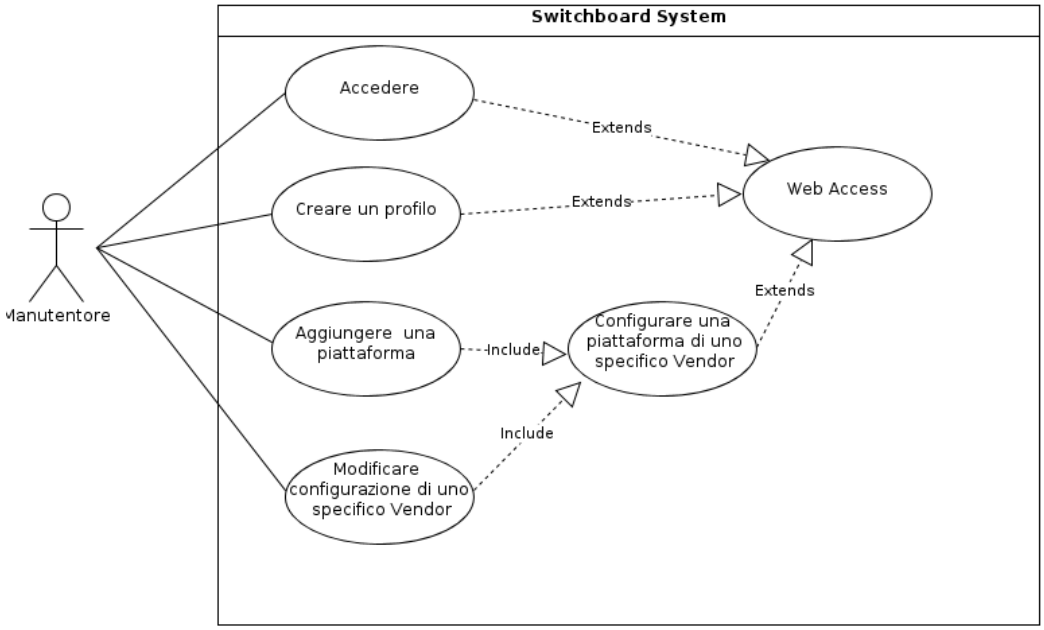

Figura 5.2: Caso d'uso del Manutentore

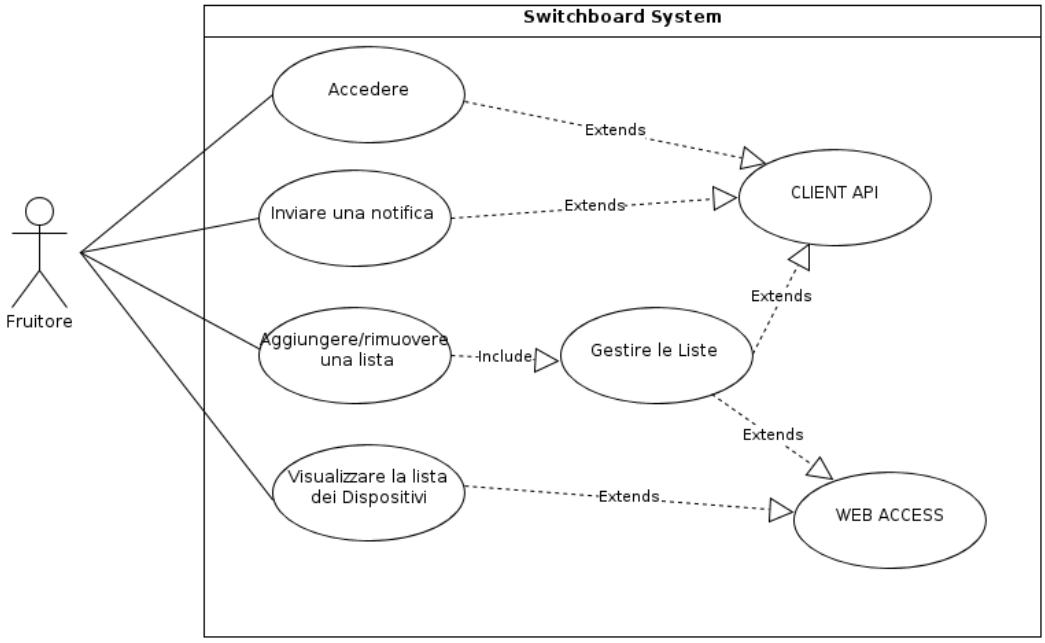

Figura 5.3: Caso d'uso del Fruitore

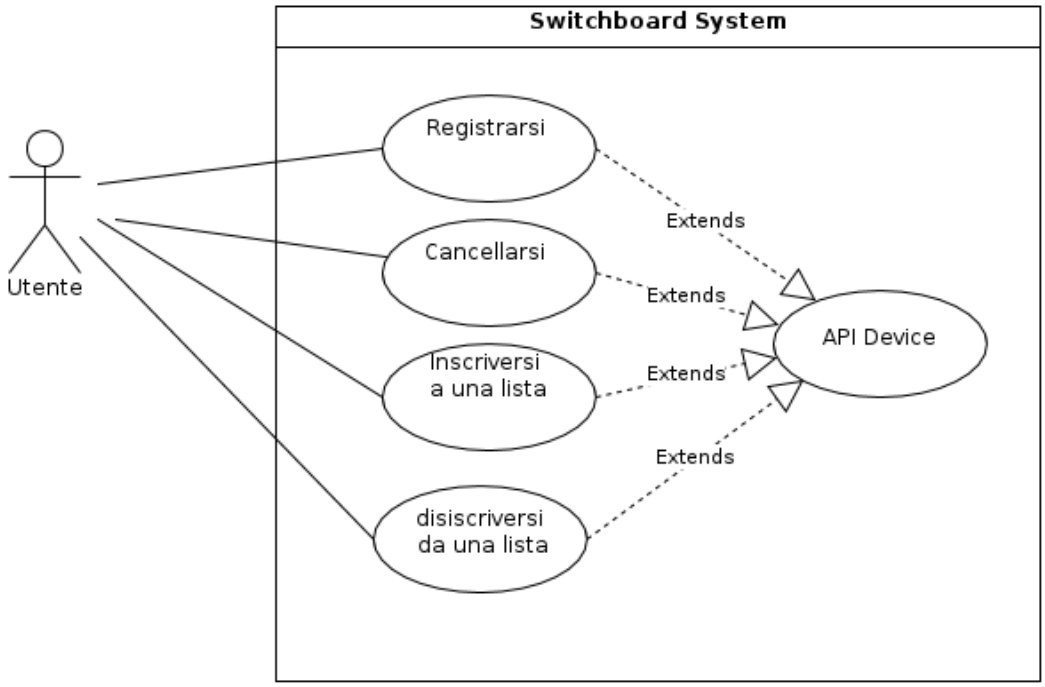

Figura 5.4: Caso d'uso dell'Utente

Forma testuale dei casi d'uso

Di seguito sono elencati in forma testuale e più dettagliata i casi d'uso principali:

Caso D'uso #1	Registrare un profilo
Obiettivo:	Registrare nel sistema una nuovo profilo e ricevere Le chiavi publish e subscribe per Interagire con Il sistema
Sistema di Riferimento:	Interfaccia Web dell'applicazione switchboard
Attore Primario:	Manutentore
Pre-condizioni:	Il profilo non è presente nel database, Il sistema è accessibile e attivo
Post-Condizioni Per Successo:	Il sistema ha registrato il profilo, restituito Le chiavi publish e subscribe e pronto a registrare i dispositivi
Post-Condizioni Per Fallimento:	Il sistema non ha concluso la procedura di registrazione e non ha restituito le chiavi di accesso
Descrizione:	1. Il manutentore accede al sistema 2. Il manutentore registra il nuovo profilo Inserendo: il nome del profilo e altre credenziali 3. Il sistema registra il nuovo profilo e visualizza le chiavi d'accesso

Caso d'uso #2	Configurare una piattaforma
Obiettivo:	Configurare una piattaforma per l'invio di notifiche
Sistema di riferimento:	Interfaccia web dell'applicazione "Switchboard"
Attore primario:	Sviluppatore
Pre-condizioni:	almeno un profilo è presente nel sistema, avere acquisito le credenziale dal fornitore del servizio di push notifiche
Post-Condizioni per Successo:	Il sistema non restituisce errori
Post-Condizioni per Fallimento:	Il sistema ha restituito eccezioni
Descrizione:	1. Accedere al proprio account e scegliere il profilo che si vuole configurare 2. accedere alla pagina di configurazione relativa alla piattaforma che si vuole configurare 3. Inserire i dati negli appositi campi 4. salvare la configurazione

Caso d'uso #3	Registrare altri profili sullo stesso account
Obiettivo:	aggiungere ad un account già esistente un altro profilo
Sistema di riferimento:	Interfaccia web dell'applicazione switchboard
Attore primario:	Sviluppatore
Pre-condizioni	L'account è già registrato al sistema , Il sistema è accessibile e attivo
Post-Condizioni per Successo:	Il sistema ha registrato il profilo, restituito le chiavi publish e subscribe e pronto a registrare i dispositivi
Post-Condizioni per Fallimento:	Il sistema non ha concluso la procedura di registrazione e non ha restituito le chiavi di accesso
Descrizione:	1. Lo Sviluppatore si autentica al sistema 2. Lo Sviluppatore registra il nuovo profilo Inserendo: il nome del profilo, l'indirizzo email dello sviluppatore 3. Il sistema registra il nuovo profilo e visualizza le chiavi d'accesso

Caso d'uso #4	Registrare un dispositivo
Obiettivo:	Registrare un nuovo dispositivo nel sistema
Sistema di riferimento:	applicazione per dispositivo, switchboard system
Attore primario:	Utente
Pre-condizioni:	L'applicazione ha integrato le libreria switchboard e ottenuto la chiave d'accesso
Post-Condizioni per Successo:	Il sistema ha registrato il dispositivo e inviato l'ack all'applicazione
Post-Condizioni per Fallimento:	l'applicazione non ha ricevuto l'ack dal server
Descrizione:	1. L'utente avvia l'applicazione 2. L'applicazione si registra al proprio server Push 3. L'applicazione invia il token ricevuto dal Vendor Server al sistema Switchboard

Capitolo 6

Implementazione di Switchboard

6.1 Scelte implementative

Nel processo di sviluppo di software la fase di analisi dei requisiti visti in 5.2 e la stesura dei casi d'uso in 5.3 serve per chiarire i confini dell'applicazione e scegliere le tecnologie più adatte alle necessita. Inoltre prima di iniziare con la stesura del codice è importante cercare patterns architetturali che più si adattano al progetto, per prevenire errori e aiutare lo sviluppatore nella realizzazione del progetto. Dalla mia analisi il pattern architetturale più vicino al progetto in esame è di tipo publish subscribe [13]; molti dei concetti e delle soluzioni proposte in questo progetto possono essere viste come un estensione di questo pattern architetturale. Dopo una prima analisi ho definito il ruolo di quattro entità importanti: *Device, Profile, Message e Configuration*:
L'entità Device rappresenta il dispositivo, ogni dispositivo è identificato da un id univoco valido nel sistema Switchboard e da un *device_id* che può essere visto come l'indirizzo univoco di una particolare applicazione installata in un particolare dispositivo; In aggiunta a questi dati durante il processo di registrazione il sistema raccoglie ulteriori informazioni relative al disposi-

tivo come: versione switchboardAPI, tipo e versione del sistema operativo (BlackBerry, iPhone, Andorid, Windows) queste informazioni servono per adattare le notifiche alle diverse versioni in caso di aggiornamenti delle specifiche da parte delle case produttrici. L'entità *Profile* invece rappresenta in maniera univoca un uno profilo cioè l'insieme di tutte le opzioni (credenziali, configurazioni) richieste per una applicazione. In profile troviamo le chiavi di accesso del sistema per le interfacce Client e Device.
Configuration invece rappresenta l'insieme di credenziali per instaurare un canale di comunicazione con un particolare Vendor Server.
L'ultima entità in gioco è *Message* in essa vengono salvati i dati relativi alla notifica, è stato indispensabile creare un entità che astrae il concetto di notifica perché in base alle diverse piattaforme l'utente deve essere in grado di utilizzare tutte le opzioni previste dalla piattaforma; ex. in iOS è possibile inviare badge piuttosto che notifiche testuali. Nel sotto package *org.switchboard.connector* in esso sono contenute le classi incaricate ad instaurare la comunicazione con i Vendor Servers. Nel sotto package *org.switchboard.controller* sono presenti le classi ProfileController e DeviceController in accordo con i principi del pattern Model View Controller (MVC) gestiscono e processano rispettivamente le richieste http provenienti dalle interfacce dei Client e dei Dispositivi. Invece nel sotto package *org.switchboard.service* sono presenti le classi: DeviceService e ProfileService che hanno il compito di elaborare le richieste provenienti dai Controller e interfacciarsi con il database.

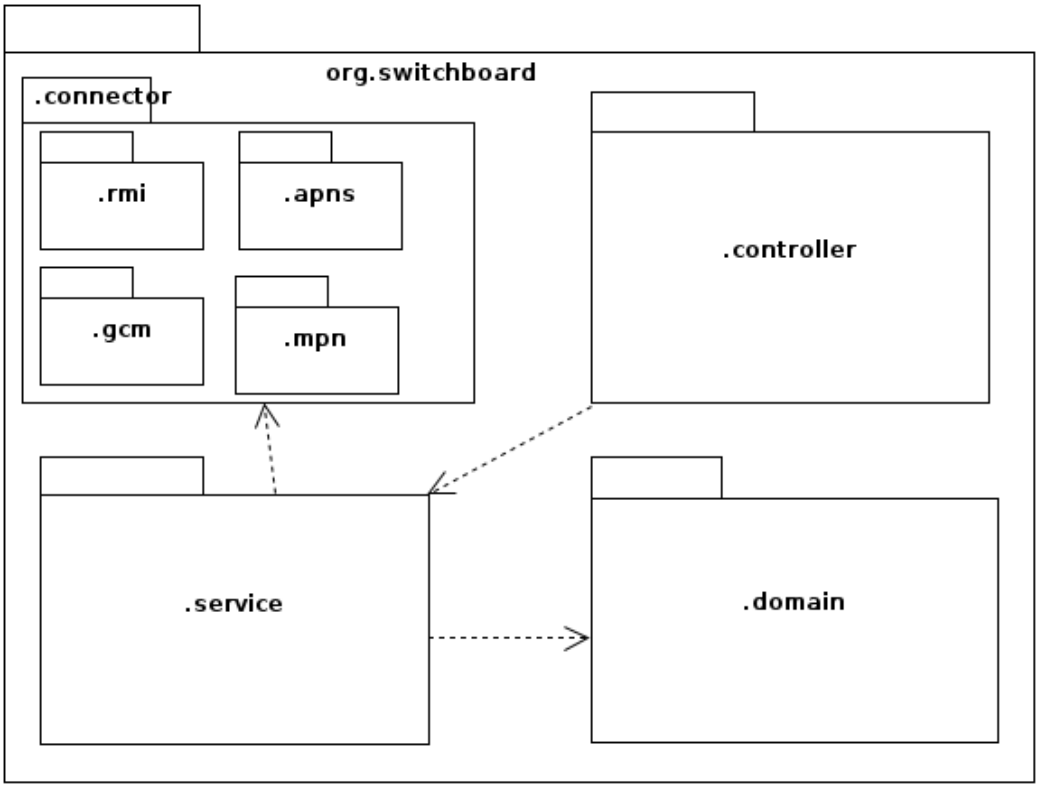

Figura 6.1: Diagramma dei package del sistema Switchboard

6.2 Principio di funzionamento

In questo paragrafo descriverò il funzionamento del sistema, verrano utilizzati dei diagrammi di sequenza per descrivere in dettaglio le componenti coinvolte. La prima azione che andremo ad analizzare è la registrazione di un nuovo profilo schematizzato in Figura 6.2; In accordo con il capitolo 5 faremo riferimento con la parola Client agli utenti che usufruisco del sistema tramite le Client API, con Device ai dispositivi che usufruiscono del servizio di push notification tramite le Device API e con Configurazione all'insieme delle credenziali per instaurare una connessione con uno specifico Vendor Server. La registrazione avviene tramite l'interfaccia web, il manutentore ha il compito di registrare un nuovo profilo nel sistema tramite questa interfaccia. Il primo passo è l'inserimento dei dati tramite la view addprofile gestita dal

ProfileController il sistema provvederò a salvare in persistenza i dati dell'applicazione, tramite il ProfileService e restituire al controller le chiavi univoche alfanumeriche *device key* e *cliente key*. Queste chiavi servono per accedere al sistema tramite le API esposte rispettivamente dal lato Client e lato Device. Le chiavi di accesso hanno validità finché il profilo esiste nel sistema e non possono essere modificate. Una aspetto importante riguardante la *device key* è che le registrazioni da parte dei dispositivi di una determinata piattaforma non sarà permessa finché non sarà aggiunta una configurazione corretta della stessa piattaforma. Questa limitazione è risultata essenziale per il corretto funzionamento del sistema, per evitare che dispositivi non registrate precedentemente presso i Vendor Server, quindi non abilitate alla ricezione delle notifiche push, siano presenti nella lista dei dispositivi associati ad un profilo.

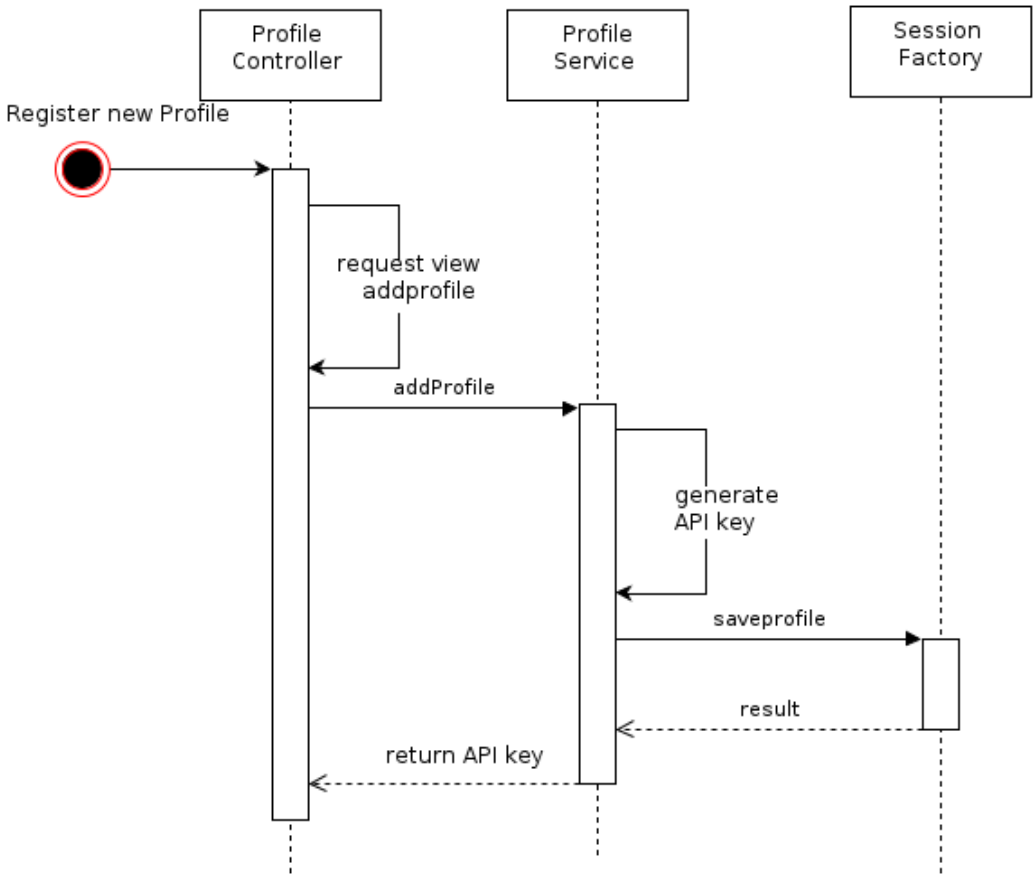

Figura 6.2: Diagramma di sequenza che rappresenta l'aggiuntai di un nuovo profilo

Il passo successivo è aggiungere al profilo creato una nuova configurazione, diagramma in Figura 6.3 quest'azione viene svolta tramite l'interfaccia web; dopo aver richiesto l'accesso al pagina di configurazione del profilo, in questa pagina si può scegliere dalla lista delle piattaforme supportate quale piattaforma si vuole configurare, che nel momento della della stesura sono: Andorid, Black Berry, Apple iOS e Microsoft Windows Phone. Da questo punto di vista le piattaforma discostano le une dalle altre per le credenziali utilizzate, ad esempio la piattaforma Google Cloud Messaging per i dispositivi Android necessita solo di una chiave detta API key ottenibile dalla console di google per instaurare la connessione con il server GCM. La piattaforma

di RIM per BlackBerry necessita di un indirizzo univoco e la copia APP ID e password ottenibili dopo una lunga registrazione e l'approvazione da parte dell'azienda stessa. questo argomento è stato affrontato in dettaglio nel capitolo 4. Una volta aggiunta almeno una configurazione il sistema abilita la registrazione dei dispositivi ed è pronto a ricevere istruzioni dal lato Client e registrare dispositivi dal lato Device.

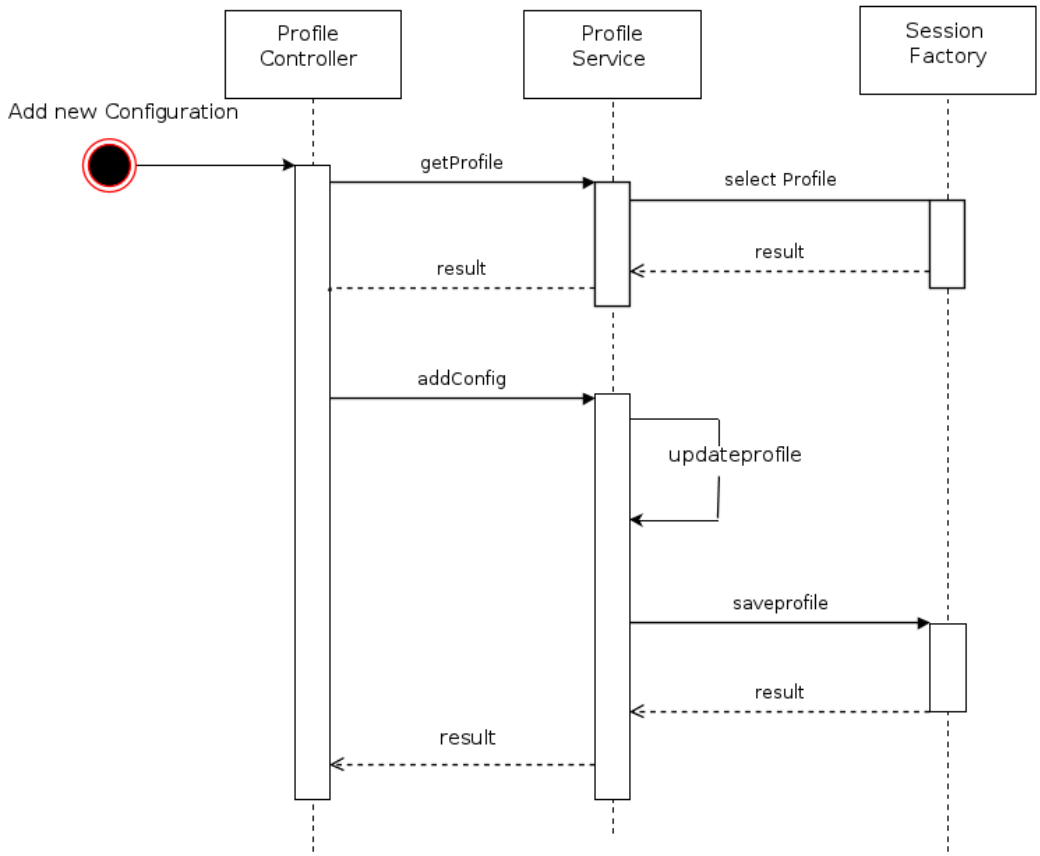

Figura 6.3: Diagramma di sequenza che rappresenta l'aggiunta di una configurazione

L'ultimo diagramma che andremo analizzare Figura 6.4 mostra come le differenti componenti interagiscono tra loro. L'azione descritta rappresenta l'invio di una notifica da parte del Client tramite le API del sistema, la richiesta è contenuta in un oggetto JSON. Il primo componente coinvolto è il

sistema switchboard che ha il compito di gestire la richiesta e controllare se effettivamente il Client è autorizzato a inviare le notifiche e se i dati sono validi. Se la richiesta viene accettata dal sistema, esso interroga il database per avere la lista dei dispositivi associati al profilo e prepara la notifica . Essa verrà adattata dai vari connettori per ogni specifico Vendor Server ed inviata agli stessi che prenderanno in gestione la notifica e la consegneranno al dispositivo. Successivamente il sistema switchboard ricontatterà i Vendor Server per sincronizzare la lista dei dispositivi abilitati alla ricezione delle notifiche con la lista dei dispositivi registrati in switchboard questo passaggio è di vitale importanza per evitare overhead delle richieste; ad esempio richieste che contengano riferimenti a dispositivi che non sono più registrati o che non hanno più installato l'applicazione di riferimento. Come già detto nel capitolo 4 nessuna piattaforma presa in esame fornisce un meccanismo per garantire la ricezione della notifica da parte del dispositivo quindi il sistema non si preoccupa di restituire un feedback al Client. Per correttezza va specificato che fa eccezione la piattaforma BlackBerry con il suo servizio riservato ad un utenza enterprise, che non verrà trattato in questa tesi.

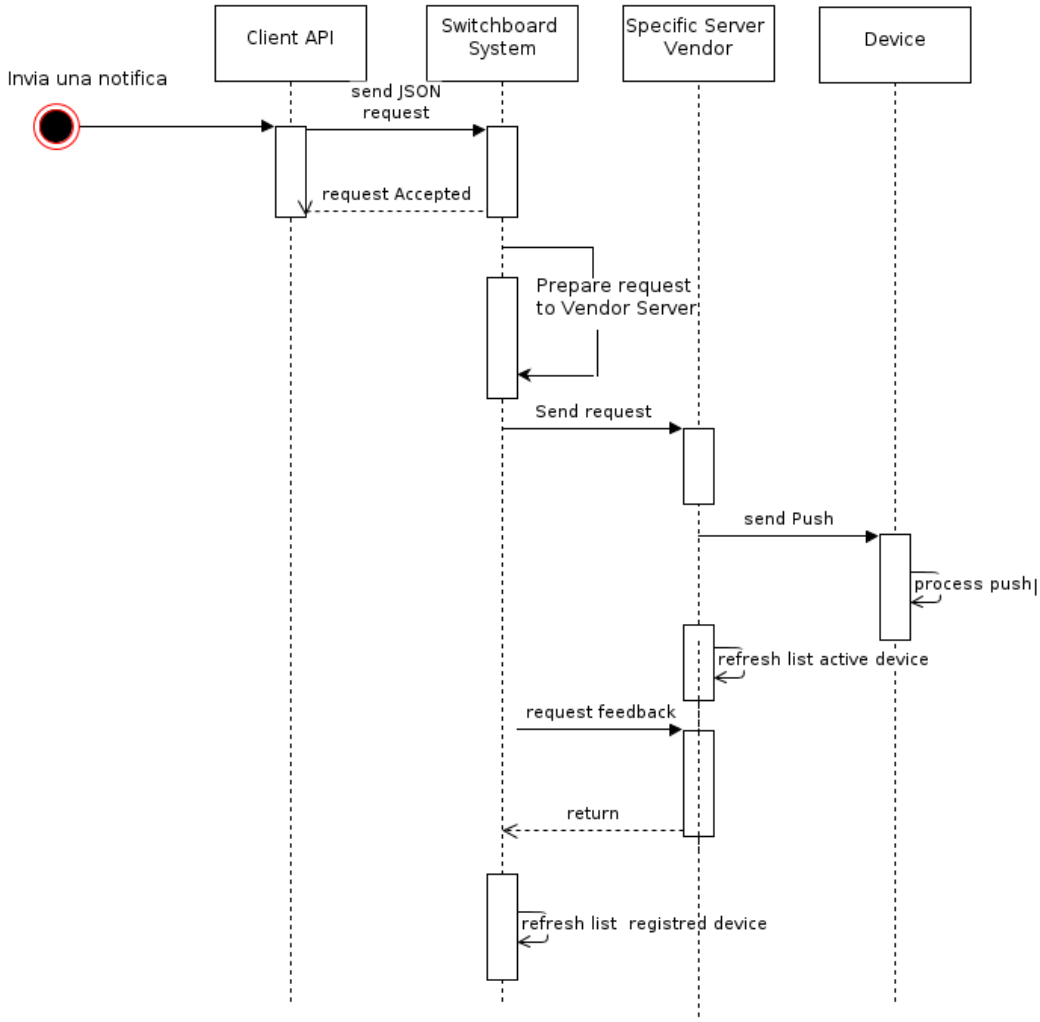

Figura 6.4: Diagramma di sequenza che rappresenta l'invio di una push notification

6.3 API del sistema

Di seguito vedremo come è strutturata una richiesta POST HTTP per inviare un notifica
Resource URL: `http://switchboard:8080/Client/v1/request.json`

Esempio di body di un messaggio

```
{  "CLIENTKEY" : "jd4802k6ge0phop9",
"DEVICETYPE" : "ANDROID",
"TIMETOLIVE" : "WEEK" ,
 "TEXT" :{ "test":,"test"
  "test1":,"test1"
  "test1":,"test1"}
 }
 "NOTIFICATIONTYPE" : "text"}
```

Parametri

Per la creazione di una notifica ho mi sono basato sulle linee guida dalle piattaforme Andorid e iPhone, come formato ho preferito JSON molto più leggere è intuitivo del XML per alleggerire il lavoro del parse del sistema. inoltre in questo modo per le due piattaforme sopra citate è possibile incapsulare direttamente il messaggio nella richiesta verso i Server sottraendo i parametri che svolgono una funzione solo nel contesto di switchboard. Nella Tabella 6.3 sono descritti i parametri utilizzabili per interagire con il sistema.

Parametri	Descrizione	Tipo	Richiesto
CLIENTKEY	chiave ricevuta al momento della registrazione di un nuovo profilo	String	YES
DEVICETYPE	con questo parametro specifichiamo il tipo di dispositivo a cui inviare la notifica: ALL (tutti i dispositivi) ANDROID (solo dispositivi Android) APPLE (solo dispositivi Apple) BLACKBERRY (solo dispositivi BlackBerry) WINDOWS (solo dispositivi Windows Phone)	String	YES
TIMETOLIVE	Con questo parametro ci riferiamo al tempo che la notifica rimane valida nelle code di invio dei Vendor Server in caso il dispositivo non è connesso alla rete dati. Il sistema accetta quattro possibili varianti: NOW (la notifica verrà ricevuta solo dai dispositivi connessi alla rete dati in quel momento) 24H (La notifica interesserà tutti i dispositivi che si connetteranno alla rete dati nelle 24 ore) WEEK (La notifica ha validità una settimana) MAX(viene impostato il massimo valore disponibile per ogni piattaforma).	String	NO
TEXT	la forma testuale della notifica	String	NO
NOTTYPE	con questo parametro possiamo scegliere il tipo di notifica che è possibile inviare, se una delle piattaforme non è compatibile con il tipo di notifica, per quella specifica piattaforma verrà utilizzata la tipologia standard TEXT (notifica testuale) ALLARM (notifica sonora) BADGE (supportato solo da dispositivi Apple)	String	NO

Response format

Codice	Descrizione
accepted	la richiesta è stata accettata dal sistema
no profile associated	Questa situazione si può verificare se il parametro CLIENTKEY è errato o se il profilo associato è stato cancellato
error parse json parameter	Errore relativo al parametro riportato nel messaggio, siamo nel caso in cui non viene rispettata la specifica della richiesta. Ex. TIMETOLIVE : un tot
no supported platform	quest'errore si può verificare nella situazione in cui viene specificata una piattaforma tramite il parametro DEVICETYPE ma nel sistema non sono presenti dispositivi appartenenti a quella piattaforma

Di seguito vedremo come è strutturata una richiesta di registrazione da parte di un dispositivo

Resource URL: `http://switchboard:8080/Device/v1/request.json`

Esempio di body di un richiesta di registrazione

```
{  "ACTION" :  REGISTER
"DEVICEKEY" : "jd4802k6ge0phop9",
"TYPE" : "ANDROID",
"DEVICE_ID" : "gskjfgkasdjfgkjsadbhfkjsadbijfn" ,
 "OS VERSION" : "4.1 "
 "API_VERION" : "v0.1"}
```

Parametri	**Descrizione**	**Tipo**	**Valore Richiesto**
ACTION	parametro che specifica l'intento della richiesta al momento della stesura è possibile inserire solo REGISTER	String	YES
DEVICEKEY	chiave ricevuta al momento della registrazione di un nuovo profilo	String	YES
TYPE	con questo parametro viene specificato il tipo di dispositivo che vuole registrarsi: ANDROID (dispositivi Android) APPLE (solo dispositivi Apple) BLACKBERRY (solo dispositivi BlackBerry)	String	YES
DEVICE_ID	questo parametro essenziale per l'invio della notifica da parte del sistema è un identificativo che rappresenta univocamente il dispositivo.	String	YES
OS_VERSION	questo parametro si riferisce alla versione del sistema operativo del dispositivo	String	NO
API_VERSION	questo parametro si riferisce alla versione delle API di switchboard	String	NO

Tabella 6.1: Parametri API lato Client switchboard

Codice	Descrizione
registred	la registrazione è avvenuta con successo
no profile associated	Questa situazione si può verificare se il parametro DEVICEKEY è errato o se il profilo associato è stato cancellato
error parse json parameter	Errore relativo al parametro riportato nel messaggio, siamo nel caso in cui non viene rispettata la specifica della richiesta.

Conclusioni

Nel corso di questo lavoro ho approfondito gli aspetti fondamentali di una tecnologia che nei dispositivi mobile è ancora giovane. La Tecnologia Push viene incontro alle crescenti esigenze delle applicazioni di interagire con l'utente anche quando non sono in primo piano. Si è visto come il moderno trand Bring your own device (BYOD) in un contesto business abbia preso piede tanto da spingere aziende a sviluppare applicazioni multipiattaforma. Questo insieme di aspetti ha spinto il mio lavoro a non soffermarsi su una specifica piattaforma, ma ad ampliare la visione verso un buon sottoinsieme di esse, che in primis supportano la tecnologia push e secondo luogo raggiungano un certo livello di maturità per presentarsi in un contesto business. In una prima fase si è esplorata la tecnologia push, visti quali sono i benefici che questa tecnologia apporta e i rischi legati ad un uso scorretto. Successivamente si è aperto il discorso verso il concetto di Cloud visti i diverse tipologie (*pubbliche, private, ibride*) classificati i modelli di servizio (*IaaS, PaaS, SaaS*) per poi concentrarsi sulle realtà delle private cloud, poiché è proprio in questa branca che il mio studio si vuole posizionare. Nell'esplorazione dello stato dell'arte sono state prese in considerazione realtà di private cloud e non, che offrono servizi per dispositivi mobile e integrano la tecnologia Push. Nel capitolo sulle piattaforme si è preso in esame le piattaforme più di successo che in quest'ultimo periodo coprono se non la totalità la maggior fetta del mercato mobile. Si è visto come le case produttrici offrono agli sviluppatori questo servizio dal punto di vista delle caratteristiche, standard e delle limitazioni. Importante per

questo lavoro è stato comparare l'architettura e le caratteristiche di ognuna di esse. Dopo questa lunga fase di knowledge e di test sulle piattaforme per comparare i risultati e comprendere come questi servizi si potessero sfruttare al meglio, ho cercato di concretizzare questa lavoro con la progettazione e la realizzazione di una componente integrabile in una private cloud, per semplificare l'invio delle push notification verso le differenti piattaforme. L'idea è appunto di gestire e nascondere la parte di comunicazione con i server Push delle case produttrici ed esporre delle semplici API che permettano di inviare push notification verso i dispositivi registrati al servizio. Un private cloud aziendale potrebbe utilizzare questo servizio come inizializzatore per una connessione protetta con altri servizi offerti dalla cloud.

Sviluppi futuri

La tecnologia Push è relativamente nuova nel mondo mobile, quindi si può auspicare che le case produttrici continueranno ad investire risorse per rendere più completo il servizio. Un punto critico nello stato dell'arte è che nessuna piattaforma, presa in esame, offre garanzie sulla ricezione e sull'ordine dei messaggi da parte dei dispositivi. Nei sistemi distribuiti questa criticità è molto discussa e non mancano algoritmi basati su feedback e ritrasmissione dei messaggi, per rendere affidabile un servizio con queste caratteristiche. Quindi si potrebbe pensare di rendere affidabile l'invio di questi messaggi. Inoltre le linee guida di alcune case produttrici al riguardo di questo servizio sconsigliano di utilizzare questo servizio per inviare dati sensibili perché non è garantita la sicurezza. Quindi un'altro passo futuro potrebbe essere di aggiungere al sistema ed alle API rilasciate per le piattaforme, funzioni comuni di codifica dei dati. Data l'entità del progetto e la natura multipiattaforma il prototipo realizzato verrà rilasciato con licenza open source.

Bibliografia

[1] Aspetti gestionali e implicazioni normative nell'implementazione del cloud computing nella pubblica amministrazione. http://nexa.polito.it/2012-cloud-pa.

[2] Blog post urban airship hybridcloud. we are up.

[3] Docmentazione ufficiale apple push notification services. http://developer.apple.com/library/mac/documentation/NetworkingInternet/Conceptual/R

[4] Docmentazione ufficiale blackberry push service.

[5] Docmentazione ufficiale google cloud messaging. http://developer.android.com/guide/google/gcm/index.html.

[6] Ibm con acquisizione di worklight punta sempre di piu' sul mobile. http://www-03.ibm.com/press/it/it/pressrelease/36688.wss.

[7] Ibm worklight official documenation. http://publib.boulder.ibm.com/infocenter/wmbhelp/v8r0m0/index.jsp?topic=

[8] Private cloud computing for enterprises: Meet the demands of high utilization and rapid change. http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns836/ns976/white_p 543729.pdf.

[9] Push access protocol. http://www.openmobilealliance.org/tech/affiliates/wap/wap-247-pap-20010429-a.pdf.

[10] Universal device service 6.1 feature and technical overview. http://docs.blackberry.com/en/admin/deliverables/44623/Universal_Device_Service_6.1_-_Feature_and_Technical_Overview.pdf.

[11] Urban airship official documetation. https://docs.urbanairship.com/display/DOCS/Home.

[12] Nadeen El Ajou. Bring your own device trend is ict industry's hottest talking point at gitex technology week. http://www.ameinfo.com/bring-own-device-trend-ict-industrys-312613.

[13] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Comput. Surv.*, 35(2):114–131, June 2003.

[14] WAP Forum. Wap push architectural overview. technical specifications. http://www.openmobilealliance.org/tech/affiliates/wap/wap-250-pusharchoverview-20010703-a.pdf, July 2002.

[15] Martin Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3 edition, 2003.

[16] Jarle Hansen, Tor-Morten Gronli, and Gheorghita Ghinea. Cloud to device push messaging on android: A case study. In *Proceedings of the 2012 26th International Conference on Advanced Information Networking and Applications Workshops*, WAINA '12, pages 1298–1303, Washington, DC, USA, 2012. IEEE Computer Society.

[17] YoungSuk Kim, JoonWoo Lee, SeongRae Park, and ByoungCheol Choi. Mobile advertisement system using data push scheduling based on user preference. In *Proceedings of the 2009 conference on Wireless Telecommunications Symposium*, WTS'09, pages 280–284, Piscataway, NJ, USA, 2009. IEEE Press.

[18] Steve Mansfield-Devine. Interview: BYOD and the enterprise network. *Computer Fraud & Security*, 2012(4):14–17, April 2012.

[19] Peter Mell and Tim Grance. The NIST Definition of Cloud Computing. Technical report, July 2009.

[20] Zhi Xu and Sencun Zhu. Abusing notification services on smartphones for phishing and spamming. In *Proceedings of the 6th USENIX conference on Offensive Technologies*, WOOT'12, pages 1–1, Berkeley, CA, USA, 2012. USENIX Association.

Ringraziamenti