Università degli Studi di Padova

Dipartimento di
Ingegneria dell'Informazione

Corso di Laurea in Ingegneria Informatica

Una applicazione mobile multipiattaforma per la consultazione di albi pretori

Laureando

Francesco Boschetto

Relatore

Prof. Carlo Fantozzi

Anno Accademico 2012/2013

A mio papà per avermi sempre sostenuto ed incoraggiato,
a mamma che mi è sicuramente stata vicino nei momenti di difficoltà,
a Silvia per la pazienza e la comprensione
e a tutti quelli che hanno reso possibile la conclusione di questo percorso.

Grazie.

Francesco Boschetto

Sommario

L'oggetto di questa relazione è lo sviluppo di un'applicazione per piattaforme Android e iOS, svolto durante il tirocinio presso l'azienda Euro Servizi Srl. L'applicazione permette la consultazione dell'albo pretorio del Comune di Chiampo, con la possibilità di navigare e ricercare tra le pratiche in corso di pubblicazione, visualizzarne i dettagli e i documenti allegati. Nel primo capitolo viene descritta l'azienda e si introduce il portale al quale l'applicazione si interfaccia. Nel secondo capitolo vengono illustrate le piattaforme e gli strumenti utilizzati per sviluppare l'applicazione. Nel terzo capitolo viene fatta una panoramica sugli aspetti e requisiti comuni che dovrà avere l'applicazione nelle rispettive piattaforme. Nel quarto capitolo vengono approfonditi i dettagli dell'applicazione sviluppata per la piattaforma Android, e nel quinto infine quelli della piattaforma iOS.

Indice

Capitolo 1

Introduzione

1.1 Euro Servizi Srl

Euro Servizi Srl nasce nel 2001 dall'esperienza maturata dai suoi componenti fin dal 1994 nell'ambito della pubblica amministrazione locale. Le attività prevalenti della società sono rivolte ai servizi a corredo del software applicativo: conversioni banche dati, formazione tecnica ed applicativa, supporto all'avviamento, affiancamento in supporto alla normale attività dell'Ente.

1.1.1 Consulenza

La consulenza agli enti locali è stata la prima attività dell'azienda e grazie alla notevole esperienza dei consulenti, il personale è in grado di offrire consulenza a 360 gradi ai clienti, accompagnandoli nelle trasformazioni e nelle innovazioni tecnologiche. Le competenze riguardano i seguenti campi:

- Sistemistico (hardware e software di base): dalla configurazione di server e postazioni, al disegno di soluzioni globali, inclusi cablaggio e la telefonia.

- Funzionale (specifiche attività di analisi): grazie alla notevole esperienza, le diverse figure presenti sono in grado di analizzare tutti i principali processi dell'Ente Locale.

- Normativo: possibilità di erogare con competenza servizi di consulenza su molti ambiti tipici della PAL, ad esempio privacy, sicurezza, tributi, gestione economica e finanziaria, controllo di gestione e Carta d'Identità Elettronica (CIE). Caratteristica fondamentale dell' approccio del personale è il fatto di mantenere sempre una visione unitaria dell'Ente.

- Web/E-government: l'azienda possiede specifiche competenze oltre che sul sistema a portale facente parte della suite di prodotti software, anche sui seguenti progetti e-governament: People, From-Ci (Comune di Vicenza), Sirv-Interop (Regione Veneto).

1.1.2 Sviluppo Software

Nel corso del tempo l'attività aziendale si è spostata dalla consulenza verso lo sviluppo software, specializzandosi in soluzioni per gli enti rivolte alla cittadinanza: i portali dei servizi. Tutte le soluzioni a portale rispettano le indicazioni delle normative specifiche (legge Stanca, etc.) ma soprattutto sono pensate per risultare semplici e intuitive ai loro utenti, ovvero i cittadini. Queste soluzioni erogano servizi su più livelli.

- Livello pubblico: comprende servizi tipicamente informativi normalmente associati alla trasparenza dell'azione amministrativa degli enti.
- Livello registrato (per tutti previa registrazione): include servizi ancora informativi come, ad esempio, la newsletter ma anche per presentare istanze o segnalazioni.
- Livello confermato (per utenti registrati e confermati dall'Ente) : servizi con un elevato livello di interattività, come consultazioni demografiche o dello stato di avanzamento delle proprie pratiche.
- Livello autorizzato (per utenti esplicitamente autorizzati dall'ente al singolo servizio): per gestioni particolari come le posizioni tributarie.
- Livello dedicato: servizi visibili solo a categorie particolari di utenti, ad esempio servizi dedicati ad amministratori comunali.

A fianco dello sviluppo di soluzioni per i portali dei cittadini, pur privilegiando lo sviluppo in ambiente open-source, l'azienda realizza applicativi in ambiente Microsoft DotNet, Microsoft Access, PHP, Ruby, Python e Java.

1.2 Portale del cittadino

I portali dei servizi al cittadino realizzati da Euro Sevizi sono pensati in modo personalizzato per ciascun cliente, ed offrono un numero tale di componenti da coprire la totalità delle esigenze in questo contesto. Essi sono sviluppati rispettando la Legge Stanca e più in generale considerando la problematica

dell'accessibilità.

Figura 1.1: Portale del cittadino, Comune di Chiampo.

La suite dei servizi per la Pubblica Amministrazione Locale è così composta:

- Content Managment System (CMS);
- Portale dei Servizi e repository utenti;
- Newsletter;
- Albo Pretorio;
- Demografia;
- Interrogazione Fornitori;
- Consultazione Atti Formali, ad esempio decreti e delibere;
- Consultazione Stato avanzamento pratiche;
- Presentazione pratiche via web;

- Segnalazioni OnLine, ad esempio segnalazione di guasti di impianti comunali;
- Tributi OnLine;
- Portale per amministratori (ordini del giorno online);
- Agende di servizio per prenotazioni appuntamenti da portale;

L'applicazione da sviluppare nelle due versioni, Android e iOS, deve permettere all'utente la consultazione degli atti pubblicati nell' albo pretorio.
L'albo pretorio è il luogo in cui vengono pubblicati gli atti comunali per i quali è obbligatoria la pubblicità legale.
Per poter sviluppare l'applicazione il personale dell'azienda ha dovuto aggiungere nuove funzioni al proprio server in modo da permettere la comunicazione tra l'applicazione e il portale.
Nei prossimi capitoli vengono illustrati gli strumenti utilizzati e i requisiti dell'applicazione. Successivamente, nei capitoli 4 e 5, vengono esposti i dettagli implementativi, rispettivamente in Android e iOS.

Capitolo 2

Piattaforme e strumenti utilizzati

2.1 Android

Android è una piattaforma open-source per dispositivi mobili costituita da un sistema operativo basato su una versione modificata del kernel Linux, da un insieme di librerie native, dall'Android Runtime, da un Application Framework e da un set di applicazioni fondamentali.
Lo sviluppo di applicazioni Android avviene in linguaggio Java utilizzando delle librerie custom apposite, non conformi a Java SE o ME. Il codice sorgente viene compilato in bytecode Java e poi nel formato proprietario DEX. I file DEX sono indipendenti dalla piattaforma e vengono eseguiti in una macchina virtuale proprietaria, chiamata Dalvik. Il formato DEX e la macchina virtuale Dalvik sono state progettate in modo da essere ottimizate per sistemi limitati in termini di memoria e velocità di calcolo.
Per porzioni di codice critiche in termini di performance è possibile sviluppare parti di applicazioni utilizzando i linguaggi nativi di Android, C e C++, e le librerie libc (C library), libm (math library), libz (ZLib compression library) e liblog. Questo è permesso dal Native Development Kit (NDK), che si rivela molto utile nel caso di operazioni che fanno uso intensivo della CPU ma occupano poca memoria come, ad esempio, l'elaborazione dei segnali e simulazioni fisiche.

2.1.1 SDK

Il Software Development Kit (SDK) di Android fornisce gli strumenti di sviluppo necessari per sviluppare e testare applicazioni Android: include un debugger, librerie API, un insieme di emulatori basati su QEMU, documen-

tazione, codici sorgenti di esempio e tutorial. Attualmente le piattaforme supportate per lo sviluppo sono calcolatori con sistema operativo Linux, Mac OS X 10.5.8 o successivo, Windows XP o successivo. L'IDE (Integrated Development Environment) supportato ufficialmente è Eclipse, nel quale bisogna installare un plugin aggiuntivo, l'ADT plugin, per integrare l'SDK.

2.1.2 Versioni

L'attuale distribuzione delle varie versioni di Android è illustrata nel seguente grafico, basato sui dati raccolti dai dispositivi che hanno effettato l'accesso a Google Play (vedi Figura 2.1). Questa informazione risulta molto utile a chi intende sviluppare applicazioni Android: grazie ad essa si può dare priorità alle specifiche dell'applicazione in base ai dati sui dispositivi attualmente in mano agli utenti.

Version	Codename	API	Distribution
1.5	Cupcake	3	0.1%
1.6	Donut	4	0.3%
2.1	Eclair	7	2.7%
2.2	Froyo	8	10.3%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	50.6%
3.1	Honeycomb	12	0.4%
3.2		13	1.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	27.5%
4.1	Jelly Bean	16	5.9%
4.2		17	0.8%

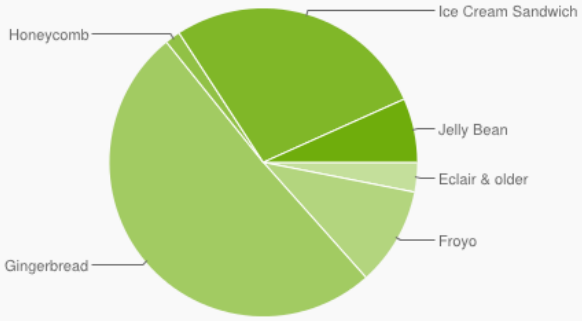

Figura 2.1: Distribuzione versioni Android, dati ufficiali Dicembre 2012 [1].

In particolare è importante l'API level, che è un valore intero che identifica univocamente la versione dell'API Framework offerta da una data versione della piattaforma Android. Le applicazioni usano l'API Framework per interagire con il sistema operativo. Esso consiste in:

- Un set base di package e classi,

- Un insieme di elementi e attributi XML per dichiarare il file Manifest, all'interno del quale sono elencate le informazioni e le componenti dell'applicazione e i permessi necessari al corretto funzionamento della stessa,
- Un insieme di elementi e attributi XML per dichiarare e accedere alle risorse,
- Un set di Intent, oggetti tramite i quali si descrivono delle azioni specifiche che devono essere eseguite dal dispositivo come, ad esempio, telefonare, inviare un sms etc.

2.2 iOS

iOS è una piattaforma sviluppata e distribuita da Apple Inc. per i propri dispositivi mobili iPhone, iPod Touch e iPad. La piattaforma è costituita da un sistema operativo derivato da UNIX per il quale Apple non fornisce la licenza per installazioni in dispositivi non proprietari. iOS è formato dai seguenti quattro livelli di astrazione.

- **Core OS layer** che comprende il kernel, i driver, il file system, gestione della rete e del risparmio energetico.
- **Core Service layer** che comprende i servizi base e servizi per la gestione delle specifice harware.
- **Media layer** che comprende la gestione della grafica bidimensionale e tridimensionale e gestione audio/video.
- **Cocoa Touch** che è un Framework ad alto livello per lo sviluppo delle applicazioni.

2.2.1 SDK

L'iOS Software Development Kit contiene gli strumenti e le interfacce necessarie per sviluppare, installare e testare applicazioni che appariranno nella schermata principale del dispositivo. Esse vengono scritte in linguaggio Objective C. Lo strumento principale per poter sviluppare applicazioni iOS è XCode,una suite che contiene l'IDE e una serie di strumenti che permettono di gestire progetti, scrivere ed editare codice sorgente, creare gli eseguibili e testare le applicazioni. A differenza dell'SDK Android, in cui è presente un emulatore, Apple fornisce un simulatore che è quindi più limitato. XCode è disponibile esclusivamente su calcolatori Apple con Mac OS X.

2.3 Elaborazione della grafica

2.3.1 Adobe Photoshop CS5

Photoshop è un software proprietario, prodotto da Adobe Systems Incorporated, per l'elaborazione di immagini digitali e fotoritocco.
Nello sviluppo e progettazione dell'applicazione, nelle due piattaforme, Photoshop è stato utilizzato per la personalizzazione della grafica in particolare dello sfondo, delle barre del titolo e navigazione, dei pulsanti, loghi applicazione e degli screenshot.

Capitolo 3

Requisiti e specifiche dell'applicazione

L'albo pretorio è il luogo dove vengono esposte le deliberazioni, le ordinanze, i manifesti, gli avvisi e gli atti che devono essere portati a conoscenza di tutta la cittadinanza oppure gli atti destinati ai cittadini che, per assenza o irreperibilità, non sono stati loro consegnati.
L'applicazione Albo Pretorio utilizza i dati del portale del cittadino prodotto dall'azienda per i Comuni. Il progetto è pensato in modo che ci sia una applicazione per ogni Comune che ne faccia richiesta, perciò è stata sviluppata in modo che possano esserne pubblicate più versioni, una per ogni Comune. La prima versione dell'applicazione, nonché l'applicazione modello, è stata pubblicata per il Comune di Chiampo. Nel momento in cui un'altro Comune, cliente dell'azienda, decida di renderla disponibile, ne verrà pubblicata una versione che andrà a collegarsi al portale di quel comune, modificando solo dei parametri e se necessario la grafica.

3.1 Caratteristiche e funzionalità

L'applicazione in questione assume che il dispositivo sia connesso ad internet e deve avere le seguenti funzionalità e requisiti.

- Una schermata principale che permetta di consultare l'elenco delle pratiche in corso di pubblicazione e, attraverso un campo di testo, permetta di filtrare istantaneamente le pratiche che hanno in oggetto o nel tipo dell'atto il testo inserito.

- Una schermata di dettaglio che appare una volta selezionato un elemento dell'elenco principale e che contenga l'oggetto del'atto, il tipo,

la data di affissione, quella di scadenza, l'ente interessato e l'elenco degli eventuali allegati.

- La possibilità di visualizzare o scaricare gli allegati di un'affissione, se presenti.
- La possibilità di essere avvisati tramite notifiche push della pubblicazione di una nuova affissione.

3.2 Comunicazione con il server

Per comunicare con il portale vengono effettuate chiamate in formato JSON. Per permettere tale comunicazione, il personale dell'azienda che gestisce il portale Web ha dovuto integrare il server con l'aggiunta di apposite funzioni.

3.2.1 I file JSON

JSON (JavaScript Object Notation) è un formato leggero di file per l'interscambio di dati, facile per gli umani da leggere e scrivere e allo stesso tempo facile per i calcolatori da generare e analizzare. Il formato è basato su un sottoinsieme del linguaggio JavaScript. JSON è ideale per lo scambio di dati poiché, anche se è un formato testuale completamente indipendente da altri linguaggi di programmazione, usa convenzioni che sono familiari ai programmatori in linguaggi della famiglia del C come C,C++,C#, Java, JavaScript, Perl, Pyton e molti altri. JSON è basato su due strutture:

- L'**oggetto**, che rappresenta un insieme di coppie chiave/valore.Esso inizia con (parentesi graffa sinistra) e finisce con (parentesi graffa destra). Ogni nome è seguito da : (due punti) e la coppia di nome/valore sono separata da , (virgola).
- L'**array**, che rappresenta una raccolta ordinata di valori.Esso comincia con [(parentesi quadra sinistra) e finisce con] (parentesi quadra destra). I valori sono separati da , (virgola).

Un valore può essere una stringa tra virgolette, o un numero, un booleano vero o falso o nullo, oppure un oggetto stesso o un'array.

3.2.2 Esempio di file JSON gestito dall'applicazione

Di seguito viene mostrato un esempio di utilizzo del file JSON per la visualizzazione del dettaglio di un'affissione all'albo.

```
{"affissione": {
                "obj_modified":"2012-11-08T12:53:11+00:00",
                "cr_date":"2012-11-08T12:53:11+00:00",
                "ente":"COMUNE DI VICENZA",
                "id_annuale":"2012001127",
                "numero_protocollo":"20656",
                "tipo_atto": "ordinanze",
                "codice_pratica_collegata:"0",
                "numero_atto":null,
                "anno":"2012",
                "data_affissione":"2012-11-08",
                "data_scadenza":"2013-03-23",
                "oggetto":"PROVVEDIMENTI TEMPORANEI DI
                    LIMITAZIONE DELLA CIRCOLAZIONE STRADALE
                    PER LA CIRCOLAZIONE STRADALE PER LA
                    PREVENZIONE E RIDUZIONE DEI LIVELLI DI
                    CONCENTRAZION DEGLI INQUINANTI NELL
                    ATMOSFERA URBANA CITTA DI VICENZA",
                "id":3603,
               },
 "allegati": [
               {
                   "file":"allegato1352375472.pdf",
                   "descrizione":"Allegato",
                   "url":"3603/allegati/allegato1352375472.pdf"
               }
              ]
}
```


Listing 3.1: Esempio file JSON descrizione affissione.

3.3 Gli allegati

Ogni affissione presente nell'albo pretorio può avere uno o più allegati. Gli allegati possono essere di vari formati quali file Adobe Pdf, file di Microsoft Word (.doc) nonché immagini. L'estensione e quindi il formato degli allegati viene indicato nel file JSON dell'affisione assieme all'url dell'allegato.

3.4 Notifiche Push

Gli utenti che utilizzano l'applicazione devono poter essere aggiornati sulle nuove affissioni tramite notifiche push. Le notifiche push sono brevi messaggi testuali che appaiono sul display del dispositivo per informare l'utente di

novità, messaggi e avvisi, permettendo, selezionandoli, di aprire l'applicazione interessata. Per ricevere notifiche, il dispositivo deve essere connesso ad internet tramite Wi-Fi o rete 3G.

Capitolo 4

Sviluppo Android

Figura 4.1: Le due schermate principali dell'applicazione Android.

L'applicazione Android è composta da due activity[1] principali. Una è la MainActivity che è la prima schermata dell'applicazione e permette la visualizzazione, la navigazione e la ricerca nell'elenco degli atti dell'albo. L'altra è la DetailActivity che è la schermata che permette di visualizzare i dettagli della pratica selezionata e di scaricarne gli allegati.
Dopo una accurata valutazione, tenendo conto della distribuzione delle versioni di Android attive al momento dello sviluppo, è stato scelto di progettare l'applicazione in modo da essere compatibile con le versioni di precedenti di Android a partire dalla versione Froyo 2.2 (API Level 8).

4.1 Gestione dell'interfaccia grafica

Il linguaggio che viene utilizzato per la definizione dell'interfaccia grafica delle applicazioni Android è l' XML, con il quale vengono definiti gli oggetti grafici e le relative proprietà. L'ambiente di sviluppo permette di costruire l'interfaccia utente attraverso un layout grafico in modo da limitare la scrittura manuale di codice XML.

4.1.1 Supporto delle risoluzioni

Il sistema operativo Android è installato in una grande varietà di dispositivi che hanno differenti risoluzioni. L'ambiente di sviluppo fornisce quindi gli strumenti per costruire l'interfaccia utente in modo tale da poter essere visualizzata correttamente su tutti i dispositivi. I concetti che vengono utilizzati sono quelli di *dimensione* dello schermo e *densità* dello schermo. La prima è la dimensione fisica del display del dispositivo, che viene misurata considerando la diagonale dello schermo, mentre la seconda è la quantità di pixel all'interno di un area fisica dello schermo, indicata come dpi (dots per inch), ovvero punti per pollice. La piattaforma generalizza le dimensioni del display in quattro gruppi: *small, normal, large, xlarge* e le densità in: *ldpi (low), mdpi (medium), hdpi(high), xhdpi(extra high)* (vedi Figura 4.2).
Inoltre, viene introdotta l'unità di misura detta *Density-indipendent pixel* (dp), un unità di misura di pixel virtuale che si usa per definire il layout dell'interfaccia utente in modo tale da esprimere le dimensioni e la posizione dei componenti in modo indipendente dalla densità. Sarà il sistema operativo a scalare i dp e le grafiche in base alla densità corretta. Il ridimensionamento bitmap può però causare immagini sfuocate o con effetto pixel. Per evitare

[1] Un' activity è una componente di una applicazione Android che fornisce una schermata con un' interfaccia utente.

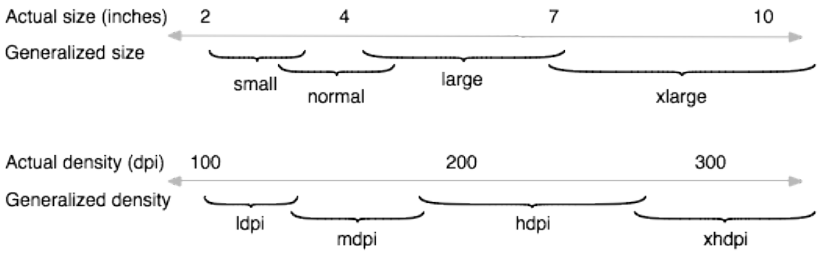

Figura 4.2: Divisione in gruppi in base a dimensione e densità.

ciò è necessario creare più risorse grafiche in base ai quattro gruppi di densità. Nel caso, ad esempio, di immagini personalizzate bisogna creare, per ognuna, quattro versioni differenti. L'immagine di base è quella destinata al gruppo mdpi (medium density), le altre tre versioni dovranno essere con una diversa scala: 0.75x per ldpi (low density), 1.5x per hdpi (high density) e 2.0x per xhdpi. Una volta inserite nel progetto all'interno della categoria di densità corretta, sarà il sistema operativo ad utilizzare automaticamente le immagini adatte in base al dispositivo sul quale è installata l'applicazione.

4.2 Gestione delle connessioni al server

Per la comunicazione con il server, sia per le chiamate a file JSON, sia per il download degli allegati, viene utilizzata la classe astratta AsyncTask fornita dalle API Android. Questa classe permette di eseguire operazioni in background e visualizzarne i risultati nell'interfaccia grafica principale evitando di gestire i thread manualmente. Un processo asincrono è definito come una operazione che viene eseguita in un thread separato in background in modo tale che una volta conclusa vengano pubblicati i risultati nel thread dell'interfaccia grafica. Per poter fare ciò bisogna creare una classe che estenda la classe astratta AsyncTask e che ne implementi i metodi. La classe AsyncTask usa i tipi generici (generics) di Java. I generics permettono di definire delle astrazioni sui tipi di dati definiti nel linguaggio.
AsyncTask è definita da 3 generics, *AsyncTask<Params, Progress, Result>* che hanno il seguente significato.

Params è il tipo dei parametri che vengono passati al task al momento dell'esecuzione.

Progress è il tipo dell'unità di misura che indica il progresso dell'operazione.

Result è il tipo del risultato della operazione in background.

I metodi principali di AsyncTask che devono essere implementati sono i seguenti:

onPreExecute() , invocato nel thread principale dell'interfaccia grafica prima che l'operazione sia eseguita, che può servire a visualizzare una progress bar o a settare parametri che servono all'operazione da eseguire.

doInBackground(Params...) , invocato nel thread separato in background subito dopo onPreExecute(), nel quale viene eseguita l'operazione vera e propria.Gli eventuali parametri vengono passati direttamente a questo metodo e alla sua conclusione esso deve ritornare il risultato dell'operazione. All'interno di questo metodo si può eventualmente usare publishProgress(Progress...) per pubblicare una o più unità di progresso.

onProgressUpdate(Progress...) , invocato nel thread di interfaccia grafica dopo un eventuale esecuzione del metodo publishProgress(Progress...), che può essere usato per aggiornare eventuali label o progress bar in modo di informare l'utente sullo stato dell'operazione.

onPostExecute(Result) , invocato nel thread di interfaccia grafica dopo la conclusione dell'operazione,nel quale si elabora il risultato che viene restituito da doInBackround(Params...) e passato al metodo.

Perché possa funzionare correttamente bisogna che l'istanza della classe che estende AsyncTask sia creata nel thread di interfaccia grafica. Per far partire l'operazione bisogna chiamare il metodo execute(Params...) dell'istanza sempre nell'thread UI senza invocare manualmente i quattro metodi illustrati sopra.
Nell'applicazione, il codice che esegue la connessione vera e propria al server viene quindi eseguito nel thread separato mostrandone, nel caso del download degli allegati, il progresso mentre, nel caso di chiamate a JSON, solo una finestra di caricamento in corso senza indicazioni sul progresso.

4.3 Gestione JSON: la libreria Gson di Google

Google-gson è una libreria Java che permette di convertire un oggetto Java nell'equivalente rappresentazione JSON e, viceversa, convertire una stringa JSON nell'equivalente oggetto Java. Per utilizzarla bisogna scaricarla dal sito code.google.com e importarla nel progetto Android.
Nell'applicazione sono state create delle classi modello che rispecchiano i contenuti dei file JSON che vengono scaricati dal server, cioè classi che abbiano

gli attributi con lo stesso nome delle chiavi del JSON e i metodi getter e setter. In questo modo, per convertire la stringa JSON in un oggetto basta creare un'istanza della classe Gson e chiamarne il metodo fromJson(). Così facendo si può accedere efficentemente ai dati ricevuti manipolando oggetti Java. Di seguito viene illustrato un esempio di come viene utilizzata la libreria Gson nell'applicazione. La classe Affissione.java rappresenta il modello di un affissione all'albo (vedi Listing 4.1).

```
public class Affissione{
    private String anno;
    private String codice_pratica_collegata;
    private String data_affissione;
    private String data_atto;
    private String data_protocollo;
    private String data_scadenza;
    private String ente;
    private String giorni_affissione;
    private String id;
    private String oggetto;

    ...
    ...

    public String getAnno(){
      return this.anno;
    }

    public void setAnno(String anno){
      this.anno = anno;
    }

    public String getCodice_pratica_collegata(){
      return this.codice_pratica_collegata;
    }


    public void setCodice_pratica_collegata(String codice_pra...
      this.codice_pratica_collegata = codice_pratica_collegata;
    }

    ...
    ...

}
```

Listing 4.1: Frammento di codice della classe Affissione.java.

```
...
  Gson gson = new Gson();
  Affissione affissione;
  affissione = gson.fromJson(stringaJSON, Affissione.class);
...
```

Listing 4.2: Esempio di conversione da JSON all'oggetto affissione.

4.4 Gestione degli allegati

Per quanto riguarda il download e la visualizzazione degli allegati è stato deciso in fase di sviluppo di far gestire all'applicazione il download dei file non affidandosi al browser web di Android. Questa decisione è stata presa perché prima di iniziare il download si vuol verificare se nel dispositivo è presente un'applicazione in grado di consentire la visualizzazione dell'allegato. Dopo aver verificato che ci sia un applicazione in grado di aprire il file, si comincia il download vero e proprio. I file vengono scaricati nella cartella pubblica di download all'interno della memoria esterna del dispositivo e restano salvati fino alla chiusura dell'applicazione in modo che, se l'utente seleziona un allegato già scaricato, non ci sia la necessità di riscaricare il file. Nel caso non ci sia alcuna applicazione installata per aprire il file viene visualizzato un *toast*, cioè un piccolo pop-up che resta sul display per pochi secondi, in modo da informare l'utente che non è possibile aprire l'allegato e non viene scaricato nulla (vedi Figura 4.3).
Per effettuare il controllo e per aprire eventualmente il file una volta scaricato vengono usati gli Intent. Un Intent è una descrizione astratta di un'azione specifica eseguibile dal dispositivo, come ad esempio telefonare, inviare un sms, aprire un'activity e così via. Inoltre, per verificare la presenza un'applicazione in grado di visualizzare l'allegato viene utilizzata la classe PackageManager che fornisce informazioni sulle applicazioni installate nel dispositivo. La funzione che ritorna un valore booleano, vero nel caso in cui il controllo vada a buon fine, falso in caso contrario, è illustrata nel frammento di codice seguente (vedi Listing 4.3).

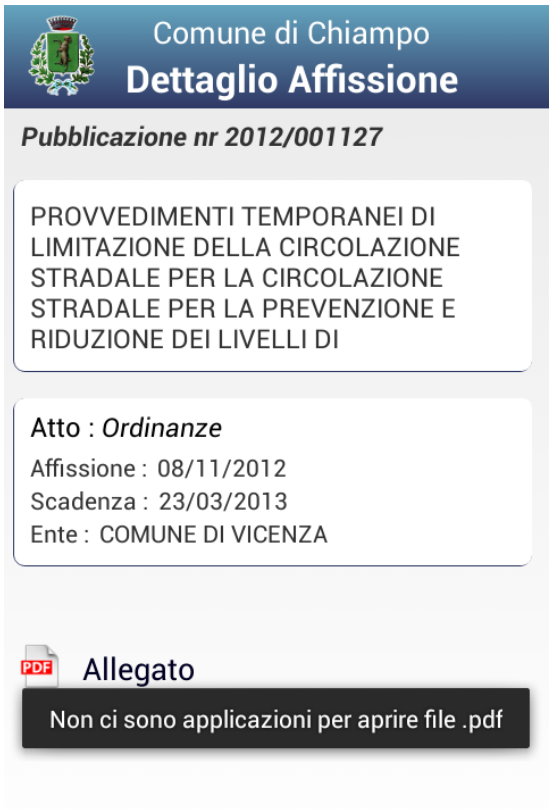

Figura 4.3: Nessuna applicazione presente per aprire file pdf.

```
public boolean intentFound(String estensioneFile){

  Intent testIntent = new Intent(Intent.ACTION_VIEW);
  testIntent.setDataAndType(null, "application/"+estensioneFile);
  PackageManager pm = getPackageManager();
  ResolveInfo info;
  info = pm.resolveActivity(testIntent,Intent.FLAG...);
  if(info != null)
    return true;
  return false;

}
```

Listing 4.3: Verifica della presenza di una applicazione per aprire un file.

Se il controllo dà esito positivo inizia il download del file; in caso di spazio insufficente viene interrotta l'operazione e viene avvisato l'utente con un messaggio *toast*. Il file viene scaricato in una directory pubblica, nella memoria esterna, accessibile a tutte le applicazioni. Questo è possibile utilizzando la funzione *getApplicationContext().getExternalFilesDir().getPath()* che restituisce il percorso della directory pubblica.
Il codice che esegue il download dell'allegato è inserito nel metodo doInBackround() di AsyncTask (Vedi paragrafo 4.2). L'utente è informato sullo stato del download attraverso un pop-up con una barra di progresso e l'indicazione dello stato dell'operazione in percentuale. Durante il download non è possibile navigare tra le altre pratiche ma si deve aspettare il termine del download oppure è possibile interrompere l'operazione con il tasto BACK del dispositivo (vedi Figura 4.4).
Una volta completato il download dell'allegato, viene lanciato un Intent per aprirlo con l'applicazione corretta alla quale viene passato il percorso del file. Nel caso siano presenti più applicazioni adibite allo scopo il sistema operativo notifica l'utente chiedendo con cosa aprire il file selezionato (vedi Figura 4.5).

```
...

  Intent fileIntent = new Intent( Intent.ACTION_VIEW );

  fileIntent.setDataAndType( filePath ,
                             "application/"+estensioneFile );
  startActivity(fileIntent);

...
```

Listing 4.4: Lancio di un intent per l'aperura dell'allegato.

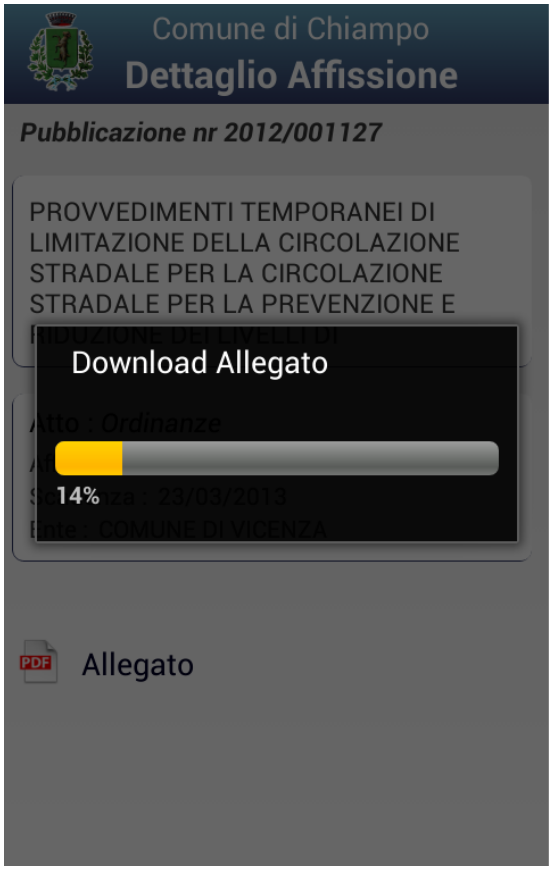

Figura 4.4: Pop-up che indica lo stato del download dell'allegato.

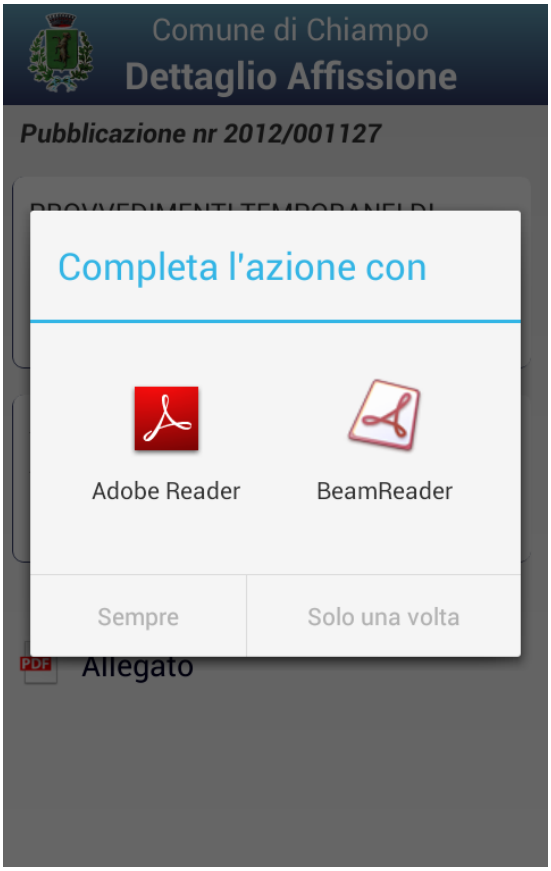

Figura 4.5: Più applicazioni in grado di aprire il tipo di file selezionato.

4.5 Notifiche Push: GCM

L'applicazione informa l'utente di nuove affissioni tramite notifiche push. L'utente può decidere se attivare o meno le notifiche utilizzando il menù dell'applicazione che appare premendo il pulsante menu del dispositivo nella schermata principale dell'applicazione. Per utilizzare le notifiche nell'applicazione è necessario utilizzare il servizio *Google Cloud Messaging* (GCM).
Google Cloud Messaging for Android è un servizio che permette agli sviluppatori di inviare dati dai propri server alle loro applicazioni installate sui dispositivi degli utenti. I dati possono essere piccoli messaggi di testo oppure messaggi contenenti fino a 4kb di payload. Il servizio GCM gestisce l'accodamento e il salvataggio dei messaggi e l'invio di essi all'applicazione Android installata sui dispositivi riceventi. Tale servizio è completamente gratuito e non comporta alcun costo anche a chi ne fa un grande utilizzo.
Per ricevere le notifiche non occorre che l'applicazione in questione sia in esecuzione: all'arrivo del messaggio sarà il sistema operativo che informerà l'applicazione, a patto che l'applicazione abbia il permesso di ricevere notifiche. Il servizio non fornisce alcuna interfaccia grafica e neppure metodi particolari per gestire i messaggi: esso inoltra semplicemente un messaggio testuale ricevuto dal server di terze parti e lo spedisce all'applicazione Android. Sarà compito dell'applicazione gestire il messaggio all'arrivo, con la possibilità di mostrare sul display un messaggio, oppure semplicemente sincronizzando dati senza informare l'utente.
Nei dispositivi in cui sono installate versioni Android inferiori alla 3.0 il servizio GCM richiede che su di essi sia configurato l'account utente Google.

4.5.1 Architettura

Le entità fisiche coinvolte nel processo sono le seguenti.

Dispositivi Android: dispositivi in cui è installata l'applicazione Android che usa GCM. Devono essere dispositivi con almeno Android 2.2, che hanno installato Google Play e con account Google configurato.

Server Applicazione: il server di terze parti che lo sviluppatore ha configurato per la creazione e l'invio dei messaggi.

Server GCM: il server Google che fa da tramite inviando i messaggi ricevuti dal server applicazione ai dispositivi.

Le credenziali, cioè i vari ID e token che vengono usati nelle fasi del processo sono le seguenti.

Sender ID: l'identificativo del progetto. Viene fornito quando lo sviluppatore crea il progetto nella Google API Console, per poter utilizzare il servizio GCM.

Application ID: l'identificativo dell'applicazione che si registra per ricevere i messaggi. L'applicazione Android è identificata dal package name, in modo che il messaggio sia indirizzato all'applicazione corretta.

Registration ID: l'identificativo che viene restituito da GCM all'applicazione Android che fa richiesta di registrazione. Identifica una particolare applicazione su un particolare dispositivo.

Account Google: account utente Google che deve essere configurato nel dispositivo per far funzionare il servizio (per sistemi Android precedenti alla versione 3.0).

Sender Auth Token: una chiave che va salvata nel server di terze parti, così che sia autorizzato a inviare messaggi attraverso GCM.

La prima fase del processo è la registrazione, che l'applicazione esegue la prima volta che viene eseguita. In questa fase l'applicazione invia una richiesta, detta Registration Intent, al GCM Server. Essa contiene il Sender ID e l'Application ID. Se quest'operazione va a buon fine il GCM server invia un broadcast Intent, che assegna e ritorna all'applicazione il Registration ID. Tale Registration ID viene inviato dall'applicazione al server applicazione di terze parti, che lo memorizzerà in modo da avere l'identificativo di tutte le applicazioni così che esse poi possano ricevere messaggi (vedi Figura 4.6).
La seconda fase è l'invio dei messaggi. Quando il server di terze parti ha un messaggio da inviare alle applicazioni, lo invia al GCM Server, il quale nel caso il dispositivo sia offline accoda i messaggi che verranno poi inviati non appena esso torni online. Quando il dispositivo riceve il messaggio, il sistema operativo invia un Broadcast Intent all'applicazione. In questo modo esso manda in esecuzione l'applicazione così che riceva il messaggio e lo processi (vedi Figura 4.7).
L'ultima fase, la disattivazione del servizio per una applicazione, non avviene al momento esatto della disinstallazione di essa dal dispositivo. Il server Google tenta l'invio di un messaggio all'applicazione: nel caso essa sia stata disinstallata il sistema operativo notifica il server GCM, che memorizzerà il dispositivo come disabilitato.

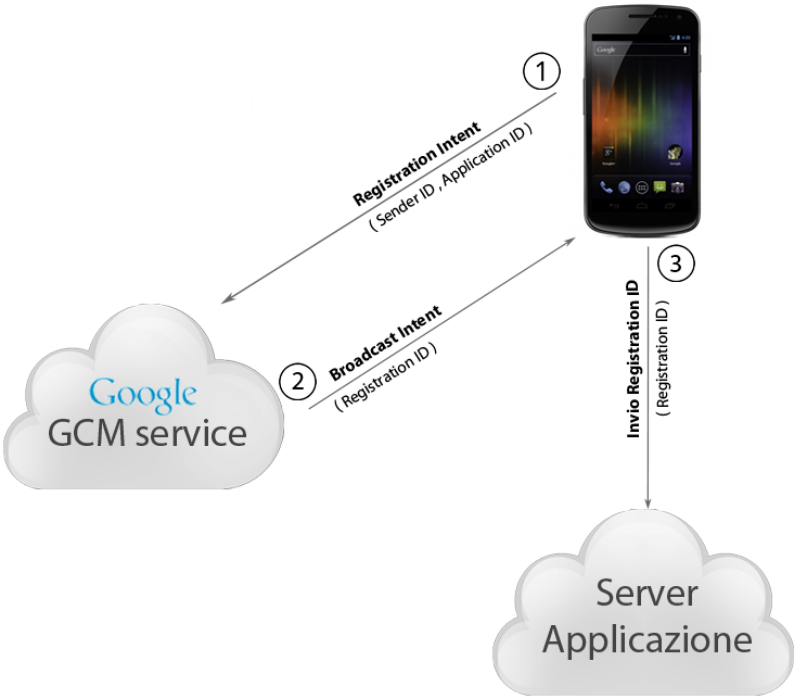

Figura 4.6: La fase di registrazione al CGM.

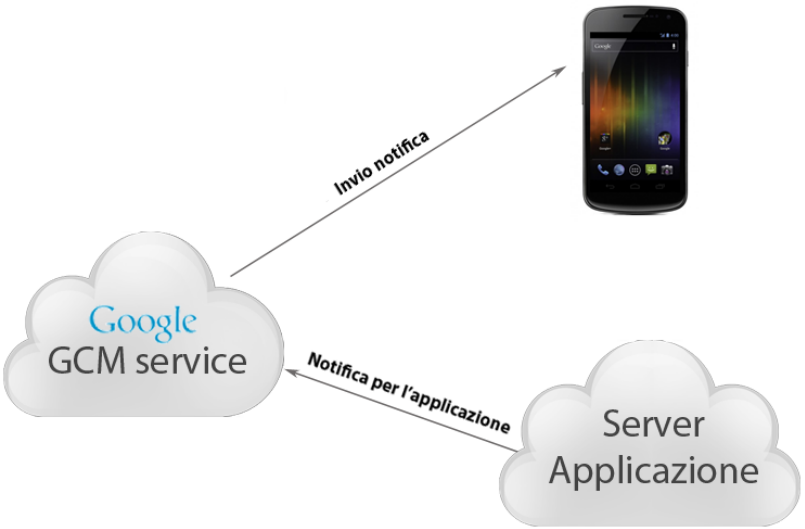

Figura 4.7: Invio di una notifica all'applicazione.

Il personale di Euro Servizi ha dovuto inserire delle funzioni aggiuntive all'interno del portale Web in modo che, nel momento in cui venga pubblicata una nuova affissione, il server invii la notifica al server GCM, il quale provvederà a recapitarla all'applicazione.

Capitolo 5

Sviluppo iOS

Figura 5.1: Prima schermata dell'applicazione iOS.

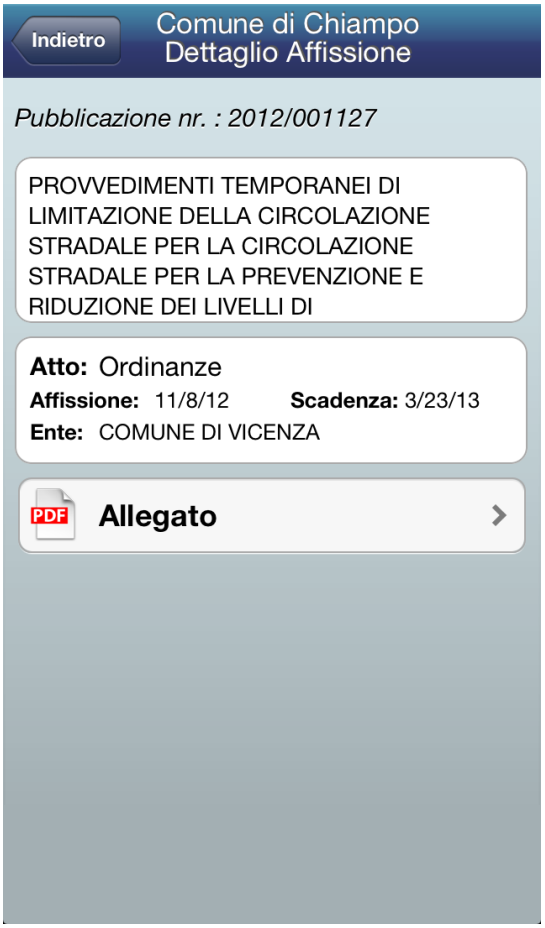

Figura 5.2: Schermata di dettaglio di una pratica.

L'applicazione iOS è formata da tre schermate. La prima è la schermata principale, adibita alla visualizzazione delle pratiche pubblicate nell'albo pretorio e alla ricerca tra di esse attraverso un campo di testo. La seconda è la schermata di dettaglio nella quale vengono visualizzati i dettagli della pratica selezionata e la lista degli eventuali allegati. Selezionando un allegato si passa infine alla terza schermata che scarica e visualizza l'allegato all'interno dell'applicazione.

5.1 Gestione dell'interfaccia grafica

5.1.1 Storyboard

Per la costruzione e la gestione dell'interfaccia grafica è stato scelto lo strumento fornito con l'avvento di iOS 5 e XCode , lo Storyboard. A differenza di Interface builder con il quale si gestisce l'interfaccia grafica attraverso i file xib, lo Storyboard permette di avere un piano di lavoro in cui poter gestire tutte le varie schermate e i passaggi tra esse. Il risultato del proprio lavoro viene memorizzato in un unico file (.storyboard). Inoltre, le transizioni tra le varie schermate e le proprietà degli oggetti grafici possono essere gestite quasi completamente in maniera grafica, riducendo notevolmente il codice da scrivere per applicazioni con più schermate. Come illustrato in Figura 5.3 il punto di partenza è indicato dalla prima freccia che indica che il primo componente è il NavigationController, un oggetto che gestisce la navigazione tra le varie schermate, dopo di che sono collegati i ViewController cioè gli oggetti che gestiscono le singole schermate.

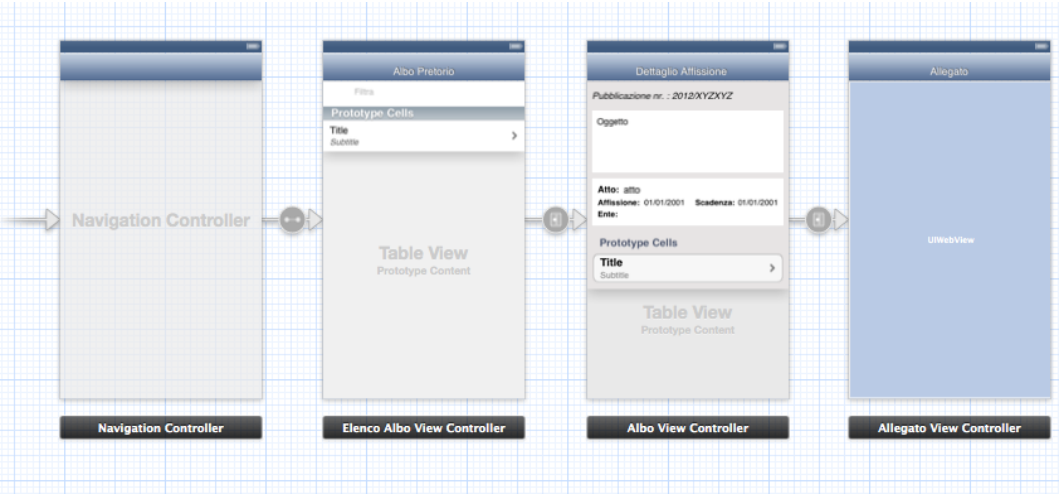

Figura 5.3: Utilizzo dello Storyboard.

5.1.2 Supporto delle risoluzioni

I dispositivi Apple a cui è destinata l'applicazione, dal punto di vista dell'interfaccia grafica, si dividono in due gruppi a seconda della tipologia di display. I dispositivi meno recenti hanno display con risoluzione di 480x320 pixel, mentre quelli più recenti sono dotati di Retina Display con una risoluzione di 960x640 pixel. In realtà l'iPhone 5 ha una risoluzione ancora maggiore, di 1136x640 pixel, ma come si vedrà in seguito si può considerare equivalente alla seconda categoria citata. Da questa situazione nasce l'esigenza di avere immagini e grafiche separate per le due categorie. Nella progettazione della grafica si sono quindi create due versioni per ogni immagine, una a risoluzione normale (in scala 1x1) e una a risoluzione maggiore (scala 2x2). A queste ultime immagini va dato lo stesso nome di quelle a risoluzione minore ma con il suffisso @2x così che automaticamente il sistema operativo usi le immagini adatte.
Per quanto riguarda la compatibilità con la risoluzione del nuovo iPhone 5, le immagini che verranno destinate ad esso automaticamente sono le immagini con suffisso @2x. Tutte a meno di un immagine, l'immagine di lancio, della quale devono essere presenti tre versioni, quella normale, quella @2x e quella destinata all'iPhone 5 che avrà nome Default-568h@2x. Perchè gli oggetti contenuti nelle View siano visualizzati correttamente, se non si hanno esigenze particolari, basta impostare correttamente il ridimensionamento automatico e l'ancoraggio degli oggetti in modo da utilizzare lo spazio aggiuntivo che si ottiene con lo schermo a 4 pollici.

5.2 Gestione delle connessioni al server

Per tutte le chiamate al server, tranne quelle per scaricare gli allegati, viene usato il framework AFNetworking [6].
AFNetworking è un framework open-source per iOS e MacOS X, sviluppato sulle basi delle classi di Foundation. Foundation è il framework che fornisce classi e funzioni primitive per Objective-C. AFNetworking, in particolare utilizza la classe NSOperation per lo scheduling e la concorrenza, e fa largo utilizzo della programmazione a blocchi[1] per convenienza e flessibilità.
Il cuore del framework AFNetworking è la classe *AFURLConnectionOperation*, sottoclasse di NSOperation, che ha il compito di eseguire chiamate di rete asincrone. Essa è la superclasse di *AFHTTPRequestOperation*, che è

[1]Un blocco (Block Code) è una porzione di codice racchiuso da parentesi graffe e preceduto dal carattere ^ , che ha lo scopo di dichiarare una funzione che verrà eseguita in un certo momento, non noto a priori, ad esempio allo scatenarsi di un evento.

specificatamente orientata verso la comunicazione tramite protocolli HTTP e HTTPS. *AFHTTPRequestOperation* crea una distinzione tra le richieste in base al successo o al fallimento di esse, determinati dal codice di stato HTTP e dalla ricezione di una risposta considerata accettabile. Dalla classe *AFHTTPRequestOperation* vengono derivate quattro classi importanti: *AFJSONRequestOperation*, utilizzabile nel caso di comunicazione tramite JSON; *AFXMLRequestOperation* utilizzabile se le risposte sono di tipo XML; *AFPropertyListRequestOperation* adibita alla ricezione di generici NSData; *AFImageRequestOperation* che può essere adoperata per la ricezione di immagini.
Si prenda come esempio la classe *AFJSONRequestOperation*, che fornisce il metodo *JSONRequestOperationWithRequest:success:failure*. Il primo parametro è di tipo *NSURLRequest* e contiene la request (URL, metodo e parametri) da passare al servizio. Il secondo parametro è il Block Code che verrà eseguito in caso di successo nella comunicazione, il terzo è il Block Code che verrà eseguito in caso di fallimento nella comunicazione.

```
...

NSURLRequest *request = [NSURLRequest
 requestWithURL:[NSURL URLWithString:@".../2219.json"]];

AFJSONRequestOperation *operation =
  [AFJSONRequestOperation JSONRequestOperationWithRequest:request
  success:
  ^(NSURLRequest *request, NSHTTPURLResponse *response, id JSON){

     NSLog(@Oggetto":%@", [JSON valueForKeyPath:@"oggetto"]);
     NSLog(@"TipoAtto:%@", [JSON valueForKeyPath:@"tipo_atto"]);
  }
  failure: nil ];

...
```

Listing 5.1: Esempio utilizzo AFJSONRequestOperation.

Nella porzione di codice precedente (vedi Listing 5.1) viene dato esempio dell'utilizzo di *JSONRequestOperation*: in tale codice viene eseguita una chiamata all' oggetto 2219.json (nel codice vanno sostituti i tre punti che lo precedono con il percorso corretto per connessione al server). Si può notare che all'interno del blocco *success* viene effettuato il log dei campi oggetto e tipo_atto dell'oggetto JSON cioè l'affissione con id 2219. Mentre il blocco *failure* è nil cioè, in caso di fallimento, non c'è un blocco di codice che viene eseguito.

Il framework fornisce inoltre la classe *AFHTTPClient*, la quale è utile, come nel caso dell'applicazione Albo Pretorio, se è necessario interagire continuamente con il server. Infatti questa classe incapsula informazioni come un URL base per le connessioni, eventuali credenziali di accesso e intestazioni HTTP, per usarle così in tutte le comunicazioni con il server.

5.3 Gestione degli allegati

Nella versione per dispositivi iOS, al contrario di quella per dispositivi Android, gli allegati non vengono scaricati ma vengono visualizzati direttamente all'interno dell'applicazione (vedi Figura 5.4).

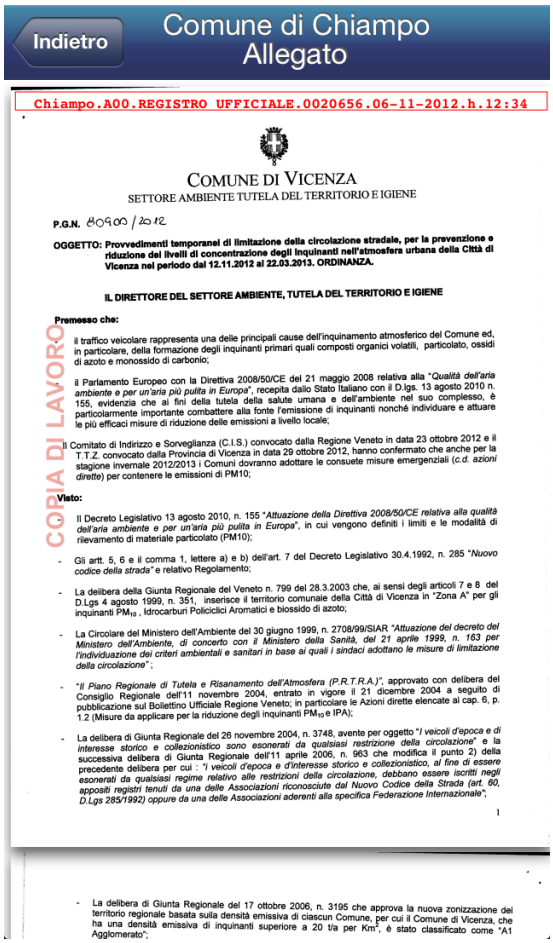

Figura 5.4: Visualizzazione di un allegato.

Questo è possibile attraverso una schermata contenente una WebView, un componente dell'interfaccia grafica che permette di visualizzare contenuti web e tutti i possibili tipi di file degli allegati. Questo è possibile passando la posizione dell'allegato (tramite URL) alla WebView attraverso il metodo seguente:

```
[webView loadRequest:[NSURLRequest requestWithURL:urlAllegato]];
```

5.4 Notifiche push: APNs

Il meccanismo delle notifiche Push in iOS si basa sull'infrastuttura Apple Push Notification service (APNs).
Attraverso l'APNs, ogni dispositivo iPhone, iPad o iPod Touch stabilisce una connessione IP cifrata con il servizio e attraverso essa riceve le notifiche. Se l'applicazione a cui è indirizzata la notifica non è attiva, il sistema operativo informerà l'applicazione dei dati in attesa.
Gli sviluppatori software creano le notifiche all'interno dei loro server, definiti Providers. Quando ci sono nuovi dati o messaggi da inviare all'applicazione installata in un dispositivo, il provider prepara e invia la notifica attraverso un canale sicuro e persistente con l'APNs, che provvederà a recapitarla al dispositivo stesso. L'APNs include un componente QoS (Quality of Service) che svolge una funzione di store-and-forward. Infatti, nel caso si tenti di inviare una notifica ad un dispositivo che è momentaneamente offline il servizio salva il messaggio, per inviarlo quando il dispositivo tornerà in linea, con una limitazione: esso può memorizzare soltanto una notifica per ogni applicazione su un dato dispositivo, che è la più recente. Inoltre, il servizio mantiene le notifiche per un periodo limitato di tempo, passato il quale il messaggio viene eliminato.
Per ricevere notifiche push, l'applicazione installata su un dispositivo deve registrarsi al servizio APNs, il quale genera un *device token* . Il *device token* è un identificativo analogo a un numero di telefono, contiene informazioni che permettono all' APNs di individuare il dispositivo in cui è installata l'applicazione. Il token generato viene restituito all'applicazione; questa lo invia al provider, che lo memorizza. Il metodo utilizzato dall'applicazione per registrarsi e ottentere il token è invocato all'interno del metodo delegato *didFinishLaunchingWithOptions* della classe AppDelegate del progetto. Il metodo è il seguente:

```
[[UIApplication sharedApplication]
  registerForRemoteNotificationTypes:
  (UIRemoteNotificationTypeAlert|UIRemoteNotificationTypeSound)];
```

E' poi necessario aggiungere nella classe AppDelegate due metodi delegati: uno che viene invocato se la richiesta va a buon fine, all'interno del quale va inserito il codice per l'invio del device token al provider (vedi Listing 5.2), l'altro per gestire il caso in cui non sia possibile effettuare la registrazione a causa di un errore (vedi Listing 5.3).

```
-(void)application:(UIApplication *)application
 didRegisterForRemoteNotificationsWithDeviceToken:
(NSData *)deviceToken
```

Listing 5.2: Metodo invocato se la richiesta va a buon fine.

```
-(void)application:(UIApplication *)application
 didFailToRegisterForRemoteNotificationsWithError:
(NSError *)error
```

Listing 5.3: Metodo invocato se si verifica un errore nella richiesta.

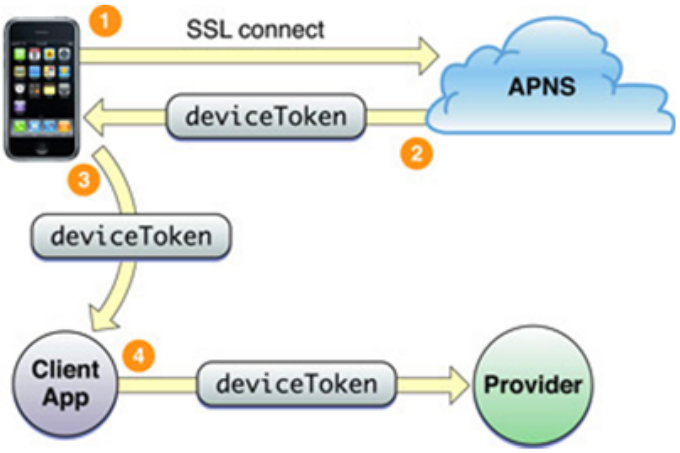

Figura 5.5: Registrazione dell'applicazione ad APNs [2].

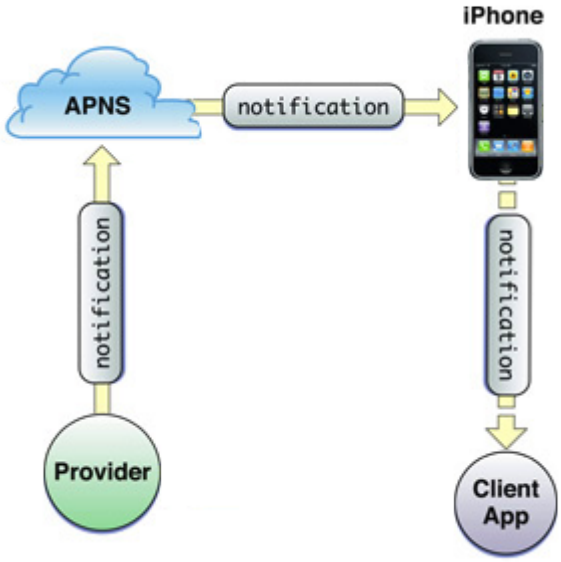

Figura 5.6: Invio di una notifica [2].

APNs è dotato anche di un servizio di feedback che serve nel caso in cui i tentativi di invio di notifiche verso un'applicazione in un determinato dispositivo vengano continuamente rifiutati. Questo accade spesso quando l'utente ha disinstallato l'applicazione. In questi casi il servizio informa il provider in modo tale da poter eliminare i relativi token.
A differenza della versione Android dell'Albo Pretorio, nella versione iOS non è presente un menù dell'applicazione per disattivare o attivare le notifiche in quanto, è possibile farlo attraverso le impostazioni di sistema sotto la voce Notifiche.
Come nel caso di Android, per implementare il servizio di notifiche push nella versione iOS, è stato necessario l'intervento del personale dell'azienda per aggiungere al portale Web nuove funzioni, in modo tale da inviare una notifica non appena è pubblicata una nuova affissione nell'albo pretorio.

Capitolo 6

Conclusioni

L'applicazione descritta, nelle due versioni, è stata pubblicata per il Comune di Chiampo (Vicenza) rispettivamente nel Google Play Store per dispositivi Android e nell' App Store Apple per dispositivi iOS, con il nome Albo Pretorio Chiampo. Questa applicazione è servita come base per un'altro progetto, attualmente in sviluppo, il cui scopo è portare altri servizi della versione Web del portale del cittadino in un'applicazione mobile, la quale comprenderà anche l'albo pretorio.
L'esperienza dello stage è stata molto utile in quanto mi ha permesso di approfondire le conoscenze sulla programmazione di dispositivi mobili apprese nel corso di Programmazione di Sistemi Embedded. Inoltre il tirocinio è stata un'opportunità per conoscere le caratteristiche e le problematiche della cooperazione all'interno di un gruppo in un'azienda e anche un modo per poter applicare sul campo e approfondire le conoscenze apprese durante il percorso di studi. Attualmente la collaborazione con l'azienda sta continuando anche dopo la fine del tirocinio.

Bibliografia

[1] Android Developer: http://developer.android.com/.

[2] Apple Developer: http://developer.apple.com/.

[3] Fantozzi, 2012: Slide del corso di Programmazione di sistemi embedded, http://esp1112.wikispaces.com/.

[4] JavaScript Object Notation (JSON) site: http://www.json.org.

[5] Google Gson Library: http://code.google.com/p/google-gson/.

[6] Mattt Thompson, 2012: AFNetworking Framework, http://github.com/AFNetworking/AFNetworking.

[7] Wikipedia: http://en.wikipedia.org/wiki/Ios, http://en.wikipedia.org/wiki/Android_(operating_system).