

UNIVERSITA' DEGLI STUDI DI TRENTO

Facoltà di Scienze Matematiche, Fisiche e Naturali



UNIVERSITY OF TRENTO - Italy

Corso di Laurea

Informatica

Final Thesis

TITLE

WhatsUp Sync: modulo per l'esposizione sincronizzata di testo e immagini su dispositivi mobili

Relatore/1st Reader:

Name Maurizio Marchese
Institution

Laureando/Graduant:

Name Simone Melchiori

Correlatore:

Name Cristhian Parra

Anno accademico 2011 - 2012

Indice

1	Introduzione	1
2	I dispositivi mobili e i più anziani	3
2.1	La programmazione per dispositivi mobili	3
2.2	I principali sistemi operativi mobile e i loro ecosistemi	4
2.3	I tablet	6
2.4	I web server	6
2.5	La tecnologia al servizio dei più anziani	7
3	WhatsUp Sync	11
3.1	Motivazione e modello di utilizzo	16
3.2	Le notifiche push	17
3.2.1	APNs	17
3.2.2	Il server Uniquush	18
3.2.3	Uniquush nelle Social API	19
3.3	Protocollo di sincronizzazione	21
3.3.1	Avvio di una sessione	21
3.3.2	Visualizzazione di una risorsa	22
3.3.3	Fine di una sessione	22
4	Conclusioni	25
4.1	Limiti dell'implementazione	25
4.2	Implementazioni iniziali	26
4.3	Possibili sviluppi futuri	27
4.4	Esperienza accumulata	29
	Bibliografia	31

Capitolo 1

Introduzione

Tra le diverse generazioni c'è una grossa differenza nei metodi di comunicazione. I più anziani prediligono una comunicazione sincrona, come per esempio il telefono, oppure la ancora più diretta comunicazione "faccia a faccia".

I più giovani invece utilizzano sempre più spesso sms, chat, oppure social network per comunicare tra loro.

Questa differenza tra le generazioni ha fatto sì che la comunicazione intergenerazionale sia diventata più difficile, anche perché i dispositivi che utilizzano i più giovani per comunicare possono risultare difficili da usare per i più anziani, come i personal computer, gli smartphone, oppure i più recenti tablet.

In particolare questi ultimi due stanno entrando sempre di più nelle tasche delle persone, grazie anche ai prezzi di vendita che stanno diventando sempre più bassi. Ad aggiungersi all'abbassamento dei prezzi vi è anche la maggiore intuitività e facilità d'utilizzo che caratterizza questi dispositivi, anche grazie ai touch screen installati.

Tuttavia, insieme ad un hardware che facilita l'utilizzo anche il software deve essere progettato in modo tale da essere utilizzato facilmente. Spesso infatti i software per dispositivi mobili non sono abbastanza intuitivi, e i più anziani hanno delle difficoltà ad utilizzarli.

WhatsUp è un esempio di software "elder friendly" pensato per essere facile da utilizzare e per fornire un canale di comunicazione tra le diverse generazioni. Con esso il più giovane può rimanere in contatto col più anziano, condividendo messaggi o foto

Il lavoro di questa tesi si inserisce proprio in questo software: un modulo per esporre in modo sincronizzato i contenuti condivisi in modo asincrono dal giovane al più anziano.

Nel primo capitolo verrà discusso della diffusione dei dispositivi mobili, in che modo hanno rivoluzionato il mercato, come funzionano, e che ruolo

possono avere per aiutare i più anziani. Nel secondo capitolo sarà esposta la parte focale di questa tesi, cioè quella riguardante il modulo implementato in WhatsUp, la sua architettura e come funziona. Nel terzo capitolo, infine, vengono esposti i limiti di questa implementazione, il percorso fatto da questa applicazione per diventare quello che è, e infine gli sviluppi futuri che questa applicazione potrà avere.

Capitolo 2

I dispositivi mobili e i più anziani

Recentemente si ha assistito ad un forte incremento della vendita di tablet e smartphone. In particolare nel mercato dei tablet, IDC, società operante nell'ambito della ricerca di mercato, ha rilevato circa 18 milioni di tablet venduti nel 2010, e circa 63 milioni del 2011[1]. Assieme ad essi, è anche aumentato il numero di sviluppatori di applicazioni per questi dispositivi, attratti dal vasto mercato che si è venuto a creare. Si contano infatti più di un milione di applicazioni presenti tra Google Play di Google e App Store di Apple.

Vedremo a seguire le peculiarità dello sviluppo di applicazioni per sistemi mobili, e come parte di queste applicazioni aiutano le persone più anziane nel migliorare la loro qualità di vita.

2.1 La programmazione per dispositivi mobili

In cosa differisce la programmazione per dispositivi mobili dalla programmazione per computer?

In primo luogo lo schermo: mentre normalmente i programmi sono organizzati in finestre, e possono essere visualizzate più finestre contemporaneamente, nella programmazione mobile viene normalmente visualizzata una schermata alla volta, che occupa tutto lo schermo. Inoltre la dimensione dello schermo non è standard, ce ne sono di svariate dimensioni e con proporzioni altezza-larghezza diverse. Questo mette lo sviluppatore di fronte alla necessità di progettare interfacce che si adattino ad ogni dimensione.

Un altro importante fattore da considerare è che le risorse hardware disponibili sono molto più limitate rispetto a quelle di un computer, nonostante ci siano enormi e continui progressi anche in questa direzione.

Bisogna programmare considerando una connettività molto mutevole, a volte il WI-FI potrebbe non essere disponibile, oppure la rete GSM potrebbe essere assente, mentre si viaggia in galleria per esempio.

Un dispositivo mobile mette a disposizione molte altre risorse, come per esempio il GPS, l'accelerometro, oppure la fotocamera. Un cattivo utilizzo di queste risorse, dovuto a una cattiva programmazione dello sviluppatore, potrebbe portare a una drastica riduzione della durata della batteria.

Lo sviluppatore ha quindi un ruolo cruciale nella programmazione di applicazioni efficienti, che non vadano a rovinare l'esperienza d'uso del dispositivo da parte dell'utente.

2.2 I principali sistemi operativi mobile e i loro ecosistemi

I principali sistemi operativi mobile, attualmente, sono Android e iOS, rispettivamente di Google e Apple. Tuttavia, definirli dei semplici sistemi operativi potrebbe sembrare riduttivo, in quanto rappresentano un vero e proprio ecosistema software. Entrambi mettono a disposizione servizi dedicati al cloud computing, oppure servizi e-mail. Ma quello che ha davvero rivoluzionato il mercato sono gli store, grazie ad essi, e a delle librerie messe a disposizione da Google ed Apple, è relativamente semplice sviluppare la propria applicazione ed immetterla sul mercato.

Inizialmente entrambe si sono rivolte esclusivamente al mercato degli smartphone, fino al 2007, anno in cui è stato rilasciato da Apple il primo iPad. A seguire vennero messi in vendita molti altri tablet concorrenti, con Android installato. Sia Android che iOS mettono a disposizione degli ambienti di sviluppo, in cui è possibile creare, debuggare e testare la propria applicazione.

Con Android è vivamente consigliato (ma non obbligatorio) utilizzare Eclipse per sviluppare le applicazioni. Per utilizzarlo è necessario scaricare un plugin, che una volta installato mette a disposizione, in Eclipse, tutte le funzioni per la compilazione, la creazione delle interfacce, e l'esecuzione dell'applicazione in un emulatore o in un dispositivo reale. Inoltre questo plugin mette anche a disposizione un logger, con il quale visualizzare eventuali messaggi di errore, messaggi di attenzione, oppure eventuali messaggi inseriti

2.2. I PRINCIPALI SISTEMI OPERATIVI MOBILE E I LORO ECOSISTEMI

appositamente nel codice per scopi di debug. Infine offre dei template per aiutare lo sviluppatore a impostare lo scheletro dell'applicazione, creando le giuste cartelle e chiamandole con il nome corretto, ci saranno una o più cartelle destinate alle immagini, ai layout delle interfacce, ad eventuali altri file che potrebbero servire all'applicazione, e infine, naturalmente, al codice sorgente.

Il codice di un'applicazione per Android è scritto in Java, per le componenti dinamiche, e in XML per le componenti statiche. Come in Java, in Android è presente una Virtual Machine che esegue un bytecode, chiamata Dalvik Virtual Machine (DVM). Il codice Java viene compilato assieme al codice XML per produrre un bytecode, che verrà emulato dalla DVM. L'utilizzo di questa è a vantaggio degli sviluppatori, il bytecode è emulato nello stesso modo da tutte le DVM, quindi ogni applicazione può essere eseguita su ogni terminale, indipendentemente dal costruttore.

Per testare l'applicazione, Android, mette a disposizione, in assenza di un dispositivo reale, un emulatore. Con esso è possibile testare il proprio programma sul computer, utilizzando mouse e tastiera. In caso l'applicazione faccia uso della posizione dell'utente esso permette di impostare delle coordinate fittizie. È possibile configurare l'emulatore in modo tale da simulare al meglio le differenze di hardware che si possono incontrare nel mondo Android. Tuttavia l'utilizzo dell'emulatore pone di fronte ad alcuni limiti, come l'impossibilità di utilizzare i sensori, come accelerometro, bussola e simili. Tale limite è facilmente aggirabile utilizzando un dispositivo reale, collegandolo al computer.

Per sviluppare applicazioni per iOS invece è necessario utilizzare Xcode, un ambiente di sviluppo disponibile solo per Mac OS X. Anche con Xcode è possibile compilare facilmente il codice e tutte le altre funzioni disponibili anche in Eclipse. È quindi possibile anche qui visualizzare eventuali messaggi di errore, ed eseguire il debug.

Xcode è strutturato in modo da favorire l'uso del pattern Model-View-Controller, implementa un editor di interfacce che genera un file con estensione .xib. A uno o più file xib è attribuito un ViewController, che gestisce le modifiche all'interfaccia.

Il codice è scritto in Objective-C, linguaggio utilizzato anche per creare applicazioni per OS X, sviluppato dalla società NeXT e migliorato da Apple dopo l'acquisto prima della licenza, e dopo dell'intera società. NeXT è anche il produttore del costruttore di interfacce che è tutt'oggi implementato in Xcode. Visto che Xcode incorpora il compilatore gcc, è anche possibile utilizzare codice scritto in C e in C++, favorendo così la riutilizzabilità di codice scritto per altri applicativi non necessariamente mobile.

Anche programmando per iOS è possibile testare le proprie applicazioni sia su di un simulatore, messo a disposizione dall'ambiente di sviluppo, sia utilizzando un dispositivo reale. Il simulatore di iOS è più semplice dell'emulatore di Android, perché visto che iOS è installato unicamente su iPhone e iPad, non ci sono le grosse differenze relative all'hardware che ci sono invece in Android. Come per l'emulatore di Android, anche il simulatore di iOS ha delle limitazioni: infatti non è possibile testare importanti funzionalità come i sensori, la fotocamera e infine le notifiche push tramite APNs. Quindi anche qui è possibile utilizzare un device reale, tuttavia, è necessario prima essere iscritti al programma degli sviluppatori Apple, pagando una quota annuale.

2.3 I tablet

L'arrivo dei tablet ha portato una vera e propria rivoluzione, perché rappresenta un punto di incontro tra le potenzialità di un computer, e la portabilità di uno smartphone. L'utilizzo inoltre è molto semplice, poiché l'interazione tramite il touch screen, e quindi il gesto di "toccare", rende tutto estremamente intuitivo.

Grazie alla connettività di cui dispongono i tablet inoltre (solitamente WI-FI e GSM) è semplice effettuare degli scambi di dati, oppure creare una piccola rete.

Con queste funzioni non c'è da stupirsi se essi vengono impiegati in diversi settori. Infatti, è possibili trovarli nella borsa di un uomo d'affari, che controlla la posta in arrivo o prepara le sue presentazioni, oppure nella cartella di uno studente al posto dei libri.

2.4 I web server

Come accennato in precedenza la quasi totalità di questi dispositivi mobili ha la possibilità di collegarsi in rete. Per sfruttare al meglio questa potenzialità però è necessario che ci siano dei server web a cui collegarsi. Infatti, molte applicazioni si utilizzano la rete per svolgere i loro "compiti". Basta pensare alla più semplice delle applicazioni che ti indica il meteo nella tua città, o ad una più complessa che ti permette di interagire con i social network.

Un web server è un'applicazione in esecuzione su un computer host, in attesa di connessioni da parte di un client, che può essere un browser, il client per le email, oppure un dispositivo mobile. La connessione avviene utilizzando il protocollo TCP/IP, mediante la porta 80. Il server web si aspetta che le richieste siano formulate in un determinato modo, a seconda

del servizio che il server offre. Le risposte che il server invia sono codificate mediante un determinato linguaggio: HTML, XML, JSON ne sono degli esempi.

Un esempio di server web che è possibile eseguire è il server Apache. Questo server è organizzato in moduli, ognuno che svolge una funzione specifica. Questi moduli sono coordinati dal core. Come in una catena di montaggio ogni modulo prende un input, svolge il proprio compito e passa il risultato al prossimo modulo. In particolare, la richiesta del client viene tradotta, viene controllata la presenza di eventuali richieste dannose, il tipo di contenuto viene verificato, e viene inviata la risposta al client dopo aver avviato eventuali procedure. Durante tutto questo un modulo si occupa di tenere traccia tutto quello che è stato fatto.

Un server web è in grado di svolgere le più svariate funzioni, ricevuta una richiesta può, per esempio, accedere a un database, prelevare i dati che gli servono, generare una pagina HTML utilizzando questi dati e inviarla al richiedente. Inoltre la pagina HTML può essere differente a seconda se la richiesta viene effettuata da un computer desktop oppure da uno smartphone. Nel secondo caso, il server potrebbe generare una pagina ottimizzata per la visualizzazione su schermi ridotti.

Se la richiesta viene effettuata da un'applicazione, invece che mandare una pagina HTML può, per esempio, mandare un XML con i dati richiesti. A questo punto sarà compito dell'applicazione richiedente elaborare il file XML e mostrare i risultati a seconda delle sue esigenze.

2.5 La tecnologia al servizio dei più anziani

La tecnologia viene incontro ai più anziani in tre modi diversi, secondo il CAST center[2], raggruppati come:

1. Tecnologie per la sicurezza
2. Tecnologie per la salute e il benessere
3. Tecnologie per i rapporti sociali

Nel primo gruppo rientrano quelle tecnologie come, per esempio, i rilevatori di cadute, sensori che rilevano il fumo o la temperatura della casa, e simili.

Nel secondo gruppo quelle tecnologie come quelle che ti indicano quale medicina prendere in quale momento, o quando effettuare determinati esercizi.

Mentre nell'ultimo gruppo tutte quelle tecnologie per migliorare i rapporti sociali, dal telefono, alle videoconferenze, a dei servizi e-mail progettati ad-hoc per i più anziani, quindi molto intuitivi e di facile utilizzo.

In generale queste tecnologie, che possono per esempio essere applicazioni, vengono progettate secondo due diversi approcci[3]: uno orientato a mitigare gli handicap che possono sorgere con l'andare dell'età, e l'altro orientato a prevenire alcune situazioni spiacevoli che si possono verificare.

Quello che ora verrà approfondito sono le tecnologie per i rapporti sociali.

Secondo uno studio [4] l'interazioni dei più anziani sono molto limitate. Risultano ristrette a delle sporadiche telefonate, perlopiù partite dal più anziano. Sono molto ridotti anche gli incontri "faccia a faccia". La maggior parte dei nonni sente i propri nipoti meno di una volta ogni due settimane, spesso le telefonate durano meno di cinque minuti e vengono cominciate dal nonno (vedi figura 2.1).

Ora verranno esposti alcuni progetti che hanno come obiettivo la risoluzione di questo problema.

Uno di questi progetti è **Bettie**: si tratta di dispositivi che permettono la creazione di una rete sociale senza bisogno di avere un computer. L'utente inserisce il "pass amico" nel device e comincia immediatamente a ricevere aggiornamenti su quella persona. Vengono mostrate tutte le informazioni aggregate, come le foto su Facebook o Twitter o i messaggi che l'amico inserito condivide sui Social Network. Una nota dolente di questo progetto, è che ogni "pass amico" deve essere acquistato. Questo può risultare molto dispendioso e laborioso per una persona anziana poco pratica nell'uso del computer che vuole rimanere in contatto con la propria famiglia.

Un altro esempio è **Epigraph**: è un altro dispositivo, pensato per essere messo in un punto della casa tale che possa essere visto facilmente. La schermata del dispositivo è divisa in un certo numero di canali, uno per ogni membro della famiglia, ogni volta che un membro della famiglia invia un messaggio all'Epigraph, tramite e-mail, SMS oppure SMS, il suo canale viene aggiornato. Uno svantaggio di questo dispositivo, è quello di avere un numero prefissato e limitato di canali, che può essere sottonumerario rispetto al numero di componenti della famiglia con cui l'anziano vorrebbe rimanere in contatto.

Eldy è un software che propone di interagire con il computer con un'interfaccia semplice e intuitiva. Viene installato su computer con sistemi operativi Windows oppure Unix-like e viene seguito automaticamente all'avvio. Eldy conta circa 400000 utenti ed è disponibile in più lingue.

Nel capitolo a seguire verrà presentato invece WhatsUp, sviluppato dal team Lifeparticipation, presso l'Università di Trento, e WhatsUp Sync, il modulo per l'esposizione sincronizzata di contenuti, oggetto di questa tesi.

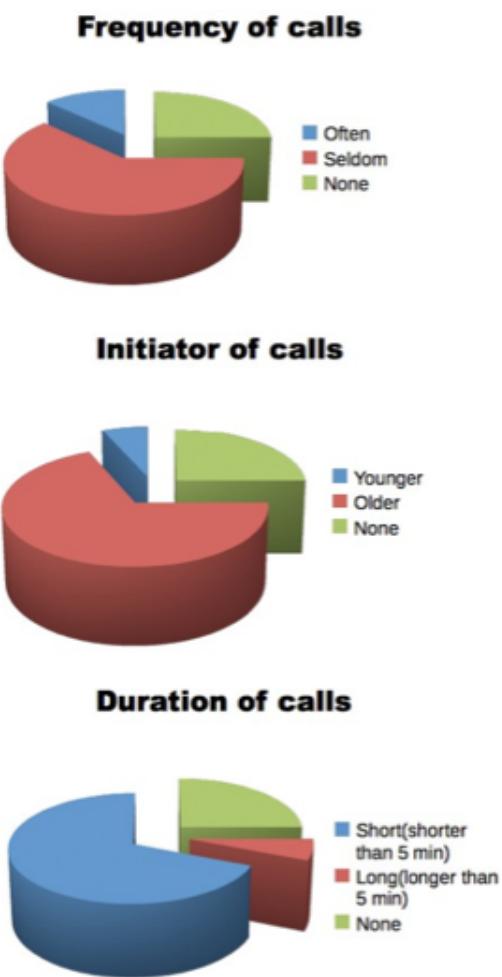


Figura 2.1: Grafici dello studio [4]

Capitolo 3

WhatsUp Sync

What'sUp è un programma, sviluppato per iOS e Android, che ha l'obiettivo di facilitare i rapporti intergenerazionali tra i giovani e gli anziani. Tramite il suo smartphone o tablet il giovane può condividere dei contenuti, che possono essere del testo, un'immagine o un'immagine con testo. Sono disponibili 5 interfacce, di complessità diverse. È possibile cambiare l'interfaccia a seconda dell'utente che usa l'applicazione. La più semplice permette all'utente solamente di vedere i contenuti che sono stati condivisi con lui, scorrendoli trascinando il dito sullo schermo. La seconda permette di vedere i contenuti scorrendoli con delle frecce ai lati dello schermo. La terza è come la seconda ma permette anche di esprimere un gradimento (positivo o negativo) sul contenuto. La quarta oltre a esprimere un gradimento permette di commentare un contenuto utilizzando la tastiera su schermo. Queste quattro interfacce possono essere utilizzate solo su tablet (solo iPad attualmente).

L'ultima interfaccia invece è riservata alla condivisione dei contenuti, è presente sia su smartphone, sia su tablet, ed è quella che deve essere utilizzata dal giovane.

I requisiti che WhatsUp vuole soddisfare sono i seguenti:

- Fornire un sistema di autenticazione degli utenti, in modo da poterli identificare in modo univoco
- Implementare un modello di Social Networking
- Fornire un sistema di interazione e di comunicazione
- Utilizzare un'infrastruttura comune per gestire le applicazioni (WhatsUp "lato giovane" e WhatsUp "lato anziano")

In figura 3.1 è possibile visualizzare il modello dei KSpace di WhatsUp. Ogni utente possiede uno spazio, che viene detto KSpace. In questo spazio, possono essere iscritti diversi altri utenti, detti Contributors, che possono inserire contenuti sul KSpace. È compito del proprietario del KSpace scegliere i Contributors. I Contributors potranno vedere soltanto i contenuti che loro stessi hanno condiviso, più gli eventuali commenti che il proprietario può lasciare.

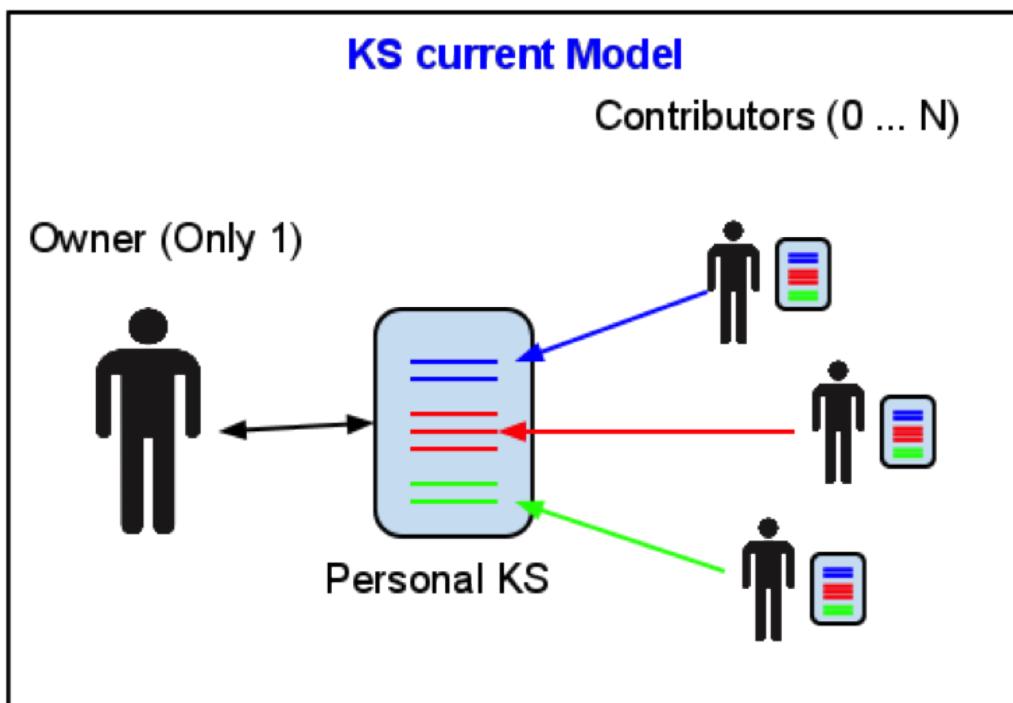


Figura 3.1: Modello dei KSpaces di WhatsUp.

In figura 3.2 è raffigurata l'architettura di WhatsUp: ci sono tre API, una per la gestione degli utenti, una per la gestione dei KSpaces e le risorse, ed una per la gestione dei file, eventualmente inviati nei KSpace. Ogni API fa riferimento ad un suo Database, e possono essere contattate da entrambi i lati dell'applicazione, quello dedicato ai giovani e quello dedicato agli anziani.

Per l'autenticazione degli utenti ci si appoggia su Facebook e il suo sistema d'autenticazione.

Il funzionalmento di WhatsUp è molto semplice (3.3): il Contributor (nel'immagine rappresentato a sinistra) seleziona a chi mandare un contenuto, può scegliere di scrivere un messaggio, scattare una fotografia, oppure ag-

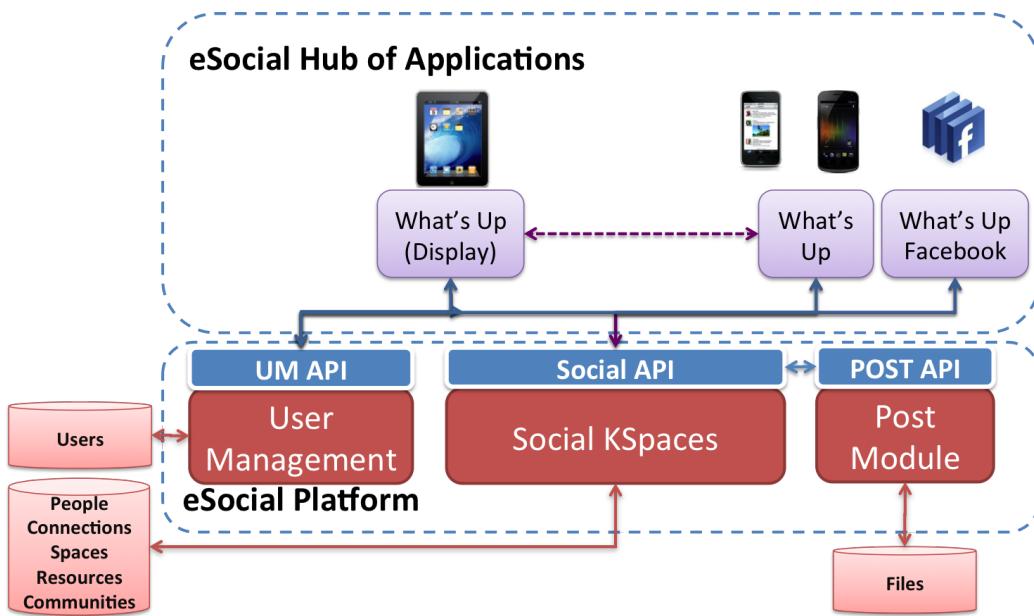


Figura 3.2: Architettura di WhatsUp.

giungerne una dalla sua galleria fotografica. Dopodichè premendo il tasto "Invia" (in alto a destra), invierà il contenuto. Il proprietario del KSpace riceverà e visualizzerà il contenuto. Nell'immagine viene utilizzata l'interfaccia completa, quindi il proprietario potrà scrivere un commento, col tasto in basso raffigurante una tastiera, esprimere apprezzamento o dissenso, con i tasti verde e rosso collocati rispettivamente in alto a destra e in alto a sinistra dello schermo, oppure scorrere le altre risorse, con le frecce ai lati dello schermo.

Con tre clic (vedi figura 3.4) è possibile iniziare una sincronizzazione, col primo clic visualizzi tutti gli utenti di cui sei Contributor, selezionandone uno visualizzi maggiori dettagli sull'utente selezionato, e da lì è possibile premere il tasto "Sync" in alto a destra. La richiesta viene mandata e l'applicazione si mette in attesa di una risposta.

La controparte visualizzerà un messaggio che notifica una richiesta di sincronizzazione (figura 3.5). Declinandola si chiude, e l'applicazione torna alla sua normale esecuzione. Accettandola parte la sincronizzazione, a questo punto l'unica opzione possibile è interromperla.

Questa funzione, implementata in WhatsUp, permette all'utente giovane di esporre a distanza i contenuti che lui ha inserito nel suo KSpace. Inoltre, grazie a questa funzione, il giovane può controllare la risorsa visualizzata dall'anziano, passando a suo piacimento alla successiva, o tornando alla



Figura 3.3: Funzionamento di WhatsUp, una volta scelto il destinatario premendo "Invia" il contenuto verrà immediatamente condiviso.



Figura 3.4: Inizio di una sincronizzazione

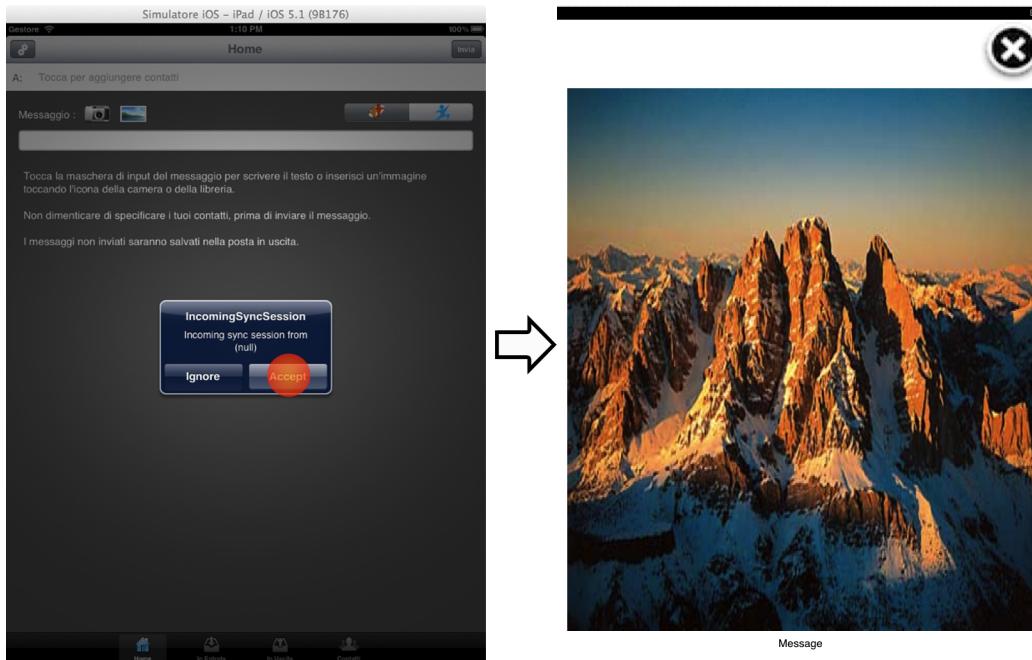


Figura 3.5: Ricezione di una richiesta

precedente.

Per implementare questa funzione è necessario tenere a mente i requisiti che questa applicazione deve avere. Per rimanere coerente con la filosofia dell'applicazione, la nuova funzione deve essere intuitiva, e di facile utilizzo, in particolare la parte che deve ricevere la richiesta di sincronizzazione ed essere quindi "controllata", perché destinata alla controparte più anziana.

Questa funzione necessiterà di un modo di ricevere dei messaggi in tempo reale, cosicché quando il controllore decide di passare alla prossima risorsa, allora il controllato passi automaticamente e in meno tempo possibile alla stessa risorsa. Dopo aver trovato un metodo per mandare i messaggi dovrà essere progettato un protocollo, in questo modo quando l'applicazione riceverà uno specifico messaggio, verrà eseguita una specifica funzione.

In seguito verrà spiegata la motivazione che ha portato alla creazione di questa funzione, il suo modello di utilizzo, e com'è stata implementata attraverso l'utilizzo delle notifiche push e di un semplice protocollo sviluppato.

3.1 Motivazione e modello di utilizzo

In certe occasioni è possibile che la visualizzazione dei contenuti senza altre informazioni non sia sufficiente: l'anziano, vedendo una fotografia, potrebbe avere qualche domanda a riguardo. Potrebbe chiederlo utilizzando un commento, ma abbiamo visto che questo non è disponibile in tutte le interfacce. Si potrebbe utilizzare un collegamento telefonico, però potrebbe risultare complicato capire quale contenuto sta visualizzando attualmente l'anziano.

Si è pensato quindi di creare uno slideshow guidato: il giovane, con l'aiuto di un collegamento telefonico, contatta l'anziano e avvia una sessione di slideshow guidato, selezionando i contenuti che vuole mostrare. A questo punto, l'anziano, riceverà una richiesta di avvio della sessione, che potrà accettare o rifiutare. Accettandola comparirà una schermata che visualizzerà il primo contenuto scelto dal giovane, e da qui in poi l'utente giovane potrà decidere di andare alla risorsa successiva, a quella precedente o terminare la sessione. Ogni volta che il giovane passerà al prossimo contenuto la schermata dell'anziano si aggiornerà visualizzando lo stesso contenuto. L'anziano ha solo la possibilità di terminare la sessione.

In questo modo il giovane può dare informazioni in più su ogni contenuto, per esempio raccontando un evento che è accaduto legato ad un'immagine, oppure focalizzando l'attenzione dell'anziano in un determinato punto della foto.

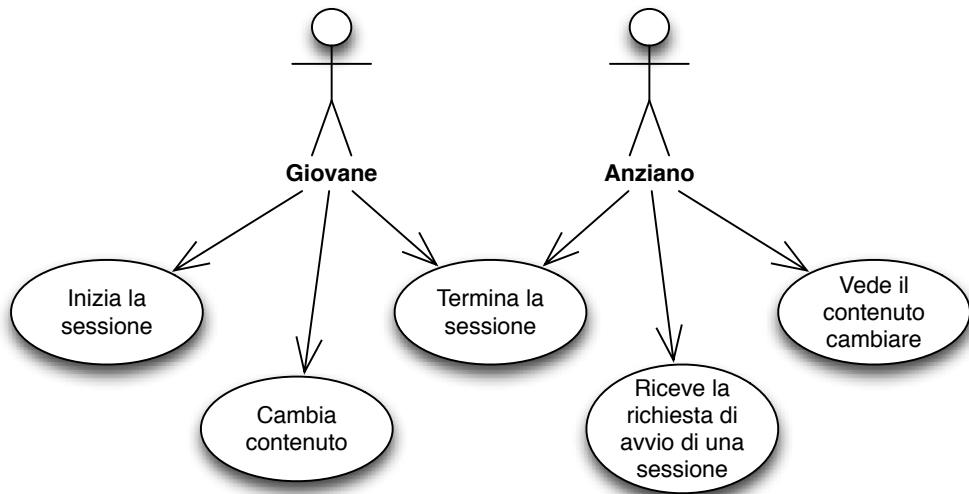


Figura 3.6: Diagramma degli Use Case del modulo Sync

3.2 Le notifiche push

Per mettere in comunicazione due dispositivi è necessario uno scambio di messaggi. In questo caso è necessario che il messaggio arrivi a destinazione in tempi molto brevi, altrimenti l'esperienza di utilizzo ne risentirebbe visto che non si potrebbe più parlare di sincronizzazione. L'idea di eseguire il polling con un intervallo brevissimo è impraticabile, perché ne risentirebbe la durata della batteria, vista la quantità delle eventuali richieste.

L'ideale sarebbe che il dispositivo rimanga in attesa di eventuali messaggi e che compia determinate azioni soltanto al ricevimento di un messaggio. In questo l'architettura di iOS e di Android viene incontro, con l'implementazione delle notifiche push con dei servizi chiamati rispettivamente "Apple Push Notification service" (APNs) e "Google Cloud Messaging" (GCM).

Per entrambi i servizi il funzionamento è analogo: quando l'applicazione viene avviata ci si registra al servizio con un identificatore univoco, quando qualcuno vuole mandare una notifica push contatta il servizio utilizzato questo identificatore e inserendo il contenuto del messaggio.

Vedremo ora nel dettaglio il funzionamento dell'APNs, come è stato implementato un substrato per mandare richieste all'APNs e al GCM tramite l'utilizzo di un server, e come viene contattato questo server con l'utilizzo delle API di WhatsUp.

3.2.1 APNs

Apple mette a disposizione due server per mandare le notifiche push, uno dedicato allo sviluppo e uno dedicato a quello che sarà il prodotto finale. Per accedere ad essi è necessario procurarsi gli opportuni certificati con i quali effettuare le richieste di invio. Quando un'applicazione che implementa le notifiche push viene avviata essa si mette in contatto con APNs e gli viene assegnato un deviceToken, una stringa alfanumerica di 64 caratteri che verrà utilizzata per inviare notifiche a quel dispositivo.

Il contenuto del messaggio è una stringa JSON che non deve superare i 256 bytes ed è strutturato così:

```

1   {
2     "aps" :{
3       "alert" : "You have a notification",
4       "badge" : 0,
5       "sound" : 0
6     },
7     "sync" : "sync protocol command",
8   }

```

- Un messaggio che verrà visualizzato a schermo se l'applicazione è chiusa (riga 3)
- Un numero che comparirà sull'icona dell'applicazione (riga 4)
- Un suono che verrà riprodotto dal dispositivo (riga 5)

Questi contenuti vengono messi nel tag "aps" del JSON inviato, è anche possibile prevedere dei tag personalizzati dove mettere informazioni che potranno essere utilizzate dall'applicazione. Nel nostro caso abbiamo previsto un tag "sync" che contiene i comandi del nostro protocollo (riga 7).

3.2.2 Il server Uniqush

Uniqush è un software libero ed open source di cui ci avvaliamo per unificare i servizi per le notifiche push. Attualmente supporta APNs e GCM, ma nel futuro supporterà anche altri servizi.

Per prima cosa bisogna creare dei servizi, in cui vengono specificate le credenziali per raggiungere i server APNs e GCM. Per creare un servizio che raggiunga APNs bisognerà specificare il percorso nel file system in cui sono collocati i certificati prelevati nel portale Apple dedicato agli sviluppatori. Per creare un servizio che raggiunga GCM bisogna specificare l'id del progetto Android e la chiave dell'API, prelevate dai portali di Google.

Dopodichè, i device, una volta avviata l'applicazione, si sottoscrivono a questi servizi, indicando un nome arbitrario e il proprio deviceToken nel caso di APNs, o il proprio account Google nel caso di GCM.

Per inviare una notifica push ad un dispositivo, infine, basta effettuare una richiesta a Uniqush indicando il nome scelto dal dispositivo, il nome del servizio a cui esso si è sottoscritto e il messaggio che si vuole inviare. Inoltre, è possibile indicare altri parametri utilizzati solo da iOS, come il messaggio a schermo, il numero sull'icona dell'applicazione e il suono da riprodurre, oppure dei parametri arbitrari che possono essere utilizzati a discrezione dello sviluppatore.

Per usufruire di tutto il potenziale di Uniqush, è molto importante creare i servizi, per APNs e GCM, aventi lo stesso nome, in questo modo possiamo mandare una notifica ad un dispositivo senza nemmeno preoccuparci se esso sia un utente iOS oppure un utente Android, basterà solo specificare il nome del servizio e il nome con cui il dispositivo che si vuole raggiungere si è sottoscritto.

Le richieste a Uniqush vengono effettuate tramite una richiesta di tipo POST specificando come PathParam l'operazione che si intende svolgere, e come QueryParam i parametri associati ad ogni operazione.

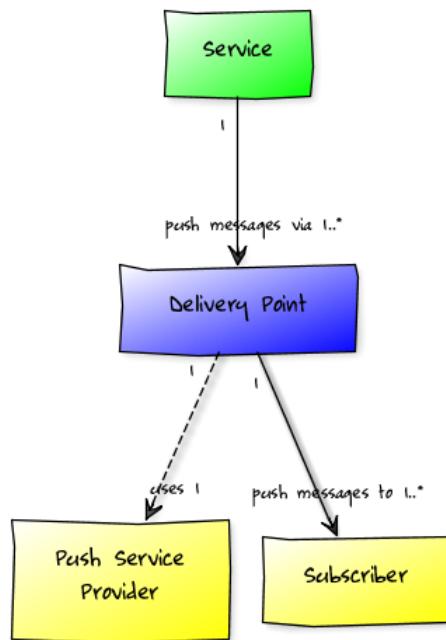


Figura 3.7: Architettura del server Uniquush

3.2.3 Uniquush nelle Social API

In questo sistema, per accedere alle funzionalità di Uniquush, bisogna passare necessariamente per le Social API di WhatsUp, questo per assicurare che possa mandare richieste solamente un utente che ha regolarmente effettuato l'accesso tramite WhatsUp. Infatti in ogni richiesta deve essere presente un authToken ottenuto dopo il login a WhatsUp.

Ogni richiesta deve essere una stringa JSON con gli opportuni elementi.

Creazione e rimozione dei servizi

Si può creare e rimuovere i servizi tramite delle richieste di tipo POST o DELETE formattate nel seguente modo:

```

1  {
2      "service" : ""          //Il servizio da creare
3      "type" : ""           //Il tipo del servizio(apns o gcm)
4      "apikey" : ""          //apikey (per gcm)
5      "projectid" : ""        //id del progetto (per gcm)
6      "cert" : ""            //percorso del certificato Apple
                           //nel filesystem del server (per apns)
7      "key" : ""             //percorso della chiave privata
                           //Apple nel filesystem del server (per apns)
  
```

```

8     "sandbox" : ""      //true o false a seconda del
9         server APNs che si vuole usare, quello per lo
          sviluppo o quello per la distribuzione (per apns
          )
9 }
```

Sottoscrizione ai servizi

Si può creare e rimuovere la sottoscrizione ai servizi tramite delle richieste di tipo POST o DELETE formattate nel seguente modo:

```

1  {
2      "service" : ""      //Il servizio a cui ci si vuole
          sottoscrivere
3      "type" : ""       //Il tipo del servizio (apns o gcm)
4      "account" : ""    //Account di Google (per gcm)
5      "regid" : ""       //Il regid ottenuto dopo l'
          autenticazione di Google (per gcm)
6      "devtoken" : ""    //Il deviceToken (per apns)
7 }
```

Non è specificato il nome con cui ci si vuole sottoscrivere in quando viene utilizzato l'userId di WhatsUp

Richiesta di invio di notifiche push

Per inviare una notifica push bisogna effettuare una richiesta di tipo POST formattata nel seguente modo:

```

1  {
2      "service" : ""      //Il servizio che si vuole
          utilizzare
3      "to" : ""           //L'userId di WhatsUp del
          destinatario
4      "command" : ""      //Il comando del protocollo che
          si vuole utilizzare
5      "params" : { "p1" : "v1", "p2" : "v2"} //I
          parametri che si vogliono inviare
6 }
```

I campi command e params sono strettamente legati al protocollo di sincronizzazione, argomento che vedremo nella prossima sezione

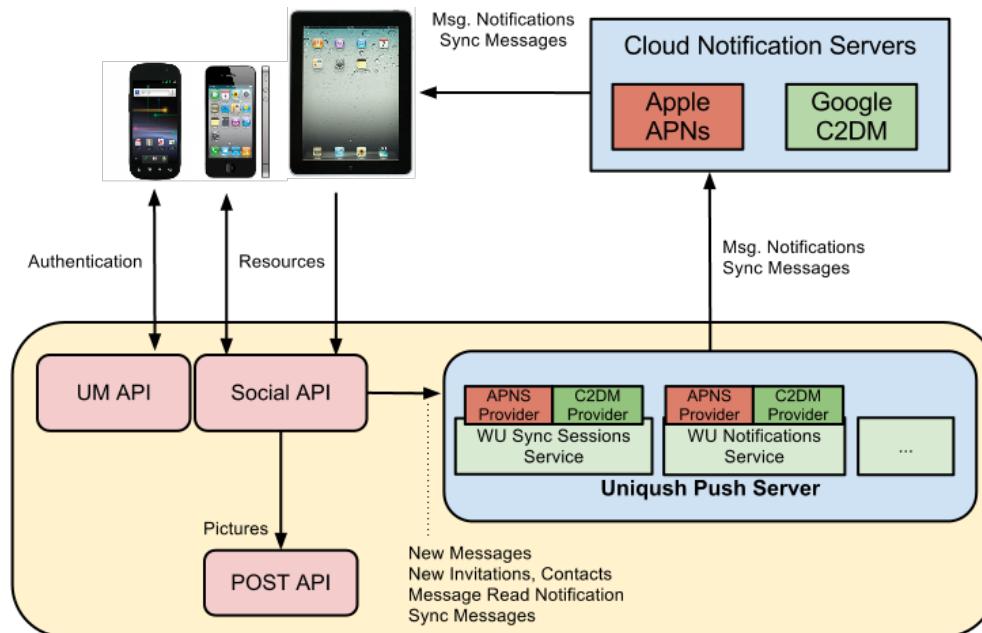


Figura 3.8: Architettura delle notifiche push

3.3 Protocollo di sincronizzazione

In questo protocollo ogni richiesta deve essere effettuata attraverso le API di WhatsUp. La struttura generale di una richiesta l'abbiamo vista nella sezione 3.2.3.

Quello che non abbiamo ancora visto è la struttura di un messaggio quando viene ricevuto da un device, i messaggi ricevuti sono formattati nel seguente modo:

```

1  {
2      "command" : ""      //Il comando del protocollo che
3          ci è stato inviato
4      "fromId": ""        //L'id del mittente
5      "params" : { "p1" : "v1", "p2" : "v2"}  //I
          parametri annessi al comando
5  }

```

Vedremo ora nel dettaglio gli stati e i comandi previsti da questo protocollo in base alle azioni che si intendono svolgere.

3.3.1 Avvio di una sessione

1. Quando la connessione è inattiva lo stato è NOT_CONNECTED.

2. Da questo stato è possibile avviare una sessione mandando il comando START_SESSION ad un destinatario specificato nel campo "to" (vedi 3.2.3). Appena mandata la richiesta di avvio di una sessione, lo stato passa immediatamente a WAIT_FOR_SESSION_ACK.
3. Quando il messaggio arriva a destinazione, il destinatario, se disponibile e accetta la richiesta, passa allo stato READY e invia il comando START_SESSION_ACK al mittente
4. Se il mittente riceve il comando START_SESSION_ACK entro 15 secondi passa allo stato READY anche lui, e la sessione è correttamente avviata, altrimenti torna allo stato NOT_CONNECTED

3.3.2 Visualizzazione di una risorsa

1. Se lo stato è READY il dispositivo che ha avviato la sessione può mandare il comando DISPLAY, specificando tra i parametri un campo "resId" che indica l'id di una risorsa di WhatsUp, lo stato passa a WAIT_FOR_DISPLAY_ACK
2. Quando il destinatario riceve il comando DISPLAY controlla l'id della risorsa, se quella risorsa è già presente in locale allora invia il comando DISPLAYING altrimenti invia il comando DISPLAY_ACK, contatta le API di WhatsUP e scarica la risorsa. A download completato invia DISPLAYING.
3. Quando viene ricevuto il comando DISPLAY_ACK il device che ha avviato la sessione passa allo stato DISPLAY_ACK RECEIVED, in attesa del comando DISPLAYING.
4. Una volta ricevuto il comando DISPLAYING il device torna allo stato READY, pronto per inviare un altro comando.
5. Se non viene ricevuto ne un DISPLAY_ACK, ne un DISPLAYING entro 15 secondi la sessione si chiude, e il dispositivo passa allo stato NOT_CONNECTED

3.3.3 Fine di una sessione

1. In qualunque momento, qualsiasi dispositivo può mandare il comando END_SESSION all'altro, e lo stato passa a NOT_CONNECTED

2. Quando il comando END_SESSION anche lo stato del destinatario passa a NOT_CONNECTED

Capitolo 4

Conclusioni

4.1 Limiti dell'implementazione

L'attuale implementazione incorre in alcuni limiti, che derivano perlopiù dall'utilizzo di APNs. Esso, infatti, limita le notifiche che un dispositivo può ricevere in un certo lasso di tempo. Quindi, mandando troppi messaggi in rapida successione, potrebbe succedere che APNs si blocchi temporaneamente, smettendo di inviare notifiche, e inviandole tutte insieme alla rinfusa in un momento impreciso. Nella documentazione ufficiale di Apple, tuttavia, non è specificato quali siano questi limiti [7]. Bisogna quindi cercare di inviare le notifiche cercando di non inviarne troppe in un lasso di tempo troppo breve, altrimenti la funzionalità di sincronizzazione potrebbe subire un momentaneo "black-out".

Una possibile soluzione a questo inconveniente può essere, per esempio, l'utilizzo di APNs solo per "svegliare" il device e notificare la ricezione di una richiesta di sincronizzazione. Accettata la richiesta, si potrebbe utilizzare un altro metodo non soggetto a limitazioni per l'invio di notifiche push, per esempio MQTT, che vedremo in seguito.

Un altro problema derivato da APNs è che non viene garantito che le richieste di invio di notifiche vengano evase in ordine. Quindi, se mando due richieste in rapida successione, è possibile che esse arrivino disordinatamente. Nel nostro protocollo, l'unico caso interessato è quello specificato nel punto 2 del paragrafo 3.3.2. Il caso è il seguente: se quando viene ricevuto il comando DISPLAY la risorsa non è presente in locale viene mandato in risposta il comando DISPLAY_ACK, viene recuperata la risorsa e in seguito inviato il comando DISPLAYING. Se la fase di recupero avviene molto velocemente i due comandi, DISPLAY_ACK e DISPLAYING ,vengono mandati a distanza

molto ravvicinata, ed APNs potrebbe invertirne l'ordine.

Per questo motivo il comando DISPLAYING viene accettato sia nello stato WAIT_FOR_DISPLAY_ACK, sia nello stato DISPLAY_ACK RECEIVED. In questo modo, se si verificasse il caso precedente, verrebbe ricevuto il messaggio DISPLAYING e lo stato tornerebbe a READY, dopodiché verrebbe ricevuto il comando DISPLAY_ACK, che nello stato READY viene ignorato.

L'ultimo limite è rappresentato dalla dimensione massima che una notifica push può avere: 256 bytes. Considerando che lo scheletro di una notifica push, ottenuto utilizzando la struttura presente nel paragrafo 3.2.1 riempendo il campo "sync" con il JSON presente nel paragrafo 3.3, notiamo che la dimensione è di circa 75 bytes, quindi restano altri 181 bytes per il contenuto del messaggio. Il comando più lungo che si può mandare è START_SESSION_ACK (17 bytes), avanzano quindi 164 bytes da utilizzare nei campi "fromid" e "params". Ogni utente è rappresentato da un "id", un numero in formato decimale. Anche se venisse raggiunto il miliardo di utenti (Facebook ne conta poco meno di 900,000,000[8]), verrebbero occupati 10 bytes, quindi ci sarebbero 154 bytes soltanto per il campo params (che tutt'ora prevede solo l'id di una risorsa)

Tutt'ora, quindi, il sistema può funzionare tranquillamente, quando il campo "params" sarà occupato con altri parametri si può considerare di rappresentare i numeri, relativi all'id di un utente e di una risorsa, in formato esadecimale, per risparmiare qualche byte.

4.2 Implementazioni iniziali

Inizialmente per implementare le notifiche push abbiamo usato MQTT, un message protocol opensource col funzionamento simile al server Uniquush. Una volta connessi a un piccolo message broker ci si sottoscriveva ad un topic, noi utilizzavamo l'id dell'utente. Quindi, per mandare un messaggio a qualcuno, bisognava usare l'id del destinatario come topic.

Il sistema funzionava correttamente, tuttavia era presente un piccolo problema architettonale. Infatti ogni dispositivo collegato apriva e manteneva aperta una connessione TCP, e col crescere del numero degli utenti c'era la possibilità che il server non riuscisse a gestire tutto il carico di lavoro. Inoltre sarebbe risultato incoerente con il resto dell'architettura, che è tutta basata su richieste HTTP.

Un altro problema di questa implementazione riguarda esclusivamente la versione per iOS. Infatti, mentre per Android è possibile programmare il proprio servizio in background che mantenga aperta la connessione TCP

anche quando l'applicazione non è in foreground, per iOS questo è possibile solo per determinati servizi, come il VOIP o la riproduzione di musica[9]. Quindi, i messaggi venivano ricevuti soltanto quando l'applicazione era in primo piano, e non era quindi possibile, per esempio, ricevere una richiesta di sincronizzazione mentre il dispositivo si trova in fase di "sleep" o mentre si stava svolgendo un'altra attività col dispositivo.

All'inizio, nel tag "sync" (vedi 3.2.1), il messaggio non era formattato come una stringa JSON ma era semplicemente una stringa di testo non formattata. Quindi per avviare una sessione, per esempio, bisognava semplicemente mettere nel tag "sync" la stringa "START_SESSION {fromId: *toId*, *resId*}".

Questo, oltre ad essere incoerente con il resto delle API (che forniscono sempre delle risposte in JSON), creava molto più disordine a livello di codice, poiché il parsing risultava era più scomodo. Con JSON invece, è possibile avvalersi di apposite librerie per fare il parsing. Qui è stata utilizzata la libreria SBJSON, in sintesi prende in input una stringa JSON e ne ricava una hash table, oppure ricava una stringa JSON da una hash table.

L'utilizzo di JSON inoltre facilita l'eventuale espansione del protocollo, poiché per aggiungere un nuovo elemento, per esempio un nuovo valore nel campo "params" (3.3), è sufficiente modificare solo una classe (che rappresenta il messaggio).

4.3 Possibili sviluppi futuri

Il lavoro svolto è solo la parte iniziale di questo progetto. Ad esso potrebbero essere aggiunte altre nuove funzioni. Vediamo ora degli esempi.

Una funzione potrebbe indicare in che punto dello schermo un utente ha cliccato. Se, per esempio, mentre si sta visualizzando un'immagine, il controllato (che attualmente non può interagire in nessun modo se non chiudendo la sessione) avesse una domanda su un punto specifico della foto, gli basterebbe toccare questo punto. In questo caso, il controllore, visualizzerebbe sul suo schermo un indicatore sul punto toccato.

Questa funzione potrebbe usarla anche il controllore però: se, per esempio, volesse focalizzare l'attenzione del controllato su un punto specifico dell'immagine, anche a lui basterebbe toccare il punto e il controllato visualizzerebbe il suddetto indicatore.

Analogamente, durante la sessione, il controllore potrebbe anche aggiungere un’etichetta al punto toccato. Supponendo che durante la sessione vengano visualizzate le foto di una vacanza, in questo caso, il controllore, potrebbe, per esempio, mettere un’etichetta su un monumento, ed essa verrebbe visualizzato anche sullo schermo del controllato. Non è da escludere nemmeno l’ipotesi di mantenere questa etichetta anche alla fine della sessione, ma questo esula dal protocollo implementato, perché richiederebbe dei cambiamenti alle API di WhatsUp per tenere memorizzate le etichette.

Un’altra possibilità è quella di permettere, in qualche modo, di dare la possibilità al controllato di bloccare l’esposizione. Premendo per esempio un bottone, viene fermata l’esposizione, e il controllore non ha più la possibilità di cambiare immagine finché il controllore non preme di nuovo il bottone. Questo può venire utile nei casi in cui il controllore età andando troppo veloce nell’esposizione delle immagini, allora il controllato lo ferma e gli chiede le spiegazione che gli servono.

Un’importante funzionalità, sarebbe quella di sfruttare Skype per effettuare le chiamate durante la sessione. Attualmente la chiamata deve essere effettuata o tramite telefono, oppure avviando manualmente Skype e la chiamata. La soluzione ideale sarebbe quella di integrare le funzionalità di Skype nell’applicazione. Sfortunatamente Skype non mette a disposizione delle librerie per effettuare l’integrazione, quindi un’integrazione completa non è possibile. È invece possibile (avendo Skype installato sul dispositivo), avviare l’applicazione da un’altra e avviare la conversazione. Non è la soluzione migliore perché una volta avviata la chiamata, bisognerebbe ritornare manualmente a WhatsUp Sync, e ciò si scontrerebbe con l’intuitività e la facilità di utilizzo a cui mira il progetto.

Non è da escludere l’ipotesi di utilizzare un altro protocollo VoIP, da integrare completamente con l’applicazione, in questo modo avviando la sessione si potrebbe avviare contemporaneamente la chiamata, e terminarla alla fine della sessione, svolgendo diversi passaggi in meno, e rendendo l’uso di questa funzionalità drasticamente più facile e intuitivo.

In sintesi, il lavoro svolto è si deve considerare come un punto di partenza dal quale è possibile aggiungere le più svariate funzioni, e ha posto una solida base per continuare a lavorarci e a renderlo migliore.

4.4 Esperienza accumulata

La partecipazione a questo progetto è stata un'ottima opportunità di mettere in pratica le cose imparate nel corso degli studi e di impararne di nuove.

Lo scoglio più grande da superare è stato apprendere l'utilizzo del framework CocoaTouch, quello utilizzato per sviluppare applicazioni per iOS, e quindi anche dell'Objective-C. Grazie all'aiuto ricevuto dal team, e alle conoscenze pregresse, relative perlopiù al framework Android, questo ostacolo è stato superato e il lavoro ha proceduto molto più velocemente.

Si è rivelato molto utile anche per comprendere al meglio le peculiarità della programmazione per sistemi mobili, poiché bisogna programmare considerando che le risorse sono limitate e fare le cose in modo tale che lavorino nel modo più efficiente possibile, anche per conservare la batteria.

Infine è stata un'ottima esperienza anche per l'aver imparato il lavoro all'interno di un team, la progettazione dell'applicazione, la pianificazione del lavoro, la divisione dei compiti e il confronto. Inoltre è stata un'importante opportunità anche per parlare la lingua inglese, poiché è stata la lingua maggiormente parlata durante i meeting.

Bibliografia

- [1] <http://www.idc.com/getdoc.jsp?containerId=prUS23228211>
- [2] Alwan, M., Wiley, D., & Nobel, J. (2007). State of Technology in Aging Services. *Aging*, (November).
- [3] Carroll, J. M., Convertino, G., Farooq, U., & Rosson, M. B. (2011). The firekeepers: aging considered as a resource. *Universal Access in the Information Society*.
- [4] Park, D.C., Gutchess, A.H., Meade, M.L., & Stine-Morrow, E.A.L. (2007). Improving cognitive function in older adults: Nontraditional approaches. *Journal of Gerontology: Psychological Sciences*, 62B(Special Issue I), 45–52
- [5] <http://developer.apple.com/library/mac/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/ApplePushService/ApplePushService.html>
- [6] <http://uniquush.org>
- [7] <http://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/WhatAreRemoteNotif/WhatAreRemoteNotif.html>
- [8] <http://www.socialbakers.com/countries/continents>
- [9] <http://developer.apple.com/library/ios/#DOCUMENTATION/iPhone/Conceptual/iPhoneOSProgrammingGuide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html>